

TIBCO BusinessEvents™

User's Guide

*Software Release 3.0.1
November 2008*

The Power to Predict™

 **TIBCO®**
The Power of Now®

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.PDF) 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.

TIB, TIBCO, TIBCO Software, TIBCO Adapter, Predictive Business, Information Bus, The Power of Now, The Power to Predict, TIBCO BusinessEvents, TIBCO BusinessWorks, TIBCO Rendezvous, TIBCO Enterprise Message Service, TIBCO PortalBuilder, TIBCO Administrator, TIBCO Runtime Agent, TIBCO General Interface, 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, JMS and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Excerpts from Oracle Coherence documentation are included with permission from Oracle and/or its affiliates. Copyright © 2000, 2006 Oracle and/or its affiliates. All rights reserved.

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.TXT FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

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

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

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

Copyright © 2004-2008 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Figures	xix
Tables	xxi
BusinessEvents Palette Resources	xxiii
Preface	xxv
Enterprise Suite and Inference Edition Features	xxvi
Related Documentation	xxvii
TIBCO BusinessEvents Documentation	xxvii
Other TIBCO Product Documentation	xxviii
Typographical Conventions	xxix
How to Contact TIBCO Support	xxxii
Chapter 1 Overview	1
What's Different About Complex Event Processing	2
Technical Requirements of a CEP System	3
A Model-Driven Approach	3
Stateful Rule Engine	4
Object Management Options	5
BusinessEvents Major Components	6
TIBCO Designer—Design-time Activities	6
TIBCO Administrator—Deploy and Runtime Activities	8
BusinessEvents Runtime Activities	8
Decision Manager and Rules Management Server	8
Design Concepts	9
Channels and Destinations	9
Events	9
Concepts and Concept Instances	11
Score Cards	12
Rules and Rule Sets	12
State Modeler	13
Object Management and Fault Tolerance	14
Query Language for Querying Cached Objects	14
Deployment Concepts	15

Deployment Files	15
Deploytime Configuration and Deployment Methods	16
Hot Deployment	17
Runtime Architecture and Flow	18
Rule Evaluation and Execution	19
Summary of Tasks	20
Design Tasks	20
Deployment and Runtime Tasks	22
Chapter 2 Working With Channels and Destinations	23
Understanding Channels and Destinations	24
Local Channels	25
Selecting a Serializer	25
Deploy-time Configuration	26
Message Acknowledgment	26
Selecting a JMS Serializer	27
BytesMessageSerializer	27
TextMessageSerializer	28
Each JMS Input Destination Runs a Session	28
Selecting a Rendezvous Serializer	29
Rendezvous Message Header	29
Basic Serializer	29
Serializer For Events With Payloads	29
Including a Payload in a Rendezvous Message	30
Working With Channels and Destinations	31
Creating a Channel	31
Creating a JMS or Rendezvous Destination	31
Reconnecting to a JMS Server	32
Creating a Local Destination	33
Using a Local Destination in a Rule	33
Channel Resource Reference	34
Configuration	34
Destination Resource Reference	37
Configuration	37
Mapping Incoming Messages to Non-default Events	42
Creating Unique JMS DurableSubscriber Name Properties	43
Changing the JMS Message Acknowledgment Mode	44
Using JMS Header Properties in Incoming and Outgoing Messages	47
Setting Certain Header Properties in Destinations	47
Setting Header Properties in Events	47
Internal Use of JMSReplyTo Header Property for Request-Response Scenario	48

JMS Header Field Names.....	49
Chapter 3 Working With Simple Events.....	51
Understanding Simple Events.....	52
Inheritance.....	52
Events and Message Acknowledgement.....	53
Time To Live.....	53
Expiry Actions.....	54
Specifying Non-Default Destinations.....	54
Working With Simple Events.....	55
SimpleEvent Resource Reference.....	56
Configuration.....	56
Properties.....	58
Payload.....	59
Extended Properties.....	61
Simple Event Attributes Reference.....	62
Chapter 4 Working With Time Events.....	63
Understanding Time Events.....	64
Repeating Time Events.....	64
Rule Based Time Events.....	64
Working With Time Events.....	65
Creating a Time Event.....	65
Configuring a Rule Based Time Event in a Rule.....	65
TimeEvent Resource Reference.....	66
Configuration.....	66
Extended Properties.....	67
TimeEvent Attributes Reference.....	68
Rule Based TimeEvent Function Reference.....	69
Chapter 5 Working With Advisory Events.....	71
Understanding and Working With Advisory Events.....	72
Working With Advisory Events.....	73
Advisory Event Attributes Reference.....	74
Chapter 6 Working With Concepts.....	77
Understanding Concepts.....	78
Understanding Concept Property History.....	80
History Size.....	80
History Policy.....	82

Understanding Concept Relationships	84
Inheritance Relationships	84
Containment Relationships	85
Reference Relationships	86
Rules Governing Containment and Reference Relationships	87
When a Contained or Referred Concept Instance is Deleted	89
Working With Concepts and Concept Relationships	90
Creating a Concept	90
Creating Concept Relationships	90
Concept Resource Reference	92
Configuration	92
Properties	93
Extended Properties	94
Concept Attributes Reference	95
Working With Concept Views	96
The Concept View User Interface	96
Using Existing Concepts and Creating New Concepts	97
Creating Relationships Between Concepts	97
Concept View Resource Reference	98
Generating XML Schemas (XSD files) for Concepts and Events	99
Configuring How to Handle Null Concept Property Values	101
Enabling Use of the Nillable Attribute	101
Enabling Null Property Values to Appear When Serializing Concepts to XML	101
Examples of Nillable Attribute and Null Properties Settings	102
Enabling and Setting Special Treatment of Numeric Null Values	103
Setting Design-time Properties for Handling of Null Property Values	103
Setting Runtime Properties for Special Treatment of Null Values	104
TIBCO Designer Property Reference for Null Property Handling	105
Chapter 7 Working With Database Concepts	107
Database Concepts Overview	108
Supported Database Products	108
Importing Database Tables or Views With the DB Import Utility	109
Database Concepts Extended Properties	113
Extended Properties for a Concept	113
Extended Properties for a Concept Property	113
Performing Database Operations	116
Setting and Unsetting a Connection	117
Testing the Database Connection Periodically	117
Defining Transactions	117
Performing Insert Operations	118
Performing Update and Delete Operations	119

Performing Database Query Operations	121
Chapter 8 Working With Scorecards	123
Understanding and Working With Scorecards	124
Creating a Scorecard	124
ScoreCard Resource Reference	125
Chapter 9 Working With Rules and Functions	127
Understanding Rules and Rule Functions	128
Rules	128
Rule Sets	128
Rule Functions	129
Virtual Rule Functions and Decision Manager	129
Understanding Conflict Resolution and Run to Completion Cycles	130
How a Rule Becomes Newly True	132
How Conflict Resolution Uses Rule Dependencies	132
Order of Evaluation for Different Types of Rule Conditions	133
Working With Rule Sets, Rules, and Rule Functions	134
Creating Rule Sets and Rules	134
Creating Rule Functions	135
Using Rule Functions	135
Working With the Rule Editor	136
Opening the Rule Editor	136
Adding Terms to the Declaration Panel	136
Specifying Conditions in the Conditions Panel	137
Specifying Actions in the Actions Panel	137
Using Global Variables	138
Using Functions	138
Using XSLT Mapper Functions	139
Working With the Rule Editor (and BAR) Search for Text Utility	141
Adding Advisory Events to Rules	141
Working With Startup and Shutdown Rule Functions	142
When Startup Rule Functions Execute	143
Creating Entities With a Startup Action in a Multi-Engine Project	143
ActiveMatrix BusinessWorks Containers	144
Working With Event Preprocessors	145
How Configured	145
Worker Threading and Queue Options	146
Locking and Synchronization Functions in Preprocessors	147
RuleFunction Resource Reference	148
Design Panel (Rule Function Editor)	148
Configuration	149

Extended Properties	150
Ruleset Resource Reference	151
Configuration	151
Rule Resource Reference	152
Design Panel (Rule Editor)	152
Configuration	153
Understanding and Working With Functions	154
Function Catalogs	154
Ontology Functions	156
Tool Tips	157
Decorations Indicating Where Functions can be Called	157
VRF Functions	158
Temporal Functions	160
Chapter 10 Working With The State Modeler	163
State Modeler Overview	164
State Machine Resource Reference	166
Name	166
Main State Machine	166
Timeouts	167
Functions	167
Inheritance	167
CallStateMachine Resource Reference	168
Call Explicitly	168
Configuring	169
State Machine States Reference	170
Start and End States	170
Simple State	170
Composite State	170
Concurrent State	171
Configuring a Start State	171
State Timeouts	171
Timeout State	171
Transitions	173
Self Transitions	173
Complex Transitions	173
Configuring a Transition	176
Entry and Exit Actions	177
BusinessEvents State Modeler Palette	178
Exporting a State Model to SVG Format	180

Chapter 11 Out-of-Process ActiveMatrix BusinessWorks Integration	181
Understanding Out-of-Process ActiveMatrix BusinessWorks Integration	182
Working With the BusinessEvents Activities	184
Receive Event Resource Reference	185
Configuration	185
Misc	186
Output	186
Send Event Resource Reference	187
Configuration	187
Input	187
Wait for Event Resource Reference	188
Configuration	188
Event	189
Input	189
Output	190
Chapter 12 In-Process ActiveMatrix BusinessWorks Integration	191
Understanding In-Process TIBCO ActiveMatrix BusinessWorks Integration	192
Feature Summary	193
Design Considerations	195
Integration Scope	195
Avoiding Threading Issues	195
Use of Global Variables and Environment Variables	195
Design Considerations Related to Container	196
Fault Tolerance With a BusinessEvents Container	197
Tips for Working With ActiveMatrix BusinessWorks Containers	197
Tips for Working With BusinessEvents Containers	197
Configuring the Environment for BusinessEvents Containers	198
Invoking a BusinessEvents Rule Function From ActiveMatrix BusinessWorks	204
Specifying Input Arguments	204
Using Synchronous Invocation	205
Using the lockWM Parameter	205
Overriding the Rule Function at Runtime	205
Specifying the Rule Service Provider for ActiveMatrix BusinessWorks Containers	206
RuleServiceProvider Configuration Resource Reference	207
Configuration	207
Working With Invoke RuleFunction Activities	208
Invoke RuleFunction Resource Reference	210
Configuration	210
Input	211
Output	212

Working With the BusinessWorks Functions	213
Configuring and Using invokeProcess().	213
Configuring and Using startProcess()	215
Configuring and Using cancelProcess()	217
Configuring and Using init()	217
Configuring and Using shutdown().	217
Deploying the Integration Project	219
When ActiveMatrix BusinessWorks is the Container	219
When BusinessEvents is the Container	220
Chapter 13 BusinessEvents Performance Profiler	221
Overview of Profiler	222
Working With the Profiler	223
To Turn Profiler On and Off Using Engine Properties	223
To Turn Profiler On and Off Using Functions	225
To Turn Profiler On and Off Using TIBCO Hawk Methods	226
Profiler Reference	228
Chapter 14 Understanding Object Management and Fault Tolerance	233
Object Management (OM) and Fault Tolerance Overview	234
Object Management	234
Fault Tolerance	235
Summary of Object Management Options	235
Summary of Object Management and Fault Tolerance Features	237
Mixing Object Management Options	237
Migrating to a Different Object Management Method	238
Object Management and Fault Tolerance Scenarios	239
In Memory Object Management and Fault Tolerance Scenarios	239
Persistence Object Management and Fault Tolerance Scenarios	240
Cache Object Management and Fault Tolerance Scenarios	241
Chapter 15 Configuring In Memory Object Management	243
Understanding In Memory-Based Object Management	244
Configuring In Memory Object Management	245
Configuring Fault Tolerance for In Memory OM Systems	246
Fault Tolerance Configuration for In Memory OM—Using TRA Files	246
Fault Tolerance Configuration for In Memory OM—Using TIBCO Administrator	249
Chapter 16 Configuring Persistence Object Management	251
Understanding Persistence-Based Object Management	252
Fault Tolerance With Persistence Object Management	253

Data Migration to Cache Object Management With Backing Store	253
Configuring Persistence Object Management	254
BusinessEvents Archive Resource Object Management Tab—Persistence Settings Reference	255
Engine Property File Settings for Persistence Object Management	260
Chapter 17 Understanding Cache OM and Multi-Engine Features	263
Cache Object Management and Multi-Engine Feature Overview	264
Distributed Cache Characteristics	264
Scaling the System	264
Reliability of Cache Object Management	264
Backing Store for Data Persistence	265
Multi-Engine Features	266
Cache Modes for Individual Entity Types	266
Characteristics of Distributed Caching Schemes	267
Failover and Failback of Distributed Cache Data	268
Limited and Unlimited Schemes	268
Distributed Cache and Multi-Engine Architecture and Terms	269
Cache Clusters	270
Cache Cluster Nodes	270
Inference Agents	270
Cache Server Nodes (Storage Nodes)	271
Query Agents	271
Load Balancing and Fault Tolerance Between Inference Agents	272
Load Balancing of Inference Agents in a Group (Multi-Engine Mode Only)	272
Fault Tolerance Between Inference Agents in a Group	272
Designing With Multiple Active Inference Agents	274
Concepts are Shared Across Agents Asynchronously	274
Scorecards are Local to the Agent	274
Events are Clustered but not Shared Across Agents in a Group	275
Understanding Concurrency and Locking Issues	276
Multi-Engine Example	276
Tuning the Threading Options	278
Cache OM Configuration Task Summary	279
Chapter 18 Understanding and Working With Cache Modes	281
Working With Cache Modes	282
Cache Plus Memory—The Default Cache Mode	282
In Memory Only—Useful for Stateless Entities	283
Cache Only Mode	284
Design Constraints With Cache Only Cache Mode	285
Explicitly Loading Objects into Working Memory, With the Cache Only Mode	285

Objects are Removed From Working Memory at End of RTC	286
Chapter 19 Configuring Cache Cluster Discovery	287
Cache Cluster Discovery Overview	288
Cluster Member Discovery Using Multicast Discovery	288
Cluster Member Discovery Using Well-Known-Addresses	289
Overriding and Adding Cluster Configuration Properties	289
Discovering Cache Cluster Members Using Multicast	290
Discovering Cache Cluster Members Using Well-Known-Addresses	292
Discovery When Hosts Have Multiple Network Cards	294
Chapter 20 Configuring Cache Cluster Settings	295
Configuring Caching Scheme, Multi-Engine, and Cluster Properties	296
Cluster Level Options, Summarized by Task	296
Cluster Level Option Details	297
Configuring Cache Cluster Logging Properties	301
Example Cluster Configuration Properties	303
Cluster Properties That Must be the Same in All Nodes	303
Cluster Properties That Can Differ Across Nodes	304
Cluster Properties Set in Some Nodes, As Needed	304
Chapter 21 Configuring Inference Agents (Cache OM)	305
Inference Agent Configuration Overview	306
Local L1 Cache	306
Access to Cached Objects	306
Multiple Agent Write and Replication Step	306
Inactive Agents and Fault Tolerance Behavior	307
Inference Agent Runtime Architecture	308
Single Agent Flow	308
Configuring Inference Agents—TIBCO Designer	309
BusinessEvents Archive Resource Object Management Tab—Cache Settings Reference	311
Configuring Inference Agents—Engine Properties	313
Defining an Agent Group	313
Defining a Unique Key for Each Agent	313
Local Storage Generally Not Enabled on Inference Agents	313
Configuring Fault Tolerance and Load Balancing for an Inference Agent Group	314
Tuning Agent Performance	314
Inference Agent Engine Property Reference	315
Example Inference Agent Configuration Properties	318

Chapter 22 Configuring Cache Servers (Cache OM)	319
Cache Server Configuration Overview	320
Deploying a Node as a Dedicated Cache Server	320
Memory and Heap Size Guideline for Cache Servers	320
Configuring a Cache Server	321
Configuring a Dedicated Cache Server Node	321
Configuring an Agent Node to Also be a Cache Server	321
Engine Property Reference for Cache Server Settings	322
Adding a Cache Server to a Running Production System	323
Chapter 23 Configuring Query Agents (Cache OM)	325
Configuring Query Agents—TIBCO Designer	326
Configuring Query Agents—Engine Properties for Performance	327
Chapter 24 Setting up a Backing Store Database	329
Backing Store Database Setup Overview	330
Backing Store Requirements	330
Before You Begin Database Setup	330
Backing Store Database Setup Tasks	330
Extra Procedure to Handle Long Identifier Names	331
After You Finish Database Setup	331
Maintaining a Backing Store—If Ontology Object Definitions Change	331
Resources Required for Setting Up the Database	332
Backing Store Database Configuration Tasks	334
Updating an Existing Backing Store Database Schema	338
Chapter 25 Project Configuration for Backing Store	341
Project Configuration for Backing Store—Overview	342
Backing Store Configuration Builds on Cache Configuration	342
Backing Store Configuration Summary	342
Backing Store Runtime Behavior	342
Additional Configuration Options	343
Summary List of Backing Store Related Properties	344
Adding a JDBC Connection Resource	345
Setting JDBC Connection and Pool Properties	347
Specifying the JDBC Connection	347
Setting Connection Pool Size	347
Enabling Backing Store and Setting Cache Characteristics	348
Enabling Backing Store Functionality	348
Specifying Limited Cache Size	348

Configuring How Backing Store Data is Loaded at Startup	349
Preloading Options for Cache Only Objects	350
Engine TRA File Reference for Backing Store Settings	351
Chapter 26 Deploytime Configuration	353
Deploytime Configuration Overview	354
Two Approaches to Configuring and Managing BusinessEvents Applications	354
Understanding Methods of Configuring Engine Properties	356
Setting Engine Properties for Deployment Outside of a Domain	356
Setting Engine Properties for Deployment to a Domain	356
Using AppManage for Scripted Deployment to a Domain	357
Defining and Using Global Variables	358
Configuring a BAR File for Deployment	360
BusinessEvents Archive Resource Reference	362
Configuration	362
Rule Sets	363
Input Destinations	364
Startup and Shutdown	366
Object Management	366
Shared Queue and Threads Properties	367
Configuring to Run Outside of a TIBCO Administrator Domain	368
Configuring to Run in a TIBCO Administrator Domain	370
Deploying Many Agents on One Machine With TIBCO Administrator	373
Customizing the List of Properties on the Advanced Tab	374
Chapter 27 Configuring Logging Settings	375
Configuring BusinessEvents Logging Settings	376
Chapter 28 Deploying a TIBCO BusinessEvents Project	379
Deploying a Project Outside a TIBCO Administrator Domain	380
Command-line Engine Startup Options	381
Deploying a Project in a TIBCO Administrator Domain	383
TIBCO Hawk Application Management Interface	387
Determining Various Names and Locations	388
Determining the Engine (Node) Name	388
Determining the TIBCO Repo URL for BusinessEvents	388
Location of TIBCO Administrator-Generated Property File	389
Default Location of Log Files and Other Files and Directories	389

Chapter 29 Configuring and Using Hot Deployment	391
Hot Deployment Overview	392
Modifications Allowed in Hot Deployment.	392
Hot Deployment in a Fault Tolerance Group (In Memory OM)	392
How Hot Deployment Occurs	392
Hot Deployment Supported Modifications	393
Enabling and Disabling Hot Deployment	395
Performing Hot Deployment in a TIBCO Administrator Domain	397
Modify the Project as Needed and Build the EAR File	397
Enable Hot Deployment (As Needed)	397
Perform Hot Deployment	397
Performing Hot Deployment Outside a TIBCO Administrator Domain	399
Modify the Project as Needed and Build the EAR File	399
Enable Hot Deployment (As Needed)	399
Add a Property to the Engine Property File	399
Perform Hot Deployment	399
Appendix A Engine Startup and Shutdown Sequence	401
Startup Sequence	401
Shutdown Sequence	402
Appendix B Object Models	403
Event Model	404
Concept Model	405
Rule and Ruleset Model	406
State Model—Overview	407
State Model—State Detail	408
Appendix C Advanced Caching Topics	409
Provided Caching Schemes	410
Overriding and Extending the Operational Deployment Descriptor	411
Overriding Element Values in Engine Property Files	411
Defining Additional Elements and System Properties in Override Files	411
Specifying Operational Override File Locations	413
How the First Tier Override File Default Location is Specified	413
How to Specify a Different Location for the First Tier Override File	414
How a Second Tier Override File Default Location is Specified and Overridden	415
Understanding Entity Caches	416
Entity Cache Names Format	416
Caches for Ontology Objects	417

Caches for Internal Entities	417
Appendix D BusinessEvents Tools Overview	419
BusinessEvents Tools Overview	420
Ontology Graphs	420
Rule Analyzer Benefits	421
Rule Debugger Benefits	421
BusinessEvents Tools—Rule Analyzer User Interface Overview	423
Overview Panel	424
Properties Panel	424
Source Panel	425
Analyzer/Dependency Panel	425
BusinessEvents Tools Graph Elements	427
BusinessEvents Tools—Rule Debugger User Interface Overview	428
Rule Execution History Panel	429
Debug Context Window Panel	429
Debug Context Window Panel Tabs	430
Appendix E Working With Rule Analyzer	433
Running Rule Analyzer	434
Using Rule Analyzer	435
Using the Analyzer/Dependency Graph	435
Using the Properties and Source Panels	436
Using Find, Filter, Export, and Printing Features	436
Rule Analyzer Filtering and Viewing Options	437
Filtering Entities in a Graph	437
Using Class Diagram and State Machine Viewing Options	438
Specify Filter Parameters Reference	439
Entity Chooser Dialog	440
Finding Entities in a BusinessEvents Tools Graph	441
Using the Entity Chooser	442
Exporting a Graph as an Image	445
Printing a Graph	447
Appendix F Working With Rule Debugger	449
Rule Debugger Setup Overview	450
Creating and Managing Profiles	450
Running Rule Debugger Against a Remote Engine	450
Providing Data to the Application	450
Running Rule Debugger	451
Configuring Rule Debugger Profiles	454

