

TIBCO ActiveMatrix BusinessWorks™ ActiveAspects Plug-in

User's Guide

*Software Release 1.2
October 2011*

TIBCO provides the two-second advantage™



Important Information

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

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

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

TIBCO, The Power of Now, TIBCO ActiveMatrix BusinessWorks, TIBCO Rendezvous, TIBCO Administrator, TIBCO Enterprise Message Service, TIBCO InConcert, TIBCO Policy Manager, and TIBCO Hawk are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

EJB, Java EE, J2EE, and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

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

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

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

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

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

Copyright © 2010 -2011 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Preface	vii
Changes from the Previous Release of this Guide	viii
Related Documentation	ix
TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Documentation	ix
Other TIBCO Product Documentation	ix
Typographical Conventions	xi
Connecting with TIBCO Resources	xiii
How to Join TIBCOCommunity	xiii
How to Access All TIBCO Documentation	xiii
How to Contact TIBCO Support	xiii
 Chapter 1 Overview	 1
Introduction	2
Aspect Oriented Programming (AOP) Terminology	3
Overview	5
Roles and Responsibilities	6
Process Join Point	7
TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in	8
Process Aspect	8
Advices, Advice Instances and Advice Implementation Instances	11
Advice Configuration Properties	12
Advice Ordering	14
Packaging and Deployment of Aspects	22
Deploying Packaged Aspects in ActiveMatrix BusinessWorks Engine	25
Advice Implementations	26
Java Annotations for Advertising Advice Implementation Metadata	26
Advice Implementation Properties	32
Scopes	35
XML Document Access	37
Packaging and Deployment of Advice Implementations	39
 Chapter 2 Point Cut Query Language	 41
Introduction	42
Query Language Primitives	43
Examples of Point Cuts Defined Using Query Language	49

Chapter 3 Asynchronous Advice Implementations	51
Introduction	52
Asynchronous Advices in ActiveMatrix BusinessWorks Engine	53
Execution Model (Successful Execution)	53
Execution Model (timeout)	53
Threading Model: Asynchronous Advice Implementations	54
Threading Model: Asynchronous Advice Implementations (Timeout)	55
Chapter 4 Hibernate Resume	57
Features of Hibernate Resume	58
Comparison between Checkpointing and Hibernate	59
Defining a Hibernate Advice Implementation	60
Example of Hibernate Advice Implementation	61
Execution of a Hibernate Advice	62
Resuming the Hibernated Job	62
Example of Resuming a Job	63
Using a Database for Hibernation	65
Modifying the Hibernated Data	66
Chapter 5 Object Sharing Between Java Activities and Advice Implementation	67
Overview	68
User Scenarios	69
API's and New Interfaces	71
Use Cases	71
Chapter 6 BWAA Palette	73
Resume	74
Configuration	74
Input	74
Output	74
Chapter 7 Monitoring and Management	75
Introduction	76
getAdviceInstances	77
getAdviceInstanceMetrics	81
getRunningAdviceInstancesCount	85
getRunningAdviceInstances	86
Appendix A ActivityTypes	89

ActivityTypes	90
Appendix B Developing gXML Applications	97
Overview	98
Developing gXML Applications	99
gXML Recipes	106
Parsing	106
Constructing a Data Model Tree Programmatically	108
Validating	117
Navigation	119
Mutation	122
Serialization	124
XPath	125
XSLT	128
XQuery	135
Validation	140
Index	145

Preface

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in (BWAA) extends the capabilities of TIBCO ActiveMatrix BusinessWorks by adding an Aspect Oriented Programming capability. This allows you to modularize and inject crosscutting concerns to enhance your ActiveMatrix BusinessWorks processes at deploy time while keeping the original ActiveMatrix BusinessWorks processes intact. It exposes a JAVA API to build jar files that can alter the execution of a BusinessWorks application.

This preface gives some information on the TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in documentation set, related documentation, and on the conventions used in TIBCO manuals.

Topics

- [Changes from the Previous Release of this Guide, page viii](#)
- [Related Documentation, page ix](#)
- [Typographical Conventions, page xi](#)
- [Connecting with TIBCO Resources, page xiii](#)

Changes from the Previous Release of this Guide

This section itemizes the major changes from the previous release of this guide.

- No major updates in this release.

Related Documentation

This section lists documentation resources you may find useful. The documentation road map shows the relationships between the books and online references in this product's documentation set.

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Documentation

The following documents form the TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in documentation set:

- *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Installation and Configuration* Read this manual for information on product installation.
- *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in User's Guide* Read this manual to learn how to develop, build, and deploy aspects in ActiveMatrix BusinessWorks.
- *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in API Reference* This manual gives information about the JAVA APIs for creating advice implementations in TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in.
- *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Getting Started* Read this manual for detailed information about creating and testing a ActiveMatrix BusinessWorks project using TIBCO Designer, setting development environment in Eclipse, advice development in Eclipse, and the expected final output.
- *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

TIBCO ActiveMatrix BusinessWorks is a pre-requisite for TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in and is used with other products. You may find it useful to read the documentation for the following TIBCO products:

TIBCO ActiveMatrix BusinessWorks

- *TIBCO ActiveMatrix BusinessWorks Concepts* Read this manual before reading any other manual in the documentation set. This manual describes terminology and concepts of ActiveMatrix BusinessWorks, and the other

manuals in the documentation set assume you are familiar with the information in this manual.

- *TIBCO ActiveMatrix BusinessWorks Getting Started* This manual steps you through a very simple example of designing, deploying, and monitoring a ActiveMatrix BusinessWorks process.
- *ActiveMatrix BusinessWorks Process Design Guide* This manual describes how to create, edit, and test business processes using ActiveMatrix BusinessWorks.
- *ActiveMatrix BusinessWorks Palette Reference* This manual describes each of the palettes available in ActiveMatrix BusinessWorks.
- *TIBCO ActiveMatrix BusinessWorks Administration* This manual describes how to use TIBCO Administrator to deploy, manage, and monitor ActiveMatrix BusinessWorks processes.
- *TIBCO ActiveMatrix BusinessWorks Installation* Read this manual for information on installing one or more components of ActiveMatrix BusinessWorks and setting up ActiveMatrix BusinessWorks domain.
- *TIBCO ActiveMatrix BusinessWorks Error Codes* This manual describes errors returned by ActiveMatrix BusinessWorks.
- *TIBCO ActiveMatrix BusinessWorks Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Products

- TIBCO Designer™ software: TIBCO Designer is an easy to use graphical user interface for design-time configuration of TIBCO applications.
- TIBCO Administrator™ software: TIBCO Administrator is the monitoring and managing interface for new-generation TIBCO products such as TIBCO ActiveMatrix BusinessWorks.
- TIBCO Adapter software
- Third-Party Documentation




Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
<i>TIBCO_HOME</i>	<p>Many TIBCO products must be installed within the same home directory. This directory is referenced in documentation as <i>TIBCO_HOME</i>. The value of <i>TIBCO_HOME</i> depends on the operating system. For example, on Windows systems, the default value is C:\tibco.</p> <p>Incompatible products and multiple instances of the same product can be installed into different installation environments.</p>
<i>BW_HOME</i>	<p>TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in installs into the ActiveMatrix BusinessWorks directory within <i>TIBCO_HOME</i>. This directory is referenced in documentation as <i>BW_HOME</i>. The value of <i>BW_HOME</i> depends on the operating system. For example on Windows systems, the default value is C:\tibco\bw\5.9.</p>
code font	<p>Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example:</p> <p>Use MyCommand to start the foo process.</p>
bold code font	<p>Bold code font is used in the following ways:</p> <ul style="list-style-type: none"> • In procedures, to indicate what a user types. For example: Type admin. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [enable disable]
<i>italic font</i>	<p>Italic font is used in the following ways:</p> <ul style="list-style-type: none"> • To indicate a document title. For example: See <i>TIBCO ActiveMatrix BusinessWorks Concepts</i>. • To introduce new terms For example: A portal page may contain several portlets. <i>Portlets</i> are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: MyCommand <i>PathName</i>

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
Key combinations	<p>Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C.</p> <p>Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.</p>
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Connecting with TIBCO Resources

How to Join TIBCOCommunity

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

How to Access All TIBCO Documentation

After you join TIBCOCommunity, you can access the documentation for all supported product versions here:

<http://docs.tibco.com/TibcoDoc>

How to Contact TIBCO Support

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

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

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

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

<https://support.tibco.com>

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

Chapter 1 **Overview**

This chapter introduces TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in.

Topics

- [Introduction, page 2](#)
- [Aspect Oriented Programming \(AOP\) Terminology, page 3](#)
- [Overview, page 5](#)
- [TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in, page 8](#)
- [Advices, Advice Instances and Advice Implementation Instances, page 11](#)
- [Advice Implementations, page 26](#)

Introduction

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in (BWAA) extends the capabilities of TIBCO ActiveMatrix BusinessWorks by adding an Aspect Oriented Programming capability. This allows you to modularize and inject crosscutting concerns to enhance your ActiveMatrix BusinessWorks processes at deploy time while keeping the original ActiveMatrix BusinessWorks processes intact. It exposes a JAVA API to build applications, which are called advice implementations that can alter the execution of ActiveMatrix BusinessWorks Application.

This chapter provides an overview of features of the TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in, TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Concepts, TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Resources, and AOP terminology.

[Chapter 2](#) describes the Point Cut Query Language used for writing point cut expressions.

[Chapter 3](#) describes the Asynchronous Advice Implementations.

[Chapter 4](#) describes the Hibernate/Resume feature of TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in.

[Chapter 5](#) describes the JAVA Object sharing between ActiveMatrix BusinessWorks Java activities and Advice implementations.

[Chapter 6](#) provides information about the BWAA palette.

[Chapter 7](#) describes the Monitoring and Management feature.

[Appendix A, ActivityTypes](#) provides information about the activity types of all TIBCO ActiveMatrix BusinessWorks activities that are useful for the user to generate expressions in Point cut query language.

[Appendix B, Developing gXML Applications](#) provides information for developing gXML applications.

Aspect Oriented Programming (AOP) Terminology

It is important to understand the terms used in the document about the product. [Table 2](#) describes the AOP terminology used throughout this document.

Table 2 AOP Terminology

Name	Description
Process Aspect	An XML resource that can alter the behavior of a TIBCO ActiveMatrix BusinessWorks application by injecting user defined code at specific points within a process. It contains the declaration of most of the concepts in this list.
Advice Implementation	User developed JAVA class that executes one or more cross-cutting concerns around an activity.
Advice Implementation Instance	A JAVA object instance of an advice implementation.
Advice	A configured advice implementation that is placed around a given activity. Advices are defined in aspect files. Advice is a design-time concept.
Advice Instance	A particular instance of an advice. Advice instance is a run-time concept.
Process Join Point	A specific point in a process that supports injection of user defined advices.
Point Cut	An expression or query that selects the join points based on certain conditions. Point cuts are defined in aspect files.
Target Activity	An activity whose input, output, or exception messages are intercepted by an advice.
Aspect Library	A JAR file that contains one or more aspects.
Aspect Implementation Library	A JAR file that contains one or more aspect implementation files.

Table 2 AOP Terminology

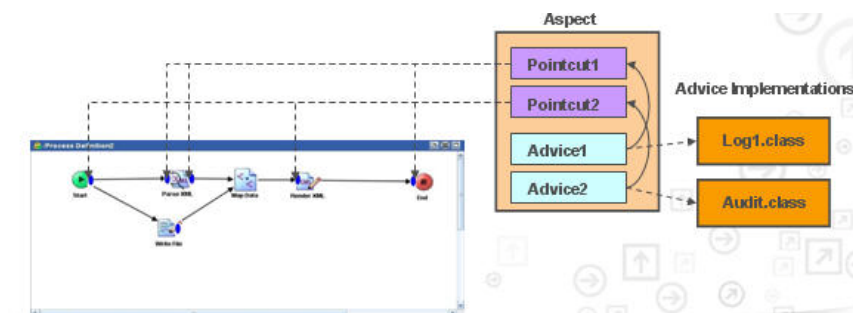
Name	Description
aspectPath	Path where aspect libraries are located.

Overview

A Process-Oriented Aspect (POA) alters the execution of a process by injecting **Advices**, which are user defined code, at specific points of the process called **Join Points**. The selection of the Join Points is made based on the expressions called **Point Cuts**.

An Aspect is the collection of Point Cuts and Advices. Aspects implement features that cut across different layers of a BW application (that is, across different BW processes). One of the key characteristics of the POA style programming is that these features can be developed, packaged and deployed independent of TIBCO ActiveMatrix BusinessWorks applications. This provides a flexible model and allows a different user, that is, the Aspect Developer to develop business logic that can alter post design-time, the behavior of a TIBCO ActiveMatrix BusinessWorks application.

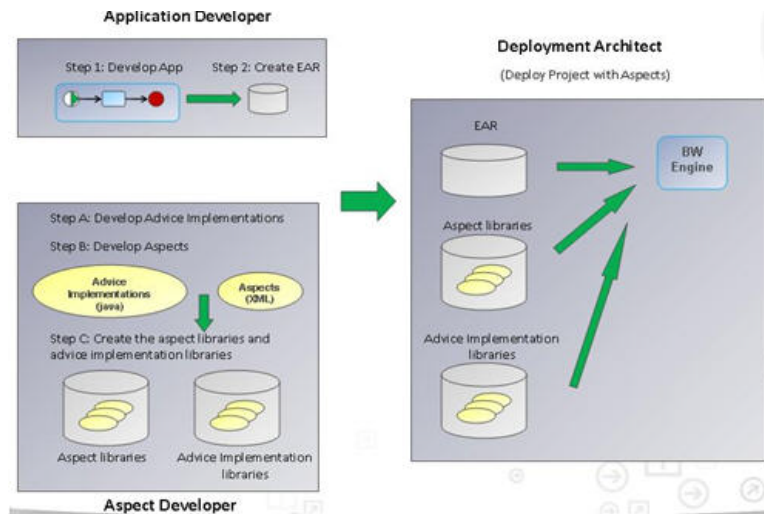
Figure 1 TIBCO ActiveMatrix BusinessWorks ActiveAspects Behavior



Aspects are packaged in Aspect libraries, which are JAR files. These are different from the JAR files that contain Advice Implementations. These JAR files *must* be available in the aspect path in order to allow the aspects to be loaded by the TIBCO ActiveMatrix BusinessWorks engine. The aspect injection process happens at run-time through a process called **In-Memory XML Weaving**.

If the Aspect libraries are not available in the aspectPath, the ActiveMatrix BusinessWorks engine executes the ActiveMatrix BusinessWorks application as is (that is, without altering the behavior defined by the TIBCO ActiveMatrix BusinessWorks developer). TIBCO ActiveMatrix BusinessWorks provides a platform for developing both TIBCO ActiveMatrix BusinessWorks applications as well as process-oriented aspects that can be injected in these applications.

Figure 2 TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Process



Roles and Responsibilities

Application Developer	This developer of an TIBCO ActiveMatrix BusinessWorks application creates processes and generates an EAR file for deployment. The application developer is aware of the potential injection of POAs but is not responsible for developing them.
Advice Implementation Developer	This developer develops the JAVA code that gets injected into the TIBCO ActiveMatrix BusinessWorks processes via the POAs. In general, this developer may have very little information about the TIBCO ActiveMatrix BusinessWorks application where the aspects are injected. This user's responsibility is to create a robust piece of code that addresses a cross-cutting concern for the TIBCO ActiveMatrix BusinessWorks application.
Aspect Developer	This developer creates the aspects that are applied to one or more TIBCO ActiveMatrix BusinessWorks applications. An extensive knowledge about the TIBCO ActiveMatrix BusinessWorks application as well as the available advices is required. These advices can be injected in the TIBCO ActiveMatrix BusinessWorks application. In some organizations the Aspect Developer may also play the role of the Advice Implementation Developer. For simplicity, this document assumes that there is only one user, the Aspect Developer, who plays both roles.
Deployment Architect	Deploys both a TIBCO ActiveMatrix BusinessWorks application and its associated aspects.

Process Join Point

A Process Join Point is a well defined point in a process flow where a special event occurs. A special event could be the beginning of the execution of an activity or the end of the execution of an activity. By inserting advices in join points, a TIBCO ActiveMatrix BusinessWorks user can alter the execution of the process. [Table 3](#) describes the join points supported by TIBCO ActiveMatrix BusinessWorks.

Table 3 TIBCO ActiveMatrix BusinessWorks Supported Join Points

Activity	Functionality
Before Activity	The advice runs before an activity is executed.
After Returning Activity	The advice runs after an activity is successfully completed.
After Throwing Activity	The advice runs after an activity throws an exception.
After Activity	<p>The advice executes after an activity either completes successfully or it throws an exception.</p> <p>This is equivalent to JAVA's finally construct.</p>

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in consists of resources.

Process Aspect

Process Aspect is a new concept in TIBCO ActiveMatrix BusinessWorks that provides a way to alter the execution of a process by injecting the user-defined code in specific points of a process. In general, Aspects are defined, packaged, and deployed completely independent of TIBCO ActiveMatrix BusinessWorks applications (like EAR files).

Two main parts of an Aspect are:

- **One or more Point Cuts** - Provide the selection or the query logic for selecting the join points where advices are inserted.
- **One or more Advices** - Provide a user-defined code that will be executed at a specified point cut.



In TIBCO ActiveMatrix BusinessWorks, an Aspect is implemented with an XML file that has a ".bwaspect" extension.

The following snippet shows the pseudo-schema of the TIBCO ActiveMatrix BusinessWorks aspect:

```
<aspect xmlns="http://schemas.tibco.com/bw/poa"
targetNamespace="xsd:anyURI"
xmlns:xsi="xsd:anyURI" xsi:schemaLocation="xsd:anyURI"
order=" xs:unsignedByte"?>
<documentation>...</documentation>?
<pointcut name="NCName"> +
<documentation>...</documentation>?
  <query queryLanguage="xsd:anyURI"?>...</query>
</pointcut>
<advice name="NCName" pointcut="NCName"> +
<documentation>...</documentation>?
<activity where="Before | AfterReturning | AfterThrowing | After"
exceptionType="xsd:string">
<implementation.java className="xsd:String">
</activity>
<properties>
<property name = "NCName">...</property>
```

```

</properties>
</advice>
</aspect>

```



An aspect has a required `@targetNamespace` attribute, whose value *must* be unique in the context of an ActiveMatrix BusinessWorks engine. At the ActiveMatrix BusinessWorks engine initialization time, if two or more aspects are found with the same `targetNamespace`, it throws a `BWAspectConfigurationException` and exits. An aspect also has an optional `@order` attribute, which is used to establish an execution order for advices that are inserted in the same join point.

A point cut is used to select all the join points where advices are injected. The selection is implemented by performing a query on the ActiveMatrix BusinessWorks Project info set. For details refer to [Chapter 2, Point Cut Query Language](#).



A point cut has a required `@name` attribute, whose value must be unique in the context of an aspect. At the ActiveMatrix BusinessWorks engine initialization time, if more than one point cuts are found with the same name, the ActiveMatrix BusinessWorks Engine throws a `BWAspectConfigurationException`.

A common usage of an aspect is to contain more than one point cut. This is useful especially when the aspect contains multiple advice definitions. At run-time, when advices are injected in the business processes, only the point cuts that are actually referenced by advices are used. Hence, all the point cuts not referenced by advices are discarded by the engine at run-time.

An advice is a configured user-defined code that gets injected in a process at run-time, thereby changing its behavior. The location of where the advice gets injected is specified by referencing a point cut, via the required `@pointcut` attribute.



An advice cannot be inserted anywhere in a process other than at the valid point cuts.

Currently supported join points can only alter the activity behaviors. There are four types of join points as given:

- Before
- AfterRunning
- AfterThrowing
- After

An `AfterThrowing` advice can also specify an exception type `QName` via the `@exceptionType` attribute.



Use `QName` attribute to select a particular type of exception. If this attribute is not set, any exception thrown by the activity triggers the execution of the advice.

An advice has a required `@name` attribute, whose value must be unique in the context of an aspect. At run-time, the ActiveMatrix BusinessWorks engine throws a `BWAspectConfigurationException` if it finds two or more advices with the same name. An advice also has an implementation, which is the actual code that gets executed at run-time.



Currently TIBCO ActiveMatrix BusinessWorks supports only JAVA for defining advice implementations.

An advice implementation JAVA class, which is specified in the `@className` attribute of the implementation element, must implement a specific contract. For more information about this contract and the API that is available to advice implementations, refer [Advice Implementations on page 26](#).



The ActiveMatrix BusinessWorks engine does not halt its execution if an exception such as a `BWAspectConfigurationException`, is thrown during its initialization process.

Aspects, point cuts, and advices can have an optional documentation element that can be used to store comments associated with these entities.

Advices, Advice Instances and Advice Implementation Instances

This section describes the difference between an advice and an advice instance.



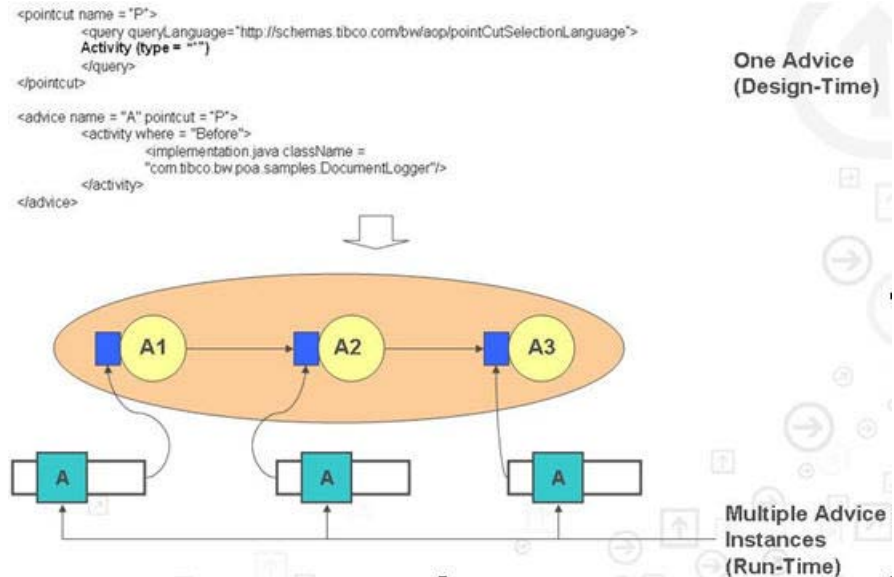
Advice is a *design-time* concept whereas Advice Instance is a *run-time* concept.

An advice defined at design-time in an aspect XML file, has a configuration that includes a reference to a point cut. At run-time, one advice instance is created for every join point that is selected by the point cut. Unless **Scoping** is specified this is the default behavior. For details refer to, [Scopes on page 35](#).

The advice instance is actually the entity that is executed at run-time, not the advice.

[Figure 3](#) shows an advice that gets instantiated and injected before all activities. Since the process has three activities, three advice instances are actually created at run-time.

Figure 3 Advice Instances Created at Run-Time



At run-time, the TIBCO ActiveMatrix BusinessWorks engine may not always create three `DocumentLogger` instances. It depends on the scope attribute of the `DocumentLogger` class.



If the scope is "Application", only one java object instance of `DocumentLogger` is actually created at run-time per TIBCO ActiveMatrix BusinessWorks application.

This is very important to understand since it represents the key difference between an advice instance and an advice implementation instance.

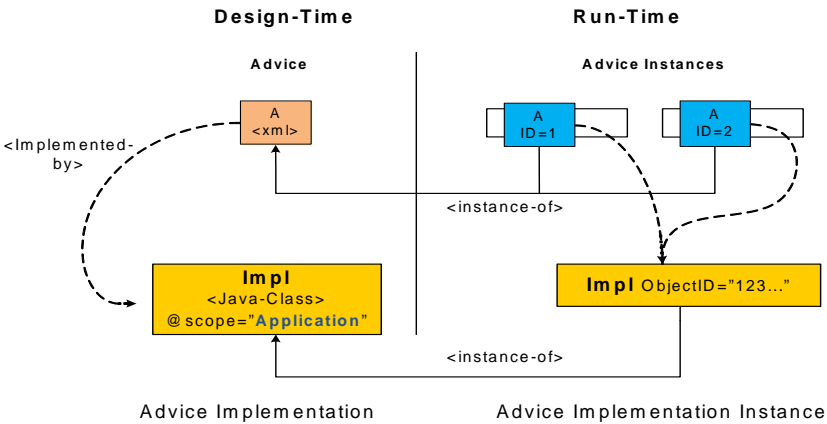


The instance of the advice implementation (such as the object of the JAVA class that provides the implementation of the advice) is not the same as the advice instance.

Three advice instances may in fact share the same advice implementation java object instance.

Figure 4 shows two advice instances sharing the same advice implementation instance.

Figure 4 Two Advices Sharing Same Advice Implementation Instance



Advice Configuration Properties

Advice implementations (that is, Java classes) can define configuration properties. These properties provide a way for advices to configure the execution of their associated Java class. This is very useful especially when different advices that share the same implementation, want to execute it using different parameter values. For details about how advice implementations can define configuration properties, refer [Advice Implementation Properties on page 32](#).

For each configuration property defined in the advice implementation, there can be a set of values in the advice configuration. If an advice implementation property is not set in an advice, the default value that is specified in the java class, if exists, is used.

Each advice has a `<properties>` element that contains these property values.



Advice implementation properties that are not required, do not need to be set in the advice configurations. For details see [Advice Implementation Properties on page 32](#).

Example 1

An advice with two properties:

```
<advice name = "Advice1" ...>
<activity where = "Before">
<implementation.java className =
"com.tibco.bw.poa.samples.JMSPropertyChangerWithConfig"/>
</activity>
<properties>
<property name = "propertyToModify">Bar</property>
<property name = "propertyToModifyValue">BarValue1</property>
</properties>
</advice>
```

In this case, the advice implementation (that is, the `JMSPropertyChangerWithConfig` java class) defines two configurable member variables:

- `propertyToModify`
- `propertyToModifyValue`

These two properties are visible and configurable from an advice. This is visible in the case of [Example 1 on page 13](#) which is `Advice1`. The `@name` attribute of the `<property>` element *must* match the name of the advice implementation property. If a match is not found, the ActiveMatrix BusinessWorks Engine throws an `AspectException` at the time of its initialization.

Another advice could share the same implementation and configure it in a different way.

Example 2

Advice2 configures its implementation to mutate the same property "Bar" but with a different value:

```
<advice name = "Advice2" ...>
<activity where = "Before">
<implementation.java className =
"com.tibco.bw.poa.samples.JMSPropertyChangerWithConfig"/>
```

```

</activity>
<properties>
<property name = "propertyToModify">Bar</property>
<property name = "propertyToModifyValue">BarValue2</property>
</properties>
</advice>

```

Example 3 **Advice3 configures its implementation to mutate a different property altogether:**

```

<advice name = "Advice3" ...>
<activity where = "Before">
<implementation.java className =
"com.tibco.bw.poa.samples.JMSPropertyChangerWithConfig"/>
</activity>
<properties>
<property name = "propertyToModify">Abc</property>
<property name = "propertyToModifyValue">Xyz</property>
</properties>
</advice>

```



The advices cannot share the same "advice implementation *instance*". However, they can certainly share the same "advice implementation".

Advice Ordering

Advices get injected in processes based on the point cuts that are associated with them. At run-time, multiple advices can be injected in the same join point. Sometimes, the order in which these advices get executed is very important and needs to be controlled by the Aspect Developer. The ActiveMatrix BusinessWorks engine executes the advices in a specific order, which is computed based on a priority order associated with each advice.

Execution of advices also depends on the process execution flow; which decides the availability of input or output XML document that is to be used by the advice.

Since the advices that are injected in a specific join point can be specified either in the same or in different aspect files, the priority order of an advice is computed based on:

1. The value of the `@order` attribute that is specified on the aspect that defines the advice.
2. The location (such as the position) of the advice element inside the aspect definition (such as the location inside the XML file). An advice with a position

that is closer to the root element `<aspect>` has a higher priority than an advice that is located farther than the root element.

The optional `@order` attribute that can be specified on an aspect is used to establish a priority order between different aspects that are applied to a BW project. All advices defined in the same aspect share the same `@order` attribute value.



This value can range between 1 and 255. The lower the number, the higher the priority of the aspect. If the `@order` attribute is not specified, 255 is used by default, which means that the aspect has the lowest priority.

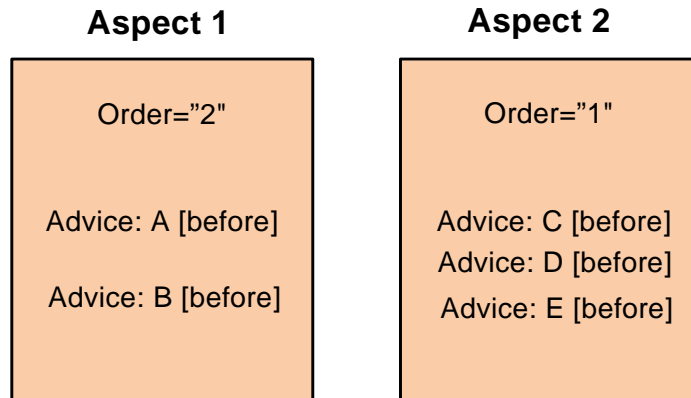


Although the `@order` attribute is `xsd:unsignedByte`, 0 is an invalid value. If 0 is used, a validation error is thrown.

The order in which the advices are specified in an aspect file is very important. The ActiveMatrix BusinessWorks engine uses this order to execute the advices that are run when a specific event occurs. This is true regardless of the event (for example, before executing an activity, after an activity returns successfully, and so on). For example, if a target activity throws an exception and there are "after" advices as well as "after throwing" advices injected after the target activity, the order in which these advices execute is influenced by the order in which they appear in the aspect file.

Figure 5 shows two aspect definitions - Aspect-1 and Aspect-2, each one defining a few "Before" advices. The order of Aspect-1 is "2" and the order of "Aspect-2" is 1, which means that Aspect-2 has a higher priority order than Aspect-1.

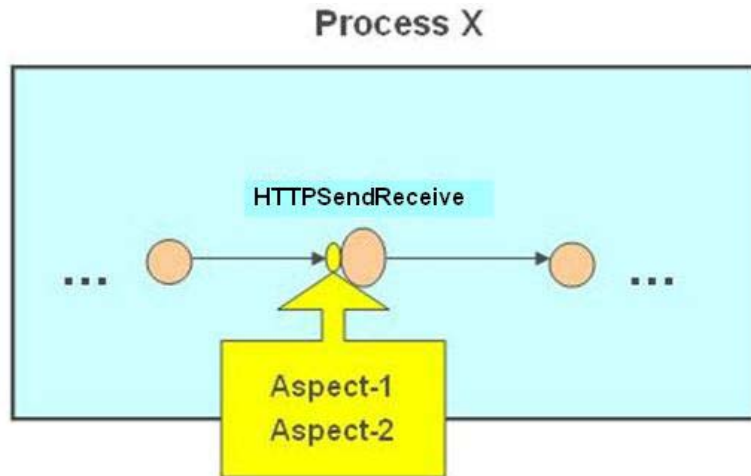
Figure 5 Aspect Definitions



Assume that the order in which these advices appear in the aspect definitions is exactly the order in which they are shown in this diagram (For example, A is before B, C is before D, and D is before E).

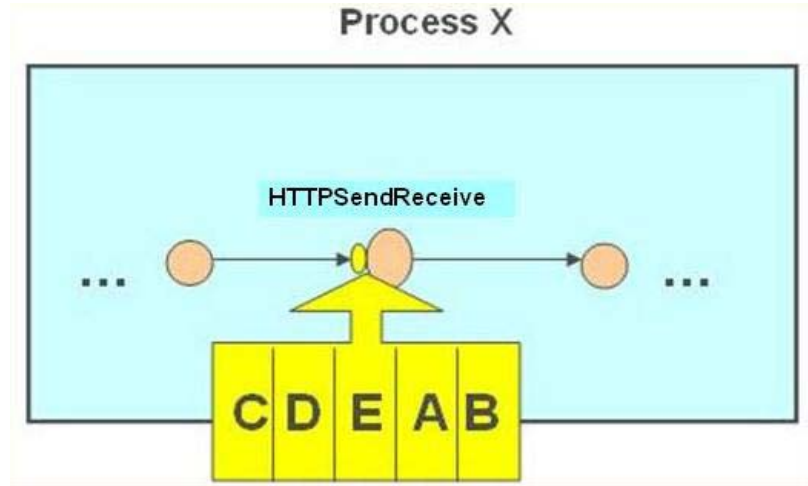
Figure 6 shows that after evaluating the point cuts associated with these advices, the ActiveMatrix BusinessWorks Engine injects these two aspects before a specific activity of a process.

Figure 6 Injecting Two Aspects Before Specific Activity in a Process



Since Aspect-2 has a higher priority than Aspect-1, all its "Before" advices will execute before the advices defined in Aspect-1. The order of execution of the advices defined in the same aspect is given by the order in which they appear in the aspect XML file. Therefore, the order in which these advices get executed is shown in Figure 7.

Figure 7 Sequence of Execution of the Advices



If two aspects have the same order, their corresponding advices execute in an order that is not deterministic.

Advice Execution Model

The advices that are injected in a specific join point execute always in sequence. As shown in [Advice Ordering on page 14](#), the order in which these advices execute is computed based on the way the aspects are configured as well as the type of the join point.

Here are the details about how these aspects get executed by focusing on other characteristics that are not related to ordering.

It is important to note that an advice always has access to the XML Document as well as the Process Context that is available in the join point where it is injected. The Process Context can provide access to other XML documents that were contributed by the previous activities (such as activities that executed before the join point).



Advices do not contribute to the Process Context's new XML documents that are visible to TIBCO ActiveMatrix BusinessWorks activities. Only activities can do that. XML documents contributed by advices are visible only to other advices that execute downstream in the process.



Do not alter this way of sharing data between advice instances that execute in the same process.

The information available to an aspect for each of the supported join point is:

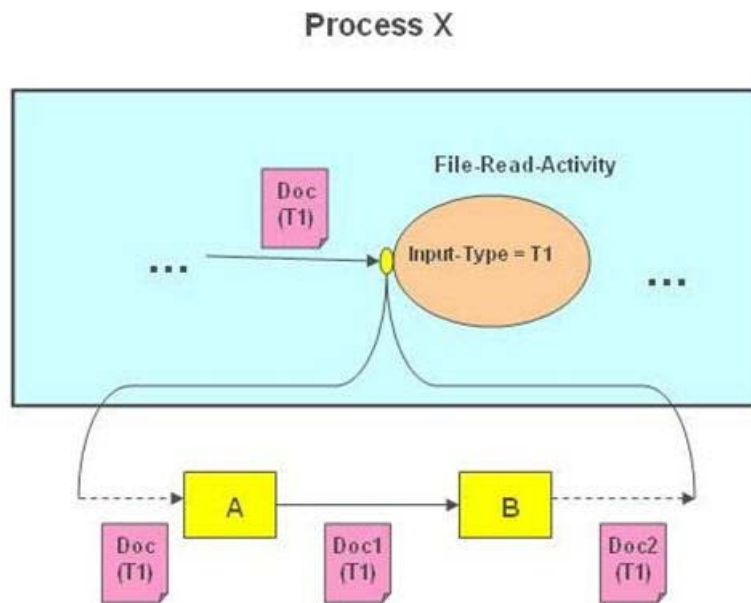
- The XML document that is passed to a "Before" advice is the same XML document that had been passed to the target activity, if the advice was not injected in the process.



The advice can alter the document but it cannot change its structure.

- The XML document that is produced by the advice must be valid against the same schema (such as the schema of the input element), which is defined by the target activity. The same rule applies to all the subsequent advices that are inserted in the same join point. This means that all the XML documents that flow through all the "Before" advices that are injected in the same join point share the same schema. This is shown [Figure 8](#).

Figure 8 XML Document Passed to a "Before" Advice



In [Figure 8](#), the File-Read-Activity's input type is T1. The advices at run-time that are injected before this activity receive the XML documents, that are valid against the same schema (such as, they are instance of T1).

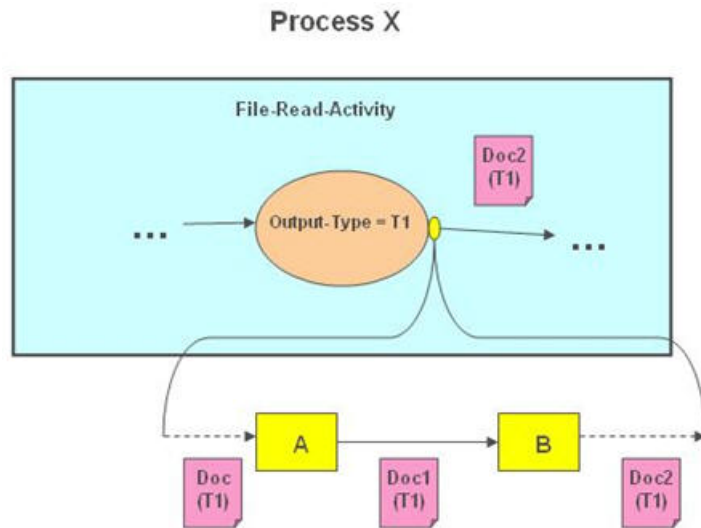
- The XML document passed to an "After Returning" advice is the XML document that is generated by the target activity, if the activity has an output type or it is a null object otherwise.



The advice can alter the document but it cannot alter its structure.

- The XML document that is produced by the advice must be valid against the same schema (such as, the schema of the output type), which is defined by the target activity.
- The same rule applies to all the subsequent advices that are inserted in the same join point. This implies that all the XML documents that flow through all the "After Returning" advices that are injected in the same join point, share the same schema. This is shown in [Figure 9](#).

Figure 9 XML Document Passed to An "After Returning" Advice



- The XML document that is passed to an "After Throwing" advice, is the XML document that represents the exception thrown by the target activity.



The advice can alter the document (for example, change the exception message, add more information to the exception, and so on) but cannot alter its structure.

- The XML document that is produced by the advice must be valid against the schema of the output exception type, which basically means that the advice cannot change the exception type that is thrown by the activity.

Since an activity can report multiple exception types, an "After Throwing" advice has to support multiple XML schema types, which usually makes it difficult to develop. If an advice is interested in a particular exception type, it can use the @exceptionType attribute to specify its QName. The value of this attribute has the following format:

```
<exceptionType> ::= <exceptionTypeQName> (", "
<exceptionTypeQName> )*
<exceptionTypeQName> ::= "{" <exceptionTypeNamespace> "}"
<exceptionTypeLocalName>
```

where,

- <exceptionTypeNamespace> is the target namespace of the XML Schema that defines the exception type.
- <exceptionTypeLocalName> is the name of the exception type.

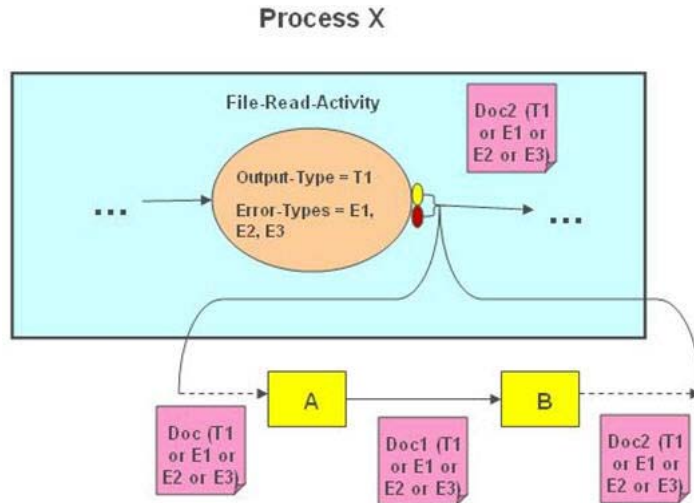
This syntax supports multiple QName values in-case an advice may want to register its interest in more than one exception type.



To provide a flexible way to select a wide range of exception types, both the exception type namespace as well as the exception type name support "*" wildcards. This is the only wildcard supported.

In such cases, the advice is executed only when an exception instance of that type is thrown by the target activity. All the subsequent advices that are inserted in the same join point follow these rules. The "After Throwing" advices should not throw back the exception that is passed to them. Instead, they should return the exception message when they finish executing. If an exception is thrown by an "After Throwing" advice, the engine treats it as any other exception thrown by other types of advices. [Figure 10](#) shows an example of two "After Throwing" advices.

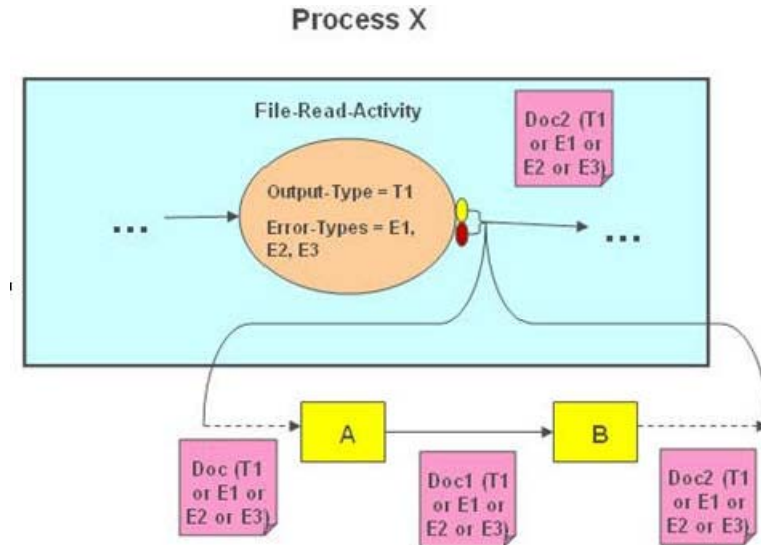
Figure 10 Example of Two "After Throwing" Advices



- The XML document that is passed to an "After" advice represents either the output of the target activity or one of its exceptions. The advice can alter the document (for example, change the output message, change the exception message, and so on) but it cannot alter its structure. The XML document produced by the advice must be valid against either the schema of the output XML document or the schema that describes the exception thrown by the activity. This basically means that the same type of document that is passed to an "After" advice is passed to any other subsequent "After" advices that might run in the same join point. While writing an "After" advice, a developer should be careful since it needs to handle different semantics (for example, successful returns as well as multiple exception types).

An "After" advice cannot use the @exceptionType attribute to register its interest in a particular exception type. If an exception is thrown by an "After" advice, the ActiveMatrix BusinessWorks engine treats it as any other exception thrown by other types of advices. [Figure 11](#) shows an example of two "After" advices.

Figure 11 Example of Two "After" Advices



Any exception thrown by an advice is propagated as a `RuntimeException` and handled by the ActiveMatrix BusinessWorks engine as thrown by the target activity. The engine processes it based on the business logic defined in the process containing the target activity.

Packaging and Deployment of Aspects

This section describes the structure of an aspect JAR and the deployment process of aspect libraries.

Aspects are packaged together in JAR files, that are referred to as "aspect JARs".

An aspect JAR is a JAR file that contains a file with the ".AMF" (Aspect Manifest File) extension in the META-INF folder.



There can be only one file with the ".AMF" extension in that folder.

The Aspect Manifest File contains information about where all the aspects are located inside the aspect JAR file. These locations can be specified in two ways:

- By specifying a folder name inside the JAR file, all aspects that are part of that folder are loaded by the ActiveMatrix BusinessWorks engine. The ActiveMatrix BusinessWorks engine does not locate aspects in the subfolders. The root folder is specified using "" (which is an empty string).

- By specifying the full name (such as, including the folder location) of an individual aspect inside the JAR file.

The AMF file is an XML file that *must* be valid against the Aspect Manifest File XML schema. When loading an aspect JAR file, the ActiveMatrix BusinessWorks engine validates the Aspect Manifest File against this schema. If validation errors are found, the ActiveMatrix BusinessWorks engine throws an `AspectException`.

The following snippet shows the Aspect Manifest File pseudo-schema:

```
<bwpoa xmlns = "http://schemas.tibco.com/bw/poa/manifest" ...>
<aspects>
<aspectsFolder>...</aspectsFolder>*
<aspect>...</aspect>*
</aspects>
</bwpoa>
```



A JAR file that contains aspects but does not contain an Aspect Manifest File, is not recognized by the ActiveMatrix BusinessWorks engine as an aspect JAR. Hence none of the aspects defined in that JAR are loaded by the ActiveMatrix BusinessWorks engine.

Examples of Aspect Manifest File

An aspect JAR with the following structure:

```
MyAspects.jar
  META-INF
    Aspects.amf
  AuditAspects
    Aspect1.bwaspect
    Aspect2.bwaspect
  ExternalMessages
    AspectX.bwaspect
    AspectY.bwaspect
    AspectZ.bwaspect
```

If the `aspects.amf` file is the following:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<bwpoa
xmlns = "http://schemas.tibco.com/bw/poa/manifest"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://schemas.tibco.com/bw/poa/manifest
bwpoa.xsd">
<aspects>
<aspectsFolder>auditAspects</aspectsFolder>
```

```
</aspects>
</bwpoa>
```

Then,

only `Aspect1.bwaspect` and `Aspect2.bwaspect` are loaded by the ActiveMatrix BusinessWorks engine. The aspects specified in the `s` folder are not loaded.

However, if the `aspects.amf` file is the following:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<bwpoa
xmlns = "http://schemas.tibco.com/bw/poa/manifest"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://schemas.tibco.com/bw/poa/manifest
bwpoa.xsd">
<aspects>
<aspectsFolder>auditAspects</aspectsFolder>
<aspectsFolder>auditAspects/ExternalMessages</aspectsFolder>
</aspects>
</bwpoa>
```

Then,

all the aspects defined in this aspect JAR are loaded by the ActiveMatrix BusinessWorks engine. The same behavior could be accomplished by specifying some or even all of the aspects individually. For example, the following manifest file would have the same result as the previous one:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<bwpoa
xmlns = "http://schemas.tibco.com/bw/poa/manifest"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://schemas.tibco.com/bw/poa/manifest
bwpoa.xsd">
<aspects>
<aspect>auditAspects/Aspect1.bwaspect</aspect>
<aspect>auditAspects/Aspect2.bwaspect</aspect>
<aspectsFolder>auditAspects/ExternalMessages</aspectsFolder>
</aspects>
</bwpoa>
```

Deploying Packaged Aspects in ActiveMatrix BusinessWorks Engine

Multiple aspect JARs can be deployed in a ActiveMatrix BusinessWorks Engine. When the engine starts up, all these aspect JARs are loaded and their aspects are weaved into the ActiveMatrix BusinessWorks Project.



All the aspect JARs loaded by the ActiveMatrix BusinessWorks engine at run-time must be located in the same folder.

The name of the folder is specified through the following JAVA system property:
aspectPath

This property can be specified in the `bwengine.tra`. For example:

```
# BW Aspect Definition Files
java.property.aspectPath
%BW_HOME%/examples/poa/Scenario1/Aspects
```

Similarly, in the case of folders specified inside aspect JARs, the ActiveMatrix BusinessWorks engine does not look at sub-folders when loading aspect JARs. The engine loads only the JARs that are part of this top level folder.

The engine writes out information about all advices that are integrated in the TIBCO ActiveMatrix BusinessWorks project. For each process, the engine analyzes the advices that are configured to be integrated and skips the ones that cannot be integrated due to product limitations. After finishing weaving of a process, the engine logs out all advices that were skipped.

For more information about the advices that cannot be skipped, read the *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Release Notes* or contact TIBCO Support.

Advice Implementations

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in provides support for executing advices developed in JAVA. TIBCO ActiveMatrix BusinessWorks defines both a JAVA API as well as a set of annotations that have to be used for creating advice implementations.

Relationship between an advice and its implementation is very important.



An advice is a configured instance of an advice implementation.

Multiple advices can be implemented with the same implementation. Therefore, there is a many-to-one relationship between advices and their associated java implementation.

An advice implementation provides the business logic of an advice.



Multiple advices can be implemented with the same Java class.

Java Annotations for Advertising Advice Implementation Metadata

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in defines a couple of java annotations that are used for advertising aspect specific metadata. These annotations are defined in the package `com.tibco.bw.poa.runtime.annotation`.

The following section describes these annotations.

The @AdviceImpl Java Annotation

This annotation is used for tagging java classes as advice implementations. This annotation is specified on a class, in the following way:

```
@AdviceImpl
public class MyAdviceImplementation ... {
}
```


Table 4 describes the 5 optional parameters.

Table 4 @AdviceImpl Java Annotation Optional Parameters

Name	Type	Description
<code>scope</code>	String	Specifies the instantiation scope of the advice implementation.
<code>dataAccess</code>	String	Advertises the data access mode (for example, read-only vs. read-write).
<code>hibernatesJobs</code>	Boolean	Advertises whether the advice implementation hibernates jobs
<code>targetKind</code>	String	Advertises the kind of the target or join point where the advice implementation gets instantiated.
<code>targetFilter</code>	String	Advertises an activity type name, which can be used to further narrow down the scope of the target.

scope

Advice implementations get instantiated in different scopes, which can be controlled using the `scope` parameter. The valid values of this parameter are the following:

Values	Description
ADVICE	The advice implementation is instantiated once for every advice instance that is using it.
APPLICATION	One instance of the advice implementation is created for an application. In TIBCO ActiveMatrix BusinessWorks 5.9, the application equals to a BW Project, which means that one instance of the advice is created in a ActiveMatrix BusinessWorks engine.

If the `scope` parameter is set to a different value than the one mentioned above, the ActiveMatrix BusinessWorks engine throws an `AspectException` at run-time, when the engine gets initialized.

This example of an advice implementation that uses the application scope:

```
@AdviceImpl (
  scope = "APPLICATION"
)
```

```
public class MyAdviceImplementation ... }  
}
```



The default value of this parameter, which is used when the parameter is not explicitly set by the user, is "ADVICE".

dataAccess

Advice implementations access the data in a way that can be classified in two categories:

- Advice implementations that need **read-only** access to the data
- Advice implementations that need **read-write** access to the data

An advice implementation advertises the category it belongs to by setting the `dataAccess` parameter. The valid values of this parameter are the following:

Values	Description
READ-WRITE	The advice implementation mutates the data, hence needs read or write access to it.
READ-ONLY	The advice implementation <i>does not</i> mutate the data, hence needs read-only access to it.

If the `dataAccess` parameter is set to a different value than the ones mentioned above, the ActiveMatrix BusinessWorks engine throws an `AspectException` at run-time, when the engine gets initialized.

Following is an example of an advice implementation that mutates the XML document:

```
@AdviceImpl (  
  dataAccess = "READ-WRITE"  
)  
public class MyAdviceImplementation ... {  
  public void execute(N input, AspectProcessContext context) {  
    // the advice implementation mutates the data here...  
  }  
}
```



The default value of this parameter, which is used when the parameter is not explicitly set by the user, is "READ-ONLY".

Hence, an advice implementation that mutates the data must explicitly set the `dataAccess` parameter to "READ-WRITE". If an advice implementation that mutates the data does not set this parameter appropriately, a `ClassCastException` is thrown at run-time, when the advice implementation tries to use a mutable XML processing context.

hibernatesJobs

Advice implementations can hibernate and resume jobs. If an advice implementation uses this feature, then it must explicitly declare that it is done through the `hibernatesJobs` parameter. The valid values of this parameter are:

Values	Description
true	The advice implementation calls an API to hibernate a job.
false	The advice implementation does not call an API to hibernate a job.

Following is an example of an advice implementation that hibernates jobs:

```
@AdviceImpl (
hibernatesJobs = true
)
public class MyAdviceImplementation ... {
public void execute(N input, AspectProcessContext context) {
...
if (need_to_hibernate_job) {
context.setHibernateJobEnabled(0);
}
}
}
```

The default value of this parameter, which is used when the user doesn't explicitly set it, is false.



By default, advice implementations do not hibernate jobs.

If an advice implementation attempts to call the API to hibernate a job without explicitly setting the `hibernatesJobs` parameter to true, an `AspectException` is thrown at run-time. This exception may result in the job to be terminated abnormally, depending on how exceptions are handled at the process level.

When accessing the incoming XML document, advice implementations may or may not be dependent on a particular schema. For example, an advice implementation that changes the value of a JMS property in a JMS message sent by the `JMSQueueSendReceive` activity is dependent on the format (like XML schema) of the `JMSQueueSendReceive` activity's input message.

For this advice implementation to work, it has to be hardcoded to expect the XML document in this specific format.



Activities that are hardcoded to a specific XML schema can only be instantiated and executed in a context where the incoming document conforms to that particular schema.

A way is provided to the advice implementation developer to declare a dependency of a particular instantiation context through the use of two parameters defined as part of the `@AdviceImpl` annotation: **targetKind** and **targetFilter**.

targetKind

This is used for specifying the kind of the target or join point where the advice is instantiated. This parameter can have one of the following values:

Values	Description
ACTIVITY-BEFORE	The advice implementation must be instantiated only before activities.
ACTIVITY-AFTER-RETURNING	The advice implementation must be instantiated only after an activity, on the path that executes when the activity returns successfully.
ACTIVITY-AFTER-THROWING	The advice implementation must be instantiated only after an activity, on the path that executes when the activity throws an exception.
ACTIVITY-AFTER	The advice implementation must be instantiated only after an activity, on the path that executes regardless on whether the activity returns successfully or throws an exception
"" (empty string)	The advice implementation is not dependent on a particular context .

If the `targetKind` parameter is set to a different value than the ones mentioned above, the ActiveMatrix BusinessWorks engine throws an `AspectException` at run-time, when the engine gets initialized.

The default value of this parameter, which is used when the user doesn't explicitly set it, is `""`. By default, advice implementations do not depend on a particular context (can be executed anywhere).

targetFilter

When the `targetKind` parameter is set, the developer of the advice implementation can also specify a target filter, using the `targetFilter` parameter, which is used to further narrow down the scope of the instantiation context. Using this parameter, the developer can specify the type of the activity around which the implementation can be instantiated. For example, the developer can specify the advice implementation that can only be instantiated in the context of a `FileReadActivity`. The value of the parameter (the filter) is the same as the value of the `activity()` primitive's "type" parameter that is defined as part of the [Point Cut Query Language](#).

The default value of this parameter, which is used when the advice implementation developer does not explicitly set it, is `""`. By default, advice implementations can be executed in any context.

Following is an example of an advice implementation that must be instantiated and executed before `JMSQueueSend` activities.