Rule Debugger Profiles Settings	456
Configuring a BusinessEvents Engine for Remote Debugging	460
Appendix G BusinessEvents Tools Reference	461
BusinessEvents Tools Toolbar Reference	462
Main Toolbar Options	462
Lower Toolbar Options	464
Properties Panel Toolbar Options	465
Rule Execution History Panel Toolbar Options	465
Debug Context Window Panel Toolbar Options	466
BusinessEvents Tools Menu Reference	467
File Menu Options	468
Edit Menu Options	469
View Menu Options	470
Debug Menu Options	471
BusinessEvents Tools Preferences Reference	472
Interactive Preferences	472
Layout Preferences	473
Look and Feel Preferences	475
BusinessEvents Tools Layout Options Reference	476
Incremental Layout	476
Hierarchical Layout	476
Orthogonal Layout	477
Circular Layout	477
Symmetric Layout	477
Appendix H Deprecated and Unused Properties	479
Appendix I TIBCO Hawk Microagent Methods	481
TIBCO Hawk Methods Overview	483
Types of Methods	483
Enabling TIBCO Hawk Microagent	483
For More Information	483
activateRuleSet()	484
activateTraceRole()	485
deactivateRuleSet()	486
deactivateTraceRole()	487
execute()	488
forceOMCheckpoint()	489
getChannels()	490

getDestinations()	491
getEvent()	492
GetExecInfo()	493
getHostInformation()	494
getInstance()	495
getMemoryUsage()	496
getNumberOfEvents()	497
getNumberOfInstances()	498
getOMInfo()	499
getRuleSet()	500
getRuleSets()	501
getScorecard()	502
getScorecards()	503
getSessionInputDestinations()	504
getSessions()	505
getStatus()	506
getTotalNumberRulesFired()	507
getTraceSinks()	508
reconnectChannels()	509
resetTotalNumberRulesFired()	510
resumeChannels()	511
resumeDestinations()	512
stopApplicationInstance()	513
suspendChannels()	514
suspendDestinations()	515
Index of Engine Properties	517
Glossary	519
Index	527

Figures

Figure 1	TIBCO BusinessEvents Architecture	18
Figure 2	Channels and Destinations	24
Figure 3	Serializer and Deserializer Behavior	25
Figure 4	History Ring Buffer	81
Figure 5	History Policy	82
Figure 6	Three Types of Concept Relationships	96
Figure 7	RDBMS functions catalog	116
Figure 8	Temporal Functions Parameters	161
Figure 9	Simple State Model	164
Figure 10	Modeling an Order Process	165
Figure 11	Explicit Calls	169
Figure 12	BusinessEvents State Modeler Palette	179
Figure 13	Out-of-Process Integration with ActiveMatrix BusinessWorks	182
Figure 14	Three Main Options for Object Management	234
Figure 15	Cache Object Management and Fault Tolerance Architecture	269

Tables

Table 1	General Typographical Conventions	xxix
Table 2	Syntax Typographical Conventions	xxx
Table 3	Which Serializers to Use for JMS Message Types	27
Table 4	BusinessEvents Engine Properties for Reconnecting to a JMS Server	32
Table 5	Variables for Use with DurableSubscriberName	43
Table 6	JMS Message Acknowledgement Modes	44
Table 7	JMS Header Field Names	49
Table 8	Simple Event Payload Element Parameters	59
Table 9	Containment and Reference Concept Relationship Rules	87
Table 10	Properties for Null Property Handling	105
Table 11	Database Concept —Extended Properties	113
Table 12	Database Concept Property—Extended Properties of Type Primitive	114
Table 13	Database Concept Property—Extended Properties of Type Concept Relationship	114
Table 14	Common Arguments for VRF Functions	159
Table 15	Design Considerations Related to Container (Integration with ActiveMatrix BusinessWorks)	196
Table 16	ActiveMatrix BusinessWorks integration Properties for BusinessEvents Containers	201
Table 17	Profiler Configuration Properties	223
Table 18	Profiler Column Heading Reference	228
Table 19	In Memory OM and Fault Tolerance Scenarios	239
Table 20	Persistence and Fault Tolerance Scenarios	240
Table 21	Cache and Fault Tolerance Scenarios	241
Table 22	BusinessEvents Engine Properties for In Memory OM Fault Tolerance	248
Table 23	BusinessEvents Engine Properties for Configuring Persistence Object Management	260
Table 24	Cache Cluster Network Settings for Multicast Protocol	291
Table 25	Cache Cluster Network Settings for Well-Known Address Protocol	293
Table 26	Cache Cluster Network Settings for Multiple Network Cards	294
Table 27	Cache Cluster Properties	297
Table 28	Cache Cluster Logging Properties	301

Table 29	Inference Agent Engine Properties	315
Table 30	BusinessEvents Engine Properties for Configuring a Cache Server.	322
Table 31	BusinessEvents Engine Properties for Query Agent Local Cache	327
Table 32	Resources Required for Backing Store Implementation	332
Table 33	BusinessEvents Engine Properties for Configuring a Backing Store	351
Table 34	Shared Queue and Threads Properties	367
Table 35	BusinessEvents Engine Properties to Determine Logging Behavior.	376
Table 36	Command-line Engine Startup Options	381
Table 37	Hot Deployment Supported Modifications	393
Table 38	Provided Caching Scheme Names	410
Table 39	Internal Entity Caches	417
Table 40	Explanation of Graph Elements in the Analyzer/Dependency Panel.	427
Table 41	BusinessEvents Tools—Main Toolbar Options	462
Table 42	BusinessEvents Tools—Bottom Toolbar Options	464
Table 43	BusinessEvents Tools Properties Panel Toolbar Options.	465
Table 44	Rule Debugger Rule Execution History Panel Toolbar Options	465
Table 45	Debug Context Panel Toolbar Buttons	466
Table 46	BusinessEvents Tools File Menu Options.	468
Table 47	BusinessEvents Tools Edit Menu Options.	469
Table 48	BusinessEvents Tools View Menu Options.	470
Table 49	BusinessEvents Tools Debug Menu Options	471
Table 50	BusinessEvents Tools Interactive Preferences	472
Table 51	BusinessEvents Tools Layout Preferences.	473
Table 52	Deprecated and Unused Engine Properties	479

BusinessEvents Palette Resources

Channel Resource Reference	34
Destination Resource Reference	37
SimpleEvent Resource Reference	56
TimeEvent Resource Reference	66
Concept Resource Reference	92
Concept View Resource Reference	98
ScoreCard Resource Reference	125
RuleFunction Resource Reference	148
Ruleset Resource Reference	151
Rule Resource Reference	152
State Machine Resource Reference	166
CallStateMachine Resource Reference	168
State Machine States Reference	170
Receive Event Resource Reference	185
Send Event Resource Reference	187
Wait for Event Resource Reference	188
RuleServiceProvider Configuration Resource Reference	207
Invoke RuleFunction Resource Reference	210
BusinessEvents Archive Resource Object Management Tab—Persistence Settings Reference	255
BusinessEvents Archive Resource Object Management Tab—Cache Settings Reference	311
BusinessEvents Archive Resource Reference	362

Preface

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

BusinessEvents
The Power to Predict™

Topics

- *Related Documentation, page xxvii*
- *Typographical Conventions, page xxix*
- *How to Contact TIBCO Support, page xxxii*

Enterprise Suite and Inference Edition Features

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

Inference Edition and Enterprise Suite

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

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

Enterprise Suite Only

All of the above components plus the following:

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



The Decision Manager application is available only on Windows.

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

Related Documentation

This section lists documentation resources you may find useful.

TIBCO BusinessEvents Documentation

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

Other TIBCO Product Documentation

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

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

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
<i>TIBCO_HOME</i> <i>BE_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>Other TIBCO products are installed into an installation environment. Incompatible products and multiple instances of the same product are installed into different installation environments. The directory into which such products are installed is referenced in documentation as <i>ENV_HOME</i>. The value of <i>ENV_HOME</i> depends on the operating system. For example, on Windows systems the default value is C:\tibco.</p> <p>TIBCO BusinessEvents installs into a version-specific directory within <i>TIBCO_HOME</i>. This directory is referenced in documentation as <i>BE_HOME</i>. The value of <i>BE_HOME</i> depends on the operating system. For example on Windows systems, the default value is C:\tibco\be\3.0.</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]

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
<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 BusinessWorks Concepts</i>. To introduce new terms. For example: A portal page may contain several <i>portlets</i>. Portlets are mini-applications that run in a portal. To indicate a variable in a command or code syntax that you must replace. For example: <code>MyCommand <i>pathname</i></code>
Key combinations	<p>Key name separated by a plus sign indicate keys pressed simultaneously. For example: <code>Ctrl+C</code>.</p> <p>Key names separated by a comma and space indicate keys pressed one after the other. For example: <code>Esc, Ctrl+Q</code>.</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.

Table 2 Syntax Typographical Conventions

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

Table 2 *Syntax Typographical Conventions*

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

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 provides an overview of TIBCO BusinessEvents, including a brief introduction to complex event processing, a description of the major components, architecture, and example uses of BusinessEvents, and a task summary.

This guide explains how to work with individual elements in a TIBCO BusinessEvents project—channels, events, concepts, rules, and so on. To gain a hands-on introduction to the way all these elements work together, first complete the tutorials in *TIBCO BusinessEvents Getting Started*. Doing so will give you a good context of understanding for the in-depth information in this manual.

Topics

- [What's Different About Complex Event Processing, page 2](#)
- [BusinessEvents Major Components, page 6](#)
- [Design Concepts, page 9](#)
- [Deployment Concepts, page 15](#)
- [Runtime Architecture and Flow, page 18](#)
- [Summary of Tasks, page 20](#)

What's Different About Complex Event Processing

Complex Event Processing (CEP) is a set of technologies that allows "events" to be processed on a continuous basis.

Most conventional event processing software is used either for Business Process Management (BPM), TIBCO iProcess for example, or for Service Oriented Architecture (SOA), for example TIBCO ActiveMatrix BusinessWorks software.

CEP is unlike conventional event processing technologies, however, in that it treats all events as potentially significant and records them asynchronously. Applications that are appropriate for CEP are *event-driven*, which implies some aspect of real-time behavior. To be more specific, the typical CEP application area can be identified as having some aspect of "situation awareness," "sense and respond," or "track and trace," aspects which overlap in actual business situations.

Situation awareness is about "knowing" the state of the product, person, document, or entity of interest at any point in time. Achieving this knowledge requires continuous monitoring of events to do with the entity, events that indicate what situation or state the entity is in, or about to be in. As an example, a dashboard indicates all performance indicators for a runtime production process. All the production plant events are monitored and the situation, or health, of the production process is determined via some performance indicators that are shown in real-time to one or more operators.

Sense and respond is about detecting some significant fact about the product, person, document or entity of interest, and responding accordingly. To achieve this result the software does the following:

- Monitors events that indicate what is happening to this entity.
- Detects when something significant occurs.
- Executes the required response.

As an example, you may monitor cell phone or credit card usage, detect that a cell phone or credit card is being used consecutively at locations that are too far apart for real-time person-to-business transactions. Detection of such transactions indicates that an act of fraud is in progress. The system responds accordingly, denying the transactions, and invoking the necessary workflow to handle the situation as defined in standard procedures.

Track and trace is about tracking the product, person, document or entity of interest over time and tracing pertinent facts like location, owner, or general status. An example would be tracking events from an RFID-enabled inventory control system where at any point in time you need to know how many widgets are in what location.

"Situation awareness," "sense and respond," and "track and trace" can all be classified as types of *activity monitoring*, for which the continuous evaluation of incoming events is suitable. For this reason, CEP is often described as a generalization of Business Activity Monitoring (BAM), although the event processing task may be only indirectly related to business, as in the case of an engine monitoring application or process routing task.

Technical Requirements of a CEP System

CEP systems must be able to receive and record events and identify patterns of these events and any related data. CEP systems must also handle temporal or time-based constraints, especially for handling the non-occurrence of events. The following TIBCO BusinessEvents features satisfy these requirements:

- A rich event model, incorporating event channels (for different event mechanisms) and destinations (for different types of events).
- A pattern detection mechanism using a sophisticated, high performance, declarative rule engine.
- A state model mechanism that allows entities to be described in terms of state, and in particular allows modelling of time-out events to handle the non-occurrence of events. (State modeling is available in the Enterprise Suite version only.)

A Model-Driven Approach

TIBCO BusinessEvents can be described not only as a CEP engine but also as an event-driven rule engine or real-time rule engine. TIBCO BusinessEvents enables CEP problems to be solved through a model-driven approach, in which the developer defines the event, rule, concept (class) and state models which are then compiled so that at run-time incoming events are processed as efficiently as possible. The various models are as follows:

Event model The event model describes the inputs into BusinessEvents. Events provide information through their properties and (optionally) through an XML payload. The event model provides the primary interface between BusinessEvents and the outside world, for input as well as output. Typical event sources (or channels) are messages from TIBCO Rendezvous and TIBCO Enterprise Message Service middleware, events generated explicitly by BusinessEvents, and custom mechanisms for non-standard event sources. Events can be used to trigger rules. See [Event Model on page 404](#), for a UML diagram of the event model.

Concept model The concept model describes the data concepts used in BusinessEvents, which may be mapped from events or their payloads, or loaded by some other mechanism into BusinessEvents. The concept model is based on standard UML Class and Class Diagram principles. See [Concept Model on page 405](#), for a UML diagram of the concept model.

Rule and ruleset model Rules, arranged into rule sets, provide one of the main behavioral mechanisms in BusinessEvents. Rules are defined in terms of *declarations* (events and concepts of interest), *conditions* (filters and joins on and between the attributes of the events and concepts), and *actions*. The underlying rule engine is based on an algorithm called the Rete algorithm, which mixes all rules together into a type of look-up tree, so that any additional concept instance or event can near-instantly cause the appropriate rule or rules to fire and invoke the appropriate actions. Rules are almost always defined in general terms (concepts or classes and events), so they apply to however many combinations of those events and classes exist in memory at any one time. The combination of rule declaration and condition defines the *event pattern* required for CEP operation. Rule actions that update other concepts may cause other rules to become available for firing, a process called *inferencing* or *forward chaining*. These types of rules are generally called *Production Rules*. The appropriate UML Production Rule Representation is still under development. See [Rule and Ruleset Model on page 406](#), for a UML diagram of the rule and ruleset model.

Rule functions: Algorithms, procedures or functions may be usefully defined as parameterized rule functions and re-used as required in rule actions and other areas where a behavior can be specified.

State model: an important item of metadata for a concept or object is its *state* according to some *state machine* or *state model* representation. Typically a state model describes the *states* that an entity can hold, and the *transitions* between states that are allowed, and the *conditions* for such transitions. Internally the state model is just additional metadata, but it is more useful to describe the state model as a visual model of states and transitions. The state transition rules can be viewed as special customizations of standard rules. The state model is based on the standard UML State Model principles. See [State Model—Overview on page 407](#) and [State Model—State Detail on page 408](#), for UML diagrams of the state model.

Stateful Rule Engine

At run-time, the rule engine executes rules based on new events and data sources on a continuous basis. The rule memory is never "reset" (unless by design), so that future events can always be compared to past events. For this reason, the rule engine is described as a *stateful rule engine*. If required, the working memory can be cleared and a new set of data asserted for each "transaction," in which case the engine is operating as a *stateless rule engine*.

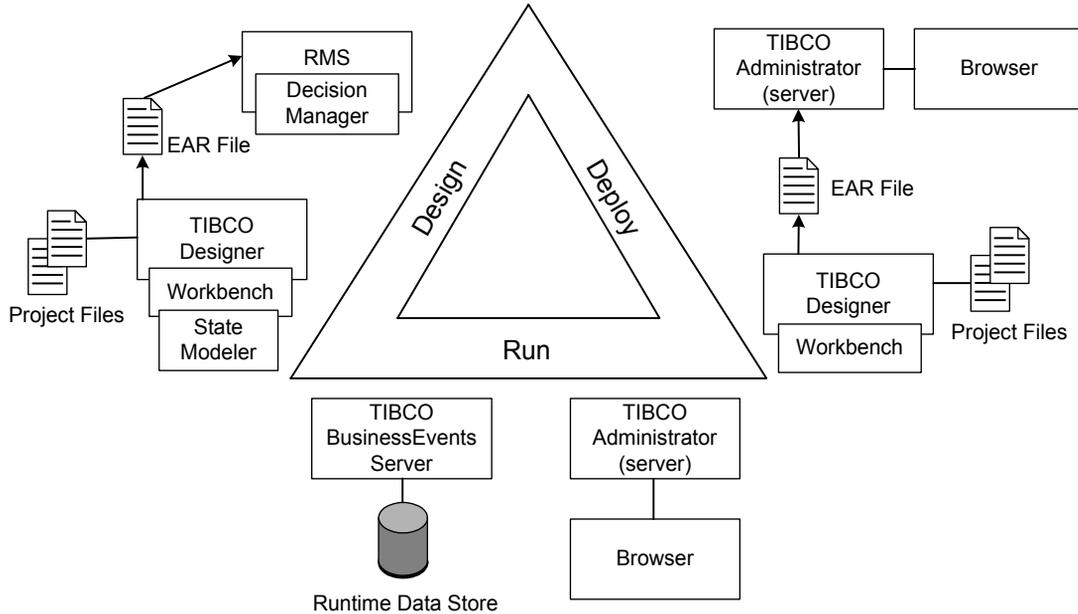
Object Management Options

To ensure resilience, BusinessEvents provides two persistence mechanisms for the events and data loaded into the system. One is called Persistence. It is a checkpoint mechanism that causes the data in the working memory to be saved to a lightweight database engine, to be restored when and as required. The other is a high performance distributed cache that allows data to be persisted and removed from memory or returned to memory, as required to handle extremely large problem domains (that would not typically fit into a runtime memory model).

These characteristics provide BusinessEvents with its enterprise and *extreme transaction processing* capabilities. Note that no rule operations are stored in the databases: this is because it is more efficient to simply rerun the rules and recreate the appropriate actions, than it is to persist the internal workings of the rule engine.

BusinessEvents Major Components

This section presents the major components of TIBCO BusinessEvents and how they are used.



TIBCO Designer—Designtime Activities

Many TIBCO products use TIBCO Designer as their design-time interface. TIBCO Designer hosts palettes for many different purposes. It also organizes project resources and makes the organization visible in graphical and folder-based ways. At design time, BusinessEvents makes use of some resources available in TIBCO ActiveMatrix BusinessWorks, such as the XML mapper. (BusinessEvents provides a development license for ActiveMatrix BusinessWorks. For runtime use, a full license is required.)

BusinessEvents Workbench Palette



The BusinessEvents Workbench palette integrates with TIBCO Designer and is used at design time. Design time activities performed using the resources in this palette include building an ontology—a set of concepts, scorecards and events that represent the objects and activities in your business—and building rules that are triggered when instances of ontology objects that fulfill certain criteria arrive in events. The output of the design-time activities is an enterprise archive (EAR) file, ready to deploy (or configure for deployment as needed).

If you are familiar with TIBCO Designer, you will know how to work with resources. If not, see tutorials in *TIBCO BusinessEvents Getting Started* to learn more.

In addition, BusinessEvents provides a palette of activities for use with ActiveMatrix BusinessWorks integrations (see [Connecting to TIBCO ActiveMatrix BusinessWorks \(Optional\)](#) below).

State Modeler

The Enterprise Suite of TIBCO BusinessEvents includes a state machine component. It allows you to model states of ontology concept instances and use those states in rules. See [Chapter 10, Working With The State Modeler, on page 163](#).

Connecting to TIBCO ActiveMatrix BusinessWorks (Optional)

TIBCO BusinessEvents communicates with TIBCO ActiveMatrix BusinessWorks through a palette of ActiveMatrix BusinessWorks Activities. Two methods of integration with ActiveMatrix BusinessWorks are available: in process and out of process.

See [Chapter 11, Out-of-Process ActiveMatrix BusinessWorks Integration, on page 181](#) and [Chapter 12, In-Process ActiveMatrix BusinessWorks Integration, on page 191](#).

TIBCO Administrator—Deploy and Runtime Activities

TIBCO Administrator provides administrative access to BusinessEvents servers that are deployed to a TIBCO Administrator administration domain. You can use TIBCO Administrator to perform deploytime configuration, deploy EAR files, and manage the deployed servers.

You can also perform many of these tasks at the command line, for deployments that are not in a TIBCO Administrator domain.

See [Chapter 26, Deploytime Configuration, on page 353](#) and [Chapter 28, Deploying a TIBCO BusinessEvents Project, on page 379](#).

BusinessEvents Runtime Activities

At runtime, one or more nodes running one or more BusinessEvents *inference agents* process the incoming events using a Rete network as the inferencing engine, and a set of rules that are triggered by conditions in incoming events.

See [Runtime Architecture and Flow on page 18](#) for more details.

When Cache object management is used, inference agents, query agents and cache server nodes co-operate to provide efficient processing. Load balancing, and fault tolerance of data and engine processes is available. See [Chapter 14, Understanding Object Management and Fault Tolerance, on page 233](#) for an introduction to these topics.

Decision Manager and Rules Management Server

Decision Manager, and its rules management server enable business users to build rules using decision tables and deploy them to a production system.

See *TIBCO BusinessEvents Decision Manager* for more details.

Design Concepts

In a rule engine, the things that the project works with such as employees, inventory, parts, and so on are *concepts* in the *ontology* (knowledge model) of the project. *Events* such as flight take-off, purchase of a mortgage, sale of stock, and so on are also part of the ontology, as are scorecards, which hold metrics. Rules are triggered by events and by changes in objects. For example, rules might cause a flight to be delayed if there is a certain problem at the airport.

Designing the ontology and the rules well is key to a good CEP (complex event processing) project.

The sections below describe the features mentioned above in greater detail.

Channels and Destinations

JMS and Rendezvous channels represent physical connections to a resource, such as a Rendezvous daemon. Destinations in a channel represent listeners to messages from that resource, and they can also send messages to the resource. Arriving messages are transformed into simple events, and simple events sent out of BusinessEvents are transformed to the appropriate type of message (JMS or Rendezvous).

TIBCO BusinessEvents also includes *Local* channels, which allow you to pass events between two *rule sessions* in the same project (Not required with multi-engine functionality).

See [Chapter 2, Working With Channels and Destinations, on page 23](#)

Events

BusinessEvents processes three kinds of events:

- **Simple Event** A representation of a single activity (usually a business activity) that occurred at a single point in time.
- **Time Event** A timer.
- **Advisory Event** A notice generated by BusinessEvents to report an activity in the engine, for example, an exception.

BusinessEvents creates instances of simple events and time events based on user-configured event definitions. The following sections provide more detail on each type of event.

Simple Events

A *simple event definition* is a set of properties related to a given activity. It includes information for evaluation by rules, meta-data that provides context, and a separate payload -- a set of data relevant to the activity. For example, say you are interested in monitoring the creation of new employee records; you might create a simple event definition that includes important fields from the employee record, perhaps the social security number, department, and salary. You could then write a rule to create an instance of this simple event each time a new employee record is created.

A *simple event* is an instance of a simple event definition. It is a record of a single activity that occurred at a single point in time.

Just as one cannot change the fact that a given activity occurred, when an event is asserted, those values are used by working memory and any changes made after that time are ignored. (Before assertion you can use an event preprocessor to enrich the event, however.) Simple events expire when their time to live has elapsed, unless BusinessEvents has instructions to consume them prior to that time.

Example 1: A temperature sensor records a reading that is above a predefined limit. The payload might include the temperature-sensor ID, the reading, and the date and time. This simple event might trigger a complex event that would immediately notify a manager.

Example 2: A customer purchases four airline tickets from San Francisco, California to San Jose, Costa Rica. The payload might include the date and time of purchase, the date and time of the flight, the purchase price, the credit card number, the flight number, the names of the four passengers, and the seat assignments. This simple event alone may include no exceptions. However, it is possible that when examined within the context of other related events, an exception may arise. For example, one or more of the passengers may have booked tickets on another flight during the same time period.