```
@AdviceImpl(
    targetKind = "ACTIVITY-BEFORE",
    targetFilter = "bw.JMSQueueSendActivity"
)
public class JMSPropertyChanger ... {
}
```

When the Aspect engine gets initialized, the engine checks these two parameters and uses them to validate the aspect configurations. If an advice is about to be instantiated in a context that is not valid, the engine throws an exception.



An exception is thrown at engine initialization time and not at run-time during the processing of an incoming request.

Important: It ensures that the engine failure is fast when aspects are not properly configured.

A more complicated example shows how all these parameters can be used together. The following is a singleton advice implementation that requires read-write access to the data to hibernates jobs. It must be instantiated and executed before JMSQueueSend activities.

```
@AdviceImpl(
    scope = "APPLICATION",
    dataAccess = "READ-WRITE",
    hibernatesJobs = true,
    targetKind = "ACTIVITY-BEFORE",
    targetFilter = "bw.JMSQueueSendActivity"
)
public class JMSPropertyChanger ... {
}
```

Enable Aspect Engine Logging

To enable tracing for Aspect Engine, set the `AspectEngine.Trace` property to true in a `cfg` file. This file should be passed as an argument to the ActiveMatrix BusinessWorks engine.

The @Property Java Annotation

This annotation is used for marking public member variables defined in advice java implementations as properties (like, advice implementation properties).

Following is an example of an advice implementation property:

```
public class AdviceImplExample<I, U, N extends I, A extends I, S,
    T, X>
    extends SyncAdvice<I, U, N, A, S, T, X> {
    @Property
    public String currency = "$";
    ...
}
```

Advice Implementation Properties

Advice implementations can define configuration properties that are used for configuring the behavior and execution of the java class. For example, an advice implementation that changes the value of a JMS property may choose to define the name of the JMS property using an advice implementation property.

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in makes these advice implementation properties visible and configurable from advices. This means that two advices that have the same implementation (that are implemented with the same java class) can specify different values for the same property. In the example in [The @Property Java Annotation](#), there could be two or more advices that want to mutate JMS properties. These advices can share the same implementation and each one can configure it with a different JMS property name.

@Property Java Annotation Optional Parameters

Name	Type	Description
required	Boolean	Whether the property is required to be set in the aspect file.

required

The advice implementation developer can declare an advice property either required or optional by setting this parameter.

A required advice property must be set in every advice that uses the implementation class that defines it. The ActiveMatrix BusinessWorks engine throws an exception at the time the engine gets initialized, if it finds a required advice implementation property that is not set in an advice.

Values	Description
true	The advice property is required.
false	The advice property is optional.

The default value, which is used when the "required" parameter is not set, is true. Therefore, by default, advice properties are required to be set in the aspect files.

Advice implementations advertise configuration properties through the use of the @Property annotation. The following example shows an advice implementation with two properties:

```

• package com.tibco.bw.poa.samples

• import com.tibco.bw.poa.runtime.annotation.Property

@AdviceImpl(
    dataAccess="READ-WRITE",
    targetKind="ACTIVITY-BEFORE",

```

```

targetFilter="bw.JMSQueueSendActivity"
)
public class JMSPropertyChangerWithConfig<I, U, N extends I, A
extends I, S, T, X> extends SyncAdvice<I, U, N, A, S, T, X> {
@Property
public String propertyToModify ="foo";

@Property (required = false)
public String propertyToModifyValue = "defaultFooValue";
...
}

```

Restrictions Imposed by TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in for Advice implementation Properties

- Only properties of string type are supported. If the Aspect Engine detects a property of a type different than string, it throws an exception at initialization time.
- Member variables that are exposed as advice configuration properties must be declared public. If the ActiveMatrix BusinessWorks Engine detects a property that is not declared public, it throws an exception at initialization time.

At run-time, when an advice is instantiated, the ActiveMatrix BusinessWorks engine injects in the advice instance, the property values that are specified in the aspect XML file.



Setter and Getter methods do not have to be available in the java class to provide access to the member variables.

The ActiveMatrix BusinessWorks engine can inject these property values by accessing the member variables directly.

All advices that share the same implementation instance (for example, when using an implementation configured with an APPLICATION scope) must have the same property values. The ActiveMatrix BusinessWorks Engine configures the implementation instance at the time the first advice that is using, it gets instantiated. If a subsequent advice that uses the same implementation instance is

instantiated, the ActiveMatrix BusinessWorks engine validates that all its properties have the same values as the properties set on the advice implementation instance. If a mismatch is found, the ActiveMatrix BusinessWorks Engine throws an exception.



To avoid unnecessary null pointer exceptions at run-time, it is highly recommended that all advice implementation properties have default values specified in the java class.

Refer [Advice Configuration Properties on page 12](#) about how property values are set in aspect files.

Scopes

At run-time, during its initialization process, the ActiveMatrix BusinessWorks Engine integrates aspects into ActiveMatrix BusinessWorks processes. As part of this process, the ActiveMatrix BusinessWorks engine instantiates advices and injects them in different join points inside processes. Since each advice has an implementation associated with it, which is represented as a Java class, the ActiveMatrix BusinessWorks engine either creates an instance of this class or takes one from a pool of already created object instances. ActiveMatrix BusinessWorks 5.9 supports binding multiple advice instances to the same advice implementation java class instance through the concept of scoping.

The two Advice Implementation Scopes supported are:

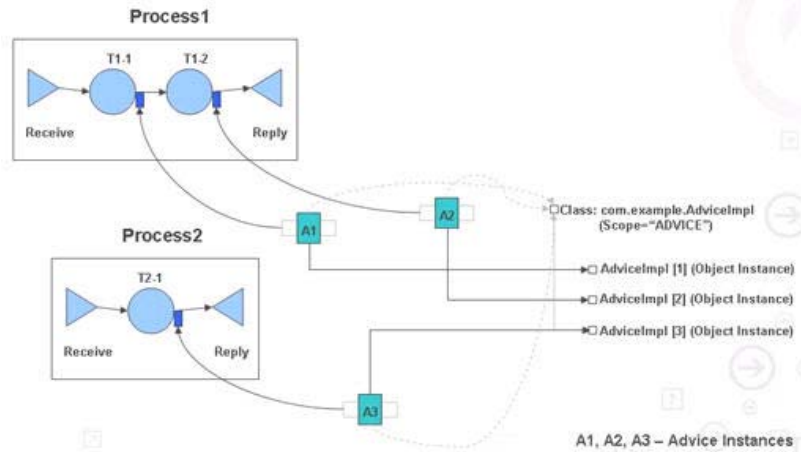
- Advice
- Application

Since scope is an attribute of the advice implementation, it is configured at the java class level in an annotation. This means that all advices using the same advice implementation java class, have the same scoping configuration.

Advice Scope Mode

The **Advice** scope is used when a new advice implementation object instance must be created for every advice instance. This mode is usually used when multiple advices that share the same implementation do not need to share any state. [Figure 12](#) shows this Advice Scope mode.

Figure 12 Advice Scope Mode



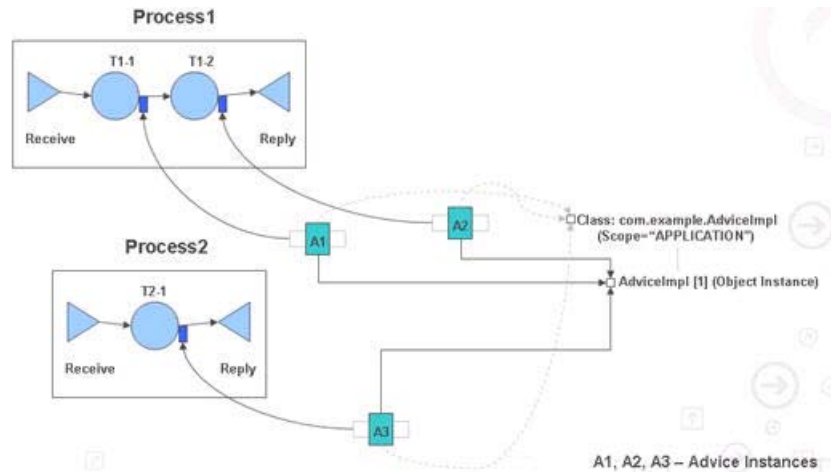
Despite one advice implementation object instance being created for every advice instance, the developer of the advice implementation java class will still have to be aware of the data concurrency issues.

Since an advice instance runs in a multi-threaded environment, potentially serving multiple process instances (such as, jobs) at the same time, each job is executed in a different thread.

Application Scope Mode

The **Application** scope is used when the developer of the advice implementation java class wants to ensure that only one advice implementation object instance is created for the entire application. In TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in, it means that there is one object instance per engine. Hence, all advice instances that are implemented with the same implementation share the same advice implementation object instance. [Figure 13](#) shows the Application Scope mode.

Figure 13 Application Scope Mode



XML Document Access

When it executes, an advice implementation has access to the XML document that is available in a particular context (like, join point). This XML Document is build based on a schema, which is again dependent on the context. For example, when it runs before an activity, the XML Document must be valid against the XML Schema that defines the input type of the activity.

In order to provide a deterministic behavior that allows the engine to fail fast when configuration issues are detected, TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in provides a declarative way of specifying metadata about advice implementations. This metadata is primarily driven by the following two main questions, a developer of an advice implementation needs to answer.

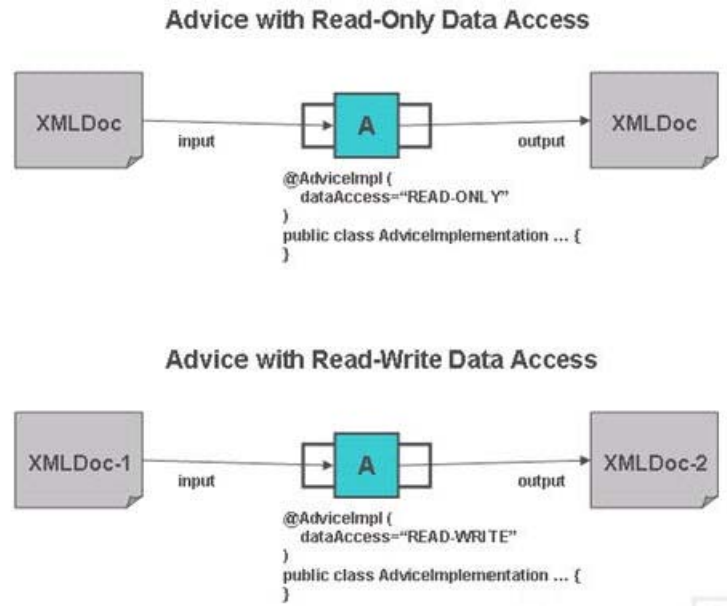
- Whether the advice implementation mutates the XML document?
- Whether the advice implementation expects the document to be valid against a particular XML schema?

Not all advice implementations mutate the incoming XML document. If the engine knows that an advice does not mutate the XML document, it can perform some optimizations, such as not requiring a revalidation of the document after the execution of the advices.

In TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in, an advice cannot mutate the incoming XML document unless it explicitly states that it does it, in the advice implementation metadata. For details about how to configure an advice to allow the mutation of the XML document, refer [The @AdviceImpl Java Annotation on page 26](#).

Figure 14 shows two advices, one that mutates the XML document and another that does not mutate the XML document.

Figure 14 Read-Only vs. Read-Write Data Access



The XML document is passed as an input parameter to the `execute()` method that is defined as part of the Advice Implementation Java class. The following is the signature of the method:

```
public N execute(N inputDoc, AspectProcessContext context)
throws AspectException;
```

The XML document is the first input parameter (like, "inputDoc"). TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in uses gXML (Generic XML) as the data model, which allows it to support multiple underlying XML tree models such as DOM, Axiom, and so on. For more information about gXML and general information on how to manipulate an XML document with gXML, refer to *References* in the *TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in API Reference*.

In order to manipulate an XML document, the advice implementation must get access to the gXML processing context object, which is an instance of `org.xml.sa.GxProcessingContext`. This object can be retrieved by an advice implementation from the advice's context, in the following way:

```
GxProcessingContext<I,U,N,A,S,T,X> pContext =
getAdviceContext().getGxProcessingContext();
```

This object is then used for traversing and pulling information from the XML document. This object cannot be used for mutating an XML document, though. In order to mutate an XML document, the advice implementation needs to get a mutable gXML processing context, which is an instance of `org.xml.sa.GxProcessingContextMutable`. Since this class extends `GxProcessingContext`, the way to retrieve a mutable processing context is very similar to the way to retrieve the immutable processing context (like, note the extra type casting):

```
GxProcessingContextMutable<I,U,N,A,S,T,X> pContext =
GxProcessingContextMutable) getAdviceContext().getGxProcessingContext();
```

If an advice implementation type casts the returned value of `getGxProcessingContext()` to a mutable processing context without explicitly setting the `@AdviceImplementation` or `dataAccess` parameter to "READ-WRITE", the previous call throws a `ClassCastException` at run-time. That is since the engine injects a mutable processing context in an advice instance only if its implementation is mutable.

Packaging and Deployment of Advice Implementations

Advice implementations are packaged in JAR files and need to be available in the CLASSPATH at run-time in order for the engine to properly instantiate advices. It does not really matter how these advice implementations are packaged in JAR files. What is important is that all the advice implementations (like, java classes) are referenced by advices to be available in the CLASSPATH at run-time, at the time the ActiveMatrix BusinessWorks engine gets initialized.

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in does not support loading a JAR file in the CLASSPATH at run-time, after the engine has been initialized. Therefore, unless it gets restarted, the ActiveMatrix BusinessWorks engine cannot execute an advice implemented with a java class that was not available in the CLASSPATH at initialization time.

The TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in installer creates a "lib" folder under `BWAA_HOME` and adds it to the CLASSPATH. This folder can be used for storing all the advice implementation libraries (that is, JAR files).

Chapter 2 **Point Cut Query Language**

This chapter defines Point Cut Query Language and provides the necessary information needed to build complex expressions.

Topics

- [Introduction, page 42](#)
- [Query Language Primitives, page 43](#)
- [Examples of Point Cuts Defined Using Query Language, page 49](#)

Introduction

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in defines a simple query language used for writing point cut expressions. This language consists of a small set of primitives. Each one of these primitives is designed to narrow down the scope of the search based on certain conditions (for example, search for all File Read activities, search for all processes that have their namespace starting with "http://example.com/", and so on).

Query Language Primitives

The query language defines the following four primitives:

- **activity** (selection-expression) - Selects all activities that match a specific expression.
- **process** (selection-expression) - Selects all processes that match a specific expression.
- **project** (selection-expression) - Narrows down the scope of the search to projects that match a specific expression.
- **engine** (selection-expression) - Narrows down the scope of the search to engines that match a specific expression.

These primitives can be combined to form complex expressions. The selection-expression, which is specified as part of these primitives, has a generic syntax that is not dependent on the actual primitive. Each primitive defines a set of properties that can be used in the expression. For example, the `activity()` primitive defines a `type` property that can be used to filter out activities based on their type. The evaluation of the expression results in the selection of a set of join points.



If there are no join points selected, the application runs without executing any advices.

Following is the Backus-Naur Form (BNF) definition of the point cut expression.

```

<point-cut-expression> ::= <primitive> ( "&&" <primitive>
)*
<primitive> ::= "activity(" <selection-expression> ")" |
"process(" <selection-expression> ")" |
"project(" <selection-expression> ")" | "engine("
<selection-expression> ")"
<selection-expression> ::= <selection-expression-part> ( <and-or>
<selection-expression-part> )*
<selection-expression-part> ::=
<selection-expression-part-simple> |
<selection-expression-part-wrapped>
<selection-expression-part-simple> ::= ( <propertyName> "="
<propertyValue> ) |
( "(" <property-name> "=" <property-value> ")" )
<selection-expression-part-wrapped> ::= "("
<selection-expression-part-simple> ")"
<and-or> ::= "&&" | "||"

```

<propertyName> - see the tables
<propertyValue> - see the table



Despite all these primitives being optional, one of them must always be specified in a point cut expression.

- **If the `activity()` primitive does not appear in the expression**, the ActiveMatrix BusinessWorks engine selects all activities that are defined as part of the project. In this case the ActiveMatrix BusinessWorks Engine treats this as having activity (`name=""`) in the expression.
- **If the `process()` primitive does not appear**, the ActiveMatrix BusinessWorks Engine treats this as having process (`name=""`) in the expression.
- **If the `project()` primitive does not appear**, the ActiveMatrix BusinessWorks Engine treats this as having project (`name=""`) in the expression.
- **If the `engine()` primitive does not appear**, the ActiveMatrix BusinessWorks Engine treats this as having engine (`name=""`) in the expression.

Properties Defined for `activity()` primitive

Property Name	Property Type	Property Value	Description
name	String		The name of the activity.
type	String		The type of the activity.
kind	String	"event-source" "signal-in" "activity"	The flavor of the activity (for example, event-source, signal-in or regular activity).
description	String		The description of the activity.

type

"type" is an ID that uniquely identifies a particular type of activity. To get the ID of a particular activity, refer to the table available in [Appendix A, ActivityTypes](#). The rest of this section describes the algorithm for building these IDs. This is useful to know since there are activities that do not get shipped out of the box with TIBCO ActiveMatrix BusinessWorks and therefore they are not mentioned in [Appendix A, ActivityTypes](#).

This ID is computed in the following way:

```
<type> ::= "bw.<resource-type-suffix>"
```

where

the `resource-type-suffix` is given by the sequence of the characters that appear after the last "." character in the `<pd:resourceType>` element, which is serialized as part of every activity's configuration. To find the `resource-type-suffix` for a specific activity, the user has to open a process definition that contains that activity and check the `<pd:resourceType>` element that is serialized as part of its configuration.

For example, here is the configuration XML of a File Read Activity that appears in a process definition:

```
<pd:activity name="Read File">
    <pd:type>com.tibco.plugin.file.FileReadActivity</pd:type>

<pd:resourceType>ae.activities.FileReadActivity</pd:resourceType>
    <pd:x>224</pd:x>
    <pd:y>173</pd:y>
    <config>
        <encoding>binary</encoding>
    </config>
    <pd:inputBindings>
        <ns1:ReadActivityInputClass>
            <fileName>
                <xsl:value-of
select="&quot;C:\test\foo.xml&quot;;"/>
            </fileName>
        </ns1:ReadActivityInputClass>
    </pd:inputBindings>
</pd:activity>
```

This example shows that the `resource-type-suffix` of the File Read Activity is "FileReadActivity". This means that this activity's type is "bw.FileReadActivity" in Point cut query language.

kind

"kind" is used to filter out activities based on their flavors. There are three flavors of activities, each one identified with a specific ID:

- **event-source** - An activity that is an event source or process starter (for example, File Event Source, JMS Queue Receiver, and so on).
- **signal-In** - An activity that is a signal-in (for example, File Signal-In, and so on).

- **activity** - An activity that is neither an event-source nor a signal-in (for example, File Read Activity, HTTP Send Receive Activity, and so on).

Since these properties are defined for the `activity()` primitive, they can only be used in the context of this primitive.

Properties Defined for `process()` primitive

Property Name	Property Type	Property Value	Description
name	String		The name of the process.
tns	String		The target namespace of the process.
kind	String	"sub-process" "regular"	The flavor of the process.
description	String		The description of the process.

kind

"kind" is used to filter out processes based on their flavors. There are two flavors of processes, each one identified with a specific ID:

- **sub-process** - A process that does not have an event source or process starter.
- **regular** - A process that has an event source or process starter.

Like the properties defined for the `activity()` primitive, the properties defined for the `process()` primitive can only be used in the context of this primitive.

Properties Defined for `project()` primitive

Property Name	Property Type	Property Value	Description
name	String		The name of the project.

This primitive is used for filtering out projects based on their name. This can be useful in organizations where all aspects are developed, packaged, and deployed together.

Properties Defined for engine() primitive

Property Name	Property Type	Property Value	Description
name	String		The name of the engine.

This primitive is used for filtering out engines based on their names. Similar to `project()` primitive, this primitive can be useful in organizations where all aspects are developed, packaged, and deployed together.

Use of Escape Character

While using double quote as part of the property value in a pointcut query, use backward slash ('\') as an escape character.

For example, `activity (description= "this is how \" is used as part of property value")`.

Escape characters should be used for all the xml predefined entities (<, >, &, ' and ") in the aspect file.

Wildcard Support

In order to provide more flexibility and to simplify the writing of point cut expressions, the query language supports wildcards ("*") in property values. These can be used either to specify the entire property value (for example, `name="*")` or to specify a part of it (for example, `name = "JMS*")`.



Property names are case insensitive and property values are case-sensitive.

This means that `activity (Name = "foo")` evaluates the same as `activity (name = "foo")`. However, `activity (name = "foo")` does not evaluate the same as `activity (name = "Foo")`.

Parentheses Support

Parentheses up to level two are supported in selection-expression of this query language.

Correct Expression (level one):

```
activity ( (name="JMSReceiver" || name= "HTTPReceiver" ) &&
(type="bw.JMSQueueEventSource" || type = "bw.httpEventSource" ) )
```

Invalid Expression (level three):

```
process (name = "Test*" || (name = "notify" || (name =  
"waitnotifyprocess*" && tns = "http://waitnotifytns/*"))))
```

Supported Operators

The following operators are supported.

EQUAL Comparison operator “=” (EQUAL) can be used while comparing property values. For details, refer to [Examples of Point Cuts Defined Using Query Language](#).

NOT EQUAL Comparison operator “!=” (NOT EQUAL) can be used while comparing property values. For details, refer to step 8 in the [Examples of Point Cuts Defined Using Query Language](#).

Examples of Point Cuts Defined Using Query Language

1. Select all the **FileEventSource** activities.

```
<pointcut name = "allFileEvsActivities">
  <query queryLanguage =
    "http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
    activity ( type = "bw.FileEventSource" )
  </query>
</pointcut>
```

2. Select all activities of type **JMS**.

```
<pointcut name = "allJMSActivities">
  <query queryLanguage =
    "http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
    activity ( type = "bw.JMS*" )
  </query>
</pointcut>
```

3. Select all **FileRead** and **FileWrite** activities.

```
<pointcut name = "fileActivities">
  <query queryLanguage =
    "http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
    activity ( type = "bw.FileReadActivity" || type =
      "bw.FileWriteActivity" )
  </query>
</pointcut>
```

4. Select all the **Event Source** activities.

```
<pointcut name = "eventSourceActivities">
  <query queryLanguage =
    "http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
    activity ( kind = "event-source" )
  </query>
</pointcut>
```

5. Select all activities that have "**@TODO**" in their description.

```
<pointcut name = "allTODOActivities">
  <query queryLanguage =
    "http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
    activity ( description = "*@TODO*" )
  </query>
```

```
</pointcut>
```

6. Select all the activities that belong to processes that have their namespace starting with "http://example.org/".