See [Chapter 3, Working With Simple Events, on page 51](#).

Time Events

A time event is an event definition that triggers the creation of event instances based on time. There are two ways to configure a time event:

- **Rule based** A rule schedules the creation of a time-event instance at a given time.
- **Time-interval based (Repeat Every)** BusinessEvents creates a time-event instance at regular intervals.

See [Chapter 4, Working With Time Events, on page 63](#).

Advisory Events

The BusinessEvents engine automatically asserts advisory events when it catches an exception that originates in user code but that is not caught with the `catch` command of the BusinessEvents Exception type (available in the rule language).

See [Chapter 5, Working With Advisory Events, on page 71](#).

Concepts and Concept Instances

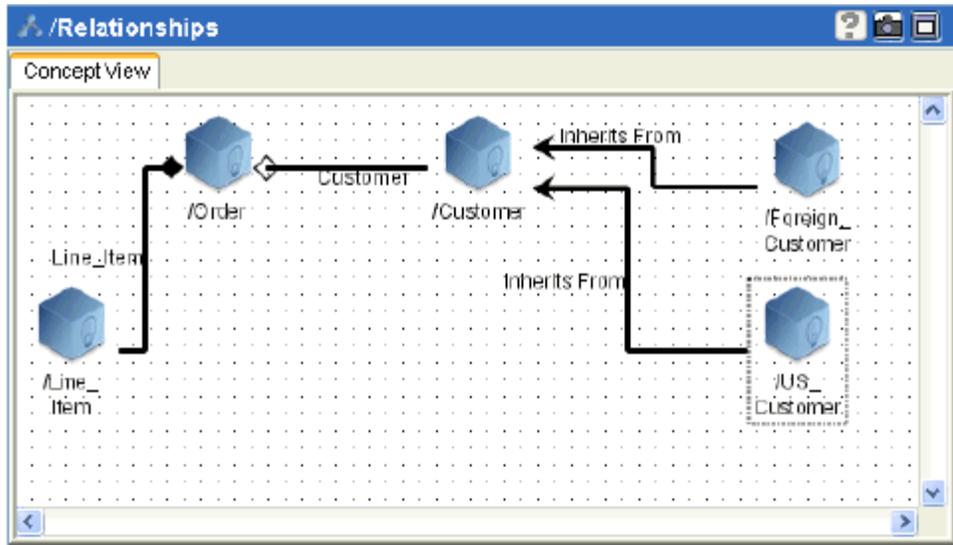
Concepts have these characteristics:

- Each concept is a definition of a set of properties that represent the data fields of an entity.
- Concepts can describe relationships among themselves. For example, an `order` concept might have a parent/child relationship with an `item` concept. A `department` concept might be related to a `purchase_requisition` concept based on the shared property, `department_id`. (See [Concept Views on page 12](#))
- Concepts can include a state model. (Enterprise Suite only. See [State Modeler on page 13](#).)
- Concepts can be created by importing table and view data from databases, and you can update the database definitions using RDBMS functions.

See [Chapter 6, Working With Concepts, on page 77](#) and [Chapter 7, Working With Database Concepts, on page 107](#).

Concept Views

A *concept view* is a visualization tool, providing an easy way to work with a group of concepts and their relationships. Here is an example of a concept view:



Within a concept view you can create concepts and define relationships between concepts.

See [Working With Concept Views on page 96](#).

Score Cards

A score card serves as a static variable that is available throughout the project. You can use a ScoreCard resource to track key performance indicators or any other information. Use rules to view its value, use its value, or change its value. Note that unlike concepts and event definitions, which describe types of instances, each scorecard is both the description and the instance.

See [Chapter 8, Working With Scorecards, on page 123](#).

Rules and Rule Sets

Rules define what constitutes unusual, suspicious, problematic, or advantageous activity within your enterprise applications. Rules also determine what BusinessEvents does when it discovers these types of activities. You can execute actions based on certain conditions on simple events, concept instances, time events, score cards, or a combination of these objects.

All rules exist inside of *rule sets*. Rule sets are groups of rules. At deployment time, you can select rule sets to use at runtime; you cannot select individual rules.

See [Chapter 9, Working With Rules and Functions, on page 127](#).

Functions

BusinessEvents offers four types of functions for use in rules:

- Standard — These functions are provided with BusinessEvents.
- Ontology — BusinessEvents generates these functions based on the resources in your project.
- Custom — You can write custom functions using Java and integrate them into BusinessEvents for use in rules.
- Rule Function — In addition to Java-based custom functions, you can use rule function resources to write functions using the BusinessEvents rule language.

Standard functions include a set of temporal functions, which allow you to perform calculations based on a sampling of a property's values over time. These functions make use of the history for that property. For more information, see [Temporal Functions on page 160](#).

See the following for more information:

[Chapter 9, Working With Rules and Functions, on page 127](#).

TIBCO BusinessEvents Functions Reference, and *TIBCO BusinessEvents Language Reference*, which lists the standard functions and explains how to add custom functions.

State Modeler

The State Modeler feature is available only in TIBCO BusinessEvents Enterprise Suite.

State Modeler is a UML-compliant application that allows you to model the life cycle of a concept instance — that is, for each instance of a given concept, you can define which states the instance will pass through and how it will transition from state to state.

States have entry actions, exit actions, and conditions, providing precise control over the behavior of BusinessEvents. Transitions between states also may have rules. Multiple types of states and transitions maximize the versatility and power of State Modeler.

See [Chapter 10, Working With The State Modeler, on page 163](#)

Object Management and Fault Tolerance

An important aspect of most BusinessEvents applications is management of the objects created and modified at runtime.

Different projects have different object management requirements. For some, it is acceptable to destroy the objects once the rule engine cycle that needs them has completed. They require only memory-based object management. For others, the instances have value beyond the rule engine cycle and need to be persisted.

Related to object management is configuration of fault tolerance features. BusinessEvents supports a variety of object management and fault tolerance options.

Cache Object Management and Multi-Engine Mode

Cache object management enables BusinessEvents to run in multi-engine mode. In this mode, load balancing ruleset chaining features are available at the agent level. This configuration provides a flexible way to load balance rule sets across multiple engines. It simplifies scalability, enabling you to run multiple inference agents on multiple nodes within the same cluster configuration.

To gain an understanding of the options, read [Chapter 14, Understanding Object Management and Fault Tolerance, on page 233](#). Chapters following explain the topics in more detail.

Query Language for Querying Cached Objects

Available in TIBCO BusinessEvents Enterprise Suite only, the query features enable you to perform snapshot queries and continuous queries using an SQL-like query language. You can then send query results to outbound destinations or use them in rules as desired. Details are provided in *TIBCO BusinessEvents Language Reference*.

Deployment Concepts

The BusinessEvents design-time project is deployed as a BusinessEvents application. This section discusses two methods of deployment, and describes the archives that contain the deployable files.

See [Chapter 26, Deploytime Configuration, on page 353](#) and [Chapter 28, Deploying a TIBCO BusinessEvents Project, on page 379](#) for more details on this topic.

Deployment Files

Each BusinessEvents engine runs one BusinessEvents application at a time, in one JVM (Java Virtual Machine). However, each application may contain multiple BusinessEvents archive files (BAR files), each of which deploys as a rule session within the overall rule server.

Enterprise Archive Resource (EAR)

The Enterprise Archive Resource (EAR) is a standard TIBCO Designer component that is used when deploying a BusinessEvents application. The EAR file contains one shared archive resource (SAR) file and one or more BusinessEvents archive resource (BAR) files. If the application integrates ActiveMatrix BusinessWorks functionality, it could also contain the process archive (PAR) file for the ActiveMatrix BusinessWorks processes.

Shared Archive Resource A shared archive is automatically included in the enterprise archive. Certain resources are added automatically to the shared archive, and others are manually added. See [Adding a JDBC Connection Resource on page 345](#) for an example use in BusinessEvents.

For more information about Enterprise Archive Resources and project deployment in general, see *TIBCO Designer User's Guide*.

BusinessEvents Archive Resource (BAR)

A BusinessEvents archive is the design-time equivalent of a BusinessEvents agent at runtime. Different kinds of agents play different roles in a large deployment: inference agents perform rule evaluation, query agents perform queries, and cache server agents are deployed as cache server nodes when the Cache object management option is used. You can include multiple rule sessions by including multiple BusinessEvents Archives within one Enterprise Archive resource.

A BusinessEvents Archive resource provides information that is used during the deployment stage. Use a BusinessEvents Archive resource to perform various tasks (depending on the type of agent you are configuring):

- Select one or more sets of rules deployment (see [Rule Sets, page 128](#))
- Enable or disable individual input listeners (see [Understanding Channels and Destinations, page 24](#))
- Specify event preprocessors for destinations, and specify thread settings to handle preprocessing more efficiently (see [Working With Event Preprocessors, page 145](#))
- Specify startup and shutdown actions (see [Working With Startup and Shutdown Rule Functions, page 142](#))
- Configure the object manager for various object management options, which affect the availability, performance and memory usage of the deployed application (see [Chapter 17, Understanding Cache OM and Multi-Engine Features, page 263](#) and chapters following)

For more information about configuring the BusinessEvents Archive resource see [BusinessEvents Archive Resource Reference on page 362](#).

Deploytime Configuration and Deployment Methods

How you configure for deployment depends in part on whether you are deploying to a TIBCO Administrator domain or not.

You can run applications at the command-line without deploying to a TIBCO Administrator domain. In this case, you set configuration parameters in the engine property file (`be-engine.tra`) This approach is useful for testing purposes, but is not recommended for production purposes.

For production purposes, it is recommended that you deploy projects to a TIBCO Administrator domain. TIBCO Administrator provides a user interface for setting deploytime parameters, as well as management and monitoring features. It also provides a command-line utility called AppManage (documented in TIBCO Administrator documentation). You can also set configuration parameters in the engine property file.

See [Configuring to Run Outside of a TIBCO Administrator Domain on page 368](#) and [Configuring to Run in a TIBCO Administrator Domain on page 370](#) for more details.

Hot Deployment

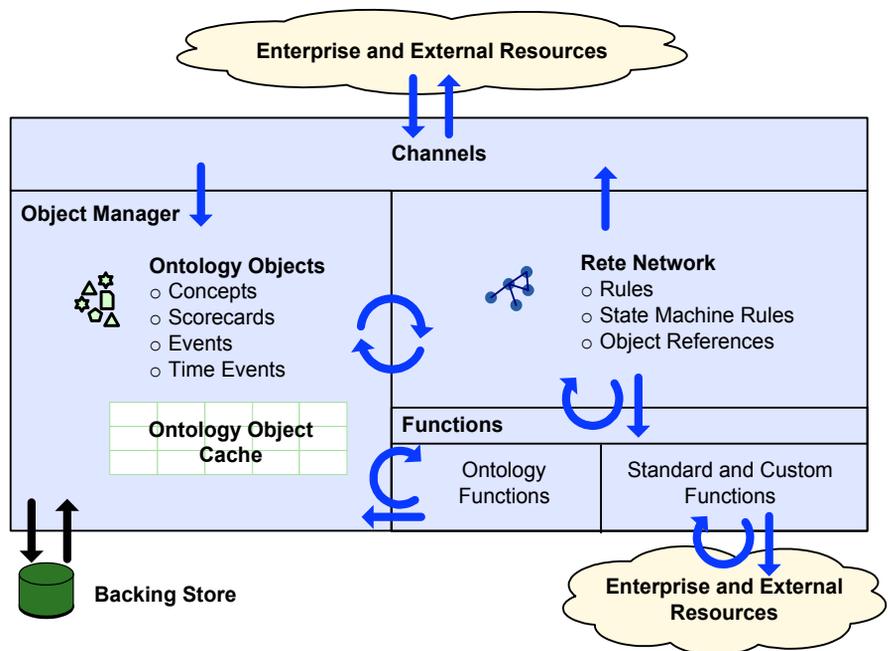
Depending on the changes made to your BusinessEvents project, you may be able to replace an EAR file for a BusinessEvents project with an updated one, without stopping the BusinessEvents engine. This feature is referred to as hot deployment. For more information about the BusinessEvents hot deployment feature, including the project changes that are supported, see [Chapter 29, Configuring and Using Hot Deployment, on page 391](#).

Runtime Architecture and Flow

BusinessEvents has three layers of functionality:

- **Rules Evaluation and Execution** based on the state of objects, events and temporal aspects. This functionality is achieved by configuring one or more inference agents with the appropriate rule sets. Each inference agent maintains a Rete network to remember past matches.
- **Lifecycle Management of Objects and Events** including distribution, clustering, persistence and recoverability. Various options are available to achieve the levels of functionality appropriate for business needs, from in-memory only storage of objects, to advanced caching features and a backing store (database).
- **Querying the Cache** A query agent (not shown below) is an optional component available in TIBCO BusinessEvents Enterprise Suite. It enables visibility into cache data and enables cache data to be returned and used in BusinessEvents or externally.

Figure 1 TIBCO BusinessEvents Architecture



Rule Evaluation and Execution

Information from enterprise applications and other sources flows into BusinessEvents through channels as messages. BusinessEvents transforms messages to events, based on event definitions (event types) and asserts them into working memory.

All the rules whose conditions match information in the events are assembled into a *rule agenda* and the first rule executes. Rule **actions** create and modify the facts, which are processed into a structure known as a *Rete network*, an in-memory network of objects based on the Rete algorithm which enables fast matching of facts with rule dependencies.

BusinessEvents uses a forward-chaining inferencing engine, based on the Rete algorithm. Every time the facts change—due to rule actions or the arrival of new information—the inferencing engine updates the rule agenda. The process goes on until there are no more changes to process. This is known as a run to completion cycle or RTC. See [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#) for more details

(Note that State Machine, is present only in TIBCO BusinessEvents Enterprise Suite).

Summary of Tasks

Having read the earlier sections in this overview, you should now have a general idea about when to use CEP, what's in a BusinessEvents project, and how it works at runtime. You will gain a deeper and more detailed understanding as you work with the product over time. This section concludes the overview with a brief summary of tasks with pointers to the relevant chapters in this guide that provide more detailed concepts, procedures, and reference materials.



It is strongly recommended that you complete the tutorials in *TIBCO BusinessEvents Getting Started* before beginning to work on your production projects. The tutorials give you a hands-on experience of building, deploying, and running a simple project.

Design Tasks

Design the Ontology

See the following chapters for background information and instructions about building your project's ontology:

[Chapter 2, Working With Channels and Destinations, page 23](#)

[Chapter 3, Working With Simple Events, page 51](#)

[Chapter 4, Working With Time Events, page 63](#)

[Chapter 5, Working With Advisory Events, page 71](#)

[Chapter 6, Working With Concepts, page 77](#)

[Chapter 7, Working With Database Concepts, page 107](#)

[Chapter 8, Working With Scorecards, page 123](#)

Build Rules

See [Chapter 9, Working With Rules and Functions, page 127](#), for details about building rules and rule functions.

Design State Machines (Enterprise Suite Only)

See chapter [Chapter 10, Working With The State Modeler, page 163](#) for details about using the state modeler to define transitions and states for concepts.

Integrate with TIBCO ActiveMatrix BusinessWorks (Optional)

If you have a production license for ActiveMatrix BusinessWorks, you can use the integration features to take advantage of ActiveMatrix BusinessWorks features. See the following chapters:

[Chapter 11, Out-of-Process ActiveMatrix BusinessWorks Integration, page 181](#)

[Chapter 12, In-Process ActiveMatrix BusinessWorks Integration, page 191](#)

Integrate with Decision Manager (Optional)

Virtual rule functions whose signatures are defined in TIBCO Designer are implemented in Decision Manager using decision tables, and the classes are then loaded into BusinessEvents engines at startup or are hot deployed to a running system. See *TIBCO BusinessEvents Decision Manager* for details.

Test the Design

As you design your project, you can analyze and test it. Three tools are provided for this purpose: Rule Analyzer, Rule Debugger and Performance Profiler. They are documented in the following chapters:

[Appendix D, BusinessEvents Tools Overview, page 419](#)

[Appendix E, Working With Rule Analyzer, page 433](#)

[Appendix F, Working With Rule Debugger, page 449](#)

[Appendix G, BusinessEvents Tools Reference, page 461](#)

[Chapter 13, BusinessEvents Performance Profiler, page 221](#)

Manage BusinessEvents Objects

Runtime activities can generate an enormous amount of object instance data. Object management refers to various ways that BusinessEvents can manage the data. Fault tolerance is a related topic. See the following chapters for details:

[Chapter 14, Understanding Object Management and Fault Tolerance, page 233](#) through [Chapter 25, Project Configuration for Backing Store, page 341](#)

Build Cache Queries (Enterprise Suite Only)

The query language features enable you to query cache data and use the results externally or in BusinessEvents rules. See *TIBCO BusinessEvents Language Reference* for details.

Deployment and Runtime Tasks

After your project design is implemented and tested, you will deploy and manage the runtime system. See the following chapters for details:

[Chapter 26, Deploytime Configuration, page 353](#)

[Chapter 27, Configuring Logging Settings, page 375](#)

[Chapter 28, Deploying a TIBCO BusinessEvents Project, page 379](#)

[Chapter 29, Configuring and Using Hot Deployment, page 391](#)

Chapter 2

Working With Channels and Destinations

This chapter focuses on understanding and configuring channels and destinations.

Topics

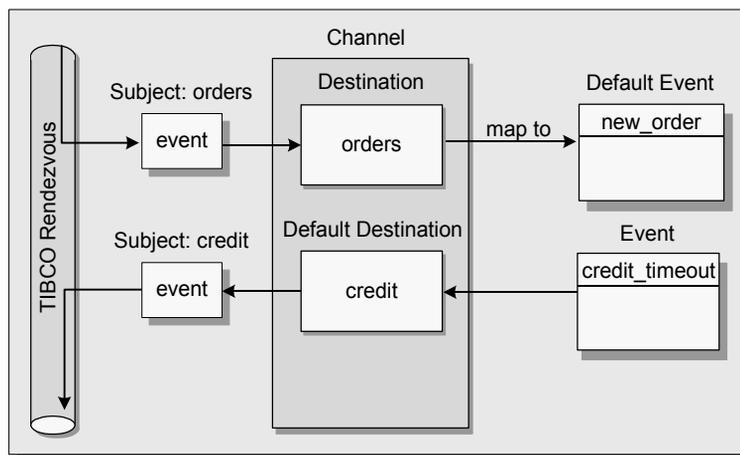
- [Understanding Channels and Destinations, page 24](#)
- [Selecting a JMS Serializer, page 27](#)
- [Selecting a Rendezvous Serializer, page 29](#)
- [Working With Channels and Destinations, page 31](#)
- [Channel Resource Reference, page 34](#)
- [Destination Resource Reference, page 37](#)
- [Mapping Incoming Messages to Non-default Events, page 42](#)
- [Creating Unique JMS DurableSubscriber Name Properties, page 43](#)
- [Changing the JMS Message Acknowledgment Mode, page 44](#)
- [Using JMS Header Properties in Incoming and Outgoing Messages, page 47](#)

Understanding Channels and Destinations

Channels (other than local channels) represent physical connections to a resource, such as a Rendezvous daemon. Destinations in a channel represent listeners to messages from that resource, and they can also send messages to the resource. All destinations for a particular channel use the same server.

For example, in [Figure 2](#) below, the channel is configured to listen to the flow of messages on Rendezvous. The *orders* destination directs BusinessEvents to map messages coming in on the subject, *orders*, to the *new_order* simple event.

Figure 2 Channels and Destinations



You can choose from the following types of channels:

- **TIBCO Rendezvous channels** connect TIBCO BusinessEvents to TIBCO Rendezvous sources and sinks.
- **JMS channels** connect TIBCO BusinessEvents to TIBCO Enterprise Message Service provider sources and sinks
- **Local channels** connect multiple rule sessions at runtime.

Local Channels

Local channels serve a different function from other kinds of channels. Other kinds of channels pass information into and out of the BusinessEvents application. Local channels are referenced in functions and are used to send events between rule sessions. Local channels are used in applications that have multiple rule sessions (multiple agents) in one node, and that use In Memory object management. They are not required or used with Cache object management.

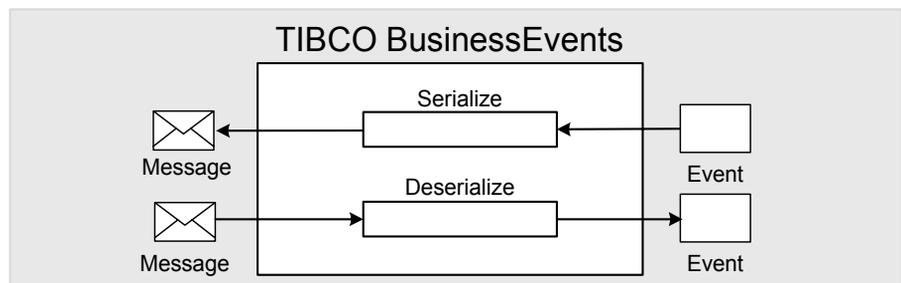
Local channels pass the same event object between the rule sessions (agents). Consuming the event in one rule session does not affect other sessions that also received the event over a local channel. A use count is maintained for each event to track how many sessions have received the event but not consumed it. The use count of the event is incremented depending on the number of sessions it is routed to. When an event is consumed in one agent, BusinessEvents deletes the reference to the event in that agent and it decrements the use count of the original event instance.

See [Creating a Local Destination on page 33](#) and [Using a Local Destination in a Rule on page 33](#) for details on using local channels.

Selecting a Serializer

BusinessEvents uses serializers to convert events to messages and a deserializer to convert incoming messages to events. Serializers and deserializers are provided for JMS and Rendezvous. (Local channels do not require serializers.)

Figure 3 Serializer and Deserializer Behavior



When you configure a destination, you select the appropriate serializer. For JMS messages, header properties are mapped to event properties, and message bodies are mapped to event payloads. Which serializer you choose depends on the type of message body and payload. Similarly, for Rendezvous destinations, you would choose the appropriate serializer, depending on the presence of a payload, and the type of payload if present.

For more details see [Selecting a JMS Serializer on page 27](#) and [Selecting a Rendezvous Serializer on page 29](#)

Deploy-time Configuration

The following channel-related configuration tasks are performed at deploy time.

- Configuring event preprocessors (rule functions) for a destination and defining worker thread options for each preprocessor (see [Working With Event Preprocessors on page 145](#) and [BusinessEvents Archive Resource Reference on page 362](#)).
- Enabling and Disabling Destination Listeners: You can enable and disable destination listeners when configuring a BusinessEvents archive for deployment. See notes for the Enable field, in the section [BusinessEvents Archive Resource Reference on page 362](#).

Message Acknowledgment

When messages come through channels, BusinessEvents creates events using the messages, and asserts the events into working memory. Depending on the message type, BusinessEvents acknowledges the receipt of these messages. Some messages do not require acknowledgement, for example reliable Rendezvous messages do not require acknowledgment.

Message Acknowledgment Timing for Each Object Management Type

The timing of the message acknowledgment is handled differently by each object management type (see [Chapter 14, Understanding Object Management and Fault Tolerance, on page 233](#) for details on each type):

Persistence All received messages are acknowledged after data is written to disk, during a checkpoint.

Cache Messages are acknowledged at the end of each conflict resolution cycle, also known as a run to completion (RTC) cycle.

In memory The message is acknowledged when the simple event is consumed, that is, when `consumeEvent(event)` is called, or when the event expires (at the end of the time to live period, if set).

Selecting a JMS Serializer

This section explains the purpose of the provided JMS serializer classes. Choose the serializer that handles the JMS message types that will be sent to the destination you are configuring.

Table 3 Which Serializers to Use for JMS Message Types

Message Type	BytesMessageSerializer	TextMessageSerializer
Message	✓	✓
BytesMessage	✓	
MapMessage	✓	✓
ObjectMessage	Not Supported	Not Supported
StreamMessage	Not Supported	Not Supported
TextMessage		✓

For `MapMessage` messages, you create properties whose names match the message keys.

Both serializers support reading and writing application header properties and JMS header properties. The difference between the serializers is in how they handle payloads. `BytesMessageSerializer` decodes the body as a sequence of bytes and parses them to create an XML structure according to the Payload definition in the event. With `TextMessageSerializer`, the text from the message is decoded as XML string.

See [Using JMS Header Properties in Incoming and Outgoing Messages on page 47](#).

BytesMessageSerializer

For incoming messages of type JMS `BytesMessage`, the serializer converts the message bodies to event payloads. The payloads are XML String type, but are not human-readable. However, you can access them using XPath functions.

For outgoing events, the serializer converts XML payloads to JMS `BytesMessage` message bodies.

The `BytesMessageSerializer` class is the default serializer.

TextMessageSerializer

For incoming messages, the `TextMessageSerializer` serializer converts JMS `TextMessage` messages to event payloads. The payloads are XML String type, and are human-readable. You can access them using XPath functions.

For outgoing events, the serializer converts XML payloads to JMS `TextMessage` messages.

Each JMS Input Destination Runs a Session

Every JMS destination that is configured to be an input destination runs in its own JMS Session. This provides good throughput on queues and topics for processing, and less connections.

Selecting a Rendezvous Serializer

This section explains the behavior of the provided Rendezvous serializer classes.

Note that the Rendezvous serializers use UTF8 encoding.

The type of the event from which the message is serialized is added to the Rendezvous message header using `_nm_` and `_ns_` fields so that if the message is used in BusinessEvents again, the correct event type is used to deserialize the message (ignoring the default event specified for the destination).

Rendezvous Message Header

For Rendezvous messages, the only header that BusinessEvents interprets is `_sendsubject_` which is of type String. It is a read-only property. The event has to define this property to receive the value. The value is the actual Subject on which the message was sent.

Basic Serializer

The basic serializer, `TibRvMsgSerializer` serializer, is for efficient handling of events and messages that do not have payloads. It ignores payloads in messages and in events if any exist. (See [Including a Payload in a Rendezvous Message on page 30](#).)

If you want pass payloads between the `_payload_` field of a Rendezvous message and an event payload, use the `TibRvMsgSerializerWithXMLPayload` serializer (see [Serializer For Events With Payloads on page 29](#)).

First level Rendezvous property values are used as values for matching event properties. Any additional (non-matching) Rendezvous properties are ignored.

Serializer For Events With Payloads

The `TibRvMsgSerializerWithXMLPayload` serializer allows you to move XML documents between Rendezvous message `_payload_` fields and event payloads.

If the `_payload_` field is of an unsupported type, or is missing, or if the event has not been configured for a payload, the payload is ignored.

Deserializing from Rendezvous Message to Event

- First level Rendezvous property values are used as values for matching event properties. Any additional (non-matching) Rendezvous properties are ignored.
- The `_payload_` field contents are passed into the event payload. `TIBRVMSG_STRING`, and `TIBRVMSG_XML` (wire format types) are supported.
- If the event defines a payload, but the incoming Rendezvous message does not have a `_payload_` field, BusinessEvents attempts to map the entire message as the payload.

Serializing from Event to Rendezvous message

- Event properties are transformed to first level Rendezvous message properties.
- The event payload is passed to the Rendezvous message `_payload_` field.

Including a Payload in a Rendezvous Message

To include a payload in a Rendezvous message, ensure that the message has this field:

`_payload_`

When BusinessEvents receives a Rendezvous message with this field, and when you use the `TibRvMsgSerializerWithXMLPayload` serializer, BusinessEvents deserializes the content of the field into the payload of the simple event.

The Rendezvous `_payload_` field can be of these types: String, byte array, or `TibrvXML`.

Working With Channels and Destinations

The procedure for creating channels and destinations is the same, though the configuration options are different.

Creating a Channel

1. Open the folder where you want to store the channel, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > Channel**.
2. In the Configuration tab, name the channel and give it a description.
3. In the Driver field, select the JMS, Rendezvous, or Local driver as appropriate.
4. For JMS and Rendezvous channels: From the Method of Configuration drop-down list, select one of the following:
 - **Resource** Select Resource if you have a shared resource in your project whose properties you want to reuse for this channel.
 - **Properties** Select Properties to configure this channel resource using properties. See [Channel Resource Reference on page 34](#) for details.



Resource names and directory names The path to the resource and the resource name cannot contain any of the keywords or other reserved words listed in the *TIBCO BusinessEvents Language Reference*, and they cannot use spaces.

5. Click **Apply** and save the project.

Creating a JMS or Rendezvous Destination

1. Open the channel in which you want to create the destination, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > Destination**.
2. In the Configuration tab, name the destination and give it a description.
3. In the Default Event field, browse to and select the event to be created (by default) from incoming messages received by this destination.
4. In the serializer field, select the appropriate serializer class. See [Selecting a JMS Serializer on page 27](#) and XREF for guidance.
5. Complete the rest of the properties for the type of destination you are creating. See [Destination Resource Reference on page 37](#) for details.
6. Click **Apply** and save the project.

Default Event

Reconnecting to a JMS Server

The following engine properties enable you to define how BusinessEvents attempts to reconnect to a JMS server in the event of a disconnection.

Table 4 BusinessEvents Engine Properties for Reconnecting to a JMS Server

Property	Notes
<code>com.tibco.tibjms.connect.attempts</code>	<p>Specifies the number of reconnection attempts, and the interval between each attempt to connect to the JMS server.</p> <p>The value must use the format: <i>attempts, retry interval</i>. For example: 10, 500 means 10 attempts, with a 500 millisecond interval between each retry attempt.</p> <p>This property is used only for channels that have a TIBCO Enterprise Message Service provider.</p> <p>Note: Use <i>either</i> <code>be.jms.reconnect.timeout</code> or <code>com.tibco.tibjms.connect.attempts</code>. If you set both the properties, then <code>com.tibco.tibjms.connect.attempts</code> takes precedence.</p> <p>Default value: 2, 500</p>
<code>be.jms.reconnect.timeout</code>	<p>Specifies the retry interval (in seconds) for reconnecting to the JMS server when the connection is broken.</p> <p>A value of 0 (zero) means don't retry. Any other value means keep retrying (with no limit to number of retries), and use the specified interval between each attempt.</p> <p>Note: Unacknowledged messages (Events) are resent to the BusinessEvents engine, which may result in duplicate events.</p> <p>Note: Use <i>either</i> <code>be.jms.reconnect.timeout</code> or <code>com.tibco.tibjms.connect.attempts</code>. If you set both the properties, then <code>com.tibco.tibjms.connect.attempts</code> takes precedence.</p> <p>Default value: 0 (zero)</p>

Table 4 BusinessEvents Engine Properties for Reconnecting to a JMS Server (Cont'd)

Property	Notes
be.jms.reconnect. msgCodes	<p>Specifies a case-insensitive character pattern that matches all error messages or error codes that will cause a reconnect attempt.</p> <p>This property is used for JMS channels with providers other than TIBCO Enterprise Message Service.</p> <p>Default value: * (that is, the wildcard matched by any characters.)</p>

Creating a Local Destination

1. Open the local channel in which you want to create the destination, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > Destination**.
2. In the Configuration tab, name the destination and give it a description.
3. Specify a size for the event queue and a timeout for events in the queue. See [Destination Resource Reference on page 37](#) for details.
4. Click **Apply** and save the project.

Using a Local Destination in a Rule

Local destinations are used in rules to route events to an appropriate rule session.

Various criteria may determine which rule session is appropriate for an event, depending on the way the project is configured. In the example provided (*BE_HOME/Examples/MultipleSessionsAndLocalChannel*) large orders are routed to one rule session and small orders to another.

To route an event to a local channel, use the `Event.routeTo()` function. You can use this function for other purposes too.

In the provided example, events containing small orders are sent to the rule session that deals with small orders as follows:

```
Event.routeTo(order, "/Channels/Local/toSmall", "");
```

The signature of this function is as follows:

```
SimpleEvent routeTo(SimpleEvent event, String destinationPath,  
String properties)
```

See *TIBCO BusinessEvents Language Reference* and *TIBCO BusinessEvents Functions Reference* for more details on functions.

Channel Resource Reference



Channels allow BusinessEvents to listen to and send out messages. Channels contain destinations. After you add a destination to a channel, you can no longer change the driver type.

You can configure channels of three types, using the appropriate driver:

- **TIBCO Rendezvous channels** connect TIBCO BusinessEvents to TIBCO Rendezvous
- **JMS channels** connect TIBCO BusinessEvents to TIBCO Enterprise Message Service providers.
- **Local channels** connect multiple rule sessions at runtime (used in test environments only).



Local channels are in memory; information in a local channel could be lost if the BusinessEvents engine fails.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. See the section Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Driver	No	Select the driver for the type of channel you are configuring: <ul style="list-style-type: none"> • Local • TIBCO Rendezvous • JMS

Field	Global Var?	Description
Method of Configuration	No	<p>Resource Select Resource if you have a shared resource in your project whose properties you want to reuse for this channel.</p> <p>Note The path to the resource and the resource name cannot contain any of the keywords or other reserved words listed in the section Keywords and other Reserved Words in the <i>TIBCO BusinessEvents Language Reference</i>, and they cannot use spaces.</p> <p>Properties Select Properties to configure this channel resource using properties. See Configuration for TIBCO Rendezvous Channels on page 35 and Configuration for JMS Channels on page 36.</p>
Resource	No	If you choose Resource as the method of configuration, browse to and select the resource you want to use.
Configuration for TIBCO Rendezvous Channels		
Service	Yes	<p>The name of the service or port number through which Rendezvous sends messages. In most cases you can leave this field empty, accepting the default value.</p> <p>For more information about the Rendezvous service parameter, see <i>TIBCO Rendezvous Concepts</i> or <i>TIBCO Rendezvous Administration</i>.</p>
Network	Yes	The network over which Rendezvous sends messages. In most cases you can leave this field empty, accepting the default value. For more information about the network parameter, see <i>TIBCO Rendezvous Administration</i> .
Daemon	Yes	The location of the Rendezvous daemon, which is usually expressed as a client socket number, for example, "6555." In most cases, you can leave this field empty, accepting the default value. For more information about the daemon parameter, see <i>TIBCO Rendezvous Concepts</i> .

Field	Global Var?	Description
Configuration for JMS Channels		
ProviderURL	Yes	The URL at which BusinessEvents can contact the Enterprise Message Service server. Example: <code>tcp://localhost:7222</code>
UserName	Yes	A valid username for the Enterprise Message Service server.
Password	Yes	The password assigned to the username, above, for the purpose of accessing the Enterprise Message Service server.
IsTransacted	Yes	Accepts <code>true</code> or <code>false</code> . Specify <code>true</code> if the session has transaction semantics. Specify <code>false</code> if it has non-transaction semantics. For more information about the <code>isTransacted</code> property, see TIBCO Enterprise Message Service documentation.
ClientID	Yes	The unique client ID of the connection.

Destination Resource Reference



Within each channel, destinations direct incoming and outgoing information. A channel resource is not ready to use until it has at least one destination. Destination resources are available only from within a Channel resource.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. See the section <i>Keywords and other Reserved Words in TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Default Event	No	The event to be created from incoming messages unless otherwise specified. Optional only if you always specify an event type in the incoming message. Not used for local channel.

Field	Global Var?	Description
Serializer/ Deserializer	No	<p>Specify a serializer class to convert messages to simple events and simple events to messages.</p> <p>JMS Serializer Classes:</p> <ul style="list-style-type: none"> • BytesMessageSerializer (Default) • TextMessageSerializer <p>Rendezvous serializer classes:</p> <ul style="list-style-type: none"> • TibRvMsgSerializerWithXMLPayload (Default) • TibRvMsgSerializer <p>Serializers are not used for local channels.</p> <p>See Selecting a JMS Serializer on page 27 and Selecting a Rendezvous Serializer on page 29.</p>

Configuration for TIBCO Rendezvous Destinations

See TIBCO Rendezvous documentation for more details on these settings.

Subject	Yes	The TIBCO Rendezvous subject for incoming and outgoing messages.
RVCM Pre Registration	Yes	For TIBCO Rendezvous certified message publishers, specify pre-registered listener names as a comma separated list.
LimitPolicy	Yes	<p>How you want the Rendezvous listener to behave when it receives more messages than the MaxEvents limit. Choose one of:</p> <p>Discard_None (default)</p> <p>Discard_First</p> <p>Discard_Last</p> <p>Discard_New</p> <p>When MaxEvents or DiscardAmount are zero (unlimited), the LimitPolicy must be Discard_None.</p>

Field	Global Var?	Description
MaxEvents	No	<p>Maximum number of message events that the queue can hold.</p> <p>The default values, zero (0) means an unlimited number of events.</p>
DiscardAmount	No	<p>When the queue exceeds its maximum event limit, discard a block of events. This property specifies the number of events to discard.</p> <p>The default values, zero (0) means events are never discarded.</p>

Configuration for JMS Destinations

See TIBCO Enterprise Message Service documentation for more detail on these settings.

Queue	No	<p>Specifies if the destination is a queue or a topic. Select if the destination is a queue. If the destination is a topic, do not select it.</p>
Name	Yes	<p>Required. The name of the queue or topic.</p>
Selector	Yes	<p>Specifies a filter to pick up messages from the destination. This is a standard JMS selector based on SQL92 semantics.</p>
DeliveryMode	No	<p>The delivery mode. This property instructs the server concerning persistent storage for the message. Select one of the following:</p> <p>PERSISTENT (default)—In JMS message headers this is represented by the code 2.</p> <p>NON-PERSISTENT—In JMS message headers this is represented by the code 1.</p> <p>RELIABLE— This value is an extension to the standard, used in TIBCO Enterprise Message Service. In message headers this is represented by the code 22.</p> <p>You can also set a delivery mode in an event. See Using JMS Header Properties in Incoming and Outgoing Messages on page 47.</p>

Field	Global Var?	Description
AckMode	No	<p>The acknowledgement mode. See Changing the JMS Message Acknowledgment Mode on page 44 for a table explaining the various modes.</p> <p>You can also set an acknowledgement mode in a node's engine properties. The setting in the destination overrides the engine property setting.</p> <p>Default is EXPLICIT_CLIENT_ACKNOWLEDGE</p>
Priority	No	<p>The message priority, a numerical value between 0 and 9. Larger numbers represent higher priority.</p> <p>You can also set a priority in an event. See Using JMS Header Properties in Incoming and Outgoing Messages on page 47.</p> <p>Default is 4</p>
TTL	No	<p>The length of time that message will live (in milliseconds) before expiration. If set to 0, the message does not expire.</p> <p>You can also set a TTL (<code>JMSExpiration</code>) in an event. See Using JMS Header Properties in Incoming and Outgoing Messages on page 47.</p> <p>Default is 0</p>

Field	Global Var?	Description
Durable Subscriber Name	Yes	<p>For destinations that are JMS Topics, if you provide a DurableSubscriber Name, the destination becomes a JMS durable topic subscriber with the specified name. If you do not provide a value, the destination becomes a non-durable topic subscriber.</p> <p>The value of this property can be any unique string and can include any global variables. BusinessEvents provides a set of case-sensitive variables that produce a unique DurableSubscriberName string. See Creating Unique JMS DurableSubscriber Name Properties on page 43 for details.</p> <p>Default is: <code>%%EngineName%%:%%SessionName%%:%%ChannelURI%%:%%DestinationName%%</code></p>

Configuration for Local Destinations

Local destinations don't use serializers or deserializers or default events.

Size	No	<p>The maximum number of events to be held in the queue. The default is zero (0), which allows unlimited events in the queue.</p>
TimeOut	No	<p>Time to wait when sending an event to this local destination:</p> <ul style="list-style-type: none"> -1 Waits indefinitely 0 Does not wait >0 Waits for the number of milliseconds specified <p>Default is -1.</p>

Mapping Incoming Messages to Non-default Events

Incoming messages can be mapped to a default event that you specify when you configure the destination. All messages arriving at that destination are transformed into the default event type, unless they specify a different event.

You can also map incoming messages to specified event types. Two fields in a message header instruct BusinessEvents to map the incoming JMS or TIBCO Rendezvous message to a specified event type:

- The field named `_ns_` takes a namespace as a value. The namespace points to the event type, for example, `www.tibco.com/be/Ontology/Events/MyEvent`
- The field named `_nm_` takes the name of the event itself, for example, `MyEvent`

These fields are added and filled automatically for events arriving from ActiveMatrix BusinessWorks using a Send Event activity (see [Send Event Resource Reference on page 187](#)), and for events created using BusinessEvents rules. You can also add these fields to messages coming in from other sources, if you have control of those messages.

For information about how you specify non-default destinations in events, see [Specifying Non-Default Destinations on page 54](#).

Creating Unique JMS DurableSubscriber Name Properties

For destinations that are JMS Topics, if you provide a DurableSubscriber Name when you configure the destination resource, the destination becomes a JMS durable topic subscriber with the specified name.

The value of this property can be any unique string and can include any global variables. BusinessEvents provides a set of case-sensitive variables that produce a unique DurableSubscriberName string:

```
%%Deployment%%:%%EngineName%%:%%SessionName%%:%%ChannelURI%%:%%DestinationName%%
```

The first variable, %%Deployment%%, is a standard TIBCO global variable. The other three are only for use with the DurableSubscriberName property within BusinessEvents. See [Table 5, Variables for Use with DurableSubscriberName](#), for details.



Do not attempt to use %%EngineName%%, %%SessionName%%, %%ChannelURI%%, or %%DestinationURI%% in any area of BusinessEvents software except the DurableSubscriberName property.

Table 5 Variables for Use with DurableSubscriberName

Variable	Description
%%EngineName%%	The name of the BusinessEvents engine. The name used is established using a series of checks. See Determining the Engine (Node) Name on page 388 for details.
%%SessionName%%	The name of the rule session that is associated with the durable subscriber. Session name is the same as the BAR file name, defined using a BusinessEvents Archive resource in TIBCO Designer.
%%ChannelURI%%	The path to the channel within the BusinessEvents project: <i>/folder/.../channel_name</i>
%%DestinationName%%	The destination name.

Changing the JMS Message Acknowledgment Mode

JMS channels support connection to TIBCO Enterprise Message Service destinations. The default acknowledgment mode is EXPLICIT.

You can set the acknowledgement mode in two ways:

- In the AckMode field of a destination resource
- Using an engine property in the `be-engine.tra` file

The destination setting overrides the engine property setting.

To define the acknowledgment mode using an engine property, use one of the following, depending on whether you are using topics or queues:

```
be.channel.tibjms.topic.ack.mode acknowledgment_mode_number
```

```
be.channel.tibjms.queue.ack.mode acknowledgment_mode_number
```

Where acknowledgment mode number is one of the numbers that represent an acknowledgement mode, as shown in [Table 6](#).

Note that in TIBCO Enterprise Message Service, mode names are slightly different. They are prefixed with TIBEMS-. See *TIBCO Enterprise Message Service User's Guide* for more details.

Table 6 JMS Message Acknowledgement Modes

Number	Mode	Description
1	AUTO-ACKNOWLEDGE	Specifies that the session is to automatically acknowledge consumer receipt of messages when message processing has finished.
2	CLIENT_ACKNOWLEDGE	Specifies that the consumer is to acknowledge all messages that have been delivered so far by the session. When using this mode, it is possible for a consumer to fall behind in its message processing and build up a large number of unacknowledged messages.
3	DUPS_OK_ACKNOWLEDGE	Specifies that the session is to "lazily" acknowledge the delivery of messages to the consumer. "Lazy" means that the consumer can delay acknowledgement of messages to the server until a convenient time; meanwhile the server might redeliver messages. This mode reduces session overhead. However, should JMS fail, the consumer may receive duplicate messages.

Table 6 JMS Message Acknowledgement Modes (Cont'd)

Number	Mode	Description
23	EXPLICIT_CLIENT_ACKNOWLEDGE (TIBCO Proprietary)	<p>TIBCO Enterprise Message Service extension to JMS acknowledge modes.</p> <p>This is the default.</p> <p>EXPLICIT_CLIENT_ACKNOWLEDGE is like CLIENT_ACKNOWLEDGE except it acknowledges only the individual message, rather than all messages received so far on the session.</p> <p>One example of when EXPLICIT_CLIENT_ACKNOWLEDGE would be used is when receiving messages and putting the information in a database. If the database insert operation is slow, you may want to use multiple application threads all doing simultaneous inserts. As each thread finishes its insert, it can use EXPLICIT_CLIENT_ACKNOWLEDGE to acknowledge only the message that it is currently working on.</p>
24	EXPLICIT_CLIENT_DUPS_OK_ACKNOWLEDGE (TIBCO Proprietary)	<p>TIBCO Enterprise Message Service extension to JMS acknowledge modes.</p> <p>Like DUPS_OK_ACKNOWLEDGE (TIBEMS-DUPS-OK-ACKNOWLEDGE) except it "lazily" acknowledges only the individual message, rather than all messages received so far on the session.</p>

Table 6 JMS Message Acknowledgement Modes (Cont'd)

Number	Mode	Description
22	NO-ACKNOWLEDGE (TIBCO Proprietary)	<p>TIBCO Enterprise Message Service extension to JMS acknowledge modes.</p> <p>Suppresses the acknowledgement of received messages. After the server sends a message to the client, all information regarding that message for that consumer is eliminated from the server. Therefore, there is no need for the client application to send an acknowledgement to the server about the received message. Not sending acknowledgements decreases the message traffic and saves time for the receiver, therefore allowing better utilization of system resources.</p> <p>Note Sessions created in no-acknowledge receipt mode cannot be used to create durable subscribers.</p> <p>Note Also, queue receivers on a queue that is routed from another server are not permitted to specify NO-ACKNOWLEDGE mode.</p>

Using JMS Header Properties in Incoming and Outgoing Messages

Information in this section assumes you are familiar with JMS and its header properties. Consult your JMS provider documentation for information. This section explains only how BusinessEvents supports use of these properties.