```
<pointcut name = "purchaseOrderActivities">
<query queryLanguage =
"http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
process ( tns = "http://example.org/*" )
</query>
</pointcut>
```

7. Select all the **File Write** activities with name starting with "**FileRead**", that are part of a process whose target namespace starts with http://example.org/. Select only the processes that are part of projects whose names start with "HR".

```
<pointcut name = "complex">
<query queryLanguage =
"http://schemas.tibco.com/bw/poa/pointCutSelectionLanguage">
activity (name = "FileRead*" && type =
"bw.FileReadActivity") && process ( tns =
"http://example.org/*" ) && project ( name = "HR*" )
</query>
</pointcut>
```

8. Select all the activities with name starting with "**file**". Do not select all "FileWrite" activities.

```
<pointcut name = "DoNotFileWriteActivities">
<query queryLanguage =
"http://schemas.tibco.com/bw/aop/pointCutSelectionLanguage">
activity ( name = "file*" && type !=
"bw.FileWriteActivity" )
</query>
</pointcut>
```


Chapter 3 **Asynchronous Advice Implementations**

This chapter describes the Asynchronous Advice Implementations.

Topics

- [Introduction, page 52](#)
- [Asynchronous Advices in ActiveMatrix BusinessWorks Engine, page 53](#)

Introduction

The asynchronous model is designed for advice implementations that take some time to execute. These are usually advice implementations that communicate with external systems, perform input or output operations, or perform tasks that can potentially bring down the performance of an ActiveMatrix BusinessWorks application.

An asynchronous advice implementation does *not* execute its business logic on the engine job thread. In other words, the ActiveMatrix BusinessWorks engine does *not* hold the job thread until an asynchronous advice completes its execution. This allows the engine to execute advices and/or activities that might exist in the process on *parallel tracks*, while the asynchronous advice is executing. However, the engine *does not* continue to execute the next advice in the join point, or the next activity on the same track before the asynchronous advice completes its execution.

A typical asynchronous advice implementation gets a thread from a thread pool in its `execute()` method and starts executing its business logic on it. Immediately after that, it calls `AdviceController->setPending()` with the appropriate timeout and returns from its `execute()`. After its business logic completes and before the advice thread finishes executing, the advice implementation calls `AdviceController->setReady()` with the result object. Once the job thread is available to execute the asynchronous advice, the engine calls its `postExecute()` method by passing the result object received in the `setReady()` call. The advice implementation gets the opportunity to do any final job related cleanup operations before returning the final result object back to the engine.

If the advice implementation does not complete its execution in the allotted time, which is specified in the `setPending()` call, the engine *times out* the advice implementation by calling its `cancelled()` method. An advice implementation that does not communicate with external systems, does not perform input or output operations and tasks that may take some time to execute can be implemented as synchronous advice implementation.

Asynchronous Advices in ActiveMatrix BusinessWorks Engine

This section describes the various models of asynchronous advice working in the TIBCO ActiveMatrix BusinessWorks engine.

Execution Model (Successful Execution)

Successful execution - `execute()` and `postExecute()`

- The engine calls `execute()` to start the execution.
- The advice gets a thread from a pool and runs its business logic on it.
- The advice sends the signals back to the ActiveMatrix BusinessWorks engine, when it finishes executing.
- The engine calls `postExecute()` to finish executing the implementation.

The ActiveMatrix BusinessWorks engine's job thread is not blocked until the advice finishes executing and produces its result.

- Parallel tracks, if available in the process, are executed on the jobs thread.
- The engine does not execute the next advice in the pipeline.

Execution Model (timeout)

Timed out execution: `execute()` and `cancelled()`

- The engine calls `execute()` to start the execution.
- The advice gets a thread from a pool and runs its business logic on it.
- Before the advice returns from `execute()`, it sets a timeout.
- When a timeout occurs, the engine calls `cancelled()` to finish executing the implementation.
- The advice releases any outstanding resources.

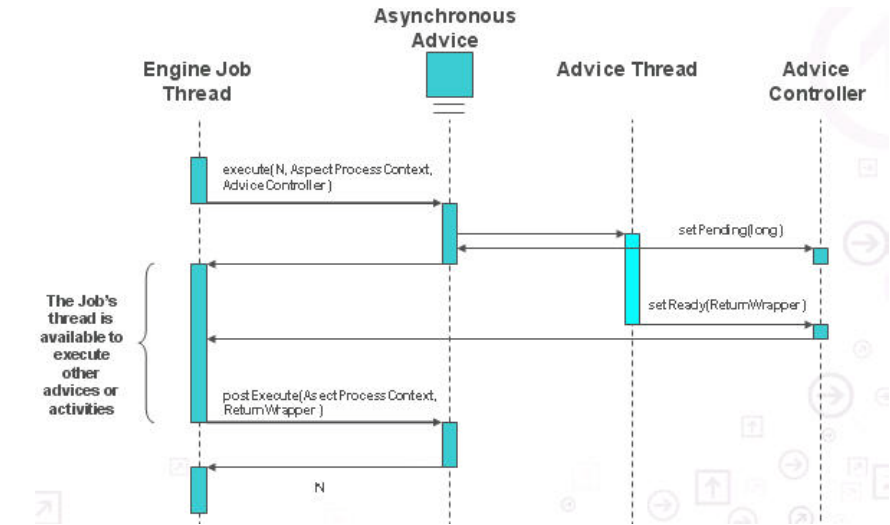
`AspectProcessContext` cannot be used by asynchronous advices on the parallel thread.

- All the logic that requires access to this object should be moved to `execute()`, `postExecute()` or `cancelled()`.

Threading Model: Asynchronous Advice Implementations

Figure 15 shows the threading model.

Figure 15 Threading Model



Asynchronous Advice Example

```

@AdviceImpl
public class AsyncTimeSleepier<I, U, N extends I, A extends I, S, T, D> extends AsyncAdvice<I, U, N, A, S, T, D> {

    @Property
    public String waitTimeInMs = "0";

    private Hashtable<String, SimpleWaitThread> outstandingThreads;

    public void init(AdviceContext<I, U, N, A, S, T, D> context)
        throws AspectException {
        super.init(context);
        outstandingThreads = new Hashtable<String, SimpleWaitThread>();
    }

    @Override
    public void execute(N input, AspectProcessContext context, AdviceController<D> controller)
        throws AspectException {
        System.out.println("-----[ASYNC ADVICE EXECUTING]-----");
        System.out.println("The async advice will wait for " + waitTimeInMs + " milliseconds");
        long waitTime = Long.valueOf(waitTimeInMs);
        controller.setPending(waitTime + 5000);
        SimpleWaitThread thread = createThread(context, input, controller, waitTime);
        thread.start();
    }

    @Override
    public N postExecute(AspectProcessContext apc, ReturnWrapper<D> result)
        throws AspectException {
        System.out.println("The async advice is back!");
        removeThread(apc);
        System.out.println("-----[ASYNC ADVICE DONE]-----");
        return result.getTailNode();
    }
}

```

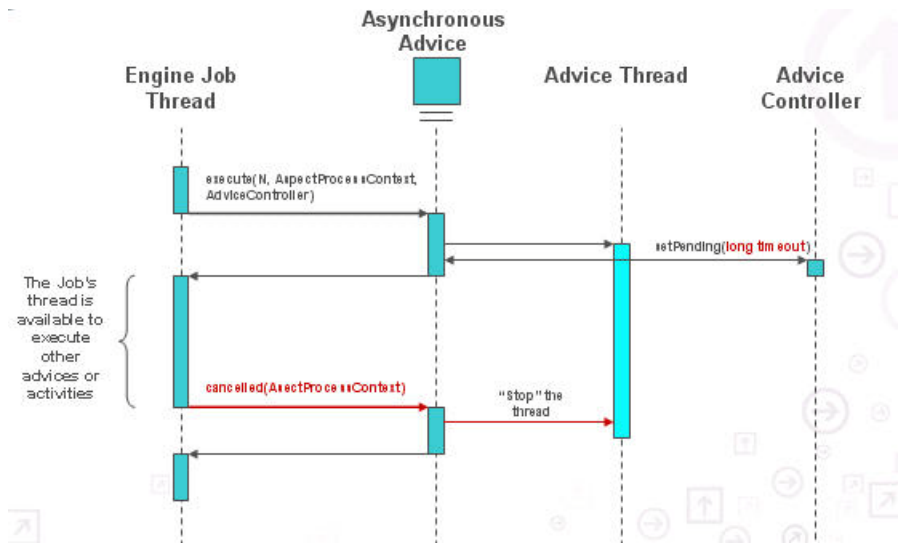
Extends AsyncAdvice

A different execute() signature

Implement postExecute()

Threading Model: Asynchronous Advice Implementations (Timeout)

Figure 16 Threading Model (Timeout)



Summary

This can be summarized as:

- An advice that takes a long time to execute should probably be asynchronous.
- An advice that uses input or output operations should probably be asynchronous.
- When implementing the `cancelled()` method, perform a graceful stop of the advice thread.
- An asynchronous advice should always use a thread pool.
- An asynchronous advice should always use a timeout and must be configured as property.
- Do not use `AspectProcessContext` on the advice (parallel) thread.

Chapter 4 **Hibernate Resume**

This chapter describes the hibernate resume feature of TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in.

Topics

- [Features of Hibernate Resume, page 58](#)
- [Comparison between Checkpointing and Hibernate, page 59](#)
- [Defining a Hibernate Advice Implementation, page 60](#)
- [Using a Database for Hibernation, page 65](#)
- [Modifying the Hibernated Data, page 66](#)

Features of Hibernate Resume

The main features of hibernate resume are:

- An advice implementation can '**Hibernate**' a process in its execute method.
- A process is said to be hibernated when the process state is written out to the disk (or database) and the process is completely removed from memory.
- Hibernation internally uses the logic similar to checkpointing for TIBCO ActiveMatrix BusinessWorks.
- A hibernated process can be started at a later stage in time using the `ResumeHibernatedProcess`, a TIBCO Hawk command exposed by the TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in on the ActiveMatrix BusinessWorks engine.

Comparison between Checkpointing and Hibernate

It is important to understand the comparison between Checkpointing and Hibernate.

[Table 5](#) shows the comparison between checkpointing (TIBCO ActiveMatrix BusinessWorks) and Hibernate (TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in)

Table 5 Comparison between Checkpointing and Hibernate

Checkpointing	Hibernate
Stores Process State.	Stores process state.
Can only be used in the form of Checkpoint Activity or implicit part of Transaction.	Any advice implementation can hibernate at any activity.
Job continues till completion.	Job stops executing and is removed from the memory.
User cannot modify the checkpointed data.	At the time when job is resumed, the advice implementation that hibernated the job, can modify the XML document that is available in the join point.
Only crashed or error jobs are restarted.	User manually decides which jobs to resume.
A restarted job will continue from the next activity.	A resumed job will continue from the same activity (or next activity depending on which join point the job was hibernated).

Defining a Hibernate Advice Implementation

To define a hibernate advice implementation, you must:

- Advertise in the following annotation that it is using the hibernate feature.

```
@AdviceImpl (  
    hibernatesJobs=true  
)
```

- Should call the `setHibernateJobEnabled(<timeDelay>)` on the `AspectProcessContext` object.

timeDelay

Use `<timeDelay>` attribute to set a time after which the job will be hibernated. This facility is provided for asynchronous activities on the parallel paths to complete execution. A time delay set to zero (0) will hibernate the job immediately without waiting.

Example of Hibernate Advice Implementation

```

package com.tibco.bw.pas.samples;

import org.xml.sax.GxModel;

@Aspect
public class StockAuditor<I, U, N extends I, A extends I, S, T, X> extends SyncAdvice<I, U, N, A, S, T, X> {

    @Property
    public String restrictedStock;

    private String TEXT_CONTENT_NS = "";
    private S textContentNs;
    private String TEXT_CONTENT_NAME = "textContent";
    private S textContentName;

    @Override
    public void init(AdviceContext<I, U, N, A, S, T, X> context)
        throws AspectException {

        super.init(context);

        GxProcessingContext<I, U, N, A, S, T, X> pctx = (GxProcessingContext<I, U, N, A, S, T, X>) context.getGxProcessingContext();
        GxNameBridge<S> nameBridge = pctx.getNameBridge();

        textContentNs = nameBridge.symbolize(TEXT_CONTENT_NS);
        textContentName = nameBridge.symbolize(TEXT_CONTENT_NAME);
    }

    @Override
    public N execute(N input, AspectProcessContext context) throws AspectException {

        System.out.print("StockAuditor received: ");

        GxModel<N, A, S, T> model = adviceContext.getGxProcessingContext().getModel();
        N firstChild = model.getFirstChild(input);

        String stockOrder =
            XmlUtils.getChildElementStringValue(firstChild, textContentNs, textContentName, adviceContext.getGxProcessingContext());

        System.out.println(stockOrder);

        if (stockOrder != null && stockOrder.startsWith(restrictedStock)) {
            System.out.println("This is a restricted stock. The job will hibernate to wait for managerial approval");
            context.setHibernateJobEnabled(0);
        }
        else {
            System.out.println("This is a non restricted stock. The order will be executed");
        }
        return input;
    }
}

```

Advise in Annotation

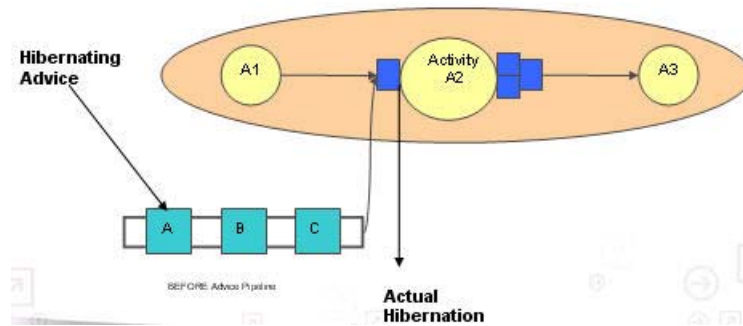
Calling the Hibernate API

Execution of a Hibernate Advice

The job may not get hibernated right after an advice calls `setHibernateJobEnabled(long)`. Hence, the engine must finish executing all advices that are part of the same join point before initiating the hibernate procedure. Even after that, the job may still not immediately get hibernated correctly.

The advice that triggers the hibernation may choose to delay it, in order to give chance to any asynchronous advice or any asynchronous activity that might exist in the job on parallel tracks to finish executing. To accomplish that, the advice implementation must call `setHibernatedJobEnabled(long)` method by passing a `timeDelay` greater than 0 (zero). A 0 (zero) `timeDelay` ensures that the hibernation is initiated right after the engine finishes executing all advices that are running in that particular join point.

Figure 17 *Hibernate Advice Execution*



Resuming the Hibernated Job

TIBCO Hawk methods have been exposed for listing and resuming the hibernated jobs. A job is resumed from the next point of execution. For example, for a "Before" hibernated advice, the job will resume from activity execution.



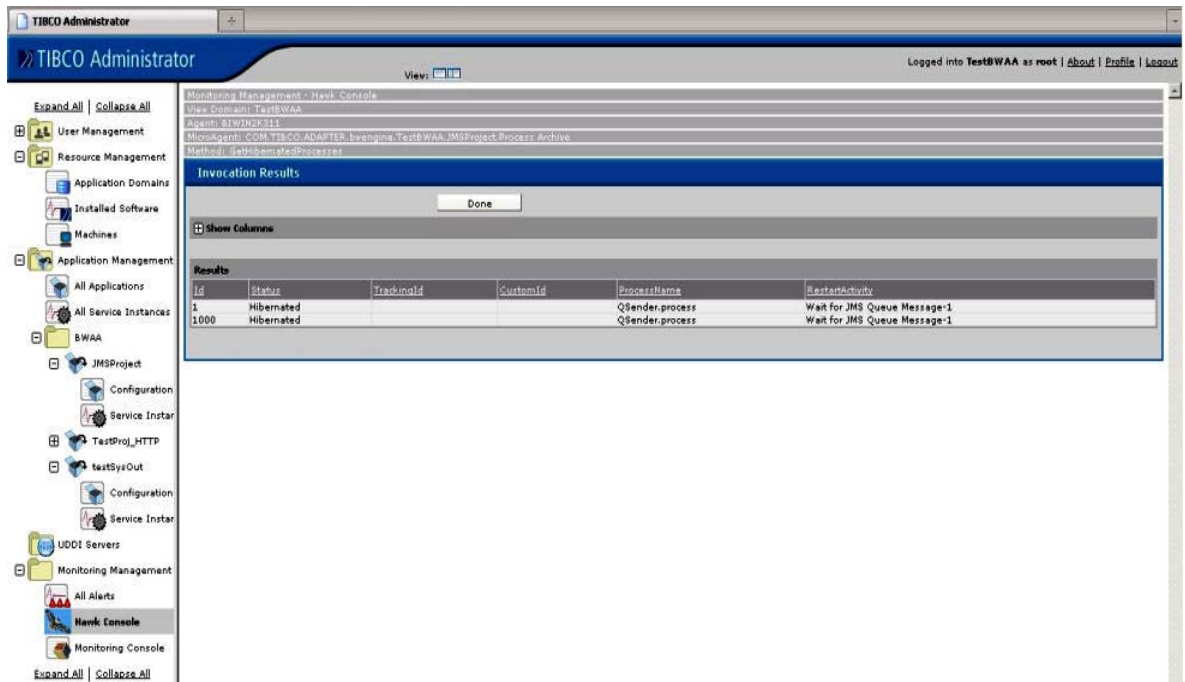
The new 'Resume' activity also allows you to resume a hibernated job. Refer [Chapter 6, BWAA Palette](#) for details.

The TIBCO Hawk methods are:

- `GetHibernatedProcesses()` - returns a Tabular Data with all hibernate jobs information
- `ResumeHibernatedProcesses(long jobId)` – resume

A job hibernated on one engine can be resumed on a different instance of the engine. However, the engine must be running prior to calling the `ResumeHibernatedProcess()`.

Figure 18 *GetHibernatedProcesses Dialog*



Example of Resuming a Job

Figure 19 shows an example of resuming a job from TIBCO Administrator Hawk console.

Figure 19 Resuming a Job

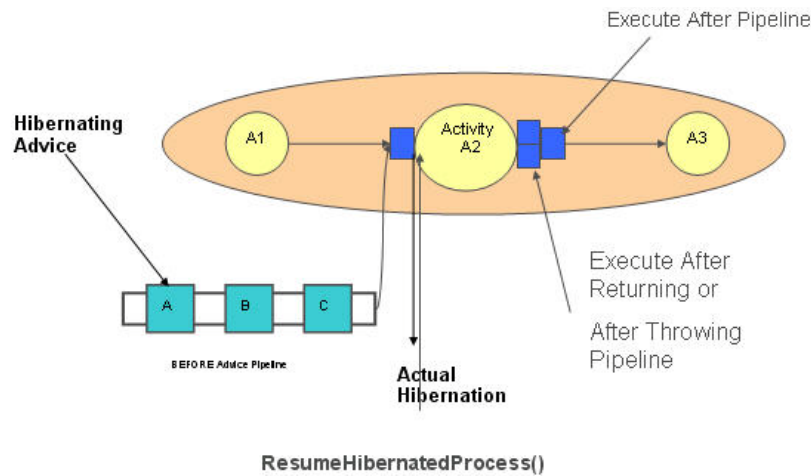
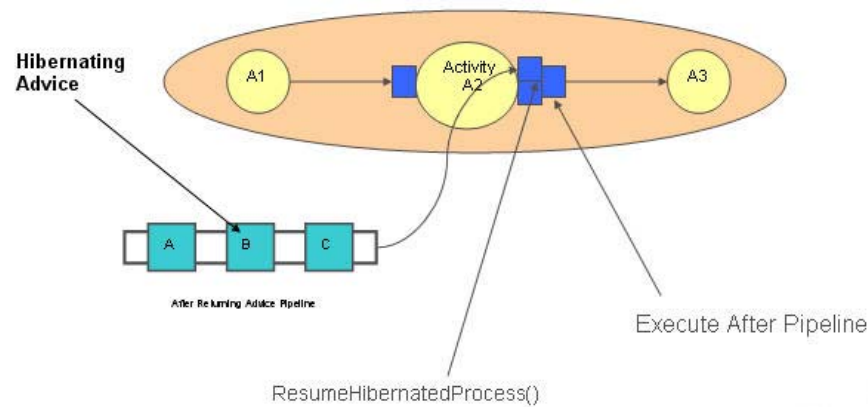


Figure 20 Resuming a Job (After Returning)



Using a Database for Hibernation

A database can be specified similar as in checkpointing. A JDBC shared resource needs to be in the project which has to hibernate jobs to a database.

Set the following ActiveMatrix BusinessWorks engine properties:

- `Engine.Hibernate.UseDatabase` – Set to true to use a database
- `Engine.Hibernate.Database.Configuration` – Path to JDBC shared resource

Table names can be specified using similar properties like Checkpointing. All databases supported for TIBCO ActiveMatrix BusinessWorks Checkpointing can be used for Hibernation.

If the database is not set, the job state is stored in a file under the `working/<hostname or enginename>/hibernate` folder.

Modifying the Hibernated Data

The hibernating advice implementation can modify the hibernated data when the job is resumed.

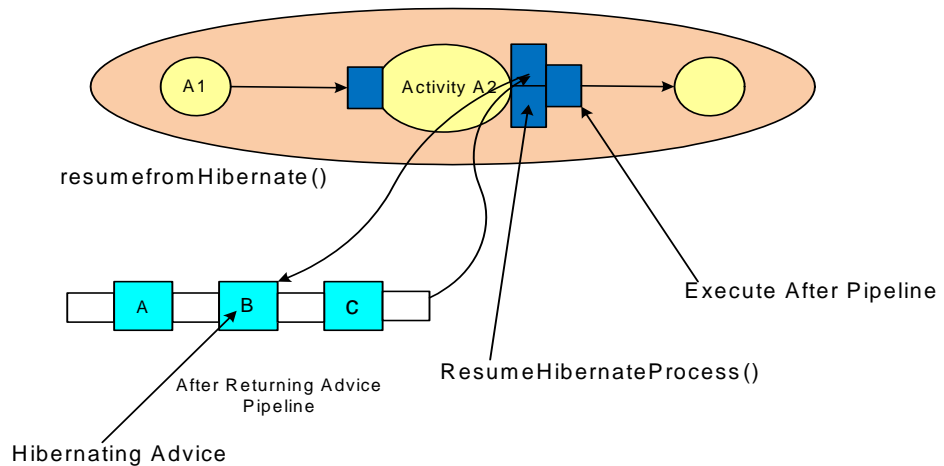
The advice implementation needs to override the following method:

```
public void resumeFromHibernate(N input, AspectProcessContext
context)
```

It should specify appropriate data-Access if it has to mutate the hibernated document. The engine calls this method only on the advice that has triggered the hibernation.

Figure 21 shows the user level view for `resumeFromHibernate()`

Figure 21 User Level View for `resumeFromHibernate()`



Chapter 5

Object Sharing Between Java Activities and Advice Implementation

This chapter provides information about the new API's which will enable the user to use this Object Sharing feature.

Topics

- [Overview, page 68](#)
- [User Scenarios, page 69](#)
- [API's and New Interfaces, page 71](#)

Overview

TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in provides a feature for the activities of the JAVA palette in TIBCO ActiveMatrix BusinessWorks and advice implementations to pass objects between each other by means of defining a "JAVA Object Reference" type for Input and/or Output schema of the activities involved.

The feature enables you to pass JAVA objects between:

- Advice implementations and JAVA activities
- An enhanced support for advice implementations with another advice implementation
- JAVA activities with other JAVA activities in a TIBCO ActiveMatrix BusinessWorks project

An Advice Implementation or a JAVA Activity can check-in and check-out objects from the engine data structure by using API's. You have to pass a unique key which will act as a unique identifier for the object in the engine data structure.



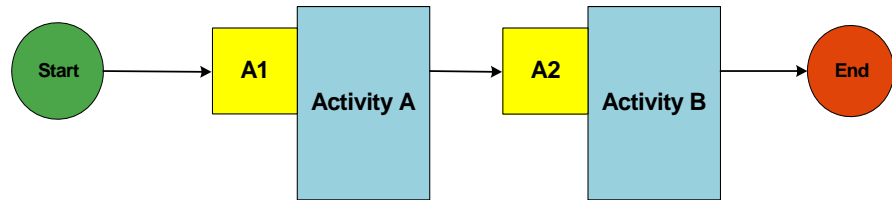
You must manually pass the key between advice implementations and activities. It is not a part of this feature. You must also ensure that the activity or advice implementation which needs to check-out the object, already has the key to it.

Also, the features provided by TIBCO ActiveMatrix BusinessWorks or TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in may or may not be used to accomplish this.

User Scenarios

This section summarises basic supported and unsupported user scenarios. [Table 6](#) and [Table 7](#) shows the user scenarios for activities and advices with implementations.

Figure 22 Basic User Scenario



Following is the basic user scenario as shown in Figure 22.

- **Activity A** and **Activity B** are supported **Java Activities**
- **A1** and **A2** are **Advices (with Implementations)** applied on **Activity A** and **Activity B** respectively
- **Activity A** and **Advice A1** will check-in JAVA objects.
- **Activity B** and **Advice A2** can check-in and check-out JAVA objects.

Table 6 User Scenarios for Activities

Action 1	Key, Value	Action 2	Key, Value	Result
Supported				
Activity A Check-in	foo, obj1	Activity B Check-out	foo	obj1 returned by Activity B.
Activity A Check-in	foo, obj1	Advice A2 Check-out	foo	obj1 returned by Advice A2.
Activity A Check-in	foo, obj1	Activity B Check-in	foo, obj2	obj1 is replaced by obj2 in the map.
Unsupported				
Activity A Check-in	foo, obj1	Advice A2 Check-out	foo, obj2	

Table 7 shows the supported and unsupported scenarios for Advices with implementations.

Table 7 User Scenarios for Advices with Implementations

Action 1	Key, Value	Action 2	Key, Value	Result
Supported				
Advice A1 Check-in	foo, obj1	Activity B Checkout	foo	obj1 returned by Activity B.
Advice A1 Check-in	foo, obj1	Advice A2 Checkout	foo	obj1 returned by Advice A2.
Advice A1 Check-in	foo, obj1	Advice A2 Check-in	foo, obj2	obj1 is replaced by obj2 in the map.
Unsupported				
Advice A1 Check-in	foo, obj1	Activity B Check-in	foo, obj2	

Summary

- Any advice implementation or JAVA activity can check-in objects with a unique-key.
- Any other advice implementation or supported JAVA activity with the knowledge of this unique key can check-out that object.

Limitations

- When an object is checked-in with a unique-key and the same unique-key is used to check-in a different object, it is only supported provided the original entity that checked-in the object is of the same type. For example, an advice implementation can overwrite an object checked-in by another advice implementation but NOT by an activity.
- An activity can overwrite an object checked-in by another activity but NOT an advice implementation.

API's and New Interfaces

The following interface has been added in the TIBCO ActiveMatrix BusinessWorks:

public interface `JavaProcessContext`

The public methods available in this interface are:

- **public void** `storeProcessObject` (String key, Serializable obj)
- **public** `Serializable` `getProcessObject` (String key)
- **public** `Serializable` `removeProcessObject` (String key)

Implement a `setJavaProcessContext` (`JavaProcessContext` object) in the Java class in which you wish to use the feature. The engine will invoke this method before any other methods are invoked on the JAVA class.



Advice Implementation developers need not use this interface for storing and retrieving objects. They should continue to use similar methods defined on the `AspectProcessContext` interface.

Use Cases

There are two cases that happen while retrieving:

- If an advice is trying to retrieve an object with the unique-key *foo*, the engine first looks up if an advice which has been executed for the same process earlier has checked-in an object with key *foo*. If it does not find an object, the engine then looks up if an activity has checked-in an object with key *foo*. If the engine does not find an object in both cases, it returns a NULL.
- Similarly, if an activity tries to retrieve an object with the unique-key *foo*, the engine first looks up if an activity which has been executed in the same process earlier has checked-in an object with key *foo*. If it does not find an object, the engine then looks up if an advice has checked-in an object with key *foo*. If the engine does not find an object in both cases, it returns a NULL.

Chapter 6 **BWAA Palette**

BWAA palette contains the activity to resume a previously hibernated job.

Topics

- [Resume, page 74](#)

Resume

Activity



Resume

This activity fetches the hibernated job from the files or database, loads it into the memory, and continues to execute it from the point of hibernation.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the activity in the process definition.
Description	No	Short description of the activity.

Input

The input for the activity is the following.

Input Item	Datatype	Description
jobID	Long	This is the job id to resume a previously hibernated job. Note: There is no static configuration for this activity as jobID is the only parameter required.



The activity will throw an `ActivityException` (`JobNotFoundException`), if there is no job with input job id to be resumed.

Output

There is no output for this activity.



However, Output tab displays the stack trace of exception detail whenever Resume activity throws an exception.

Chapter 7 **Monitoring and Management**

This chapter describes the TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in monitoring and management features.

Topics

- [Introduction, page 76](#)
- [getAdviceInstances, page 77](#)
- [getAdviceInstanceMetrics, page 81](#)
- [getRunningAdviceInstancesCount, page 85](#)
- [getRunningAdviceInstances, page 86](#)

Introduction

The Monitoring and Management feature is provided by a monitoring interface, which is implemented by a TIBCO Hawk microagent. This section describes the monitoring interface methods. These methods can be called via:

- TIBCO Hawk AMI protocol and
- JMX

The TIBCO Hawk Display, the TIBCO ActiveMatrix BusinessWorks monitor servlet, and any other TIBCO Hawk application can use the AMI protocol to invoke the monitoring interface methods on a running engine.



It is important to note that:

- Wild card characters are not supported.
- For each application the Advice Instance ID starts with zero (0).
- If there are no asynchronous advices running, the total elapsed and total execution time will be same.
- In order to provide **no value for the Long type, specify "-1"** as the value. For example, -1 as an input for `AdviceInstanceId` returns all the available `AdviceInstanceIDs` data.

getAdviceInstances

Microagent Method

Description This method provides general information about all advice instances that are created in the ActiveMatrix BusinessWorks engine.

Index Names ProcDefName, ActivityName, AdviceInstanceID

In Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AdviceInstanceID	Long	null	true	The identifier of the advice instance.	None	None
ProcDefName	String	null	true	The name of the process definition.	None	None
ActivityName	String	null	true	The name of the activity adjacent to the advice.	None	None
ActivityClass	String	null	true	The name of the activity class.	None	None
Where	String	null	true	The location where the advice instance is running (for example, before activity, after activity, after returning activity, after throwing activity)	None	ACT_BEFORE ACT_AFTER_RET, ACT_AFTER_THR, ACT_AFTER
AdviceNs	String	null	true	The namespace of the advice.	None	None
AdviceName	String	null	true	The name of the advice.	None	None
AdviceImpl	String	null	true	The advice implementation name.	None	None

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
Sync	Boolean	null	true	Whether the advice implementation is synchronous.	None	True, False
DataAccess	String	null	true	The data access mode (for example, read-only, read-write)	None	READ-ONLY, READ-WRITE
HibernatesJobs	Boolean	null	true	Whether the advice instance is capable of hibernating jobs.	None	True, False

Out Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AdviceInstanceID	Long	null	true	The identifier of the advice instance.	None	None
ProcDefName	String	null	true	The name of the process definition.	None	None
ActivityName	String	null	true	The name of the activity adjacent to the advice.	None	None
ActivityClass	String	null	true	The name of the activity class.	None	None
Where	String	null	true	The location where the advice instance is running (For example, before activity, after activity, after returning activity, after throwing activity)	None	ACT_BEFORE ACT_AFTER_RET, ACT_AFTER_THR, ACT_AFTER

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValueChoices
AdviceNs	String	null	true	The namespace of the advice.	None	None
AdviceName	String	null	true	The name of the advice.	None	None
AdviceImpl	String	null	true	The advice implementation name.	None	None
Sync	Boolean	null	true	Whether the advice implementation is synchronous.	None	True, False
DataAccess	String	null	true	The data access mode (for example, read-only, read-write)	None	READ-ONLY, READ-WRITE
HibernatesJobs	Boolean	null	true	Whether the advice instance is capable of hibernating jobs.	None	True, False

Output Example

Name	Output
AdviceInstanceID	<ul style="list-style-type: none"> 10000 10001
ProcDefName	<ul style="list-style-type: none"> Folder1/P1.process P2.process
ActivityName	<ul style="list-style-type: none"> ReadActivity HttpSend
ActivityClass	<ul style="list-style-type: none"> bw.FileRead bw.HTTPSendReceive
Where	<ul style="list-style-type: none"> ACT_BEFORE ACT_AFTER_RET

Name	Output
AdviceNamespace	<ul style="list-style-type: none">• http://aspects.com/audit• http://aspects.com/log
AdviceName	<ul style="list-style-type: none">• Audit• Log
AdviceImpl	<ul style="list-style-type: none">• com.example.AuditAdvice• com.example.LogAdvice
Sync	<ul style="list-style-type: none">• True• False
DataAccess	<ul style="list-style-type: none">• READ-WRITE• READ-ONLY
HibernatesJobs	<ul style="list-style-type: none">• False• False

getAdviceInstanceMetrics

Microagent Method

Description This method provides various metrics for a particular advice instance.

Index Names AdviceInstanceID

In Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AdviceInstanceID	Long	null	true	The identifier of the advice instance.	None	None

Out Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AdviceInstanceID	Long	null	true	The identifier of the advice instance.	None	None
ExecutionCount	Long	null	true	The number of times this advice instance has been executed by the engine.	None	None
ErrorCount	Long	null	true	The number of times this advice instance has thrown an exception.	None	None
RunningCount	Long	null	true	The number of times this particular advice instance is currently running.	None	None

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
TotEla	Long	null	true	Total wall-clock time used by all calls of this advice instance (milliseconds) including the waiting time for asynchronous advices.	None	None
TotExe	Long	null	true	Total wall-clock time used by all calls of this advice instance (milliseconds) not including the waiting time for asynchronous advices.	None	None
AvgEla	Long	null	true	Average elapsed time of all completed advice instances (milliseconds).	None	None
MinEla	Long	null	true	Minimum elapsed time of all completed advice instances (milliseconds).	None	None
MaxEla	Long	null	true	Maximum elapsed time of all completed advice instances (milliseconds).	None	None
LastEla	Long	null	true	The elapsed time of the last completed advice instance (milliseconds).	None	None

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AvgExe	Long	null	true	Average execution time of all completed advice instances (milliseconds).	None	None
MinExe	Long	null	true	Minimum execution time of all completed advice instances (milliseconds).	None	None
MaxExe	Long	null	true	Maximum execution time of all completed advice instances (milliseconds).	None	None
LastExe	Long	null	true	The execution time of the last completed advice instance (milliseconds).	None	None

Output Example

Name	Output
AdviceInstanceID	<ul style="list-style-type: none">1000010001
ExecutionCount	<ul style="list-style-type: none">31
ErrorCount	<ul style="list-style-type: none">30
RunningCount	<ul style="list-style-type: none">24
TotEla	<ul style="list-style-type: none">150090104023

Name	Output
TotExe	<ul style="list-style-type: none">• 150090• 13311
AvgEla	<ul style="list-style-type: none">• 50030• 104023
MinEla	<ul style="list-style-type: none">• 50007• 104023
MaxEla	<ul style="list-style-type: none">• 50075• 104023
LastEla	<ul style="list-style-type: none">• 50008• 50008
AvgExe	<ul style="list-style-type: none">• 50030• 100011
MinExe	<ul style="list-style-type: none">• 50007• 100011
MaxExe	<ul style="list-style-type: none">• 50075• 100011
LastExe	<ul style="list-style-type: none">• 50008• 50008

getRunningAdviceInstancesCount

Microagent Method

Description This method provides the number of synchronous and asynchronous advice instances that are executing as part of the jobs that are currently running.

Note: This method requires no input.

Out Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
SyncCount	Long	null	true	The total number of synchronous advice instances that are executing in the jobs that are currently running	None	None
AsyncCount	Long	null	true	The total number of asynchronous advice instances that are executing in the jobs that are currently running.	None	None

Example Output

Name	Output
SyncCount	<ul style="list-style-type: none">2
AsyncCount	<ul style="list-style-type: none">15

getRunningAdviceInstances

Microagent Method

Description This method provides metrics for the advice instances that are executing as part of the jobs that are currently running.

Index Name AdviceInstanceID

In Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
MinDuration	Long	null	true	Minimum elapsed wall-clock time since the advice instance started (milliseconds).	None	None
JobID	Long	null	true	The Job identifier.	None	None
MaxReturnCount	Long	null	true	The maximum number of advice instances to be returned. Note: It is important when there are multiple advice instances running.	None	None

Out Index

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AdviceInstanceID	Long	null	true	The identifier of the advice instance.	None	None
ProcDefName	String	null	true	The name of the process definition.	None	None
JobID	Long	null	true	The Job identifier.	None	None
ActivityName	String	null	true	The name of the activity adjacent to the advice.	None	None

Name	Type	Default	IsOpen	Description	ValueChoices	LegalValue Choices
AdviceNamespace	String	null	true	The namespace of the advice.	None	None
AdviceName	String	null	true	The name of the advice.	None	None
AdviceImpl	String	null	true	The advice implementation name.	None	None
Sync	Boolean	null	true	Whether the advice implementation is synchronous.	None	None
StartTime	Long	null	true	Time at which the advice instance started executing in the current job (milliseconds).	None	None
Duration	Long	null	true	The elapsed wall-clock time since the advice instance started (milliseconds) executing in the current job.	None	None

Output Example

Name	Output
AdviceInstanceID	<ul style="list-style-type: none"> 0 1
ProcDefName	<ul style="list-style-type: none"> ProcessSynch.process ProcessAsynch.process
JobID	<ul style="list-style-type: none"> 34 36
ActivityName	<ul style="list-style-type: none"> SynchLog AsynchLog

Name	Output
AdviceNs	<ul style="list-style-type: none">• http://examples.org/One• http://examples.org/One
AdviceName	<ul style="list-style-type: none">• Logger1• Logger2
AdviceImpl	<ul style="list-style-type: none">• com.tibco.test.bwaa.SleepLogger• com.tibco.test.bwaa.AsynchTimeSleeper
Sync	<ul style="list-style-type: none">• true• false
StartTime	<ul style="list-style-type: none">• 1303661907012• 1303661907011
Duration	<ul style="list-style-type: none">• 43830• 13831

Appendix A **ActivityTypes**

Topics

- [ActivityTypes, page 90](#)

ActivityTypes

The tables contains the activity types of all TIBCO ActiveMatrix BusinessWorks activities. Refer to this table in order to generate expressions in Point cut query language.

Table 8 Active Enterprise Palette Activity Types

Activity	Activity Description	Activity Type
Active Enterprise Palette		
event-source	Adapter Request-Response Server	bw.aeRRServer
event-source	Adapter Subscriber	bw.aeSubstriction
signal-in	Wait for Adapter Message	bw.aeSubSignalInActivity
signal-in	Wait for Adapter Request	bw.aeServerSignalInActivity
activity	Invoke an Adapter Request-Response Service	bw.aeOpClientReqActivity
activity	Respond to Adapter Request	bw.aeOpServerReplyActivity
activity	Publish to Adapter	bw.aePubActivity
activity	Send Exception to Adapter Request	bw.aeOpServerFaultActivity

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
File Palette		

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
event-source	File Poller	<code>bw.FileEventSourceResource</code>
signal-in	Wait for File Change	<code>bw.FileSignalInUI</code>
activity	Copy File	<code>bw.FileCopyActivity</code>
activity	Create File	<code>bw.FileCreateActivity</code>
activity	List Files	<code>bw.ListFilesActivity</code>
activity	Read File	<code>bw.FileReadActivity</code>
activity	Remove File	<code>bw.FileRemoveActivity</code>
activity	Rename File	<code>bw.FileRenameActivity</code>
activity	Write File	<code>bw.FileWriteActivity</code>
FTP Palette		
activity	FTP Change Default Directory	<code>bw.FTPChangeDefaultDirActivityUI</code>
activity	FTP Delete File	<code>bw.FTPDeleteFileActivityUI</code>
activity	FTP Dir	<code>bw.FTPDirActivityUI</code>
activity	FTP Get Default Directory	<code>bw.FTPGetDefaultDirActivityUI</code>
activity	FTP Get	<code>bw.FTPGetActivityUI</code>
activity	FTP Make Remote Directory	<code>bw.FTPMakeRemoteDirActivityUI</code>
activity	FTP Put	<code>bw.FTPPutActivityUI</code>
activity	FTP Quote	<code>bw.FTPQuoteActivityUI</code>
activity	FTP Remove Remote Directory	<code>bw.FTPRemoveRemoteDirActivityUI</code>
activity	FTP Rename File	<code>bw.FTPRenameActivityUI</code>
activity	FTP SYS Type	<code>bw.FTPSysTypeActivityUI</code>
General Palette		

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
event-source	On Event Timeout	<code>bw.onEventTimeout</code>
event-source	On Notification Timeout	<code>bw.onNotificationTimeout</code>
event-source	On Shutdown	<code>bw.onShutdown</code>
event-source	On Startup	<code>bw.onStartup</code>
event-source	On Error	<code>bw.onErrorProcess</code>
event-source	Receive Notification	<code>bw.waitStarter</code>
event-source	Timer	<code>bw.timer</code>
signal-in	Catch	<code>bw.catch</code>
signal-in	Sleep	<code>bw.sleep</code>
signal-in	Wait	<code>bw.waitActivity</code>
activity	Assign	<code>bw.assignActivity</code>
activity	Call Process	<code>bw.subprocess</code>
activity	Checkpoint	<code>bw.checkpoint</code>
activity	Confirm	<code>bw.confirm</code>
activity	Engine Command	<code>bw.enginecommand</code>
activity	External Command	<code>bw.CmdExecActivity</code>
activity	Generate Error	<code>bw.throw</code>
activity	Get Shared Variable	<code>bw.getSharedVariable</code>
activity	Inspector	<code>bw.inspectorActivity</code>
activity	Label	<code>bw.label</code>
activity	Mapper	<code>bw.MapperActivity</code>
activity	Notify	<code>bw.notifyActivity</code>
activity	Null	<code>bw.null</code>

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
activity	Rethrow	<code>bw.rethrow</code>
activity	Set Shared Variable	<code>bw.setSharedVariable</code>
activity	Write To Log	<code>bw.log</code>
HTTP Palette		
event-source	HTTP Receiver	<code>bw.httpEventSource</code>
signal-in	Wait for HTTP Request	<code>bw.httpSignalIn</code>
activity	HTTP Send Request	<code>bw.httpRequest</code>
activity	HTTP Send Response	<code>bw.httpWebResponse</code>
Java Palette		
activity	Java Code	<code>bw.javaActivity</code>
event-source	Java Event Source	<code>bw.JavaEventSource</code>
activity	Java Method	<code>bw.JavaMethodActivity</code>
activity	Java To XML	<code>bw.JavaToXmlActivity</code>
activity	XML To Java	<code>bw.XmlToJavaActivity</code>
JDBC Palette		
activity	JDBC Call Procedure	<code>bw.JDBCCallActivity</code>
activity	JDBC Get Connection	<code>bw.JDBCGetConnectionActivity</code>
activity	JDBC Query	<code>bw.JDBCQueryActivity</code>
activity	JDBC Update	<code>bw.JDBCUpdateActivity</code>
activity	SQL Direct	<code>bw.JDBCGeneralActivity</code>
JMS Palette		
event-source	JMS Queue Receiver	<code>bw.JMSQueueEventSource</code>
event-source	JMS Topic Subscriber	<code>bw.JMSTopicEventSource</code>

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
signal-in	Wait for JMS Queue Message	<code>bw.JMSQueueSignalInActivity</code>
signal-in	Wait for JMS Topic Message	<code>bw.JMSTopicSignalInActivity</code>
activity	Get JMS Queue Message	<code>bw.JMSQueueGetMessageActivity</code>
activity	JMS Queue Requestor	<code>bw.JMSQueueRequestReplyActivity</code>
activity	JMS Queue Sender	<code>bw.JMSQueueSendActivity</code>
activity	JMS Topic Publisher	<code>bw.JMSTopicPublishActivity</code>
activity	JMS Topic Requestor	<code>bw.JMSTopicRequestReplyActivity</code>
activity	Reply to JMS Message	<code>bw.JMSReplyActivity</code>
Mail Palette		
event-source	Receive Mail	<code>bw.MailEventSourceResource</code>
activity	Send Mail	<code>bw.MailActivityResource</code>
Parse Palette		
activity	Parse Data	<code>bw.ParseActivity</code>
activity	Render Data	<code>bw.RenderActivity</code>
RMI Palette		
activity	RMI Lookup	<code>bw.lookup</code>
activity	RMI Server	<code>bw.starter</code>
RV Palette		
event-source	RV Subscriber	<code>bw.RVEventSource</code>
signal-in	Wait for RV Message	<code>bw.rvSignalInActivity</code>
activity	Publish RV Message	<code>bw.RVPubActivity</code>
activity	Reply to RV Request	<code>bw.RVReplyActivity</code>
activity	Send RV Request	<code>bw.RVRequestActivity</code>

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
Service Palette		
activity	Get Context	<code>bw.getContext</code>
activity	Invoke Partner	<code>bw.invokePartner</code>
activity	Set Context	<code>bw.setContext</code>
SOAP Palette		
event-source	SOAP Event Source	<code>bw.SOAPEventSourceUI</code>
activity	Retrieve Resources	<code>bw.RetrieveResource</code>
activity	SOAP Request Reply	<code>bw.SOAPSendReceiveUI</code>
activity	SOAP Send Fault	<code>bw.SOAPSendFaultUI</code>
activity	SOAP Send Reply	<code>bw.SOAPSendReplyUI</code>
activity	MIME Parse	<code>bw.MimeParserActivity</code>
TCP Palette		
event-source	TCP Receiver	<code>bw.TCPEventSource</code>
signal-in	Wait for TCP Request	<code>bw.TCPSignalIn</code>
activity	Read TCP Data	<code>bw.TCPRead</code>
activity	TCP Close Connection	<code>bw.TCPCloseConnection</code>
activity	TCP Open Connection	<code>bw.TCPOpenConnection</code>
activity	Write TCP Data	<code>bw.TCPWrite</code>
Transaction Palette		
activity	Transaction State	<code>bw.TransactionStateActivity</code>

Table 9 Activity Types of Other ActiveMatrix BusinessWorks Palettes

Activity Kind	Activity Description	Activity Type
XML Activities Palette		
activity	Parse XML	bw.XMLParseActivity
activity	Render XML	bw.XMLRendererActivity
activity	Transform XML	bw.XMLTransformActivity
Manual Work Palette		
activity	Assign Work	bw.AssignWork
activity	Download Document	bw.DownloadDocument
activity	Get Work Status	bw.GetWorkStatus
activity	Modify Work	bw.ModifyWork
signal-in	Wait For Completion	bw.WaitForCompletion

Appendix B **Developing gXML Applications**

Topics

- [Overview on page 98](#)
- [Developing gXML Applications on page 99](#)
- [gXML Recipes on page 106](#)

Overview

A Generic Java API for XQuery Data Model (XDM) and eXtensible Markup Language (XML) Processing. gXML also provides, out-of-the-box, a cohesive suite of XML processing implementations such as XPath, XSLT, XQuery, Serialization, W3C XML Schema and Validation.

gXML is a new way of writing XML code in the Java language. The code that you write to the gXML API can be run against any Data Model that supports the gXML bridge.

This flexibility offers the following benefits:

- minimizes expensive conversion overhead
- greater opportunities for performance optimization
- greater code reuse
- minimize risks associated with locking into one Data Model

gXML currently supports Parsing, Serialization, XDM Data Model, XPath 2, XSLT 2 and XQuery, W3C XML Schema and Validation.

- A gXML bridge is provided for `org.w3c.dom.Node`.
- A gXML bridge for a high performance proprietary implementation is complete but not yet released.
- A gXML bridge for a reference implementation is complete but not yet released. A gXML bridge for AxiOM is in the works.

Developing gXML Applications

This section illustrates one way of using gXML. All gXML processors, including custom processing, run within a `GxProcessingContext` instance that provides necessary meta data. A `GxProcessingContext` instance in turn is created through a `GxApplication` instance. It is your responsibility to write a class that provides an instance of `GxApplication`. The best way to do this is to write an abstract class that implements all but the `newProcessingContext` method of `GxApplication`. This approach will allow you to write your application generically and then inject the choice of parameterization as late as possible for maximum code reuse and flexibility.

This, of course, is not the only way to use gXML. An existing architecture may force the choice of parameterization and create silos of XML processing. The degree of integration in this case may be less than is possible with a homogeneous solution.

Whatever the approach, the best way to use gXML is to write generic, parameterized, and XML processing code whenever possible.

Implementing GxApplication

```

001 package org.gxml.book.common;
002
003 import java.io.StringWriter;
004 import java.net.URI;
005 import java.net.URISyntaxException;
006
007 import junit.framework.TestCase;
008
009 import org.gxml.sa.GxApplication;
010 import org.gxml.sa.GxModel;
011 import org.gxml.sa.GxNameBridge;
012 import org.gxml.sa.GxProcessingContext;
013 import org.gxml.sa.GxSequenceHandler;
014 import org.gxml.xdm.Resolver;
015
016 import com.tibco.gxml.sa.api.common.util.PreCondition;
017 import
com.tibco.gxml.sa.processor.serialization.api.GxSerializerFactory;
018 import
com.tibco.gxml.sa.processor.serialization.impl.SerializerFactory;
019
020 public abstract class SampleApp<I, U, N extends I, A extends I, S,
T, X> extends TestCase implements GxApplication<I, U, N, A, S, T, X>
021 {
022     public Resolver getResolver()

```

```

023     {
024         try
025         {
026             return new SampleResolver(new
URI("../../plugins/org.gxml.book/resources/foo.xml"));
027         }
028         catch (final URISyntaxException e)
029         {
030             throw new AssertionError(e);
031         }
032     }
033
034     protected String serialize(final N node, final
GxProcessingContext<I, U, N, A, S, T, X> pcx)
035     {
036         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
037
038         // Configure for "pretty" printing.
039         sf.setIndent(Boolean.TRUE);
040
041         final StringWriter w = new StringWriter();
042
043         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(w);
044
045         final GxModel<N, A, S, T> model = pcx.getModel();
046
047         handler.startDocument(null);
048         try
049         {
050             model.stream(node, true, true, handler);
051         }
052         finally
053         {
054             handler.endDocument();
055         }
056
057         return w.toString();
058     }
059
060     /**
061      * Some bridge implementations may use {@link String} directly
for symbols. They must make them behave according to
062      * symbol semantics (==, toString).
063      */
064     public void assertNodeSymbolSemantics(final N node, final
GxProcessingContext<I, U, N, A, S, T, X> pcx)
065     {
066         final GxModel<N, A, S, T> model = pcx.getModel();
067         final GxNameBridge<S> nameBridge = pcx.getNameBridge();

```

```

068
069         switch (model.getNodeKind(node))
070         {
071             case ELEMENT:
072                 {
073                     assertSymbolSemantics(model.getNamespaceURI(node),
nameBridge);
074                     assertSymbolSemantics(model.getLocalName(node),
nameBridge);
075                 }
076             case TEXT:
077             case DOCUMENT:
078                 {
079
080                 }
081             break;
082             default:
083                 {
084                     throw new AssertionError(model.getNodeKind(node));
085                 }
086         }
087     }
088
089     public void assertSymbolSemantics(final S symbol, final
GxNameBridge<S> nameBridge)
090     {
091         PreCondition.assertArgumentNotNull(symbol, "symbol");
092         PreCondition.assertArgumentNotNull(nameBridge,
"nameBridge");
093         assertSame(symbol, nameBridge.symbolize(symbol.toString()));
094         assertSame(symbol,
nameBridge.symbolize(copy(symbol.toString())));
095     }
096
097     /**
098      * Do anything to manufacture a String that is equal, but not
identical (the same), as the original.
099      * <p>
100      * This method has the post-condition that the strings are equal
but not the same.
101      * </p>
102      *
103      * @param original
104      *         The original.
105      * @return A copy of the original string.
106      */
107     private String copy(final String original)
108     {
109         final String copy = original.concat("junk").substring(0,
original.length());

```

```

110          // Post-conditions verify that this actually works and isn't
"optimized" out.'
111          assertEquals(original, copy);
112          assertNotSame(original, copy);
113          // Be Paranoid
114          assertTrue(original.equals(copy));
115          assertFalse(original == copy);
116          // OK. That'll do.'
117          return copy;
118      }
119 }

```

Implementing GxCatalog

A catalog provides the means to isolate your application from the physical location of file resources. Writing a catalog simply means implementing the GxCatalog interface, so that it maps from the logical locations specified in code or XML resources to the corresponding physical location.