Setting Certain Header Properties in Destinations

The JMS destination Configuration tab allow you to configure the following three header properties:

- DeliveryMode (JMSDeliveryMode)
- Priority (JMSPriority)
- TTL (JMSExpiration)



JMS header properties defined in events take precedence over properties defined in destinations.

Setting Header Properties in Events

You can configure events created from JMS messages to contain JMS header properties. Similarly, you can use event properties to set JMS header properties in outgoing messages. To use the JMS header properties, you configure events to have properties that match the JMS header fields. You must use the names as shown in [Table 7, JMS Header Field Names, on page 49](#). You only have to configure event properties for those fields that you want to use. Incoming JMS message header properties will then populate the corresponding BusinessEvents event properties.

Similarly outgoing JMS message header properties will be populated by the corresponding BusinessEvents event properties.



You can add the JMS properties to the Base event in your project so that the properties are inherited by all other events.

Note that the JMSMessageID and JMSTimeStamp properties are generated when the message is sent. You can't set these properties manually.

Use of the JMSReplyTo property in BusinessEvents is limited, as explained below.

See [Table 7, JMS Header Field Names, on page 49](#) for details on all properties.

Internal Use of JMSReplyTo Header Property for Request-Response Scenario

BusinessEvents cannot act as a client in a JMS request-response scenario because BusinessEvents currently cannot dynamically create a destination to listen for JMS messages.

When an incoming JMS message is mapped to a BusinessEvents event, the `JMSReplyTo` property of the event (if it exists) is set to `null`.

BusinessEvents does, however, use the value of the `JMSReplyTo` JMS header property internally when it sends a response to a JMS request, as explained below.

How BusinessEvents Uses JMSReplyTo Property to Reply to Messages

In a request-response scenario, you use the `Event.ReplyEvent()` function to send a reply to an incoming message. This function takes two arguments:

- The request event, which holds details about the incoming request message. In this case, the incoming message is the JMS message which you map to an event.
- The reply event, which provides information for the reply message.

You must create these events as appropriate for the specific request-response scenario, and use them in your rules. You must also associate them with BusinessEvents destination resources, in the usual way.

Internally, BusinessEvents uses the `JMSReplyTo` property from the request (incoming) message to set the destination (`JMSDestination`) in the response (outgoing) message. (That value also overrides any value set in the BusinessEvents destination resource that is associated with the reply event.)

JMS Header Field Names

The table below shows the names you must use to define event properties corresponding to JMS header field names, as well as some details about the purpose of each property. The property names are not case sensitive.

Table 7 JMS Header Field Names

Field name	Type	Description
JMSDestination	Object	<p>The destination (queue or topic) to which the message is sent.</p> <p>See How BusinessEvents Uses JMSReplyTo Property to Reply to Messages on page 48 for details on use of this property in response messages.</p>
JMSDeliveryMode	Integer	<p>The delivery mode specified by the sender. This property instructs the server concerning persistent storage for the message. Value can be:</p> <p>2—interpreted as PERSISTENT (default).</p> <p>1—interpreted as NON-PERSISTENT.</p> <p>22—interpreted as RELIABLE. This value is an extension to the standard used in TIBCO Enterprise Message Service.</p> <p>The integer values are interpreted as the text names of delivery modes.</p> <p>You can also set a delivery mode for a destination. See Setting Certain Header Properties in Destinations on page 47.</p>
JMSExpiration	Long	<p>The length of time that message will live (in milliseconds) before expiration. If set to 0, the message does not expire.</p> <p>You can also set an expiration (TTL) for a destination. See Setting Certain Header Properties in Destinations on page 47.</p>
JMSPriority	Integer	<p>The message priority, a numerical value between 0 and 9. Larger numbers represent higher priority.</p> <p>You can also set a priority for a destination. See Setting Certain Header Properties in Destinations on page 47.</p>

Table 7 JMS Header Field Names (Cont'd)

Field name	Type	Description
JMSMessageID	String	An ID that uniquely identifies each message sent by a provider. A generated value overrides any value set in the corresponding event property.
JMSTimestamp	Long	The time when the message was handed off to a provider to be sent. The message may be sent later than this timestamp value. A generated value overrides any value set in the corresponding event property.
JMSCorrelationID	String	A correlation ID that can be used to link messages. For example, you can link a response message to a request message. Optional.
JMSReplyTo	Object	The destination (queue or topic) to send the message reply to. Optional. See Internal Use of JMSReplyTo Header Property for Request-Response Scenario on page 48 for details about use of this property in BusinessEvents.
JMSType	String	The message type identifier, if used by the provider.
JMSRedelivered	Boolean	If this field is set, it is possible that the message was delivered to the client earlier, but not acknowledged at that time.

Chapter 3

Working With Simple Events

BusinessEvents supports three sorts of events: Simple events (usually referred to just as events); time events, which are timers; and advisory events. This chapter explains how to use simple events.

Topics

- [Understanding Simple Events, page 52](#)
- [Working With Simple Events, page 55](#)
- [SimpleEvent Resource Reference, page 56](#)
- [Simple Event Attributes Reference, page 62](#)

Understanding Simple Events

A simple event defines an object that represents an activity. When the term "event" is used without the qualifier advisory or time, it refers to a simple event.



In complex event processing, the term *event* is overloaded: it means an activity that happens, and the definition of an object that represents the activity (also known as an event type), and also an instance of that event type.

See *TIBCO BusinessEvents Getting Started* for a short introduction to events, which explains use of default events and default destinations.

An event type includes three sets of information:

- Configuration, which can include a default destination, time to live (expiration period), and an expiry action
- A set of properties
- A payload (optional).

This section explains how some of the options are used



At runtime, event instances that are created using rules are not automatically asserted into working memory. You must explicitly assert such events, for example using the `Event.assertEvent()` function.

Events that are created from incoming messages, on the other hand, are automatically asserted into working memory.

Inheritance

Use of inheritance can simplify event configuration. A child event inherits:

- All the parent event's properties.
- The parent event's expiry action (if set). However, an expiry action set in the child event overrides the parent event expiry action.



A parent event cannot have a payload.

Events that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore the following restrictions apply:

- If two events are related by inheritance, you cannot create a new property in one with a name that is already used in the other.
- If two unrelated events have properties that share a name, you cannot create an inheritance relationship between the two concepts.

Events and Message Acknowledgement

See [Message Acknowledgment Timing for Each Object Management Type on page 26](#) for details about the timing of message acknowledgements.

Time To Live

Time to live options are as follows:

- Zero (0): the event expires after the completion of the first RTC cycle. Do not set to 0 if you want to correlate the event with a future event or other future occurrences of this event, as explained below.
- One or higher (>0): the event expires after the specified time period has elapsed
- A negative integer (<0): the event does not expire, and must be explicitly consumed.

Set the event's time to live so that it can trigger the rules you intend. If a rule correlates different events, you must ensure that those event instances are all in working memory concurrently. For example, consider the following example:

- A process sends eventA, eventB, and eventC.
- The TTL for all three simple events is 0.
- Rule 1 has the condition: `eventA.id == eventB.id`.
- Rule 2 has the condition: `eventC.id != null`.

At runtime, BusinessEvents behaves as follows:

1. BusinessEvents receives eventA. Because there is no eventB in the working memory, eventA doesn't trigger any rules. BusinessEvents consumes eventA.
2. BusinessEvents receives eventB. Event A has been consumed—there is no eventA in the working memory. So eventB does not trigger any rules. BusinessEvents consumes eventB.
3. BusinessEvents receives eventC. EventC triggers Rule 2 because Rule 2 depends only on eventC.

To trigger Rule 1, you must configure the time to live for eventA and eventB to ensure that both events will be in working memory concurrently. You can trigger Rule 1 in these ways:

- If you know that eventA is sent before eventB, set the TTL for eventA to a time greater than the maximum period that can elapse between sending eventA and sending eventB.
- If you don't know the order in which eventA and eventB are sent, set the TTL for both simple events to a time greater than the maximum time between the occurrence of the two simple events.

Expiry Actions

For each simple event definition, BusinessEvents allows you to specify the action(s) to take when that simple event expires, using the BusinessEvents rule language. For example, you can write an action that routes the simple event to a different destination, sends a message, or creates a new simple event. This action can be anything that is possible with the rule language. An expiry action can be inherited from the event's parent.



If an event is explicitly consumed in a rule, BusinessEvents does not execute the expiry action.

Specifying Non-Default Destinations

BusinessEvents includes two functions that allow you to send simple events out to another application: `Event.sendEvent()` and `Event.routeTo()`.

- `Event.sendEvent()` automatically sends the event to its default destination.
- `Event.routeTo` takes a destination as an argument, ignoring the event's default destination.

With `routeTo` you can direct an event to a destination on a different channel from the event's default destination. You can also override the properties of the destination, for example, the subject.

You cannot, however, override the properties of the channel, for example, the network field (which is visible in the Channel Configuration tab if you select Properties as the method of configuration).

Working With Simple Events

This section provides summary steps for configuring a simple event. See [SimpleEvent Resource Reference on page 56](#) for details on how to complete the values.

To Create a Simple Event

1. Open the folder in which you want to create the event, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > SimpleEvent**.
2. In the Configuration tab, name the event and give it a description as desired. Complete the fields using guidelines in [Configuration on page 56](#).
3. In the Properties tab, add and configure properties as explained in [Properties on page 58](#). Event properties map to message properties.
4. In the Payload tab, configure an event payload that maps to the message body, if one exists. See [Payload on page 59](#).
5. Click **Apply** and save the project.

SimpleEvent Resource Reference



SimpleEvent resources are used to define an object that represents an activity, such as sending an invoice, debiting an account, and so on.

You can modify and enrich events before they are asserted into working memory. Rule evaluation depends on event values at time of assertion, so they can be changed only before assertion.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.

Field	Global Var?	Description
Time To Live	No	<p>Specify a numerical value and a time unit. Time units available are milliseconds, seconds, minutes, hours, and days.</p> <p>The timer starts after the first rule evaluation. After the time to live (TTL) period, the event expires and is deleted from working memory.</p> <p>With Persistence object management, the expired event is also marked for deletion. See Delete Retracted Objects from Database on page 256 for more details.</p> <p>The numerical value is interpreted as follows:</p> <ul style="list-style-type: none"> • One or higher (>0): the event expires after the specified number of units elapse. • Zero (0): the event expires after the completion of the first RTC cycle. Do not set to 0 if you want to correlate the event with a future event or other future occurrences of this event. • A negative integer (<0): the event does not expire, and must be explicitly consumed.
Inherits From	No	This event inherits from the event you select here. Leave blank if you don't want to use inheritance.
Default Destination	No	Unless you specify a destination explicitly, events of this type are sent to the destination you select here. For example you can send an event to the default destination of its event type using the <code>Event . sendEvent ()</code> function.
Expiry Action	No	<p>You can specify one or more actions to perform when the event expires. Click the rule function button to display the rule function editor.</p> <p>Note: If an event is explicitly consumed in a rule, BusinessEvents does not execute the expiry action.</p>

Properties

The Properties tab has the following fields.

Field	Global Var?	Description
Name	No	<p>The name to appear as the label for the property. Names follow Java variable naming restrictions. Do not use any reserved words. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i>.</p> <p>Note: In addition to standard naming restrictions, do not begin a property name with <code>_ns_</code> or <code>_nm_</code>. These have a special use. See XREF.</p> <p>For events used in JMS channels Names beginning with <code>_jms</code> or <code>jms</code> (case insensitive) are used only for JMS header properties. See Table 7, JMS Header Field Names, on page 49 for the list of properties. See Using JMS Header Properties in Incoming and Outgoing Messages on page 47. Consult the JMS specification for more details.</p>
Type	No	<p>One of: <code>String</code>, <code>Integer</code>, <code>Long</code>, <code>Double</code>, <code>Boolean</code>, <code>DateTime</code></p> <p>Note: For properties of type <code>Double</code>, all NaN (Not a Number) values are converted to <code>0.00</code>.</p>

Payload

An event can have a payload. The payload corresponds to a message body. Payloads are defined using XML. The table below describes the payload parameters. The following parameters appear in the drop-down list in the Payload tab:

Table 8 Simple Event Payload Element Parameters

Content/Parameter	Description
Complex Element	An element that contains other elements. This is like a structure in a programming language. The complex element can contain zero or more elements of other types, including other complex elements.
Name	The name of the element.
Cardinality	Values for Cardinality: <ul style="list-style-type: none"> • Required: The payload must include an instance of this element. • Optional (?); The element is not required. • Repeating (*); The element is a list that has zero or more instances. • At least one (+): The element is a list that has one or more instances.
Element of Type	An element with a specified data type. You can specify a scalar data type (string, integer, and so on), you can reference an XML type, or you can specify the TIBCO ActiveEnterprise Any data type.
Name	The name of the element.
Cardinality	See Cardinality under Complex Element.
Type	The generic data type. For example, decimal or date/time.
Type	The specific data type. For example, float or month. Refer to the <i>TIBCO BusinessWorks Palette Reference</i> for a complete list.
XML Element Reference	A reference to an element in a stored XML schema. See TIBCO Designer documentation for more information about XML schemas.
Cardinality	See Cardinality under Complex Element.
Schema	Stored XML schema that contains the element or type you want to reference.
Element	The element within the stored XML schema that you want to reference.

Table 8 Simple Event Payload Element Parameters (Cont'd)

Content/Parameter	Description
Attribute of Type	An attribute with a specified data type. You can specify a scalar data type (string, integer, and so on), you can reference an XML type, or specify the TIBCO ActiveEnterprise Any data type.
Name	The name of the element.
Cardinality	See Cardinality under Complex Element.
Type	The generic data type. For example, decimal or date/time.
Type	The specific data type. For example, float or month. Refer to the <i>TIBCO BusinessWorks Palette Reference</i> for a complete list.
Sequence	A sequence of elements. Each item in the sequence is a structure of the sub-elements of this element.
Cardinality	See Cardinality under Complex Element.
Choice	A choice of elements. The data type of this element can be one of the sub-elements defined.
Cardinality	See Cardinality under Complex Element.
All	The data type of this element can be all of the data types of the sub-elements defined.
Cardinality	See Cardinality under Complex Element.
XML Group Reference	A reference to an XML group in a stored XML schema. See TIBCO Designer documentation for more information about XML schema.
Cardinality	See Cardinality under Complex Element.
Schema	Stored XML schema that contains the element or type you want to reference.
Model Group	Select the appropriate model group from the pull-down list.

Table 8 Simple Event Payload Element Parameters (Cont'd)

Content/Parameter	Description
Any Element	A reference to any XML Element. You can use the Coercions button to supply a reference to the XML Element for this item when it appears in the input or process data.
Cardinality	See Cardinality under Complex Element.
Validation	Select the level of validation to be performed on the XML Element: <ul style="list-style-type: none"> • Strict: Must validate by locating a declaration for the element. • Skip: Do not validate. • Lax: Validate if a declaration for the element is available.

Extended Properties

The Extended Properties tab is used internally in this release and is reserved for future use.

Simple Event Attributes Reference

You can use the following attributes in rules to return information about a simple event instance.

Attribute	Type	Returns
@id	long	The event's unique internal ID.
@extId	string	The event's unique external ID. You can specify the value in the SimpleEvent resource Name field.
@ttl	long	The event's time to live, where the assertion of the event defines the start of the time to live period. You can specify the value in the SimpleEvent resource TTL field. See SimpleEvent Resource Reference on page 56 .
@payload	string	The payload as a string value. See XREF for more on specifying the payload in a SimpleEvent resource.

Chapter 4 Working With Time Events

BusinessEvents supports three sorts of events: Simple events (usually referred to just as events); time events, which are timers; and advisory events. This chapter explains how to use time events.

Topics

- [Understanding Time Events, page 64](#)
- [Working With Time Events, page 65](#)
- [TimeEvent Resource Reference, page 66](#)
- [TimeEvent Attributes Reference, page 68](#)
- [Rule Based TimeEvent Function Reference, page 69](#)

Understanding Time Events

Time events are timers that are used to trigger rules. TIBCO BusinessEvents offers two kinds of time events, repeating and rule-based.

Repeating Time Events

You can configure a time event to repeat at a configurable time interval. For example, if you configure a time event to repeat every thirty milliseconds, then every thirty milliseconds BusinessEvents creates a new time event of that type. You can configure a repeating time event to create a specified number of events at each interval. The time interval begins during engine startup (see [Appendix A, Engine Startup and Shutdown Sequence, on page 401](#)).

Rule Based Time Events

A rule based TimeEvent resource has only a name and description. You can schedule an event to be asserted into working memory using its ontology function, `ScheduleTimeEventName()` in a rule. (see [Rule Based TimeEvent Function Reference on page 69](#)). You can schedule the event to be asserted after a period of time, and you can pass information to the event and specify its time to live. You can call the `ScheduleTimeEventName()` function in different places with different time delays.

You can use rule based time events in various ways. For example, you might write rules that check for delays in order fulfillment:

1. A new Order event is asserted, and Rule A (which has Order in its scope) creates a time event T and configures it to be asserted in sixty minutes, and passes the order ID as the closure parameter value. (The rule also sends the order details to another system.)
2. Sixty minutes after Rule A executes, timer event T is asserted.
3. The assertion of time event T triggers Rule B, which has T in its scope. Rule B checks the order status. If the order is delayed, it sends out an alert.

Working With Time Events

This section explains how to create and configure a time event.

Creating a Time Event

See [TimeEvent Resource Reference on page 66](#) for information on how to complete the values.

1. Open the folder in which you want to create the time event, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > TimeEvent**.
2. In the Configuration tab, name the event and give it a description as desired.
3. In the Type field, select the type of time event you want to create:
 - Rule Based: Finish configuring the event in the rule. See [Configuring a Rule Based Time Event in a Rule on page 65](#).
 - Repeat Every (periodic): Rule fires at specified intervals.
4. If you choose Repeat Every, configure the time event following guidelines in [TimeEvent Resource Reference on page 66](#).
5. Click **Apply** and save the project.

Configuring a Rule Based Time Event in a Rule

1. Open the rule editor for the rule where you want to use the time event.
2. On the right side of the user interface, click the vertical Ontology tab and expand the list under Ontology Functions. Expand the time event you want to configure. You see the `ScheduleTimeEventName` function.
3. Drag the time event's ontology function into the Actions panel and complete the configuration, following guidelines in [Rule Based TimeEvent Function Reference on page 69](#).
4. Click **Apply** and save the project.

TimeEvent Resource Reference



TimeEvent resources are used as timers. You can configure rule-based timers, scheduled in a rule, and repeating timers, scheduled in the TimeEvent resource. Use time events to trigger rules.



Time events configured to repeat at intervals are not supported in multiple-agent (multi-engine) configurations. Rule-based time events, however, are supported.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	<p>The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i>.</p> <p>For events used in JMS channels Names beginning with <code>_jms</code> or <code>jms</code> (case insensitive) are used only for JMS header properties. See Using JMS Header Properties in Incoming and Outgoing Messages on page 47. Consult the JMS specification for more details.</p>
Description	No	Short description of the resource.
Type	No	<ul style="list-style-type: none"> Rule Based: The time event is scheduled by a rule. Repeat Every: Time events are created periodically. Complete the rest of the settings to define the period and number of events per interval.

Field	Global Var?	Description
Repeat Every	No	<p>Used with repeating time events only. Specify the interval between time events using a numerical value and a time unit.</p> <p>BusinessEvents creates the first time event immediately after the engine starts and then creates the next time event based on the RepeatEvery interval.</p> <p>A repeating time event expires after the completion of the first RTC cycle (that is, the time to live code is set internally to zero).</p> <p>Time units available are milliseconds, seconds, minutes, hours, and days.</p>
Events/ Interval	No	<p>Used with repeating time events only. Enter the number of events to create each Repeat Every interval.</p>

Extended Properties

The Extended Properties tab is used internally in this release and is reserved for future use.

TimeEvent Attributes Reference

You can use the following attributes in rules to return information about an event instance.

Attribute	Type	Returns
@id	long	The event's unique internal ID.
@closure	String	<ul style="list-style-type: none"> For repeating time events: Not used (Null value). For rule based time events: A string that was specified when the event was scheduled.
@interval	long	<ul style="list-style-type: none"> For repeating time events: The number of units between creation of successive time events. For rule based time events: Not used (0 value).
@scheduledTime	DateTime	<p>The time scheduled for asserting an instance of this event into the working memory.</p> <ul style="list-style-type: none"> For repeating time events: calculated based on the time of the last assertion into working memory of an instance of this event, and the interval. For rule based time events: specified using the <code>ScheduleTimeEventName()</code> function's <code>delay</code> parameter. See Rule Based TimeEvent Function Reference on page 69.
@ttl	long	<ul style="list-style-type: none"> For repeating time events: always 0 (zero). For rule based time events: specified using the ontology function's <code>ttl</code> parameter. See Rule Based TimeEvent Function Reference on page 69.

Rule Based TimeEvent Function Reference

Signature Event Schedule*TimeEventName*(long delay, String closure, long, ttl)

Domain action

Description For each rule-based time event you create, an ontology function is also created to enable you to schedule the assertion of an instance of the event in a rule action. The function name follows this format: *ScheduleTimeEventName*.

Parameters

Name	Type	Description
delay	long	The event is created (and asserted) the specified number of milliseconds after the rule action executes.
closure	String	You can pass information the closure parameter to a rule triggered by the event's assertion.
ttl	long	The event's time to live. Follows the same rules as the time to live setting in a simple event. See SimpleEvent Resource Reference on page 56 .

Returns

Type	Description
Event	An instance of the event type specified in the function name (<i>ScheduleTimeEventName</i>).

Chapter 5

Working With Advisory Events

BusinessEvents supports three sorts of events: Simple events (usually referred to just as events); time events, which are timers; and advisory events. This chapter explains how to use advisory events.

Topics

- [*Understanding and Working With Advisory Events, page 72*](#)
- [*Advisory Event Attributes Reference, page 74*](#)

Understanding and Working With Advisory Events

Advisory events are asserted into working memory when certain conditions occur. Add the `AdvisoryEvent` event type to rules to be notified of such conditions. An advisory event expires after the completion of the first RTC cycle (that is, the time to live code is set internally to zero).

The three types of advisory event are described next. Each advisory event has attributes for category, type, and message (see [Advisory Event Attributes Reference on page 74.](#))

Exception

The `BusinessEvents` engine automatically asserts an advisory event when it catches an exception that originates in user code but that is not caught with the `catch` command of the `BusinessEvents` Exception type (available in the rule language). For information on working with exceptions, see *Exception Handling of TIBCO BusinessEvents Language Reference*.

The advisory event category is `Exception` and the type is the Java class name of the exception. The message attribute includes the stack trace and a message (if the exception includes a message).

BusinessEvents-ActiveMatrix BusinessWorks Integration

Advisory events are also used in the in-process `BusinessEvents-ActiveMatrix BusinessWorks` integration feature `invokeProcess()` function. Such events are asserted when the `ActiveMatrix BusinessWorks` process fails or times out (or is cancelled). See [Configuring and Using `invokeProcess\(\)` on page 213](#) for more information.

The advisory event category is `Engine` and the type is `INVOKE BW PROCESS`. The error message contains the error message from the failed `ActiveMatrix BusinessWorks` process, or the timeout message.

Engine Activated Advisory Event

An advisory event is asserted when an engine has finished starting up and executing startup functions (if any).

The advisory event category is `Engine` and the type is `engine.primary.activated`. An advisory event of this type has the message: `Engine EngineName activated`.

Working With Advisory Events

You never have to create or configure an event of type `AdvisoryEvent`. Only one event type is needed. It is called `AdvisoryEvent`, and it is available for selection from the resource list you see when you add a resource to the declaration of a rule in the rule editor.

Advisory Event Attributes Reference

The `AdvisoryEvent` event type has no properties. You can use the following attributes in rules to return information about an advisory event.

Name	Type	Description
<code>@id</code>	<code>long</code>	The event's unique internal ID.
<code>@extId</code>	<code>String</code>	Null.
<code>@category</code>	<code>String</code>	<p>Broad category of advisory:</p> <p><code>Engine</code>—Used for engine events. Also used for <code>BusinessEvents-ActiveMatrix</code> <code>BusinessWorks</code> integration projects.</p> <p><code>Exception</code>—Used for exceptions not otherwise caught (see Understanding and Working With Advisory Events on page 72).</p>
<code>@type</code>	<code>String</code>	<p>Type of advisory within the category:</p> <p><code>engine.primary.activated</code>—Used in the <code>Engine</code> category. Indicates the engine has been activated (after any startup rule functions have finished execution). See Appendix A, Engine Startup and Shutdown Sequence, on page 401.</p> <p>The exception Java class name—Used in the <code>Exception</code> category.</p> <p><code>INVOKE BW PROCESS</code>—Used in the <code>Engine</code> category. Indicates a failure or cancellation or timeout of the <code>ActiveMatrix BusinessWorks</code> process.</p>

Name	Type	Description
@message	String	<p>For the Engine category, engine.primary.activated type: The message is: Engine <i>EngineName</i> activated.</p> <p>For the Exception category, exception Java class: The stack trace of the exception, preceded by the exception message, if any.</p> <p>For the Engine category, INVOKE BW PROCESS type: The error message from the failed ActiveMatrix BusinessWorks process, or the timeout message.</p>

Chapter 6

Working With Concepts

This chapter explains how to work with concepts and concept views.

For details on database concepts, see [Chapter 7, Working With Database Concepts](#), on page 107.

Topics

- [Understanding Concepts, page 78](#)
- [Understanding Concept Property History, page 80](#)
- [Understanding Concept Relationships, page 84](#)
- [Working With Concepts and Concept Relationships, page 90](#)
- [Concept Resource Reference, page 92](#)
- [Concept Attributes Reference, page 95](#)
- [Working With Concept Views, page 96](#)
- [Concept View Resource Reference, page 98](#)
- [Generating XML Schemas \(XSD files\) for Concepts and Events, page 99](#)
- [Configuring How to Handle Null Concept Property Values, page 101](#)

Understanding Concepts

Concept types are descriptive entities similar to the object-oriented concept of a class. They describe a set of properties. For example, one concept might be Department. The Department concept would include department name, manager, and employee properties.

Runtime Behavior of Concepts

Rules at runtime can create instances of concepts. For example, when a simple event arrives, a rule can create an instance of a concept using values present in the event. Rules can also modify existing concept instance property values.

Concepts must be explicitly deleted from working memory when no longer needed or they will steadily increase memory usage. Use the function `Instance.deleteInstance()` to delete concept instances.

Depending on other factors, adding modifying, and deleting concept instances can cause BusinessEvents to evaluate or re-evaluate dependent rules, as explained in [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#).



Concepts are automatically asserted into working memory when created, except in the following cases:

- Concepts returned by database query operations. See [Asserting Concepts Returned by a Database Query on page 121](#).
- Concepts passed to a rule function in the context of in-process ActiveMatrix BusinessWorks integration projects are not automatically asserted into working memory. See [Invoking a BusinessEvents Rule Function From ActiveMatrix BusinessWorks on page 204](#).

Concept Serialization and Handling of Null Value Properties at Runtime

By default, when concept instance objects are serialized to XML, properties with null values are excluded. You can change this behavior so that null values are included. You can also change the XSD for a concept object to allow null values, using the nillable attribute. See [Configuring How to Handle Null Concept Property Values on page 101](#) for details.

Concept Property History

Each concept property includes a history, the size of which is configurable. The history size determines how many previous values BusinessEvents stores for that property. See [Understanding Concept Property History on page 80](#).

Concept Relationships

Concepts can have inheritance, containment and reference relationships with other concepts. See [Understanding Concept Relationships on page 84](#).

Concept Views

You can optionally work with concepts and their relationships using a graphical user interface called a concept view. See [Working With Concept Views on page 96](#).

Understanding Concept Property History

Each concept property includes a history, the size of which is configurable. The history size determines how many previous values BusinessEvents stores for that property.

Each concept property also includes a history policy. You can also set the history policy to record all values or only changed values.



For ContainedConcept and ConceptReference properties History is tracked when a contained or referenced concept instance changes to a different concept instance. History is *not* tracked, however, when a contained or referenced concept's properties change. See [Understanding Concept Relationships on page 84](#) for more on containment and reference relationships.

History Size

If you set the history size to one or more, BusinessEvents stores the property value when the property changes, along with a date and time stamp, up to the number specified. When the maximum history size is reached, the oldest values are discarded as new values are recorded.

If you set the history size to 0, BusinessEvents does not store historical values for the concept. It stores the value without a time and date stamp.

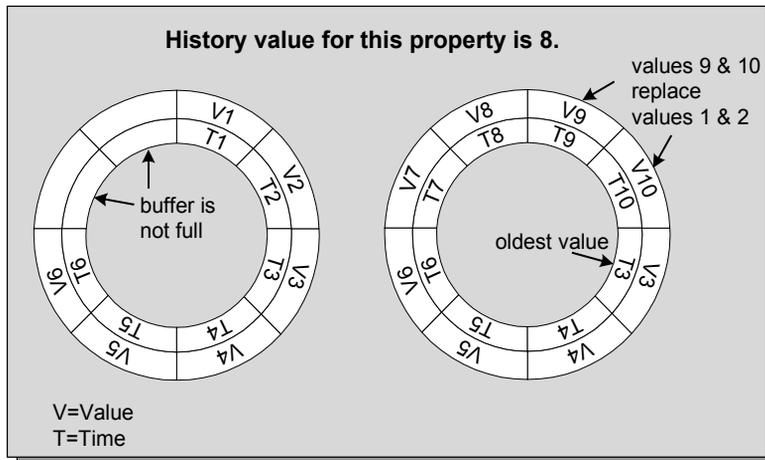
For example, consider a Customer concept:

Property Name	History	Comments
customer_name	1	These properties tend to be very stable and you may have little interest in tracking a history for them.
customer_address	1	
city	1	
state	1	
zip	1	
account_number	0	With history size 0, BusinessEvents does not record the time stamp when the value is set.
credit_limit	4	Credit limit may change more frequently and you may have an interest in tracking the changes.

Historical Values are Stored in a Ring Buffer

The historical values for a concept property are kept in a ring buffer, as illustrated in Figure 4.

Figure 4 History Ring Buffer



The ring buffer stores both the value and the time at which the value was recorded. After the ring buffer reaches maximum capacity, which is eight in this example, BusinessEvents begins replacing older values such that it always stores the n most recent values, where n is the history size. Note in [Figure 4](#) in the ring buffer on the right, after the buffer reached maximum capacity, V9 replaced V1, then V10 replaced V2, making V3 the oldest value stored in the ring buffer.

History Policy

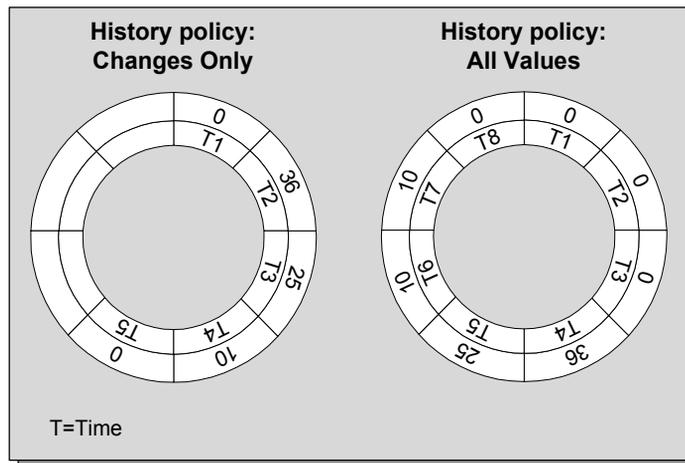
BusinessEvents can record values using either of these policies:

- **Changes Only** BusinessEvents records the value of the property every time it changes to a new value.
- **All Values** BusinessEvents records the value of the property every time an action sets the value even if the new value is the same as the old value.

Which you choose depends on what you are tracking. For example, if you are setting the history for a property that tracks how many people pass a sensor every five minutes, All Values might be the best policy. However, if you are setting the history for a property that tracks the level of liquid in a coffee pot, Changes Only might be more appropriate.

For example, look at the two ring buffers in [Figure 5](#). In both cases, the same series of values is set to the same property, but the history policy is different.

Figure 5 History Policy





History Policy and Rule Evaluation The history policy affects how frequently BusinessEvents re-evaluates rules that are dependent on the property. Each time BusinessEvents records a value, it reevaluates rules that are dependent on that property. If you track changes only, rules are re-evaluated less frequently than if you track all values.

Understanding Concept Relationships

Concepts can have inheritance, containment and reference relationships with other concepts.



Object Management Cache Modes and Concept Relationships If you use a certain cache mode for a concept, you must use the same cache mode for all of that concept's related concepts. See [Design Constraints With Cache Only Cache Mode on page 285](#) for details.

Inheritance Relationships

Definition: One concept inherits all the properties of another concept, similar to Java, where a subclass inherits all the properties of the superclass that it extends. You can define a hierarchy of inheritance, where each concept in the hierarchy extends the concept above it.

The relationship is defined by the Inherits From field in the concept resource Configuration tab.

Concepts that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore, the following restrictions apply:

- If two concepts are related by inheritance, you cannot create a new property in one with a name that already exists in the other.
- If two unrelated concepts have properties that share a name, you cannot create an inheritance relationship between the two concepts.

Also, BusinessEvents does not allow you to create an inheritance loop; for example, if Concept A inherits from Concept B, Concept B cannot inherit from Concept A.

At runtime, a rule on a parent concept also affects all its child concepts. For example, suppose the concept Coupe inherits from the concept Car. A rule on Car is therefore also a rule on Coupe.

Containment Relationships

Definition: One concept is contained inside another concept. You can define a hierarchy of containment, where each concept in the hierarchy is contained by the concept above it.

The relationship is defined using a `ContainedConcept` property in the container concept.



When working with container and contained concepts in the rule editor, the XSLT mapper and XPath builder show the entire hierarchy of properties.

In the rule editor, you can also use the `@parent` attribute to get the parent of a contained concept instance.

See [Table 9, Containment and Reference Concept Relationship Rules, on page 87](#) for rules governing the behavior of concepts linked by containment or reference. The table also helps you to choose which is the appropriate type of relationship for your needs.

Containment Example: Car with Wheels, Doors, and Engine

The following example illustrates some of the rules that are listed in [Table 9](#). To configure a concept `Car` to contain a concept `Wheel`, you add a `ContainedConcept` property, `Wheels` for example, whose value is an instance of the concept `Wheel`. The `Wheels` property provides the link between the container and contained concept:

Car (Concept)—Wheels (property)—Wheel (Concept)

The concept `Car` contains four instances of the contained concept `Wheel`, so you define the property as an array.

The concept `Car` could also contain other concepts, such as `Door` and `Engine`, defined in a similar way using `ContainedConcept` properties.

However, the contained concepts—Wheel, Door, and Engine—cannot be contained by any other concept type. They can only be contained by the Car concept. For example, the concept Wheel cannot be contained in the concept Motorbike, if it is already contained by the concept Car.



In BusinessEvents a container concept can link to a contained concept using only *one* ContainedConcept property. You can use inheritance, however, to achieve a result similar to that gained by the general programming technique of linking to multiple contained class properties. Suppose you extend the concept Wheel by creating child concepts CarWheel and MotorcycleWheel. You can then use CarWheel as the concept contained by Car, and MotorcycleWheel as the concept contained by Motorcycle. Rules that apply to Wheel also apply to CarWheel and MotorcycleWheel, because of inheritance.

Depending on your needs, another option would be to use a reference relationship instead of a containment or inheritance relationship.

Reference Relationships

Definition: One concept references another concept. A concept that is the target of a reference can itself refer to one or more other concepts. Reference relationships are not, however, hierarchical.

The relationship is defined by a ConceptReference property in the referring concept.

See [Table 9, Containment and Reference Concept Relationship Rules, on page 87](#) for rules governing the behavior of concepts linked by containment or reference. The table also helps you to choose which is the appropriate type of relationship for your needs.



Properties of concept references cannot be used in a condition.

Reference Example: Order with SalesRep and Customer

The following example illustrates some of the rules that are listed in [Table 9](#).

To configure a concept Order to reference a concept SalesRep, you add a `ConceptReference` property, `Rep` for example, whose value is the ID of concept SalesRep. The `Rep` property provides the link between the referring and referenced concepts:

Order (Concept)—Rep (property)—SalesRep (Concept)

You can also define additional reference relationships such as:

Order (Concept)—BackupRep (property)—SalesRep (Concept)

Order (Concept)—Lines (property array)—LineItem (Concept)

Order (Concept)—Cust (property)—Customer (Concept)

Customer (Concept)—Orders (property)—SalesRep (Concept)

Rules Governing Containment and Reference Relationships

This section presents the rules governing design-time and runtime use of containment and reference relationships. By comparing the rules, you can decide which type of relationship to use in a particular case.

Table 9 Containment and Reference Concept Relationship Rules

Containment One concept is contained in another	Reference One concept points to another
Design-time Rules	
One container concept can contain multiple different contained concepts, and a contained concept can also act as a container for other concepts.	
One referring concept (that is, the concept that has the <code>ConceptReference</code> property) can have a reference relationship with multiple referenced concepts, and a referenced concept can also refer to other concepts.	
(In this respect containment and reference relationships are similar.)	
A container concept can link to a contained concept using only one <code>ContainedConcept</code> property. (Some other object oriented languages do allow you to reuse object types in parent object properties.)	A referring concept link to a referenced concept using multiple <code>ConceptReference</code> properties. (That is, multiple <code>ConceptReference</code> properties can reference the same referenced concept.)
A contained concept can have only one container concept.	A referenced concept can be referred to by multiple referring concepts

Table 9 Containment and Reference Concept Relationship Rules (Cont'd)

Containment One concept is contained in another	Reference One concept points to another
Runtime Rules	
<p>When one contained instance is replaced with another, BusinessEvents deletes the instance that it replaced automatically. You do not have to delete the replaced instance explicitly.</p>	<p>When one referenced instance is replaced with another, BusinessEvents does <i>not</i> delete the instance that it replaced automatically. It may not be appropriate to delete the referenced instance. If you want to delete the referenced instance, do so explicitly.</p>
<p>When a contained instance is modified, the container instance is also considered to be modified. The reasoning can be seen by a simple example: a change to the wheel of a car is also a change to the car. Rules that test for modified instances would return the Car concept instance as well as the Wheel concept instance.</p>	<p>When a referenced instance is modified, the referring instance is <i>not</i> considered to be modified. The reasoning can be seen by a simple example: a change to the support contract for a customer is not a change to an order that references that customer.</p>
<p>When a container instance is asserted or deleted, the contained instance is also asserted or deleted, along with any other contained instances at lower levels of the containment hierarchy.</p>	<p>When a referring instance is asserted or deleted, the referenced instance is <i>not</i> also asserted or deleted.</p>

When a Contained or Referred Concept Instance is Deleted

This section highlights an important difference in behavior when history is tracked for an array property, and when history is not tracked, or the property is not an array.

Property Settings (ContainedConcept or ConceptReference Property)	Effect of deleting a Contained or Referred Concept:
Single value property, regardless of history setting.	The value of the ContainedConcept or ConceptReference property becomes null.
Multiple-value property (array), with History is set to 0 or 1 (that is, historical values are not tracked).	The array entry that held the deleted concept is removed, reducing the array size by one.
Multiple-value property (array), whose History is set to 2 or more (that is, historical values are tracked).	The array entry that held the deleted concept remains and its value is set to null, so that history can be tracked.

For more on history, see [Understanding Concept Property History on page 80](#)).

Working With Concepts and Concept Relationships

This section explains how to work with concepts, and how to set up relationships between concepts. See [Understanding Concept Relationships on page 84](#) to understand rules governing behavior at designtime and at runtime.

For details on database concepts, see [Chapter 7, Working With Database Concepts, on page 107](#).

Creating a Concept

See [Concept Resource Reference on page 92](#) for details on completing values.

1. Open the folder in which you want to store the concept, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > Concept**.
2. In the Configuration tab do the following:
 - Name the concept and give it a description as desired.
 - If the concept inherits from another concept, browse to and select the parent concept in the **Inherits From** field.
 - If you want to use a custom icon for this concept type, browse to and select the icon file in the **Icon Reference** field.
3. In the Properties tab, add properties as needed. See [Properties on page 93](#) for details. See also
4. Click **Apply** and save the project.

Creating Concept Relationships

See [Understanding Concept Relationships on page 84](#) for details about the behavior of concept relationships.

To Create an Inheritance Relationship

Do one of the following:

- In the child concept resource's Configuration tab, identify the parent concept in the **Inherits From** field.
- Within a concept view, use the inheritance link tool: . Drag a link from the child to the parent.

To Create a Containment Relationship

1. Do one of the following to add the `ContainedConcept` property:
 - In the container concept resource's Property tab, configure a property of type **ContainedConcept**.
 - Within a concept view, use the property link tool: . Drag the link from the container to the contained.
2. Configure the property as follows:

Name: Provide a helpful name such as "is a part of" to express the relationship.

isArray: Check the checkbox if this property is an array. (Note that this field is labeled Multiple in the Properties tab of the concept.)

Policy: Changes Only or All Values. See [History Policy on page 82](#)

Size: The number of historical values to keep. See [History Size on page 80](#).

To Create a Reference Relationship

1. Do one of the following:
 - In the Property tab of one of the concept resources, create a property of type `ConceptReference`.
 - Within a concept view, use the property link tool: . Drag the link from the primary concept to the concept to be referenced.
2. Configure the property as follows:

Name: Provide a helpful name such as "is a lineitem of" to express the relationship.

isArray: Check the checkbox if this property is an array. (Note that this field is labeled Multiple in the Properties tab of the concept.)

Policy: Changes Only or All Values. See [History Policy on page 82](#)

Size: The number of historical values to keep. See [History Size on page 80](#).

Editing or Deleting Concept Relationships

You can't edit or delete the relationship using the concept view graphical user interface. Instead, work with the concept properties directly. Select the concept in the concept view or in the project tree, and edit or delete the property in the Properties tab.

Concept Resource Reference



Concept resources are descriptive entities similar to the object-oriented concept of a class. They describe a set of properties. For example, one concept might be Department, and it could include department name, manager, and employee properties.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Inherits From	No	<p>If you want this concept to inherit all the properties of another concept, browse to and select that concept.</p> <p>Concepts that are related to each other directly or indirectly by inheritance cannot have distinct properties that share a common name. Therefore the following restrictions apply:</p> <ul style="list-style-type: none"> • If two concepts are related by inheritance, you cannot create a new property in one with a name that is already used in the other. • If two unrelated concepts have properties that share a name, an inheritance relationship cannot exist between them.
Icon Reference	No	You can associate an image file with a concept type. Add the image as a shared resource. Then browse to and select it here.

Properties

The Properties tab has the following fields:

Field	Global Var?	Description
Name	No	The property name.
Type	No	<p>Any of the following types: <i>String, Integer, Long, Double, Boolean, DateTime, ContainedConcept, ConceptReference</i></p> <p>When you create a property of type <i>ContainedConcept</i>, you are creating a containment relationship. The concept that you are currently configuring is the container; the concept you specify as a property is the contained concept.</p> <p>When you create a property of type <i>ConceptReference</i> you are creating a property that references another concept.</p> <p>See Understanding Concept Relationships on page 84 for more details.</p> <p>Note: For properties of type <i>Double</i>, when a backing store is used, all NaN (Not a Number) values are converted to 0.00.</p>
Multiple	No	<p>Check the Multiple checkbox if this property is an array.</p> <p>Consider, for example, an <i>Order</i> concept: In most cases, an <i>Order</i> concept would allow only one value for the <i>customer</i> property but multiple values for the <i>line_item</i> property. Selecting the Multiple checkbox creates a property array.</p>
Policy	No	<p><i>BusinessEvents</i> can record historical values using either of these policies:</p> <p>Changes Only <i>BusinessEvents</i> records the value of the property every time it changes to a new value.</p> <p>All Values <i>BusinessEvents</i> records the value of the property every time an action sets the value even if the new value is the same as the old value.</p>

Field	Global Var?	Description
History	No	<p>Determines if historical values are stored, and if so how many.</p> <p>The maximum value allowed is 1024.</p> <p>Zero (0): BusinessEvents does not store historical values for the concept. It stores the value without a time and date stamp</p> <p>One or more (>0): BusinessEvents stores the property value when the property changes, along with a date and time stamp, up to the number specified. When the maximum history size is reached, the oldest values are discarded as new values are recorded. See Understanding Concept Property History on page 80.</p> <p>Note: Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.</p>

Extended Properties

The extended properties tab is for use with database concepts. See [Chapter 7, Working With Database Concepts, on page 107](#) for details. Except for that use, the Extended Properties tab is used internally in this release and is reserved for future use.

Concept Attributes Reference

You can use the following attributes in rules to return information about a concept instance.

Entity	Attributes	Type	Returns
Concept	@id	long	The concept instance's unique internal ID.
	@extId	string	The concept instance's unique external ID.
ContainedConcept	@id	long	The contained concept instance's unique internal ID.
	@extId	string	The contained concept instance's unique external ID.
	@parent	concept	The parent concept instance. (This is treated as a concept reference in the language.)

Working With Concept Views

The ConceptView resource provides a graphical user interface that makes it easy to configure a group of related concepts and their relationships. Everything you can do in a concept view, you can also do using concept resources. Within a concept view you can

- Create and configure new concepts
- Edit existing concepts
- Create relationships between concepts: inheritance, containment, and reference.