```

001 package org.gxml.book.common;
002
003 public class SampleCatalog
004 {
005
006 }

```

Implementing GxResolver

A resolver takes a base-uri and an href and uses these two values to return a stream.

```

001 package org.gxml.book.common;
002
003 import java.io.File;
004 import java.io.FileNotFoundException;
005 import java.io.IOException;
006 import java.io.InputStream;
007 import java.net.URI;
008 import java.net.URISyntaxException;
009 import java.net.URL;
010
011 import org.gxml.xdm.Resolved;
012 import org.gxml.xdm.Resolver;
013
014 import com.tibco.gxml.sa.api.common.util.PreCondition;
015
016 public final class SampleResolver implements Resolver
017 {
018     final URI baseURI;

```

```

019
020     public SampleResolver(final URI baseURI)
021     {
022         this.baseURI = PreCondition.assertArgumentNotNull(baseURI,
023             "baseURI");
024     }
025     /**
026      * Convert a URI relative to a base URI into an input source.
027      * <p/>
028      * This default implementation requires that neither parameter
029      * be null, and performs the expected action to retrieve
030      * the input source (which may involve network access).
031      *
032      * @param baseURI
033      *         the base URI against which the target is to be
034      *         resolved; must not be null
035      * @param location
036      *         the URI to resolve; must not be null
037      * @return a pair of InputStream and resolved URI.
038      */
039     public Resolved<InputStream> resolveInputStream(final URI
040         location) throws IOException
041     {
042         PreCondition.assertArgumentNotNull(location, "uri");
043         if (location.isAbsolute())
044         {
045             return retrieve(location, location);
046         }
047         else
048         {
049             PreCondition.assertArgumentNotNull(baseURI, "baseURI");
050
051             final URI base = baseURI.normalize();
052             final URI resolved = base.resolve(location);
053
054             return retrieve(location, resolved);
055         }
056     }
057
058     private Resolved<InputStream> retrieve(final URI location, final
059     URI uri) throws IOException
060     {
061         PreCondition.assertArgumentNotNull(uri, "uri");
062
063         final URL toRetrieve;
064
065         if (!uri.isAbsolute()) // assume local file
066         {
067             final File canonFile = new
068             File(uri.toString()).getCanonicalFile();

```

```

064         toRetrieve = canonFile.toURI().toURL();
065     }
066     else
067     {
068         toRetrieve = uri.toURL();
069     }
070
071     if (toRetrieve == null)
072     {
073         throw new FileNotFoundException(uri.toString());
074     }
075
076     final InputStream stream = toRetrieve.openStream();
077     if (stream == null)
078     {
079         throw new FileNotFoundException(toRetrieve.toString());
080     }
081     try
082     {
083         return new Resolved<InputStream>(location, stream,
084         toRetrieve.toURI());
085     }
086     catch (final URISyntaxException e)
087     {
088         throw new AssertionError(e);
089     }
090 }

```

Injecting DOM

The final task in providing a concrete `GxApplication` class is to implement the `newProcessingContext` method on a derived class. This is where you get to choose the tree, atomic values, meta data and symbols that your application will use. In many cases, you will use an off-the-shelf processing context class, but it is also possible to assemble your own variety or build one entirely from scratch.

If you are going to use gXML with `org.w3c.dom.Node`, you still have choices for the atomic values that your system will use as well as the meta data implementation. This example uses atomic values that are mostly Java Wrapper types and the reference sequence type implementation, `SmSequenceType`.

```

001 package org.gxml.book.parsing;
002
003 import org.gxml.sa.GxMetaBridge;
004 import org.gxml.sa.GxNameBridge;
005 import org.gxml.sa.mutable.GxApplicationMutable;

```

```

006 import org.gxml.sa.mutable.GxProcessingContextMutable;
007 import org.gxml.xs.SmMetaBridge;
008 import org.gxml.xs.SmSequenceType;
009 import org.w3c.dom.Node;
010
011 import com.tibco.gxml.sa.api.common.datatype.StringNameBridge;
012 import com.tibco.gxml.sa.common.atom.AtomBridge;
013 import
com.tibco.gxml.sa.common.helpers.GxMetaBridgeOnSmMetaBridgeAdapter;
014 import
com.tibco.gxml.sa.common.helpers.SmAtomBridgeOnGxAtomBridgeAdapter;
015 import com.tibco.gxml.sa.xdm.dom.DomProcessingContext;
016 import com.tibco.gxml.xs.SmMetaBridgeFactory;
017
018 /**
019  * Demonstration of constructing a concrete GxApplication(Mutable)
implementation using the DOM processing context.
020  */
021 public final class DomValidatingParsingSample extends
BookValidatingParsingSample<Object, Object, Node, Object, String,
SmSequenceType<Object, String>, Object> implements
GxApplicationMutable<Object, Object, Node, Object, String,
SmSequenceType<Object, String>, Object>
022 {
023     public final GxProcessingContextMutable<Object, Object, Node,
Object, String, SmSequenceType<Object, String>, Object>
newProcessingContext()
024     {
025         // The name bridge is created along with the processing
context for maximum concurrency.
026         final GxNameBridge<String> nameBridge = new
StringNameBridge();
027         final AtomBridge<String> atomBridge = new
AtomBridge<String>(nameBridge);
028         final SmMetaBridge<Object, String> cache = new
SmMetaBridgeFactory<Object, String>(new
SmAtomBridgeOnGxAtomBridgeAdapter<Object,
String>(atomBridge)).newMetaBridge();
029         final GxMetaBridge<Object, String, SmSequenceType<Object,
String>> metaBridge = new GxMetaBridgeOnSmMetaBridgeAdapter<Object,
String>(cache, atomBridge);
030
031         final DomProcessingContext<Object, SmSequenceType<Object,
String>> pcx = new DomProcessingContext<Object, SmSequenceType<Object,
String>>(this, metaBridge, cache);
032
033         // Set the "owning" processing context on the atom bridge.
034         atomBridge.setProcessingContext(pcx);
035
036         // Return the newly constructed processing context.
037         return pcx;
038     }
039 }

```

gXML Recipes

Parsing

Parsing a Character Stream and a Byte Stream

```

001 package org.gxml.book.parsing;
002
003 import java.io.InputStream;
004 import java.io.Reader;
005 import java.io.StringReader;
006 import java.net.URI;
007
008 import org.gxml.book.common.SampleApp;
009 import org.gxml.sa.GxModel;
010 import org.gxml.sa.GxNameBridge;
011 import org.gxml.sa.GxProcessingContext;
012 import org.gxml.xdm.NodeKind;
013 import org.gxml.xdm.Resolved;
014 import org.gxml.xdm.Resolver;
015
016 import com.tibco.gxml.sa.common.helpers.DocumentBuilderFactory;
017 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilder;
018 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilderFactory;
019
020 public abstract class BookIntroParsingSample<I, U, N extends I, A
    extends I, S, T, X> extends SampleApp<I, U, N, A, S, T, X>
021 {
022     public void testCharacterStreamParse() throws Exception
023     {
024         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
    newProcessingContext();
025
026         final GxDocumentBuilderFactory<N, S> factory = new
    DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
027
028         final GxDocumentBuilder<N> builder =
    factory.newDocumentBuilder();
029
030         final String xmlString = "<e>123</e>";
031         final URI systemId = new URI("e.xml");
032         final Reader characterStream = new StringReader(xmlString);
033         final N doc = builder.parse(characterStream, systemId);
034
035         final GxModel<N, A, S, T> model = pcx.getModel();
036

```



```

037         assertEquals(NodeKind.DOCUMENT, model.getNodeKind(doc));
038
039         final N e = model.getFirstChildElement(doc);
040         assertEquals(NodeKind.ELEMENT, model.getNodeKind(e));
041         assertEquals("e", model.getLocalNameAsString(e));
042         assertEquals("123", model.getStringValue(e));
043     }
044
045     public void testByteStreamParse() throws Exception
046     {
047         final Resolver resolver = getResolver();
048
049         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
050
051         final URI systemId = new URI("email.xml");
052         final Resolved<InputStream> source =
resolver.resolveInputStream(systemId);
053
054         final GxDocumentBuilderFactory<N, S> factory = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
055
056         final GxDocumentBuilder<N> builder =
factory.newDocumentBuilder();
057
058         final N document = builder.parse(source.getResource(),
source.getSystemId());
059
060         final GxModel<N, A, S, T> model = pcx.getModel();
061
062         assertEquals(NodeKind.DOCUMENT,
model.getNodeKind(document));
063
064         final N email = model.getFirstChildElement(document);
065         assertEquals(NodeKind.ELEMENT, model.getNodeKind(email));
066         assertEquals("email", model.getLocalNameAsString(email));
067         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
068         final S namespaceURI =
nameBridge.symbolize("http://www.example.com");
069         final S localName = nameBridge.symbolize("from");
070         final N from = model.getFirstChildElementByName(email,
namespaceURI, localName);
071         assertEquals("Julie", model.getStringValue(from));
072
073         for (final N node : model.getDescendantOrSelfAxis(document))
074         {
075             assertNodeSymbolSemantics(node, pcx);
076         }
077     }
078 }

```

Constructing a Data Model Tree Programmatically

This example demonstrates constructing a tree directly using the fragment builder.

```

001 package org.gxml.book.snoopy;
002
003 import java.io.IOException;
004 import java.io.InputStream;
005 import java.io.StringReader;
006 import java.io.StringWriter;
007 import java.net.URI;
008 import java.net.URISyntaxException;
009
010 import javax.xml.namespace.QName;
011 import javax.xml.parsers.ParserConfigurationException;
012
013 import org.gxml.book.common.SampleApp;
014 import org.gxml.sa.GxException;
015 import org.gxml.sa.GxFragmentBuilder;
016 import org.gxml.sa.GxMetaBridge;
017 import org.gxml.sa.GxModel;
018 import org.gxml.sa.GxNameBridge;
019 import org.gxml.sa.GxProcessingContext;
020 import org.gxml.sa.GxSequenceHandler;
021 import org.gxml.sa.GxVariantBridge;
022 import org.gxml.xdm.NodeKind;
023 import org.gxml.xdm.Resolved;
024 import org.gxml.xdm.Resolver;
025 import org.gxml.xs.SmName;
026
027 import com.tibco.gxml.sa.api.common.lang.ExprException;
028 import com.tibco.gxml.sa.api.common.lang.ExprResult;
029 import com.tibco.gxml.sa.api.common.lang.GxExpr;
030 import com.tibco.gxml.sa.api.common.lang.GxExprContextDynamicArgs;
031 import com.tibco.gxml.sa.api.common.lang.GxExprContextStaticArgs;
032 import com.tibco.gxml.sa.api.common.lang.GxLanguageToolkit;
033 import com.tibco.gxml.sa.common.helpers.DocumentBuilderFactory;
034 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilder;
035 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilderFactory;
036 import
com.tibco.gxml.sa.processor.serialization.api.GxSerializerFactory;
037 import
com.tibco.gxml.sa.processor.serialization.impl.SerializerFactory;
038 import com.tibco.gxml.sa.processor.xquery.LanguageToolkit;
039 import com.tibco.gxml.sa.processor.xslt.GxTransform;
040 import com.tibco.gxml.sa.processor.xslt.GxTransformBuilder;
041 import com.tibco.gxml.sa.processor.xslt.GxTransformer;
042 import com.tibco.gxml.sa.processor.xslt.XSLTransformBuilder;

```

```

043 import
com.tibco.gxmmlsa.processor.org.exslt.strings.ExsltStringsFunctionGroup;
044
045 public abstract class SnoopySample<I, U, N extends I, A extends I,
S, T, X> extends SampleApp<I, U, N, A, S, T, X>
046 {
047     public void testDocumentFromString()
048     {
049         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
050
051         final N document = documentFromString(pcx);
052
053         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
054
055         sf.setIndent(true);
056
057         final StringWriter sw = new StringWriter();
058
059         final GxSequenceHandler<A, S, T> serializer =
sf.newSerializer(sw);
060
061         final GxModel<N, A, S, T> model = pcx.getModel();
062
063         model.stream(document, true, true, serializer);
064
065         // System.out.println(sw.toString());
066     }
067
068     public void testFragmentBuilder()
069     {
070         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
071
072         final N document = documentFromEvents(pcx);
073
074         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
075
076         sf.setIndent(true);
077
078         final StringWriter sw = new StringWriter();
079
080         final GxSequenceHandler<A, S, T> serializer =
sf.newSerializer(sw);
081
082         final GxModel<N, A, S, T> model = pcx.getModel();
083
084         model.stream(document, true, true, serializer);
085

```

```

086         // System.out.println(sw.toString());
087     }
088
089     private N documentFromString(final GxProcessingContext<I, U, N,
A, S, T, X> pcx)
090     {
091         final String strval = "" + "<?xml version='1.0'
encoding='UTF-8'?'>" + "<book isbn='0836217462'>" + " <title>Being a Dog
Is a Full-Time Job</title>" + " <author>Charles M. Schultz</author>" + "
<character>" + " <name>Snoopy</name>" + " <since>1950-10-04</since>" + "
</character>" + "</book>";
092
093         final GxDocumentBuilderFactory<N, S> factory = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
094
095         final GxDocumentBuilder<N> builder =
factory.newDocumentBuilder();
096
097         try
098         {
099             return builder.parse(new StringReader(strval), null);
100         }
101         catch (final IOException e)
102         {
103             throw new AssertionError();
104         }
105     }
106
107     private N documentFromEvents(final GxProcessingContext<I, U, N,
A, S, T, X> pcx)
108     {
109         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
110
111         final S NULL_NS_URI = nameBridge.empty();
112         final S BOOK = nameBridge.symbolize("book");
113         final S ISBN = nameBridge.symbolize("isbn");
114         final S TITLE = nameBridge.symbolize("title");
115         final S AUTHOR = nameBridge.symbolize("author");
116         final S CHARACTER = nameBridge.symbolize("character");
117         final S NAME = nameBridge.symbolize("name");
118         final S SINCE = nameBridge.symbolize("since");
119
120         final GxFragmentBuilder<N, A, S, T> builder =
pcx.newFragmentBuilder();
121
122         // Note: Using try...finally not only ensures that elements
get closed when errors
123         // occur, it also helps to remind you to end elements and
makes the levels in
124         // the XML more obvious.
125         builder.startDocument(null);
126         try

```

```

127     {
128         builder.startElement(NULL_NS_URI, BOOK, "", null);
129         try
130         {
131             builder.attribute(NULL_NS_URI, ISBN, "",
132 "0836217462");
133             builder.startElement(NULL_NS_URI, TITLE, "", null);
134             try
135             {
136                 builder.text("Being a Dog Is a Full-Time Job");
137             }
138             finally
139             {
140                 builder.endElement();
141             }
142             builder.startElement(NULL_NS_URI, AUTHOR, "", null);
143             try
144             {
145                 builder.text("Charles M. Schultz");
146             }
147             finally
148             {
149                 builder.endElement();
150             }
151             builder.startElement(NULL_NS_URI, CHARACTER, "",
152 null);
153             try
154             {
155                 builder.startElement(NULL_NS_URI, NAME, "",
156 null);
157                 try
158                 {
159                     builder.text("Snoopy");
160                 }
161                 finally
162                 {
163                     builder.endElement();
164                 }
165                 builder.startElement(NULL_NS_URI, SINCE, "",
166 null);
167                 try
168                 {
169                     builder.text("1950-10-04");
170                 }
171                 finally
172                 {
173                     builder.endElement();
174                 }
175             }
176             finally
177             {
178                 builder.endElement();
179             }
180         }
181     }
182 }

```

```

173         {
174             builder.endElement();
175         }
176     }
177     finally
178     {
179         builder.endElement();
180     }
181 }
182 finally
183 {
184     builder.endDocument();
185 }
186
187 return builder.getNodes().get(0);
188 }
189
190 public void testExample() throws ParserConfigurationException,
191     IOException, GxException, ExprException, URISyntaxException
192 {
193     final GxProcessingContext<I, U, N, A, S, T, X> pcx =
194     newProcessingContext();
195
196     final Resolver resolver = getResolver();
197
198     final URI xmlSystemId = new URI("hotel.xml");
199     final Resolved<InputStream> xmlInput =
200     resolver.resolveInputStream(xmlSystemId);
201
202     final GxDocumentBuilderFactory<N, S> f = new
203     DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
204
205     final GxDocumentBuilder<N> builder = f.newDocumentBuilder();
206
207     final N document = builder.parse(xmlInput.getResource(),
208     xmlInput.getSystemId());
209
210     final URI xslSystemId = new URI("hotel.xsl");
211     final Resolved<InputStream> xslInput =
212     resolver.resolveInputStream(xslSystemId);
213
214     final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
215     XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
216
217     // poem.xsl uses version="2.0", but we want to use XPath 1.0
218     // compatibility mode
219     // so that arguments to functions are converted etc.
220     compiler.setCompatibleMode(true);
221
222     final GxTransform<I, U, N, A, S, T, X> compiled =
223     compiler.prepareTransform(xslInput.getResource(),
224     xslInput.getSystemId());

```

```

215
216         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
217
218         // TODO: Extract the configuration?
219         // compiled.configure(sf);
220
221         sf.setIndent(true);
222
223         final StringWriter w = new StringWriter();
224
225         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(w);
226
227         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
228
229         transformer.transform(document, pcx, handler);
230     }
231
232     public void testVariableBinding() throws
ParserConfigurationException, IOException, GxException, ExprException,
URISyntaxException
233     {
234         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
235
236         final Resolver resolver = getResolver();
237
238         final URI xslSystemId = new URI("email.xsl");
239         final Resolved<InputStream> xslInput =
resolver.resolveInputStream(xslSystemId);
240
241         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
242
243         final GxTransform<I, U, N, A, S, T, X> compiled =
compiler.prepareTransform(xslInput.getResource(),
xslInput.getSystemId());
244
245         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
246
247         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
248         final SmName<S> varName = nameBridge.name(new QName("to"));
249         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
250         final X value = valueBridge.stringValue("David");
251
252         transformer.bindVariableValue(varName, value);
253         transformer.bindVariableValue(nameBridge.name(new
QName("http://www.example.com", "from")),
valueBridge.stringValue("Julie"));

```

```

254
255         final N documentNode = transformer.transform(null, pcx);
256
257         final GxModel<N, A, S, T> model = pcx.getModel();
258
259         assertEquals(NodeKind.DOCUMENT,
260 model.getNodeKind(documentNode));
261         final N email = model.getFirstChildElement(documentNode);
262         final N to = model.getFirstChildElementByName(email,
263 nameBridge.symbolize("http://www.example.com"),
264 nameBridge.symbolize("to"));
265         assertEquals("David", model.getStringValue(to));
266         final N from = model.getFirstChildElementByName(email, null,
267 nameBridge.symbolize("from"));
268         assertEquals("Julie", model.getStringValue(from));
269         final N again = model.getFirstChildElementByName(email,
270 nameBridge.symbolize("http://www.example.com"), null);
271         assertEquals("David", model.getStringValue(again));
272     }
273
274     public void testExternalFunctions() throws
275 ParserConfigurationException, IOException, GxException, ExprException,
276 URISyntaxException
277     {
278         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
279 newProcessingContext();
280
281         final Resolver resolver = getResolver();
282
283         final URI xmlSystemId = new URI("exslt.xml");
284         final Resolved<InputStream> xmlInput =
285 resolver.resolveInputStream(xmlSystemId);
286
287         final GxDocumentBuilderFactory<N, S> f = new
288 DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
289
290         final GxDocumentBuilder<N> builder = f.newDocumentBuilder();
291
292         final N document = builder.parse(xmlInput.getResource(),
293 xmlInput.getSystemId());
294
295         final URI xslSystemId = new URI("exslt.xsl");
296         final Resolved<InputStream> xslInput =
297 resolver.resolveInputStream(xslSystemId);
298
299         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
300 XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
301
302         final String namespaceURI = "http://exslt.org/strings";
303         final ExsltStringsFunctionGroup<I, U, N, A, S, T, X>
304 functions = new ExsltStringsFunctionGroup<I, U, N, A, S, T,
305 X>(namespaceURI, pcx);
306         compiler.setFunctionSigns(namespaceURI, functions);

```



```

292         compiler.setFunctionImpls(namespaceURI, functions);
293
294         final GxTransform<I, U, N, A, S, T, X> compiled =
295             compiler.prepareTransform(xslInput.getResource(),
296                                     xslInput.getSystemId());
297
298         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
299             SerializerFactory<I, U, N, A, S, T, X>(pcx);
300
301         // TODO: Extract the configuration.
302         // compiled.configure(sf);
303
304         sf.setIndent(true);
305
306         final StringWriter w = new StringWriter();
307
308         final GxSequenceHandler<A, S, T> handler =
309             sf.newSerializer(w);
310
311         final GxTransformer<I, U, N, A, S, T, X> transformer =
312             compiled.newTransformer();
313
314         transformer.transform(document, pcx, handler);
315
316         // System.out.println(w.toString());
317     }
318
319     public void testHotel() throws ParserConfigurationException,
320         IOException, GxException, ExprException, URISyntaxException
321     {
322         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
323             new ProcessingContext();
324
325         final Resolver resolver = getResolver();
326
327         final URI xmlSystemId = new URI("hotel.xml");
328         final Resolved<InputStream> xmlInput =
329             resolver.resolveInputStream(xmlSystemId);
330
331         final GxDocumentBuilderFactory<N, S> f = new
332             DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
333
334         final GxDocumentBuilder<N> builder = f.newDocumentBuilder();
335
336         final N document = builder.parse(xmlInput.getResource(),
337                                           xmlInput.getSystemId());
338
339         final URI xslSystemId = new URI("hotel.xsl");
340         final Resolved<InputStream> xslInput =
341             resolver.resolveInputStream(xslSystemId);
342

```

```

332         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
333
334         final GxTransform<I, U, N, A, S, T, X> compiled =
compiler.prepareTransform(xslInput.getResource(),
xslInput.getId());
335
336         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
337
338         // TODO: Extract the configuration.
339         // compiled.configure(sf);
340
341         sf.setIndent(true);
342
343         final StringWriter w = new StringWriter();
344
345         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(w);
346
347         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
348         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
349         final SmName<S> varName = nameBridge.name(new
QName("MessageData"));
350         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
351         final X value = valueBridge.node(document);
352
353         transformer.bindVariableValue(varName, value);
354
355         transformer.transform(null, pcx, handler);
356
357         // System.out.println(w.toString());
358     }
359
360     public void testHelloWorld() throws Exception
361     {
362         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
363         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
364
365         final GxLanguageToolKit<I, U, N, A, S, T, X> xtk = new
LanguageToolKit<I, U, N, A, S, T, X>(pcx);
366
367         final GxExprContextStaticArgs<I, U, N, A, S, T, X> senv =
xtk.newStaticContextArgs();
368         final String NAMESPACE = "http://www.peanuts.com";
369
370         senv.getInScopeNamespaces().declarePrefix("nuts",
nameBridge.symbolize(NAMESPACE));
371

```

```

372         final SnoopyFunctionGroup<I, U, N, A, S, T, X>
peanutsFunctionGroup = new SnoopyFunctionGroup<I, U, N, A, S, T,
X>(NAMESPACE, pcx);
373         senv.setFunctionSigns(NAMESPACE, peanutsFunctionGroup);
374         senv.setFunctionImpls(NAMESPACE, peanutsFunctionGroup);
375
376         final GxMetaBridge<A, S, T> metaBridge =
pcx.getMetaBridge();
377
378         final ExprResult<I, U, N, A, S, T, X> prepared =
xtk.prepare("nuts:GetVariableProperty('foo','bar')",
metaBridge.emptyType(), senv);
379
380         final GxExpr<I, U, N, A, S, T, X> expr = prepared.getExpr();
381
382         final GxExprContextDynamicArgs<I, U, N, A, S, T, X> darg =
xtk.newDynamicContextArgs();
383
384         final String strval = expr.stringFunction(xtk.emptyFocus(),
darg, pcx);
385
386         assertEquals("Bingo!", strval);
387     }
388 }

```

Validating