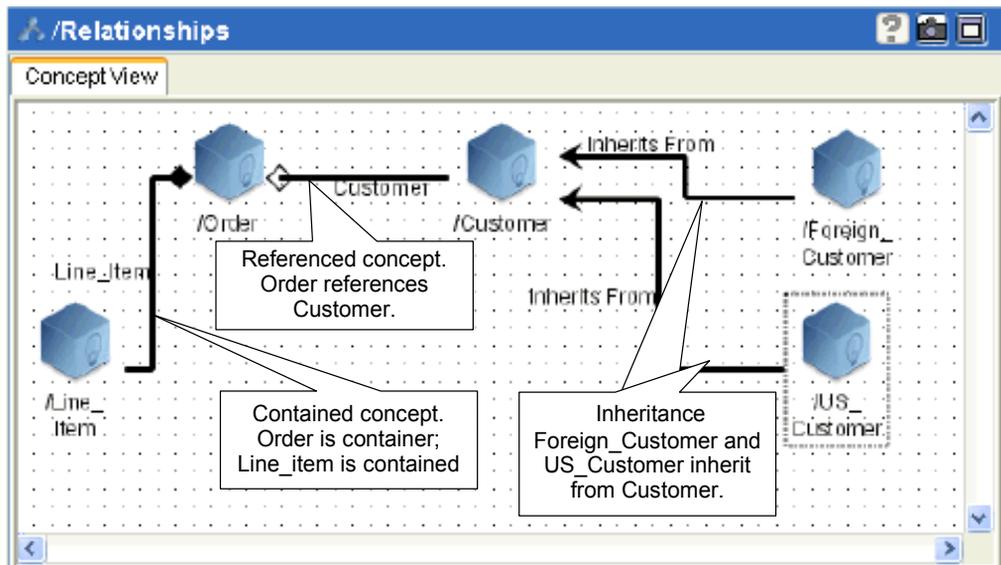


Deleting a concept view does not delete concepts When you delete a configured ConceptView resource, the concepts within it are not deleted or affected by the deletion.

The Concept View User Interface

The following figure shows the three types of relationships, displayed in a concept view. Notice the difference in appearance of each link.

Figure 6 Three Types of Concept Relationships



Using Existing Concepts and Creating New Concepts

You can drag existing concepts from the project tree into the concept view (design panel). Any relationships already defined in their properties are reflected in the graphical display.

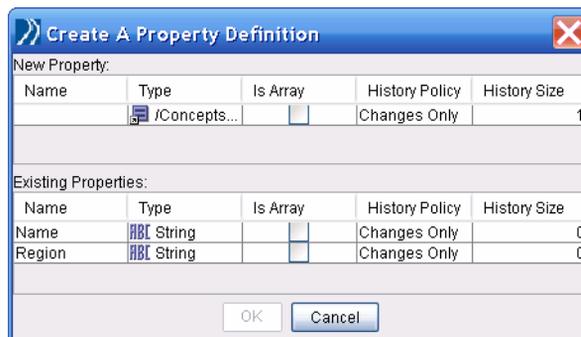
You can create new concepts in a concept view using the Create Concept () button. The new concept is added to the same folder that contains the concept view. You can configure the new concept using the configuration and property tabs, as well as using the relationship links in the concept view.

Creating Relationships Between Concepts

When you create relationships between concepts in a concept view, appropriate concept properties are added to the concept resources.

Creating Reference Properties

When you connect two concepts using the Create Reference Property link, a dialog appears where you can give the relationship property a name and perform any other configuration of the property as appropriate. For example, suppose you put two concepts in the design panel, an Account concept and an Account Rep concept. You then select the Reference relationship link, and drag the cursor from AccountRep to Account, finishing with a click on Account. You would then see this dialog:



New Property:				
Name	Type	Is Array	History Policy	History Size
/Concepts...		<input type="checkbox"/>	Changes Only	1

Existing Properties:				
Name	Type	Is Array	History Policy	History Size
Name	String	<input type="checkbox"/>	Changes Only	0
Region	String	<input type="checkbox"/>	Changes Only	0

The lower panel provides a view-only list of existing properties. In the upper panel, you can complete the configuration of the relationship property.

Concept View Resource Reference



The Concept view resource provides an optional feature offering a convenient, graphical way to configure concepts and relationships between concepts.

ConceptView Resource Toolbar

The ConceptView resource provides a toolbar with the following items. To use an item, click the item, and then click in the design panel



Add Existing Concept Adds an existing concept to the concept view. After clicking the tool, click in the design panel to bring up a list of concepts that are not already referenced by the current concept view. Select the desired concept from the list.



Create Concept and add it to Concept View Creates a new concept resource and immediately references it in the current concept view. After clicking the tool, click in the design panel once for each concept you want to add.



Create Inheritance Link Creates a relationship between two concepts such that one concept inherits from the other. After clicking the tool, click and drag from the child concept (the one that will inherit) to the parent (the concept from which the child will inherit).



Create Reference Property Link Creates a relationship between two concepts such that one concept references the other concept through a property. After clicking the tool, click and drag from the main concept to the property concept.



Create Contained Property Link Creates a relationship between two concepts such that one concept contains the other concept within a property. After clicking the tool, click and drag from the container to the contained.

Generating XML Schemas (XSD files) for Concepts and Events

The Generate Schema utility lets you export concepts and events to XML Schema Definition (XSD) files, one per entity, in a specified location. The files use the same folder structure as the project from which they are exported. In addition, `_BaseConcept.xsd` and `_BaseEvent.xsd` are generated in the root of the selected directory.

XML schemas are used for interoperability between BusinessEvents and third party tools or SOA platforms that use well-defined XML for message communication, transformation, and validation.

In the XSD files, concepts are represented as follows:

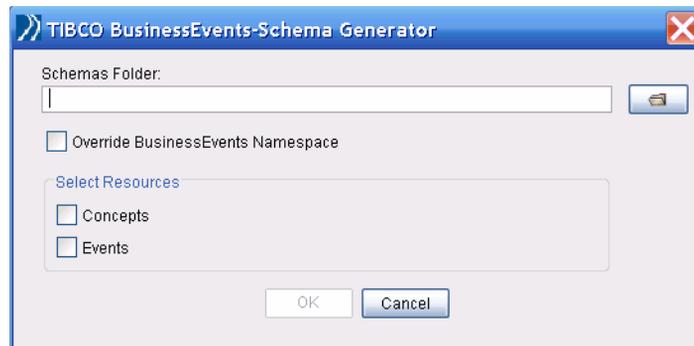
- Each concept is exported to its own complex type using its own namespace.
- Referenced concepts have a corresponding `ref` attribute in the parent's complex type.
- Contained concepts have a corresponding `type` attribute in the parent's complex type.



Determining How Null Concept Property Values are Handled A TIBCO Designer property determines whether the `xsd:nillable` attribute is set on all elements in concept XSD files. See [Enabling Use of the Nilable Attribute on page 101](#) for details.

To Generate an XML Schema (XSD File)

1. From the TIBCO Designer window menu bar, go to **Tools > BusinessEvents Tools > Generate Schema**. The Schema Generator dialog displays.



2. In the Schemas Folder field, browse to and select the folder where you want to put the schema files.

3. Select the **Override BusinessEvents Namespace** checkbox to specify a different namespace.

Provide a different namespace to avoid conflicts with the source concepts and events. If you do not provide a namespace, the default BusinessEvents namespace is used. (BusinessEvents events and concepts have a hidden schema.) If the source entities and generated schema files are in the same folders, use of the default BusinessEvents namespace results in a namespace conflict. In this case, you must provide a namespace.

4. In the Select Resources panel, select Concepts or Events or both. Schemas for all concepts or all events in the project (or both) are generated accordingly.
5. Click **OK**.
6. In all cases, an informational message in the confirmation dialog warns that generating schema in the current project with BusinessEvents namespace will result in conflicts. Click **Yes** to continue.

The XSD files for the selected resources (all concepts, all events, or both) are generated in subdirectories of the selected directory. Subdirectory names match the project folders. The `_BaseConcept.xsd` and `_BaseEvent.xsd` files are generated in the root of the selected directory.

Configuring How to Handle Null Concept Property Values

This section explains three cases for special handling of null concept property values:

- [Enabling Use of the Nillable Attribute on page 101](#)
- [Enabling Null Property Values to Appear When Serializing Concepts to XML on page 101](#)
- [Enabling and Setting Special Treatment of Numeric Null Values on page 103](#)

This section also provides related procedures and a reference to the properties used to define the handling of null concept property values.

Enabling Use of the Nillable Attribute

The presence of the `xsd:nillable` attribute in an XSD element means that the corresponding element in the XML file permits null values.

Setting `java.property.tibco.be.schema.nil.attrs=true` in `designer.tra` causes the `xsd:nillable` attribute ("`xsd:nillable=true`") to be set on all elements in the BusinessEvents concept XSD. When an element in the XML file generated using that XSD has a null value, the `xsi:nil="true"` attribute is set on that element.

When set to false, the `xsd:nillable` attribute is not added and the corresponding XML file does not treat empty elements as null values.

In the absence of the `xsd:nillable` attribute in the XSD element, a corresponding empty element in the XML file is assumed to have a value. Elements that have no value are treated as empty strings ("").



Effect on schema generation tool The setting for this property affects the concept XSD files output by the schema generation tool (see [Generating XML Schemas \(XSD files\) for Concepts and Events on page 99](#)).

Enabling Null Property Values to Appear When Serializing Concepts to XML

By default concept properties with null values are excluded when concept objects (instances) are serialized to XML. You can override this behavior.

Setting the following property to false in the TIBCO Designer TRA file causes properties with null values to be included in the XML representation of a concept:

```
java.property.tibco.be.schema.exclude.null.props=false
```

This property is set in the `designer.tra` file.

Examples of Nillable Attribute and Null Properties Settings

These examples illustrate the effect of the following properties on concept serialization:

```
java.property.tibco.be.schema.nil.attrs
java.property.tibco.be.schema.exclude.null.props
```

If Null Properties are Excluded

```
java.property.tibco.be.schema.nil.attrs= true or false
java.property.tibco.be.schema.exclude.null.props=true
```

Suppose a Customer concept instance has no value for its CustomerName property. By default, the CustomerName property is excluded from the XML output. The output might look like the following:

```
<CustomerID>111</CustomerID>
<Country>Japan</Country>
<City>Tokyo</City>
```

If null properties are excluded when concepts are serialized, the `java.property.tibco.be.schema.nil.attrs` property has no effect on concept serialization.

If Null Properties are Included and the Nillable Attribute is Set

```
java.property.tibco.be.schema.nil.attrs=true
java.property.tibco.be.schema.exclude.null.props=false
```

The output for the Customer concept instance shown above would be as follows, where there is no value for the CustomerName element in the concept instance:

```
<CustomerID>111</CustomerID>
<Country>Japan</Country>
<City>Tokyo</City>
<CustomerName
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:nil="true"/>
```

If Null Properties are Included and the Nillable Attribute is not Set

```
java.property.tibco.be.schema.nil.attrs=false
java.property.tibco.be.schema.exclude.null.props=false
```

In this case, each null property is considered to be an empty string, and is represented, for example, as follows:

```
<CustomerName/>
```

Enabling and Setting Special Treatment of Numeric Null Values

If you enable null values to be output to XML (see [Enabling Null Property Values to Appear When Serializing Concepts to XML on page 101](#)), then you may also want to configure additional properties for defining how to treat null values for numeric types, as explained in this section.

BusinessEvents does not implicitly support null values for numeric types. This can lead to interoperability issues when working with external sources, such as databases, which do permit blank (null) values.

To address such issues, you can enable special treatment of numeric null values at design time. At runtime, BusinessEvents then uses a special numeric value for each numeric datatype to represent a null value. Default special values are provided and you can override the defaults at runtime (see [Table 10, Properties for Null Property Handling, on page 105](#)).

The special numeric values that indicate null are used in BusinessEvents when serializing and deserializing a concept to and from its XML representation, and when performing various operations on database concepts and the database tables to which they are linked.

The special numeric values that indicate null appear only in BusinessEvents. The appropriate null value is used in the XML or database representation of the concept property.

Conversely, when deserializing or importing a concept instance, BusinessEvents represents numeric null values using the special numeric values that indicate null in the concept instance.

To enable special treatment of numeric null values, set the following property in `designer.tra`:

```
java.property.tibco.be.schema.treat.null.values=true
```

At runtime you can override the default values using the following properties in `be-engine.tra`:

```
java.property.tibco.be.property.int.null.value=value
java.property.tibco.be.property.long.null.value=value
java.property.tibco.be.property.double.null.value=value
```

Setting Designtime Properties for Handling of Null Property Values

This section provides the procedure for setting properties in the TIBCO Designer properties file to set all properties described earlier in this section. After you change any of these properties, you must rebuild and redeploy the EAR file for the project.

To Set Designtime Properties for Handling of Null Property Values

1. Close all instances of TIBCO Designer.
2. Open the `TIBCO_DESIGNER_HOME/bin/designer.tra` file for editing.
3. Set the following properties as desired:
 - To enable null property values to appear when serializing concepts to XML, add the following property and set it to false:


```
java.property.tibco.be.schema.exclude.null.props=false
```
 - To enable use of the nillable attribute in the concept XSD, add the following property and set it to true:


```
java.property.tibco.be.schema.nil.attrs=true
```
 - To enable special handling of null properties, add the following property and set it to true:


```
java.property.tibco.be.schema.treat.null.values=true
```

Also see [Setting Runtime Properties for Special Treatment of Null Values on page 104](#)
4. Save the file.

Setting Runtime Properties for Special Treatment of Null Values

If you have enabled special treatment of null numeric properties ([Enabling and Setting Special Treatment of Numeric Null Values on page 103](#)), you can override the default special numeric values that indicate numeric null values in BusinessEvents as follows.

1. Open the `BE_HOME/bin/be-engine.tra` file for editing.
2. add the following properties and provide the special values as desired:


```
java.property.tibco.be.property.int.null.value=value
java.property.tibco.be.property.long.null.value=value
java.property.tibco.be.property.double.null.value=value
```

Choose values that will not be misinterpreted as literal values.
3. Save the file.
4. Open TIBCO Designer and build the EAR file for the project.
5. Deploy the EAR file.

TIBCO Designer Property Reference for Null Property Handling

Set the following properties in the `designer.tra` file as needed to configure the output for your needs before you generate the EAR file.

Table 10 Properties for Null Property Handling

Property	Notes
Properties Set in <code>designer.tra</code>	
<code>java.property.tibco.be.schema.nill.attrs</code>	<p>Setting this property to true causes the <code>xsd:nilable</code> attribute ("<code>xsd:nilable=true</code>") to be set on all elements in the BusinessEvents concept XSD.</p> <p>See Enabling Use of the Nilable Attribute on page 101</p> <p>Possible values are true and false.</p> <p>Default is false.</p>
<code>java.property.tibco.be.schema.exclude.null.props</code>	<p>When this property is set to true, null-valued concept properties are not output when the concept is serialized to XML.</p> <p>When set to false, null-valued concept properties are output to XML.</p> <p>See Enabling Null Property Values to Appear When Serializing Concepts to XML on page 101.</p> <p>Possible values are true and false.</p> <p>Default is true.</p>
<code>java.property.tibco.be.schema.treat.null.values</code>	<p>Setting this property to true causes BusinessEvents to use special numeric values that indicate null for numeric datatypes. The special numeric values are set using the properties listed next.</p> <p>See Enabling and Setting Special Treatment of Numeric Null Values on page 103.</p> <p>Possible values are true and false.</p> <p>Default is false.</p>

Table 10 Properties for Null Property Handling (Cont'd)

Property	Notes
Properties Set in be-engine.tra	
<pre>java.property.tibco.be.property .int.null.value</pre>	<p>These properties define a special numeric value that indicates null. Use a value that will not be confused with an actual numeric value.</p>
<pre>java.property.tibco.be.property .long.null.value</pre>	<p>These properties are used only if <code>java.property.tibco.be.schema.treat.null.values</code> is set to true.</p>
<pre>java.property.tibco.be.property .double.null.value</pre>	<p>Default values for each numeric datatype are the following Java constants:</p> <pre>int: Integer.MIN_VALUE long: Long.MIN_VALUE double: Double.MIN_VALUE</pre> <p>For Integer and Long these constants represent the most negative value. For Double the constant represents smallest positive nonzero value (4.9e-324).</p>

Working With Database Concepts

This chapter explains how to work with database concepts and their associated catalog of RDBMS functions.

The database concepts feature is available only in TIBCO BusinessEvents Enterprise Suite.

Topics

- [*Database Concepts Overview, page 108*](#)
- [*Importing Database Tables or Views With the DB Import Utility, page 109*](#)
- [*Database Concepts Extended Properties, page 113*](#)
- [*Performing Database Operations, page 116*](#)

Database Concepts Overview

Database concepts are BusinessEvents concepts that you create by mapping tables or views from a database to BusinessEvents concepts. One table or view maps to one BusinessEvents database concept definition. A row in the table or view maps to one database concept instance.

Database concepts are ordinary BusinessEvents concepts with additional functionality. They enable you to perform database operations such as insert, update, delete and query, using rules and rule functions. In this way, you can keep the database concepts synchronized with their database equivalents (see [Performing Database Operations on page 116](#) for more details).

Database concepts are created using the DB Import utility. This utility introspects the specified database schema and generates BusinessEvents concepts.

Optionally, you can import database constraints, that is, relationships between tables. In the BusinessEvents project the table relationships become relationships between concepts.

Optionally, you can create a simple event for each concept you import. The payload of each simple event contains the corresponding concept's schema.

Handling of Null Value Properties

In order to work with external databases, concept instance are serialized to XML. By default, when concept instance objects are serialized to XML, properties with null values are excluded. You can control handling of null properties in the XML representation of serialized concepts. Note that for numeric datatypes, some special handling may be required for interoperability. See [Configuring How to Handle Null Concept Property Values on page 101](#) for details.

Supported Database Products

The feature has been tested with Oracle 10g Enterprise Edition (see the readme file for specific version information) using the following drivers:

- TIBCO's Oracle driver
- Oracle's thin driver

Importing Database Tables or Views With the DB Import Utility

DB Import is a TIBCO Designer-based utility that allows you to connect to a database and import schemas. This utility enables you to select tables or views to import and creates corresponding BusinessEvents database concepts. By default the utility names the database concepts using the table or view name. You can, however, provide different names as desired.

The utility provides an option to create one event for each table or view. The generated event's payload corresponds to the schema of the concept created for that table or view. These events can then be used to perform operations such as `queryUsingPrimaryKeys()`.

Task A Set Up the Database Connection

The first step is to set up a JDBC Connection resource in the project and configure it for the database whose tables or views you want to import.

Instructions are provided in a different section of this guide. See [Adding a JDBC Connection Resource on page 345](#), and adapt the instructions as needed.

Remember to include the JDBC Connection resource as a shared resource (as explained in that section).



You can import concepts into the project from multiple data sources using different JDBC connections.

Task B Import Database Tables and Views

1. From the TIBCO Designer window menu bar, go to **Tools > BusinessEvents Tools > DB Import**. The DB Import wizard displays, showing the Specify Database Connection dialog:

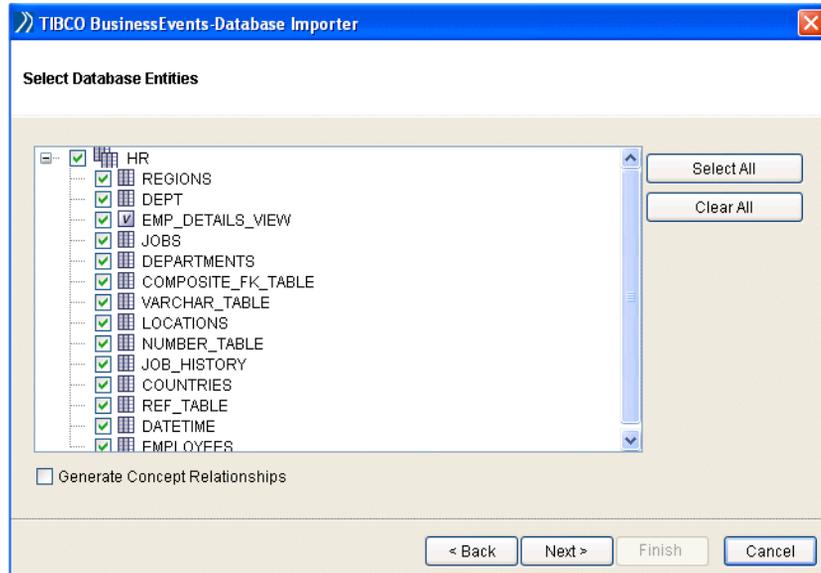
The screenshot shows a dialog box titled "TIBCO BusinessEvents-Database Importer" with the subtitle "Specify Database Connection". The dialog contains the following fields and buttons:

- JDBC Resource URI: [Text Field] [Browse Button]
- Connect Using User: [Text Field]
- Connect Using Password: [Text Field]
- Owner Name: [Text Field]
- Login Timeout(sec): [Text Field]
- Buttons: < Back, Next >, Finish, Cancel

2. In the JDBC Resource URI field, Click the **Browse** button and select a **JDBC Resource URI**. The database connection details display. Override the connection details if needed, then click **Next**. You see the Project Resource Location dialog.

Note that the database schema name is used automatically as a project folder name.
3. In the Concept Folder field, browse to and select or create the project folder for the database concepts.
4. (Optional) Check the **Generate Events** checkbox if you want to create a simple event for each imported concept. When you check the Generate Events checkbox, additional fields appear for you to specify details for the events.
 - a. In the Events Folder field, browse to and select or create the project folder for the events.
 - b. In the Event Destination URI field, click the Browse button and select the default destination for the events.

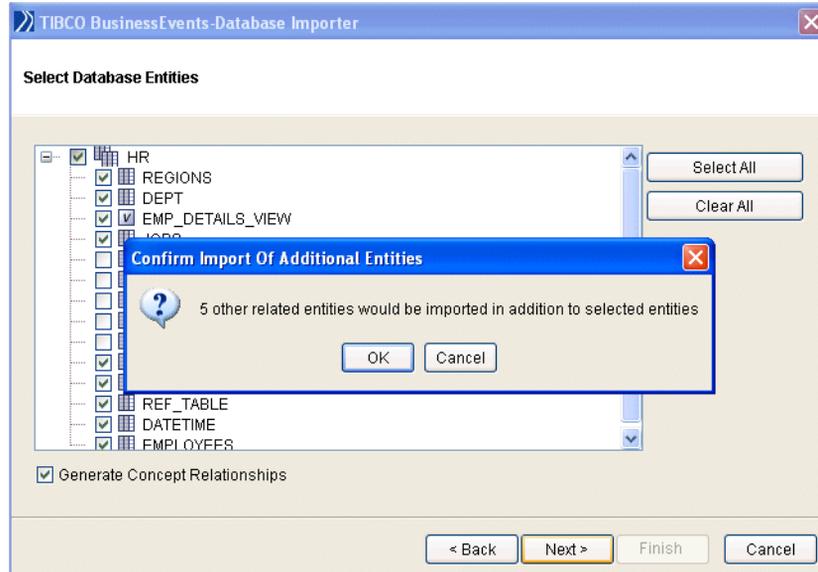
- Click **Next**. You see the Select Database Entities dialog.