```

001 package org.gxml.book.parsing;
002
003 import java.io.InputStream;
004 import java.net.URI;
005
006 import javax.xml.namespace.QName;
007
008 import org.gxml.book.common.SampleApp;
009 import org.gxml.sa.GxApplication;
010 import org.gxml.sa.GxAtomBridge;
011 import org.gxml.sa.GxModel;
012 import org.gxml.sa.GxNameBridge;
013 import org.gxml.sa.GxProcessingContext;
014 import org.gxml.xdm.Resolved;
015 import org.gxml.xdm.Resolver;
016 import org.gxml.xs.SmComponentBag;
017 import org.gxml.xs.SmExceptionCatcher;
018 import org.gxml.xs.SmMetaLoadArgs;
019 import org.gxml.xs.SmName;
020
021 import com.tibco.gxml.sa.common.helpers.DocumentBuilderFactory;

```

```

022 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilder;
023 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilderFactory;
024 import
com.tibco.gxml.sa.common.helpers.SmAtomBridgeOnGxAtomBridgeAdapter;
025 import com.tibco.gxml.xs.W3cXmlSchemaParser;
026
027 public abstract class BookValidatingParsingSample<I, U, N extends I,
A extends I, S, T, X> extends SampleApp<I, U, N, A, S, T, X>
028 {
029     public void testValidatingParse() throws Exception
030     {
031         final GxApplication<I, U, N, A, S, T, X> app = this;
032
033         final Resolver resolver = app.getResolver();
034
035         final SmMetaLoadArgs args = new SmMetaLoadArgs();
036
037         final SmExceptionCatcher errors = new SmExceptionCatcher();
038
039         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
app.newProcessingContext();
040
041         final Resolved<InputStream> resource =
getResolver().resolveInputStream(new URI("email.xsd"));
042
043         final W3cXmlSchemaParser<A, S> parser = new
W3cXmlSchemaParser<A, S>(new SmAtomBridgeOnGxAtomBridgeAdapter<A,
S>(pcx.getAtomBridge()));
044
045         final SmComponentBag<A, S> components =
parser.parse(resource.getLocation(), resource.getResource(),
resource.getSystemId(), errors, args, pcx);
046
047         pcx.register(components);
048
049         pcx.lock();
050
051         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
052
053         assertEquals(0, errors.size());
054
055         final URI xmlURI = new URI("email.xml");
056         final Resolved<InputStream> xmlInput =
resolver.resolveInputStream(xmlURI);
057
058         final GxDocumentBuilderFactory<N, S> factory = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
059
060         // Enable validation of the XML input.
061         factory.setValidating(true, nameBridge.name(new
QName("http://www.example.com", "email")));
062

```

```

063         final GxDocumentBuilder<N> builder =
factory.newDocumentBuilder();
064
065         // TODO: Need to catch errors...
066         // builder.setExceptionHandler(errors);
067
068         final N doc = builder.parse(xmlInput.getResource(),
xmlInput.getSystemId());
069
070         assertEquals(0, errors.size());
071
072         // System.out.println(serialize(doc, pcx));
073
074         final GxModel<N, A, S, T> model = pcx.getModel();
075         final GxAtomBridge<A, S> atomBridge = pcx.getAtomBridge();
076
077         final N email = model.getFirstChildElement(doc);
078         final S namespaceURI =
nameBridge.symbolize("http://www.example.com");
079         final N sent = model.getFirstChildElementByName(email,
namespaceURI, nameBridge.symbolize("sent"));
080         assertNotNull("model.getFirstChildElementByName", sent);
081         final SmName<S> typeName = model.getTypeName(sent);
082         assertNotNull("model.getTypeName", typeName);
083         assertEquals("dateTime", typeName.toQName().getLocalPart());
084         final A dateTime = model.getTypedValue(sent).get(0);
085
086         //
assertTrue(metaBridge.sameAs(metaBridge.handle(pcx.getTypeDefinition(ty
pe))),
087         // metaBridge.getType(SmNativeType.DATETIME)));
088
089         assertEquals("2008-03-23T14:49:30-05:00",
atomBridge.getC14NForm(dateTime));
090     }
091 }

```

Navigation

```

001 package org.gxml.book.parsing;
002
003 import java.io.InputStream;
004 import java.net.URI;
005
006 import org.gxml.book.common.SampleApp;
007 import org.gxml.sa.GxModel;
008 import org.gxml.sa.GxNameBridge;
009 import org.gxml.sa.GxProcessingContext;
010 import org.gxml.xdm.Resolved;
011 import org.gxml.xdm.Resolver;

```

```

012
013 import com.tibco.gxml.sa.common.helpers.DocumentBuilderFactory;
014 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilder;
015 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilderFactory;
016
017 public abstract class BookNavigationParsingSample<I, U, N extends I,
A extends I, S, T, X> extends SampleApp<I, U, N, A, S, T, X>
018 {
019     public void testBooksByNealStephenson() throws Exception
020     {
021         final Resolver resolver = getResolver();
022
023         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
024
025         final URI systemId = new URI("books.xml");
026         final Resolved<InputStream> source =
resolver.resolveInputStream(systemId);
027
028         final GxDocumentBuilderFactory<N, S> factory = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
029
030         final GxDocumentBuilder<N> builder =
factory.newDocumentBuilder();
031
032         final N doc = builder.parse(source.getResource(),
source.getSystemId());
033
034         final GxModel<N, A, S, T> model = pcx.getModel();
035
036         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
037
038         final S namespaceURI =
nameBridge.symbolize("http://www.example.com/books");
039
040         final N inventory = model.getFirstChildElementByName(doc,
namespaceURI, nameBridge.symbolize("inventory"));
041
042         for (final N book : model.getChildElementsByName(inventory,
namespaceURI, nameBridge.symbolize("book")))
043         {
044             boolean found = false;
045
046             for (final N author : model.getChildElementsByName(book,
namespaceURI, nameBridge.symbolize("author")))
047             {
048                 if (model.getStringValue(author).equals("Neal
Stephenson"))
049                 {
050                     found = true;
051                     break;
052                 }

```

```

053         }
054
055         if (found)
056         {
057             final N title =
058                 model.getFirstChildElementByName(book, namespaceURI,
059                 nameBridge.symbolize("title"));
058
059                 System.out.println(model.getStringValue(title));
060         }
061     }
062 }
063
064 public void testPurchaseOrder() throws Exception
065 {
066     final Resolver resolver = getResolver();
067
068     final GxProcessingContext<I, U, N, A, S, T, X> pcx =
069         newProcessingContext();
070     final GxModel<N, A, S, T> model = pcx.getModel();
071     final GxNameBridge<S> nameBridge = pcx.getNameBridge();
072
073     final URI systemId = new URI("PurchaseOrder.xml");
074     final Resolved<InputStream> source =
075         resolver.resolveInputStream(systemId);
076
077     final GxDocumentBuilderFactory<N, S> factory = new
078         DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
079
080     final GxDocumentBuilder<N> builder =
081         factory.newDocumentBuilder();
082
083     final N po = builder.parse(source.getResource(),
084         source.getSystemId());
085
086     final N root = model.getFirstChildElement(po);
087
088     final N items = model.getFirstChildElementByName(root, null,
089         nameBridge.symbolize("items"));
090
091     double total = 0;
092     for (final N item : model.getChildElementsByName(items,
093         null, nameBridge.symbolize("item")))
094     {
095         System.out.println("partNum:" +
096             model.getAttributeStringValue(item, nameBridge.empty(),
097             nameBridge.symbolize("partNum")));
098
099         final N price = model.getFirstChildElementByName(item,
100             null, nameBridge.symbolize("USPrice"));
101         total +=
102             Double.valueOf(model.getStringValue(price)).doubleValue();
103     }

```

```

093         System.out.println("Grand total = " + total);
094     }
095 }

```

Mutation

```

001 package org.gxml.book.mutable;
002
003 import java.math.BigDecimal;
004
005 import javax.xml.XMLConstants;
006
007 import org.gxml.book.common.MutableApp;
008 import org.gxml.sa.GxAtomBridge;
009 import org.gxml.sa.GxNameBridge;
010 import org.gxml.sa.mutable.GxModelMutable;
011 import org.gxml.sa.mutable.GxProcessingContextMutable;
012 import org.gxml.xdm.NodeKind;
013
014 /**
015  * This sample illustrates the use of the optional mutability API.
016  *
017  * @author dholmes
018  *
019  * @param <I>
020  * @param <U>
021  * @param <N>
022  * @param <A>
023  * @param <S>
024  * @param <T>
025  * @param <X>
026  */
027 public abstract class MutableSample<I, U, N extends I, A extends I,
028     S, T, X> extends MutableApp<I, U, N, A, S, T, X>
029 {
030     /**
031      * This is a test of basic mutability through the optional
032      * mutability API.
033      * Line 2
034      * Line 3
035      * Line 4 // OK
036      */
037     public void testIntroduction() throws Exception
038     {
039         final GxProcessingContextMutable<I, U, N, A, S, T, X> pcx =
040             newProcessingContext();
041         final GxAtomBridge<A, S> atomBridge = pcx.getAtomBridge();
042         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
043     }
044 }

```



```

040
041      /* // Create a new document. */
042      final N documentNode = pcx.newDocument();
043
044      final GxModelMutable<N, A, S, T> model = pcx.getModel();
045
046      assertEquals(NodeKind.DOCUMENT,
047        model.getNodeKind(documentNode));
048
049      /* // Every node in the tree has an owner which is a document
050        node. */ OK */
051      final N owner = model.getOwner(documentNode);
052
053      assertTrue(model.isSameNode(documentNode, owner));
054
055      final S namespaceURI =
056        nameBridge.symbolize("http://www.example.com");
057      final S localName = nameBridge.symbolize("foo");
058      final String prefix = "x";
059      final N documentElement = model.createElement(owner,
060        namespaceURI, localName, prefix);
061
062      // Append the document element to the documentNode.
063      model.appendChild(documentNode, documentElement);
064
065      model.setNamespace(documentElement, prefix, namespaceURI);
066
067      model.setAttribute(documentElement, nameBridge.empty(),
068        nameBridge.symbolize("version"), XMLConstants.DEFAULT_NS_PREFIX,
069        atomBridge.wrapAtom(atomBridge.createDecimal(BigDecimal.valueOf(2.7)))));
070
071      // Append four text nodes to the document element.
072      model.appendChild(documentElement, model.createText(owner,
073        "Hello"));
074      model.appendChild(documentElement, model.createText(owner, "
075        "));
076      model.appendChild(documentElement, model.createText(owner,
077        "World"));
078      model.appendChild(documentElement, model.createText(owner,
079        "!"));
080
081      // Compress the four contiguous text nodes into a single text
082      node.
083      model.normalize(documentNode);
084
085      @SuppressWarnings("unused")
086      final String strval = serialize(documentNode, pcx);
087      //System.out.println(strval);
088  }
089 }

```

Serialization

```

001 package org.gxml.book.serialization;
002
003 import java.io.StringWriter;
004
005 import javax.xml.namespace.QName;
006
007 import org.gxml.book.common.SampleApp;
008 import org.gxml.sa.GxModel;
009 import org.gxml.sa.GxProcessingContext;
010 import org.gxml.sa.GxSequenceHandler;
011 import org.gxml.xdm.Emulation;
012
013 import
com.tibco.gxml.sa.processor.serialization.api.GxDocumentSerializer;
014 import
com.tibco.gxml.sa.processor.serialization.api.GxDocumentSerializerFacto
ry;
015 import
com.tibco.gxml.sa.processor.serialization.api.GxSerializerFactory;
016 import
com.tibco.gxml.sa.processor.serialization.impl.DocumentSerializerFactor
y;
017 import
com.tibco.gxml.sa.processor.serialization.impl.SerializerFactory;
018
019 public abstract class IntroSerializationSample<I, U, N extends I, A
extends I, S, T, X> extends SampleApp<I, U, N, A, S, T, X>
020 {
021     public void exampleUsingDocumentSerializer(final N node, final
GxProcessingContext<I, U, N, A, S, T, X> pcx)
022     {
023         final GxDocumentSerializerFactory<N, S> sf = new
DocumentSerializerFactory<I, U, N, A, S, T, X>(pcx);
024
025         // Configure for "pretty" printing.
026         sf.setIndent(Boolean.TRUE);
027         sf.setMethod(new QName("xml"));
028         sf.setOmitXmlDeclaration(false);
029
030         final StringWriter sw = new StringWriter();
031
032         final GxDocumentSerializer<N> serializer =
sf.newDocumentSerializer(sw);
033
034         serializer.serialize(node);
035
036         System.out.print(sw.toString());
037     }
038

```

```

039     public void exampleUsingSequenceHandler(final N node, final
GxProcessingContext<I, U, N, A, S, T, X> pcx)
040     {
041         final GxSerializerFactory<I,U,N,A, S, T,X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
042
043         // Configure for "pretty" printing.
044         sf.setIndent(Boolean.TRUE);
045         sf.setMethod(new QName("xml"));
046         sf.setOmitXmlDeclaration(false);
047         sf.setEmulation(Emulation.C14N);
048
049         final StringWriter sw = new StringWriter();
050
051         final GxSequenceHandler<A, S, T> serializer =
sf.newSerializer(sw);
052
053         final GxModel<N, A, S, T> model = pcx.getModel();
054
055         model.stream(node, true, true, serializer);
056
057         System.out.print(sw.toString());
058     }
059 }

```

XPath

```

001 package org.gxml.book.xpath;
002
003 import org.gxml.book.common.SampleApp;
004 import org.gxml.sa.GxMetaBridge;
005 import org.gxml.sa.GxNameBridge;
006 import org.gxml.sa.GxProcessingContext;
007 import org.gxml.sa.GxVariantBridge;
008 import org.gxml.xdm.Emulation;
009 import org.gxml.xs.SmName;
010 import org.gxml.xs.SmNativeType;
011
012 import com.tibco.gxml.sa.api.common.lang.ExprResult;
013 import com.tibco.gxml.sa.api.common.lang.GxExpr;
014 import com.tibco.gxml.sa.api.common.lang.GxExprContextDynamicArgs;
015 import com.tibco.gxml.sa.api.common.lang.GxExprContextStaticArgs;
016 import com.tibco.gxml.sa.api.common.lang.GxFocus;
017 import com.tibco.gxml.sa.api.common.lang.GxLanguageToolkit;
018 import com.tibco.gxml.sa.processor.xquery.LanguageToolkit;
019 import
com.tibco.xmlsa.processor.org.exslt.math.ExsltMathFunctionGroup;
020

```

```

021 public abstract class XPathSample<I, U, N extends I, A extends I, S,
    T, X> extends SampleApp<I, U, N, A, S, T, X>
022 {
023     public void testGettingStarted() throws Exception
024     {
025         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
            newProcessingContext();
026
027         // For demonstration purposes, register the language toolkit
            with the processing context.
028         pcx.register("xyz", new LanguageToolKit<I, U, N, A, S, T,
            X>(pcx));
029
030         @SuppressWarnings("unchecked")
031         // Immediately get back the registered processor.
032         GxLanguageToolKit<I, U, N, A, S, T, X> xtk =
            pcx.getProcessor("xyz", GxLanguageToolKit.class);
033
034         final GxExprContextStaticArgs<I, U, N, A, S, T, X> sarg =
            xtk.newStaticContextArgs();
035
036         final GxMetaBridge<A, S, T> metaBridge =
            pcx.getMetaBridge();
037
038         final ExprResult<I, U, N, A, S, T, X> prepared =
            xtk.prepare("concat('Hello', ' ', 'World', '!')",
            metaBridge.emptyType(), sarg);
039         final GxExpr<I, U, N, A, S, T, X> expr = prepared.getExpr();
040
041         final GxExprContextDynamicArgs<I, U, N, A, S, T, X> darg =
            xtk.newDynamicContextArgs();
042
043         final String strval = expr.stringFunction(xtk.emptyFocus(),
            darg, pcx);
044
045         assertEquals("Hello, World!", strval);
046     }
047
048     public void testBindingVariables() throws Exception
049     {
050         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
            newProcessingContext();
051
052         final GxLanguageToolKit<I, U, N, A, S, T, X> xtk = new
            LanguageToolKit<I, U, N, A, S, T, X>(pcx);
053
054         final GxExprContextStaticArgs<I, U, N, A, S, T, X> statArgs
            = xtk.newStaticContextArgs();
055         statArgs.setEmulation(Emulation.MODERN);
056
057         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
058         final SmName<S> varName = new
            SmName<S>(nameBridge.symbolize("x"), nameBridge);

```

```

059
060         final GxMetaBridge<A, S, T> metaBridge =
pcx.getMetaBridge();
061         statArgs.bindVariableType(varName,
metaBridge.getType(SmNativeType.STRING));
062
063         final String es = "concat('Hello', ' ', ' ', $x, '!!')";
064         final T sfocus = metaBridge.emptyType();
065
066         final ExprResult<I, U, N, A, S, T, X> prepared =
xtk.prepare(es, sfocus, statArgs);
067
068         final GxExprContextDynamicArgs<I, U, N, A, S, T, X> dynArgs
= xtk.newDynamicContextArgs();
069         dynArgs.setEmulation(Emulation.MODERN);
070
071         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
072         final X value = valueBridge.stringValue("World");
073         dynArgs.bindVariableValue(varName, value);
074
075         final GxExpr<I, U, N, A, S, T, X> expr = prepared.getExpr();
076         final GxFocus<I> dfocus = xtk.emptyFocus();
077         final String strval = expr.stringFunction(dfocus, dynArgs,
pcx);
078
079         assertEquals("Hello, World!", strval);
080     }
081
082     public void testEXSLT() throws Exception
083     {
084         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
085         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
086
087         final GxLanguageToolKit<I, U, N, A, S, T, X> xtk = new
LanguageToolKit<I, U, N, A, S, T, X>(pcx);
088
089         final GxExprContextStaticArgs<I, U, N, A, S, T, X> sarg =
xtk.newStaticContextArgs();
090         sarg.getInScopeNamespaces().declarePrefix("math",
nameBridge.symbolize("http://exslt.org/math"));
091         final ExsltMathFunctionGroup<I, U, N, A, S, T, X>
exsltMathFunctionGroup = new ExsltMathFunctionGroup<I, U, N, A, S, T,
X>("http://exslt.org/math", pcx);
092         sarg.setFunctionSigns("http://exslt.org/math",
exsltMathFunctionGroup);
093         // The function implementations can be provided now or just
prior to execution.
094         sarg.setFunctionImpls("http://exslt.org/math",
exsltMathFunctionGroup);
095

```

```

096         final GxMetaBridge<A, S, T> metaBridge =
pcx.getMetaBridge();
097
098         final ExprResult<I, U, N, A, S, T, X> prepared =
xtk.prepare("math:exp(1)", metaBridge.emptyType(), sarg);
099
100         final GxExpr<I, U, N, A, S, T, X> expr = prepared.getExpr();
101
102         final GxExprContextDynamicArgs<I, U, N, A, S, T, X> darg =
xtk.newDynamicContextArgs();
103         // Here we also (redundantly) provide the function
implementations just prior to execution.
104         darg.setFunctionImpls("http://exslt.org/math",
exsltMathFunctionGroup);
105
106         final String strval = expr.stringFunction(xtk.emptyFocus(),
darg, pcx);
107
108         assertEquals("2.7182818284590455", strval);
109     }
110
111     public void testExpressionType() throws Exception
112     {
113         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
114
115         final GxLanguageToolKit<I, U, N, A, S, T, X> xtk = new
LanguageToolKit<I, U, N, A, S, T, X>(pcx);
116
117         final GxExprContextStaticArgs<I, U, N, A, S, T, X> sarg =
xtk.newStaticContextArgs();
118
119         final GxMetaBridge<A, S, T> metaBridge =
pcx.getMetaBridge();
120
121         final ExprResult<I, U, N, A, S, T, X> prepared =
xtk.prepare("'Hello'", metaBridge.emptyType(), sarg);
122         /* final GxExpr<I, U, N, A, S, T, X> expr =
*/prepared.getExpr();
123         /* final GxExprInfo<T> info = */prepared.getInfo();
124     }
125 }

```

XSLT

```

001 package org.gxml.book.xslt;
002
003 import java.io.IOException;
004 import java.io.InputStream;
005 import java.io.StringReader;

```

```

006 import java.io.StringWriter;
007 import java.net.URI;
008 import java.net.URISyntaxException;
009
010 import javax.xml.namespace.QName;
011 import javax.xml.parsers.ParserConfigurationException;
012
013 import org.gxml.book.common.SampleApp;
014 import org.gxml.sa.GxException;
015 import org.gxml.sa.GxMetaBridge;
016 import org.gxml.sa.GxModel;
017 import org.gxml.sa.GxNameBridge;
018 import org.gxml.sa.GxProcessingContext;
019 import org.gxml.sa.GxSequenceHandler;
020 import org.gxml.sa.GxVariantBridge;
021 import org.gxml.xdm.NodeKind;
022 import org.gxml.xdm.Resolved;
023 import org.gxml.xdm.Resolver;
024 import org.gxml.xs.SmName;
025 import org.gxml.xs.SmNativeType;
026
027 import com.tibco.gxml.sa.api.common.lang.ExprException;
028 import com.tibco.gxml.sa.common.helpers.DocumentBuilderFactory;
029 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilder;
030 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilderFactory;
031 import
com.tibco.gxml.sa.processor.serialization.api.GxSerializerFactory;
032 import
com.tibco.gxml.sa.processor.serialization.impl.SerializerFactory;
033 import com.tibco.gxml.sa.processor.xslt.GxTransform;
034 import com.tibco.gxml.sa.processor.xslt.GxTransformBuilder;
035 import com.tibco.gxml.sa.processor.xslt.GxTransformer;
036 import com.tibco.gxml.sa.processor.xslt.XSLTransformBuilder;
037 import
com.tibco.gxmlsa.processor.org.exslt.strings.ExsltStringsFunctionGroup;
038
039 public abstract class XSLTSample<I, U, N extends I, A extends I, S,
T, X> extends SampleApp<I, U, N, A, S, T, X>
040 {
041     public void testExample() throws ParserConfigurationException,
IOException, GxException, ExprException, URISyntaxException
042     {
043         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
044         final GxMetaBridge<A, S, T> metaBridge =
pcx.getMetaBridge();
045         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
046
047         final Resolver resolver = getResolver();
048
049         final URI xmlSystemId = new URI("hotel.xml");

```

```

050         final Resolved<InputStream> xmlInput =
resolver.resolveInputStream(xmlSystemId);
051
052         final GxDocumentBuilderFactory<N, S> f = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
053         f.setIgnoreComments(false);
054
055         final GxDocumentBuilder<N> builder = f.newDocumentBuilder();
056
057         final N document = builder.parse(xmlInput.getResource(),
xmlInput.getSystemId());
058
059         final URI xslSystemId = new URI("hotel.xsl");
060         final Resolved<InputStream> xslInput =
resolver.resolveInputStream(xslSystemId);
061
062         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
063
064         compiler.setCompatibleMode(true);
065         // compiler.setRestrictedMode(true); // XSLT 2.0 subset for
mapper.
066
067         // Specify the static type for the context item:
068         // document-node(element(*,xs:untyped))
069         final T documentType =
metaBridge.documentType(metaBridge.elementType(new SmName<S>(null,
null, nameBridge), metaBridge.getType(SmNativeType.UNTYPED), false));
070         compiler.setFocus(documentType);
071
072         final GxTransform<I, U, N, A, S, T, X> compiled =
compiler.prepareTransform(xslInput.getResource(),
xslInput.getSystemId());
073
074         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
075
076         // TODO: Extract output configuration.
077         // compiled.configure(sf);
078
079         sf.setIndent(true);
080
081         final StringWriter w = new StringWriter();
082
083         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(w);
084
085         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
086
087         transformer.transform(document, pcx, handler);
088
089         @SuppressWarnings("unused")

```



```

090         final String s = w.toString();
091         // System.out.println(s);
092     }
093
094     @SuppressWarnings("unused")
095     private void bar(final GxProcessingContext<I, U, N, A, S, T, X>
pcx)
096     {
097         try
098         {
099             final GxTransformBuilder<I, U, N, A, S, T, X> builder =
new XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
100
101             final GxTransform<I, U, N, A, S, T, X> transform =
builder.prepareTransform(new StringReader("<x xsl:version='1.0'
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'></x>"), new URI(""));
102
103             final GxTransformer<I, U, N, A, S, T, X> transformer =
transform.newTransformer();
104
105             final N document = transformer.transform(null, pcx);
106
107             final GxModel<N, A, S, T> model = pcx.getModel();
108
109             final N element = model.getFirstChild(document);
110
111             final String name = model.getLocalNameAsString(element);
112
113             // System.out.println("XSLT: " + name);
114         }
115         catch (final Throwable e)
116         {
117             e.printStackTrace();
118         }
119     }
120
121     public void skipVariableBinding() throws
ParserConfigurationException, IOException, GxException, ExprException,
URISyntaxException
122     {
123         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
124
125         final Resolver resolver = getResolver();
126
127         final URI xslSystemId = new URI("email.xsl");
128         final Resolved<InputStream> xslInput =
resolver.resolveInputStream(xslSystemId);
129
130         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
131

```

```

132         final GxTransform<I, U, N, A, S, T, X> compiled =
compiler.prepareTransform(xslInput.getResource(),
xslInput.getId());
133
134         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
135
136         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
137         final SmName<S> varName = nameBridge.name(new QName("to"));
138         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
139         final X value = valueBridge.stringValue("David");
140
141         transformer.bindVariableValue(varName, value);
142         transformer.bindVariableValue(nameBridge.name(new
QName("http://www.example.com", "from")),
valueBridge.stringValue("Julie"));
143
144         final N documentNode = transformer.transform(null, pcx);
145
146         final GxModel<N, A, S, T> model = pcx.getModel();
147
148         assertEquals(NodeKind.DOCUMENT,
model.getNodeKind(documentNode));
149         final N email = model.getFirstChildElement(documentNode);
150         final N to = model.getFirstChildElementByName(email,
nameBridge.symbolize("http://www.example.com"),
nameBridge.symbolize("to"));
151         assertEquals("David", model.getStringValue(to));
152         final N from = model.getFirstChildElementByName(email, null,
nameBridge.symbolize("from"));
153         assertEquals("Julie", model.getStringValue(from));
154         final N again = model.getFirstChildElementByName(email,
nameBridge.symbolize("http://www.example.com"), null);
155         assertEquals("David", model.getStringValue(again));
156     }
157
158     public void skipExternalFunctions() throws
ParserConfigurationException, IOException, GxException, ExprException,
URISyntaxException
159     {
160         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
161
162         final Resolver resolver = getResolver();
163
164         final Resolved<InputStream> xmlInput =
resolver.resolveInputStream(new URI("exslt.xml"));
165
166         final GxDocumentBuilderFactory<N, S> f = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
167
168         final GxDocumentBuilder<N> builder = f.newDocumentBuilder();

```

```

169
170         final N document = builder.parse(xmlInput.getResource(),
xmlInput.getSystemId());
171
172         final Resolved<InputStream> xslInput =
resolver.resolveInputStream(new URI("exslt.xsl"));
173
174         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
175
176         final String namespaceURI = "http://exslt.org/strings";
177         final ExsltStringsFunctionGroup<I, U, N, A, S, T, X>
functions = new ExsltStringsFunctionGroup<I, U, N, A, S, T,
X>(namespaceURI, pcx);
178         compiler.setFunctionSigns(namespaceURI, functions);
179         compiler.setFunctionImpls(namespaceURI, functions);
180
181         final GxTransform<I, U, N, A, S, T, X> compiled =
compiler.prepareTransform(xslInput.getResource(),
xslInput.getSystemId());
182
183         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
184
185         // TODO: Extract configuration.
186         // compiled.configure(sf);
187
188         sf.setIndent(true);
189
190         final StringWriter w = new StringWriter();
191
192         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(w);
193
194         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
195
196         transformer.transform(document, pcx, handler);
197
198         // System.out.println(w.toString());
199     }
200
201     public void skipHotel() throws ParserConfigurationException,
IOException, GxException, ExprException, URISyntaxException
202     {
203         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
204
205         final Resolver resolver = getResolver();
206
207         final Resolved<InputStream> xmlInput =
resolver.resolveInputStream(new URI("hotel.xml"));
208

```

```

209         final GxDocumentBuilderFactory<N, S> f = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
210
211         final GxDocumentBuilder<N> builder = f.newDocumentBuilder();
212
213         final N document = builder.parse(xmlInput.getResource(),
xmlInput.getSystemId());
214
215         final Resolved<InputStream> xslInput =
resolver.resolveInputStream(new URI("hotel.xsl"));
216
217         final GxTransformBuilder<I, U, N, A, S, T, X> compiler = new
XSLTransformBuilder<I, U, N, A, S, T, X>(pcx);
218
219         final GxTransform<I, U, N, A, S, T, X> compiled =
compiler.prepareTransform(xslInput.getResource(),
xslInput.getSystemId());
220
221         final GxTransformer<I, U, N, A, S, T, X> transformer =
compiled.newTransformer();
222         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
223         final SmName<S> varName = nameBridge.name(new
QName("MessageData"));
224         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
225         final X value = valueBridge.node(document);
226
227         transformer.bindVariableValue(varName, value);
228
229         final N documentNode = transformer.transform(null, pcx);
230
231         final GxModel<N, A, S, T> model = pcx.getModel();
232
233         assertEquals(NodeKind.DOCUMENT,
model.getNodeKind(documentNode));
234         final N searchHotelRequest =
model.getFirstChildElement(documentNode);
235         final N parameters =
model.getFirstChildElementByName(searchHotelRequest,
nameBridge.symbolize("http://xmlns.example.com/1189038295781"),
nameBridge.symbolize("parameters"));
236         final N searchHotel =
model.getFirstChildElementByName(parameters,
nameBridge.symbolize("http://www.xyzcorp/procureservice/QueryGDS_Europe
/"), nameBridge.symbolize("searchHotel"));
237         final N country =
model.getFirstChildElementByName(searchHotel,
nameBridge.symbolize("http://www.xyzcorp/procureservice/QueryGDS_Europe
/"), nameBridge.symbolize("country"));
238         assertEquals("USA", model.getStringValue(country));
239     }
240 }

```

XQuery

```

001 package org.gxml.book.xquery;
002
003 import java.io.StringWriter;
004 import java.math.BigInteger;
005 import java.net.URI;
006
007 import javax.xml.namespace.QName;
008
009 import org.gxml.book.common.SampleApp;
010 import org.gxml.sa.GxAtomBridge;
011 import org.gxml.sa.GxNameBridge;
012 import org.gxml.sa.GxProcessingContext;
013 import org.gxml.sa.GxSequenceHandler;
014 import org.gxml.sa.GxVariantBridge;
015 import org.gxml.xs.SmName;
016
017 import com.tibco.gxml.sa.api.common.lang.GxXQConnection;
018 import com.tibco.gxml.sa.api.common.lang.GxXQDataSource;
019 import com.tibco.gxml.sa.api.common.lang.GxXQExpression;
020 import com.tibco.gxml.sa.api.common.lang.GxXQPreparedExpression;
021 import
com.tibco.gxml.sa.processor.serialization.api.GxSerializerFactory;
022 import
com.tibco.gxml.sa.processor.serialization.impl.SerializerFactory;
023 import com.tibco.gxml.sa.processor.xquery.XQEngine;
024 import com.tibco.gxml.sa.processor.xquery.XQErrorCatcher;
025
026 /**
027  * Introduction to XQuery.
028  */
029 public abstract class XQuerySample<I, U, N extends I, A extends I,
S, T, X> extends SampleApp<I, U, N, A, S, T, X>
030 {
031     public void testExample() throws Exception
032     {
033         // Obtain a new processing context from the application.
034         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
035
036         final GxXQDataSource<I, U, N, A, S, T, X> ds = new
XQEngine<I, U, N, A, S, T, X>(pcx);
037
038         final GxXQConnection<I, U, N, A, S, T, X> conn =
ds.getConnection();
039
040         final String expression = "<x>{text{for $i in (1,2,3,4)
return $i * 2}}</x>";
041

```

```

042         final GxXQPreparedExpression<I, U, N, A, S, T, X> expr =
conn.prepareExpression(expression);
043
044         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
045         sf.setMethod(new QName("xml"));
046         sf.setOmitXmlDeclaration(true);
047         final StringWriter sw = new StringWriter();
048         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(sw);
049
050         expr.executeQuery(handler);
051
052         final String actual = sw.toString();
053         assertEquals(expression, "<x>2 4 6 8</x>", actual);
054     }
055
056     public void testGettingStarted() throws Exception
057     {
058         // Obtain a new processing context from the application.
059         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
060
061         final GxXQDataSource<I, U, N, A, S, T, X> ds = new
XQEngine<I, U, N, A, S, T, X>(pcx);
062
063         final GxXQConnection<I, U, N, A, S, T, X> conn =
ds.getConnection();
064
065         final GxXQExpression<I, U, N, A, S, T, X> expr =
conn.createExpression();
066
067         final String es = "for $n in fn:doc('catalog.xml')//item
return fn:data($n/name)";
068
069         final URI systemId = new URI("catalog.xml");
070
071         expr.setBaseURI(systemId);
072
073         @SuppressWarnings("unused")
074         final X value = expr.executeQuery(es);
075     }
076
077     public void testHelloWorld() throws Exception
078     {
079         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
this.newProcessingContext();
080
081         final GxXQDataSource<I, U, N, A, S, T, X> ds = new
XQEngine<I, U, N, A, S, T, X>(pcx);
082

```

```

083         final GxXQConnection<I, U, N, A, S, T, X> conn =
ds.getConnection();
084
085         conn.setScriptingMode(true);
086
087         final String expression = "declare variable $x external;
concat('Hello, ', $x, '!')";
088
089         final GxXQPreparedExpression<I, U, N, A, S, T, X> expr =
conn.prepareExpression(expression);
090
091         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
092         sf.setOmitXmlDeclaration(true);
093         sf.setIndent(false);
094         sf.setMethod(new QName("xml"));
095         final StringWriter sw = new StringWriter();
096         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(sw);
097
098         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
099         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
100
101         final SmName<S> varName = new
SmName<S>(nameBridge.symbolize("x"), nameBridge);
102         final X value = valueBridge.stringValue("World");
103
104         expr.bindVariableValue(varName, value);
105
106         expr.executeQuery(handler);
107
108         String actual = sw.toString();
109         assertEquals(expression, "Hello, World!", actual);
110     }
111
112     public void testMergeTextNodes() throws Exception
113     {
114         // Obtain a new processing context from the application.
115         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
116
117         final GxXQDataSource<I, U, N, A, S, T, X> ds = new
XQEngine<I, U, N, A, S, T, X>(pcx);
118
119         final GxXQConnection<I, U, N, A, S, T, X> conn =
ds.getConnection();
120
121         // final String expression = "";
122         final String expression = "count((element elem {1, 'string',
1,2e3})/text())";
123

```

```

124         final GxXQPreparedExpression<I, U, N, A, S, T, X> expr =
conn.prepareExpression(expression);
125
126         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
127         sf.setMethod(new QName("xml"));
128         sf.setOmitXmlDeclaration(true);
129         final StringWriter sw = new StringWriter();
130         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(sw);
131
132         expr.executeQuery(handler);
133
134         final String actual = sw.toString();
135         assertEquals(expression, "1", actual);
136     }
137
138     public void testProblem() throws Exception
139     {
140         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
this.newProcessingContext();
141
142         final GxXQDataSource<I, U, N, A, S, T, X> ds = new
XQEngine<I, U, N, A, S, T, X>(pcx);
143
144         final GxXQConnection<I, U, N, A, S, T, X> conn =
ds.getConnection();
145
146         final XQErrorCatcher messages = new XQErrorCatcher();
147
148         conn.setErrorHandler(messages);
149         conn.setCompatibleMode(false);
150         conn.setScriptingMode(true);
151
152         final String expression =
"(xs:untypedAtomic('1'),xs:untypedAtomic('2')) =
(xs:untypedAtomic('2.0'),2.0)";
153
154         final GxXQPreparedExpression<I, U, N, A, S, T, X> expr =
conn.prepareExpression(expression);
155
156         final X value = expr.executeQuery();
157
158         final GxVariantBridge<I, N, A, X> variantBridge =
pcx.getVariantBridge();
159         switch (variantBridge.getNature(value))
160         {
161             case ITEMS:
162             {
163                 @SuppressWarnings("unused")
164                 final Iterable<I> items =
variantBridge.getItemSet(value);

```



```

165             // System.out.println(items);
166         }
167         break;
168         case ATOM:
169         {
170             @SuppressWarnings("unused")
171             final A atom = variantBridge.getAtom(value);
172             @SuppressWarnings("unused")
173             final GxAtomBridge<A, S> atomBridge =
pcx.getAtomBridge();
174             // System.out.println(atomBridge.getC14NForm(atom));
175         }
176         break;
177         case STRING:
178         {
179             @SuppressWarnings("unused")
180             final String strval = variantBridge.getString(value);
181             // System.out.println(strval);
182         }
183         break;
184         case INTEGER:
185         {
186             @SuppressWarnings("unused")
187             final BigInteger integer =
variantBridge.getInteger(value);
188             // System.out.println(integer);
189         }
190         break;
191         default:
192         {
193             throw new
AssertionError(variantBridge.getNature(value));
194         }
195     }
196 }
197
198 public void testTyping() throws Exception
199 {
200     final GxProcessingContext<I, U, N, A, S, T, X> pcx =
this.newProcessingContext();
201
202     final GxXQDataSource<I, U, N, A, S, T, X> ds = new
XQEngine<I, U, N, A, S, T, X>(pcx);
203
204     final GxXQConnection<I, U, N, A, S, T, X> conn =
ds.getConnection();
205
206     conn.setScriptingMode(true);
207
208     final XQErrorCatcher messages = new XQErrorCatcher();
209

```

```

210         conn.setErrorHandler(messages);
211
212         final String expression = "declare variable $x external;
contains(string(number($x)), 'NaN')";
213
214         final GxXQPreparedExpression<I, U, N, A, S, T, X> expr =
conn.prepareExpression(expression);
215
216         final GxSerializerFactory<I, U, N, A, S, T, X> sf = new
SerializerFactory<I, U, N, A, S, T, X>(pcx);
217         sf.setOmitXmlDeclaration(true);
218         sf.setIndent(false);
219         sf.setMethod(new QName("xml"));
220         final StringWriter sw = new StringWriter();
221         final GxSequenceHandler<A, S, T> handler =
sf.newSerializer(sw);
222
223         final GxNameBridge<S> nameBridge = pcx.getNameBridge();
224         final GxVariantBridge<I, N, A, X> valueBridge =
pcx.getVariantBridge();
225
226         final SmName<S> varName = new
SmName<S>(nameBridge.symbolize("x"), nameBridge);
227         final X value = valueBridge.doubleValue(5.0);
228
229         expr.bindVariableValue(varName, value);
230
231         expr.executeQuery(handler);
232
233         String actual = sw.toString();
234         assertEquals(expression, "false", actual);
235     }
236 }

```

Validation

```

001 package org.gxml.book.validation;
002
003 import java.io.InputStream;
004 import java.net.URI;
005 import java.util.LinkedList;
006 import java.util.List;
007
008 import org.gxml.book.common.SampleApp;
009 import org.gxml.sa.GxFragmentBuilder;
010 import org.gxml.sa.GxModel;
011 import org.gxml.sa.GxProcessingContext;
012 import org.gxml.xdm.Resolved;
013 import org.gxml.xdm.Resolver;

```

```

014 import org.gxml.xs.SmException;
015 import org.gxml.xs.SmExceptionCatcher;
016 import org.gxml.xs.SmExceptionHandler;
017 import org.gxml.xs.SmMetaLoadArgs;
018
019 import com.tibco.gxml.sa.common.helpers.DocumentBuilderFactory;
020 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilder;
021 import com.tibco.gxml.sa.common.helpers.GxDocumentBuilderFactory;
022 import
com.tibco.gxml.sa.common.helpers.SmAtomBridgeOnGxAtomBridgeAdapter;
023 import com.tibco.gxml.sa.processor.validation.GxContentValidator;
024 import com.tibco.gxml.sa.processor.validation.GxValidatorCache;
025 import
com.tibco.gxml.sa.processor.validation.GxValidatorCacheFactory;
026 import
com.tibco.gxml.sa.processor.validation.ValidatorCacheFactory;
027 import com.tibco.gxml.xs.W3cXmlSchemaParser;
028
029 public abstract class ValidationSample<I, U, N extends I, A extends
I, S, T, X> extends SampleApp<I, U, N, A, S, T, X>
030 {
031     public void testByteStreamValidation() throws Exception
032     {
033         // Load a top-level schema into the processing context.
034         final List<Resolved<InputStream>> resources = new
LinkedList<Resolved<InputStream>>();
035         resources.add(getResolver().resolveInputStream(new
URI("PurchaseOrder.xsd")));
036
037         final SmExceptionCatcher errors = new SmExceptionCatcher();
038         final SmMetaLoadArgs args = new SmMetaLoadArgs();
039
040         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
041
042         final W3cXmlSchemaParser<A, S> parser = new
W3cXmlSchemaParser<A, S>(new SmAtomBridgeOnGxAtomBridgeAdapter<A,
S>(pcx.getAtomBridge()));
043
044         for (final Resolved<InputStream> resource : resources)
045         {
046             pcx.register(parser.parse(resource.getLocation(),
resource.getResource(), resource.getSystemId(), errors, args, pcx));
047         }
048
049         pcx.lock();
050
051         // Create a validator...
052         final GxValidatorCacheFactory<A, S, T> vcf = new
ValidatorCacheFactory<I, U, N, A, S, T, X>(pcx);
053         final GxValidatorCache<A, S, T> vc =
vcf.newValidatorCache();

```

```

054         final GxContentValidator<A, S, T> validator =
vc.newContentValidator();
055
056         // Set the downstream event handler which contains
annotations and typed content.
057         // validator.setGxContentHandler(/* ...*/null);
058         validator.setExceptionHandler(errors);
059
060         // The document node that we wish to validate.
061         final Resolved<InputStream> xmlInput =
getResolver().resolveInputStream(new URI("PurchaseOrder.xml"));
062
063         final GxDocumentBuilderFactory<N, S> factory = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
064
065         final GxDocumentBuilder<N> builder =
factory.newDocumentBuilder();
066
067         final N document = builder.parse(xmlInput.getResource(),
xmlInput.getSystemId());
068
069         // Stream the document into the validator.
070         final GxModel<N, A, S, T> model = pcx.getModel();
071
072         model.stream(document, true, true, validator);
073
074         if (errors.size() > 0)
075         {
076             // You've got errors.'
077         }
078     }
079
080     public void testTreeValidation() throws Exception
081     {
082         final Resolver resolver = getResolver();
083
084         // Load a top-level schema into the processing context.
085         final List<Resolved<InputStream>> resources = new
LinkedList<Resolved<InputStream>>();
086         resources.add(getResolver().resolveInputStream(new
URI("PurchaseOrder.xsd")));
087
088         final SmExceptionCatcher errors = new SmExceptionCatcher();
089         final SmMetaLoadArgs args = new SmMetaLoadArgs();
090
091         final GxProcessingContext<I, U, N, A, S, T, X> pcx =
newProcessingContext();
092         final W3cXmlSchemaParser<A, S> parser = new
W3cXmlSchemaParser<A, S>(new SmAtomBridgeOnGxAtomBridgeAdapter<A,
S>(pcx.getAtomBridge()));
093         for (final Resolved<InputStream> resource : resources)
094         {

```

```

095         pcx.register(parser.parse(resource.getLocation(),
resource.getResource(), resource.getSystemId(), errors, args, pcx));
096     }
097     pcx.lock();
098     // The document node that we wish to validate.
099     @SuppressWarnings("unused")
100     final URI xmlLocation = new URI("PurchaseOrder.xml");
101     final URI xmlSystemId = new URI("PurchaseOrder.xml");
102     final Resolved<InputStream> xmlInput =
resolver.resolveInputStream(xmlSystemId);
103
104     final GxDocumentBuilderFactory<N, S> factory = new
DocumentBuilderFactory<I, U, N, A, S, T, X>(pcx);
105
106     final GxDocumentBuilder<N> builder =
factory.newDocumentBuilder();
107
108     final N documentIn = builder.parse(xmlInput.getResource(),
xmlInput.getSystemId());
109
110     @SuppressWarnings("unused")
111     final N documentOut = validate(documentIn, errors, pcx);
112
113     if (errors.size() > 0)
114     {
115         // You've got errors.'
116         for (@SuppressWarnings("unused")
117             final SmException error : errors)
118         {
119             // System.out.println(error.getLocalizedMessage());
120         }
121     }
122 }
123
124 /**
125  * This static function illustrates a helper function for
126  * validating a document tree. <br/>
127  * Note that we assume that the processing context is already
128  * loaded with meta-data.
129  *
130  * @param node
131  *         The input document.
132  * @param errors
133  *         The error handler.
134  * @param pcx
135  *         The processing context.
136  */
137 public static <I, U, N extends I, A extends I, S, T, X> N
validate(final N node, final SmExceptionHandler errors, final
GxProcessingContext<I, U, N, A, S, T, X> pcx)
138 {

```

```

137         final GxValidatorCacheFactory<A, S, T> vcf = new
ValidatorCacheFactory<I, U, N, A, S, T, X>(pcx);
138
139         // We already have a tree as input so we'll use the content
validator'
140         // and stream the document in as a bunch of events (a bit
like SAX, but not lexical).
141         final GxValidatorCache<A, S, T> vc =
vcf.newValidatorCache();
142
143         final GxContentValidator<A, S, T> validator =
vc.newContentValidator();
144
145         validator.setExceptionHandler(errors);
146
147         final GxModel<N, A, S, T> model = pcx.getModel();
148
149         // We want to produce a node so we'll need a fragment builder
at the output.'
150         final GxFragmentBuilder<N, A, S, T> builder =
pcx.newFragmentBuilder();
151
152         // Connect the pieces together so that the validation output
builds a tree.
153         validator.setGxContentHandler(builder);
154
155         // Make it so!
156         model.stream(node, true, true, validator);
157
158         // Practice safe coding: We don't know what might happen if
there are errors.'
159         final List<? extends N> nodes = builder.getNodes();
160         if (nodes.size() > 0)
161         {
162             return nodes.get(0);
163         }
164         else
165         {
166             return null;
167         }
168     }
169

```

Index

Symbols

@Property Java Annotation Optional Parameters 33

A

Active Enterprise Palette 90
 ActivityTypes 90
 Advice Configuration Properties 12, 12
 Advice Execution Model 17, 17
 Advice Implementation Properties 32, 32
 Advice Implementations 26, 26
 Advice Ordering 14, 14
 Advice Scope Mode 35
 Advices, Advice Instances and Advice Implementa-
 tion Instances 11, 11
 API Functionality 71
 API's and New Interfaces 71
 Application Scope Mode 36
 argument, parameter, and option 89
 Aspect Oriented Programming (AOP) Terminology 3,
 3
 Asynchronous Advice Example 54, 54

B

BusinessWorks Active Aspects Plugin Resources 8
 BWAA Palette 73

C

Comparing Checkpointing and Hiberante 59
 Comparing Checkpointing and Hibernate 59

Configuration 74
 Connecting with TIBCO Resources xiii
 Constructing a Data Model Tree
 Programmatically 108
 customer support xiii

D

dataAccess 28
 Defining a Hibernate Advice Implementation 60, 60
 Deploying Packaged Aspects in BW Engine 25, 25
 Developing gXML Applications 99

E

Example of Hibernate Advice Implementation 61, 61
 Example of Resuming a Job 63, 63
 Examples of Aspect Manifest File 23
 Examples of Point Cuts Defined Using Query
 Language 49, 49
 Execution Model (Successful Execution) 53, 53
 Execution Model (timeout) 53, 53
 Execution of a Hibernate Advice 62, 62

F

Features 58, 58
 File Palette 90
 FTP Palette 91

G

General Palette [91](#)
 gXML Recipes [106](#)

H

Hibernate Resume with Asynchronous Advices [66](#)
 hibernatesJobs [29](#)
 How to Access All TIBCO Documentation [xiii](#)
 How to Contact TIBCO Support [xiii](#)
 How to Join TIBCOCommunity [xiii](#)
 HTTP Palette [93](#)

I

Implementing GxApplication [99](#)
 Implementing GxCatalog [102](#)
 Implementing GxResolver [102](#)
 Injecting DOM [104](#)
 Input [74](#)
 Introduction [2](#), [2](#), [42](#), [42](#), [52](#), [52](#), [76](#)

J

Java Annotations for Advertising Advice Implementation Metadata [26](#), [26](#)
 Java Palette [93](#)
 JDBC Palette [93](#)
 JMS Palette [93](#)

K

kind [45](#), [46](#)

L

Limitations [70](#)

M

Mail Palette [94](#)
 Manual Work Palette [96](#)
 Modifying the Hibernate Data [66](#), [66](#)
 Mutation [122](#)

N

Navigation [119](#)
 NoTitle [97](#), [97](#), [97](#)

O

Object Sharing Between Java Activities and Advice Implementation [67](#)
 Other TIBCO Product Documentation [ix](#)
 Other TIBCO Products [x](#)
 Output [74](#)
 Overview [1](#), [5](#), [5](#), [68](#), [98](#)

P

Packaging and Deployment of Advice Implementations [39](#), [39](#)
 Packaging and Deployment of Aspects [22](#), [22](#)
 Parentheses Support [47](#)
 Parse Palette [94](#)
 Parsing [106](#)
 Parsing a Character Stream and a Byte Stream [106](#)
 Process Aspect [8](#), [8](#)
 Process Join Point [7](#), [7](#)
 Properties Defined for activity() primitive [44](#)

Properties Defined for engine() primitive [47](#)
 Properties Defined for process() primitive [46](#)
 Properties Defined for project() primitive [46](#)

Q

Query Language Primitives [43](#)

R

Related Documentation [ix](#)
 required [33](#)
 Restrictions Imposed by TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in for Advice implementation Properties [34](#)
 Resume [74](#)
 Resuming the Hibernated Job [62, 62](#)
 RMI Palette [94](#)
 Roles and Responsibilities [6](#)
 RV Palette [94](#)

S

sample tables [90](#)
 scope [27](#)
 Scopes [35, 35](#)
 Serialization [124](#)
 Service Palette [95](#)
 SOAP Palette [95](#)
 Summary [55, 55, 70](#)
 support, contacting [xiii](#)
 Supported [69, 70](#)
 Supported Operators [48](#)

T

tables [90](#)

targetFilter [31](#)
 targetKind [30](#)
 TCP Palette [95](#)
 technical support [xiii](#)
 terminology conventions [89](#)
 The @AdviceImpl Java Annotation [26, 26](#)
 The @Property Java Annotation [32, 32](#)
 Threading Model
 Asynchronous Advice Implementations [54, 54](#)
 Asynchronous Advice Implementations (Timeout) [55, 55](#)
 TIBCO ActiveMatrix BusinessWorks [ix](#)
 TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in [8](#)
 TIBCO ActiveMatrix BusinessWorks ActiveAspects Plug-in Documentation [ix](#)
 TIBCO_HOME [xi](#)
 timeDelay [60](#)
 Transaction Palette [95](#)
 type [44](#)
 Typographical Conventions [xi](#)

U

Unsupported [69, 70](#)
 Use of Escape Character [47](#)
 User Scenarios [69](#)
 Using a Database for Hibernation [65, 65](#)

V

Validating [117](#)
 Validation [140](#)

W

Wildcard Support [47](#)
 Working of Asynchronous Activities in BW Engine [53](#)
 Working of Asynchronous Advices in BW Engine [53](#)

X

XML Activities Palette [96](#)

XML Document Access [37](#), [37](#)

XPath [125](#)

XQuery [135](#)

XSLT [128](#)