- Select the database entities (tables or views) from which you want to create database concepts (and events if you selected that option).
- (Optional) Select the **Generate Concept Relationships** checkbox if you want to create relationships between concepts (concept reference or contained concept) based on database constraints.

The utility imports any additional database entities that were not in the original selection but that must be imported because they have a relationship

to the selected subset of entities. If this is the case, a message such as the following displays:



Click **OK** to import the selected entities along with their related entities, and click **Next**.

8. At the Concept Name For Database Entity dialog, database schema names are provided as default names of the BusinessEvents concepts. Edit the BusinessEvents concept names as desired, then click **Finish**.
9. The utility creates the concepts (and events if you chose that option). Browse your project tree to verify that the expected concepts and events have been created.

Database Concepts Extended Properties

Database concepts have additional properties related to their database source. Individual concept properties also have extended properties. The extended properties for a concept are available on the Extended Properties tab in TIBCO Designer. The extended properties for a property are available using a pop-up window accessed from the Properties tab for the concept.

Do not modify any extended properties, except to manually keep the extended property definitions synchronized with their database source. For example, if the DBA changes the name of a column in a database, change the value of the COLUMN_NAME extended property to match. You don't have to re-import the concept.

Extended Properties for a Concept

The following table describes the extended properties for a database concept.

Table 11 Database Concept —Extended Properties

Property Name	Description
SCHEMA_NAME	Name of the database schema from where this concept originates
OBJECT_NAME	Name of the database table or view
OBJECT_TYPE	<ul style="list-style-type: none"> • T for table • V for view
PRIMARY_KEY_PROPS	A comma separated list of names of primary key properties. Note that these should be set to the property names rather than column names.
JDBC_RESOURCE	A JDBC URI to which this DB concept maps.

Extended Properties for a Concept Property

A database concept property also has extended properties. These properties are of two types, primitive properties and concept relationship properties. To view the extended properties for a concept property, follow the steps provided in the section, [To View the Extended Properties of a Database Concept Property on page 115](#).

The following tables describe the two types of extended properties.

Table 12 Database Concept Property—Extended Properties of Type Primitive

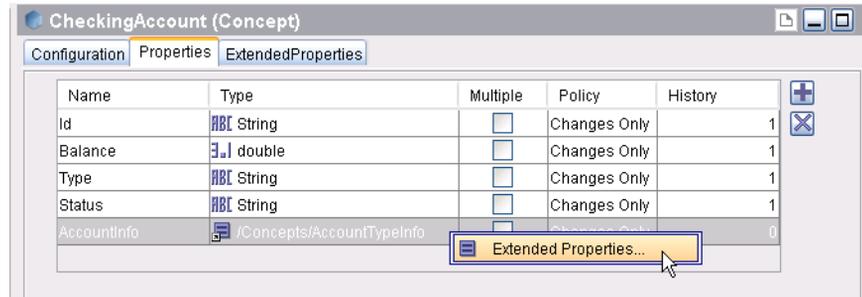
Property Name	Description
COLUMN_NAME	Name of the column in the database.
DATA_TYPE	Data type as specified in the database.
LENGTH	Length as defined in the database.
PRECISION	Precision as defined in the database.

Table 13 Database Concept Property—Extended Properties of Type Concept Relationship

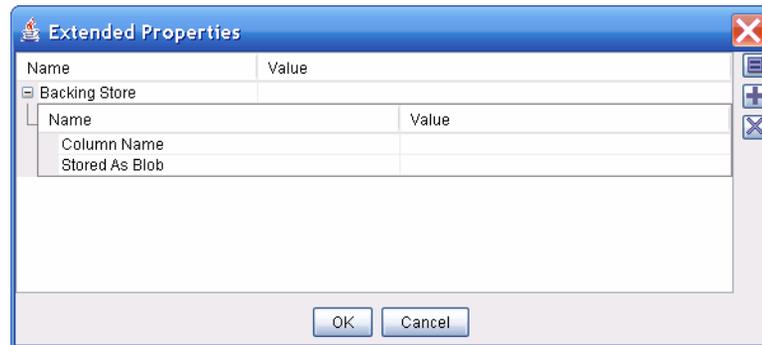
Property Name	Description
REL_TYPE	C - If the concept held in this property is a contained concept. R - If it is a reference. Note that all the relationships are modeled as references, even those defined as contained concepts.
REL_KEYS	A table containing join keys mapping source and target concepts.
NAME	Property name in the concept.
VALUE	Property name in the property's concept that corresponds to the join key in the current concept.

To View the Extended Properties of a Database Concept Property

1. In the Properties tab, right-click in the row of the property whose extended properties you want to display. You see the Extended Properties button:



2. Click the **Extended Properties** button. You see the Extended Properties dialog, showing the property's extended properties. The example shows extended properties for a ConceptReference property:



As explained in the introduction, only make modifications to keep the properties synchronized with their database equivalents. Do not make any other changes.

Performing Database Operations

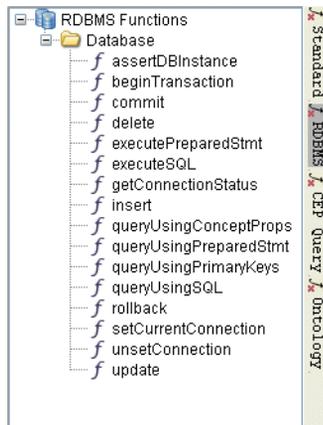
This section explains how to use RDBMS functions in BusinessEvents to connect to a database and perform database operations in rules and rule functions.



- Ensure that the JDBC Connections that were used to import databases are added to the Shared Resources in the EAR configuration. (See [Adding a JDBC Connection Resource on page 345](#) for an example of a JDBC Connection.) If these connection resources are not added to the shared resources, they will not be available at runtime, and the database concept features will not work. See
- In the rules, you need to get a handle to these connections for performing database operations.

Database concepts enable you to manipulate the database using a rule or rule function. The database concepts feature has an O/R (object to relational) mapping feature that enables you to simply act on the concepts (database concepts are BusinessEvents concepts with database behavior) and delegate the persistence of these objects to the catalog functions. The RDBMS functions catalog allows you to perform basic CRUD operations on database tables using database concepts.

Figure 7 RDBMS functions catalog



Setting and Unsetting a Connection

In a rule, use `setCurrentConnection()` once before performing any database operation. Use `unsetConnection()` once after all database operations are performed. You must call `unsetConnection()` even in case of a failure within the rule function. If you don't unset the connection, the connection is not returned to the database connection pool, which amounts to a resource leak.

Use of `setCurrentConnection()` may result in an exception if the underlying database is disconnected.

Testing the Database Connection Periodically

You can test database connections periodically using the following engine properties

Property	Description
<code>tibco.be.jdbc.test.connections.interval</code>	Interval in seconds at which the database connection is tested. Default is 60 seconds.
<code>tibco.be.jdbc.connection.retry.count</code>	Specifies how often the connection is retried. 0 (zero) means do not retry. -1 means a background thread will keep testing the connection status at the specified interval, and attempt to recover if the connection fails. Default is 0 (zero).

Defining Transactions

By default, all database statements are individually committed. For example, if an insert call results in multiple insert statements, then each one gets committed individually. To ensure that all statements are grouped inside a transaction, call the `beginTransaction()` function explicitly.

You have to explicitly commit the transaction using `Database.commit()` otherwise it results in a rollback.

Example

```

try
{
    Database.setCurrentConnection ("/MyDbConnection");
    Database.beginTransaction ();
    Concept instance = Instance.createInstance("/someconcept");
    Database.insert(instance)
    Database.commit();
    Database.unsetConnection();
} catch (Exception e) {
    Database.rollback();
    Database.unsetConnection();
}

```

Performing Insert Operations

The insert function `Database.insert()` inserts an object and its concept properties (if any) recursively into the database. You can insert all related objects at once instead of performing individual inserts. The join keys are internally managed.

In the case of concept references, foreign keys in the referencing concept are updated with primary keys in the referenced concept.

In the case of contained concept properties, foreign keys in the contained concept are updated with primary keys in the container concept.

If columns in the database are modified by the database during inserts, these changes are also made in the concept instances. This is usually the case when primary keys are automatically generated or when columns have default values.

Use Of Sequences During Inserts

If values of primary keys are to be acquired using an Oracle sequence, do the following.

Create an XML file with an extension `.sequences.xml` and add it to the project as a shared resource.

The format of the XML file is as follows:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<unique_identifiers>
  <unique_identifier entity = "conceptURI"
                    property = "property-name"
                    unique_identifier = "sequence-name" />
</unique_identifiers>
```

The `entity` attribute holds the URI of the concept whose property should use the Oracle sequence.

The `property` attribute is the name of the property that holds the primary key.

The `unique_identifier` attribute holds the name of the Oracle sequence to use.

When a record is inserted into the database, this file is consulted and an Oracle sequence value is generated for the specified primary key property.

Performing Update and Delete Operations

Primary Key Required for Update and Delete Operations

Each instance of a database concept maps to one row in a database table.

In order for the database to perform updates or deletes on the BusinessEvents objects, or for BusinessEvents to perform updates or deletes on the database tables, the software must be able to uniquely identify the row. Therefore, you can only perform delete and update operations if the table has at least one primary key.

If you attempt to perform an update for a row that has no primary key, an exception is thrown.

To find out whether a table has primary keys or not, open the project in TIBCO Designer, and check the `PRIMARY_KEY_PROPS` extended property, which is on the Extended Properties tab for the concept. If this property has no value, no primary keys exist and you can't perform update or delete operations.

Performing Update Operations on the Database

The `Database.update()` function updates the database with values contained in the concept.

Example

```

try
{
Database.setCurrentConnection ("/MyDbConnection");
Database.beginTransaction ();
/*the instance passed to update operation is an instance of the
dbconcept*/
Database.update(instance)
Database.commit();
Database.unsetConnection();
} catch (Exception e) {
Database.rollback();
Database.unsetConnection();
}

```

Performing Delete Operations on the Database

The `Database.delete()` function deletes a record corresponding to the concept instance in the database. If `cascade` is set to `true`, it deletes all database records corresponding to the contained concept property references and nulls out foreign key references in database records corresponding to the concepts that refer to the concept being deleted.

Example

```

try
{
Database.setCurrentConnection ("/MyDbConnection");
Database.beginTransaction ();
/*the instance passed to delete operation is an instance of the
dbconcept*/
Database.delete(instance)
Database.commit();
Database.unsetConnection();
} catch (Exception e)
{
Database.rollback();
Database.unsetConnection();
}

```

Performing Database Query Operations

Several RDBMS catalog functions enable you to query the database.

Asserting Concepts Returned by a Database Query

Concepts returned by these query operations are not automatically asserted. You must explicitly assert these concepts as required, using the `Database.assertDBInstance(concept, deep)` function to assert a database concept.

When a database concept is asserted with the *deep* parameter set to `true`, all the referenced concepts are also asserted.

Chapter 8 **Working With Scorecards**

This brief chapter explains how to work with scorecards.

Topics

- *[Understanding and Working With Scorecards, page 124](#)*

Understanding and Working With Scorecards

A scorecard is a special type of concept. A scorecard serves as a static variable that is available throughout the project. You can use a scorecard resource to track key performance indicators or any other information.

Unlike concepts and events, each scorecard resource is itself a single instance — it is not a description for creation of instances. You create the scorecard at design time. Its values can be viewed and updated using rules.

It is more accurate to say there is one instance of a scorecard per rule session. When a BusinessEvents engine runs with multiple rule sessions, each rule session has its own instance of the score card.

It is not necessary to add scorecards to the declaration of a rule. Because there is only one instance of each scorecard in a deployed BusinessEvents application. Any change causes all rules that use the scorecard in their conditions to be evaluated.

After configuring your ScoreCard resource, use rules to gather the information you need in the scorecard. To access the ScoreCard in a rule, use this syntax:

```
folder . folder . ScoreCard . property
```

For Example:

```
int i = sales.eastern.statistics.numOrders;
```

Creating a Scorecard

Configuring a scorecard is similar to configuring a concept, except that scorecards don't have relationships.

1. Open the folder in which you want to store the scorecard, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > ScoreCard**.
2. In the Configuration tab do the following:
 - Name the scorecard and give it a description as desired.
 - If you want to use a custom icon for this scorecard, browse to and select the icon file in the Icon Reference field.
3. In the Properties tab, add properties as needed. See [Concept Resource Reference on page 92](#) for details on the fields.
4. Click **Apply** and save the project.

ScoreCard Resource Reference



Scorecard configuration is the same as concept configuration, except that scorecards have no relationships with each other. Scorecards, therefore, have none of the properties used for setting up relationships. Scorecard properties can be of any primitive type.

See [Concept Resource Reference on page 92](#) for details about scorecard properties.

Working With Rules and Functions

This chapter explains how to work with rule sets, rules, rule functions, and other kinds of functions.

For more detailed information about working with the BusinessEvents rule language, see *TIBCO BusinessEvents Language Reference*.

Topics

- [Understanding Rules and Rule Functions, page 128](#)
- [Understanding Conflict Resolution and Run to Completion Cycles, page 130](#)
- [Working With Rule Sets, Rules, and Rule Functions, page 134](#)
- [Working With the Rule Editor, page 136](#)
- [Working With Startup and Shutdown Rule Functions, page 142](#)
- [Working With Event Preprocessors, page 145](#)
- [RuleFunction Resource Reference, page 148](#)
- [Ruleset Resource Reference, page 151](#)
- [Rule Resource Reference, page 152](#)
- [Understanding and Working With Functions, page 154](#)

Understanding Rules and Rule Functions

This section introduces some main points to understand about rules and rule functions. Also see [Understanding and Working With Functions on page 154](#) for details on the many functions you can use in rules and rule functions.



Effect of Cache Only Cache Mode If you are using the Cache Only cache mode (advanced cache option) for one or more entities, you must consider how to handle the cache-only entities when you write rules and preprocessor rule functions. See [Explicitly Loading Objects into Working Memory, With the Cache Only Mode on page 285](#).

Rules

A rule includes a declaration of entity types, and (optionally) one or more separate conditions, which evaluate to true or false, and an action, which is eligible to execute only when all conditions evaluate to true.

Within the **Actions** panel, semicolons separate statements; one statement can span multiple lines; one line can include multiple statements. Statements in a rule action might execute tasks, create an instance, modify property values in an instance, create and send a simple event, and so on.

Rule Priority

BusinessEvents rules are declarative rather than procedural: there is no inherent order for processing. However, a priority property allows you to specify which rules should execute first.

TIBCO recommends that you use priorities prudently; that is, unless there is reason to use a priority setting, let the rule engine determine the sequence of execution. This lessens the complexity of the rule maintenance task, and takes advantage of the power of the rule engine.

See [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#).

Rule Sets

A rule set is a container for rules. All rules exist within a rule set. Grouping rules into rule sets enables you to deploy a selection of rule sets in a BAR (BusinessEvents archive). Your project design determines which rules to put in each rule set and which rule sets to deploy using different agents.

Rule Functions

A rule function is a function you write in the BusinessEvents rule language for use in a project. To create a rule function, you use a RuleFunction resource. All rule functions created for a project are available project-wide and are not contained in a rule set.

Rule functions can take arguments and can return a value. The return type can be set to void, indicating that the rule function does not return anything.

You can use rule functions in the same way as the standard (provided) functions, custom functions, and ontology functions (which are created automatically from the entity types). You can use rule functions in rule conditions and rule actions.

You can assign rule functions to be used as event preprocessors (see [Working With Event Preprocessors on page 145](#)), and as startup and shutdown actions (see [Working With Startup and Shutdown Rule Functions on page 142](#)).

Virtual Rule Functions and Decision Manager

Virtual rule functions are created in the TIBCO Designer project using RuleFunction resources. A virtual rule function has arguments but no body or return type, and the Is Virtual checkbox on the Configuration tab is checked.

The implementations for virtual rule functions are provided in the Decision Manager business user interface. Business users select a virtual rule function. they drag and drop entities from a project explorer to form rows in a decision table. Each row forms a simple rule, for example, Condition: age less than 18; Action: refuse credit.

Implementations of the virtual rule functions are known as *decision table classes*. Because these classes originate in the Decision Manager application, they are also known as *external classes*, from the point of view of the BusinessEvents nodes.

The classes are placed in one or more known locations and are loaded at system startup, or using JMX while the system is running. See notes for `be.engine.cluster.externalClasses.path` in [Table 27, Cache Cluster Properties, on page 297](#).

You can also unload previously deployed classes, using RMI. See Chapter 2, RMS and Decision Manager Configuration in *TIBCO BusinessEvents Decision Manager* for the procedure and for configuration details.

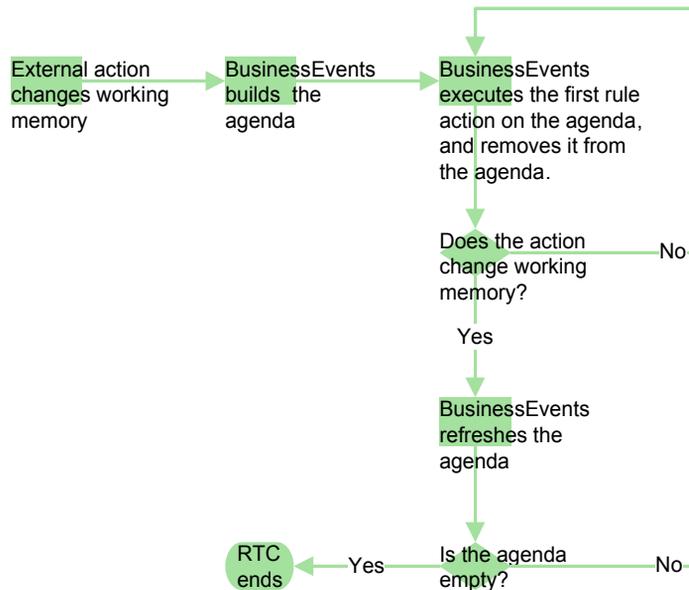
You can use virtual rule functions in BusinessEvents just as you would use any other rule function. At runtime the implementations provide the body of the function.

Additionally, a category of functions enables you to work with the specific features of virtual rule functions. See [VRF Functions on page 158](#) for details.

Understanding Conflict Resolution and Run to Completion Cycles

This section helps you to understand what triggers rules to execute, and why a rule may not execute, so that you can design rules more effectively.

The flowchart below illustrates one *run to completion* (RTC) cycle, which generally begins when an external action causes changes to working memory. One RTC cycle ends when there are no more rule actions to execute as a result of that initial change. This is also known as *forward chaining*, or *inferencing*. During one RTC no new external actions can affect the working memory. An RTC is composed of one or more *conflict resolution cycles*. A conflict resolution cycle begins when BusinessEvents builds (or refreshes) a *rule action agenda*, which is used to determine which rule action to execute, and ends when a rule action is executed (or the agenda is empty). If the rule action changes working memory, another conflict resolution cycle begins. Sections following the flow chart provide more detail.



A change in working memory initiates the first conflict resolution cycle in the RTC

Working memory changes The first of the conflict resolution cycles is initiated by change in working memory caused by an external action such as a message arriving at a destination.

All other changes to working memory during one run to completion cycle (RTC) occur because of rule actions. When there are no remaining rule actions on the agenda, the run to completion cycle ends. The next cycle begins with another change from an external source.

BusinessEvents builds the agenda To build the rule action agenda, BusinessEvents examines all rules that are *newly true* because of the change to working memory and compares them with rule dependencies. See [How a Rule Becomes Newly True on page 132](#) and [How Conflict Resolution Uses Rule Dependencies on page 132](#) for more details. The agenda's entries are ordered according to rule priority and other criteria.

Executing a rule ends each conflict resolution cycle

BusinessEvents executes the first rule on the agenda and removes it from the agenda As a result, one of the following occurs:

- The rule action does not change working memory and BusinessEvents executes the next rule entry in the agenda (if there is one).
- OR
- The rule action does change working memory and BusinessEvents refreshes the rule action agenda to account for the changes.

Note that events are not sent to destinations until the entire RTC is complete.

BusinessEvents refreshes the agenda Any of the following can occur:

- Rules that have become newly true are added to the agenda.
- Rules that have become false are dropped from the agenda.
- Rules that were newly true at the last conflict resolution cycle and are still true remain in the agenda. (In other words, rules are newly true for the duration of the run to completion cycle unless they become false.)

As a result, either the agenda is empty and the RTC ends, or the first rule in the refreshed agenda is executed and a new cycle of conflict resolution begins.

An empty agenda marks the end of one RTC

An empty agenda ends the RTC At some point, no more actions remain to be executed. The conflict resolution has run to completion. At the end of one RTC the following happens (depending on how the project has been configured):

- Events are sent to destinations.
- Cache OM: Changes are saved to the cache and written to the backing store.
- Cache OM, cache-only cache mode: At the end of the RTC, all cache-only objects are removed from working memory.
- Persistence OM: One transaction is completed and saves changes (enabling rollback in case of failures).
- Profiler: profiler data is updated.

How a Rule Becomes Newly True

As stated above, the rule action agenda is built by examining all rules that are *newly true*. A rule is *newly true* if it has become true due to a change in working memory.

A rule that was false and becomes true because of the changes in working memory during the RTC is newly true.

Less obviously, a rule that was already true can also become newly true. For example, a rule may already be true because a condition that specifies a range is satisfied. It becomes newly true if the property value in working memory changes but is still within the range. For example, the condition `c . b < 10`; is true if working memory includes a `c . b` with value 9. It is newly true if an action at the end of a conflict resolution cycle changes the value from 9 to 8.

In the case of a rule with no conditions, it is the assertion of an object in the scope (declaration) of the rule that makes the rule newly true.

A rule remains newly true until it is executed or it is removed from the agenda, or the RTC ends.

A rule is removed from the agenda because a change in working memory during an RTC means that the facts no longer satisfy its dependency set, for example because a concept instance is deleted or a concept property changes value.

How Conflict Resolution Uses Rule Dependencies

Before any data enters the system, BusinessEvents builds the Rete network, which embodies all the *rule dependencies*, using the rule conditions (if any). All the dependencies in a rule are called its *dependency set*. A rule's dependency set is everything needed to determine the truth of all the conditions. For example, a rule has this condition:

```
c . name == "Bob";
```

Where `c` is a concept of type `/Customer`. In this case, the dependency set of the rule contains only the name property of the concept type `/Customer`. As another example, suppose a rule has these conditions:

```
b . num < 10 ;
hasAGoldMembership(c) ;
```

Where `b` is another concept type and `num` is one of its properties. The dependency set for that rule is `b . num` and `c`.

Testing the Truth of a Rule's Conditions Using the Dependency Set

During a conflict resolution cycle, BusinessEvents tests each rule's dependency set against the new set of facts. If the facts match the rule dependencies, the rule conditions are all true and the rule action is added to the rule action agenda. The structure of the Rete network enables very quick matching between facts and rule dependency sets.

If BusinessEvents cannot calculate dependencies on the properties of an entity from the rule condition, for example if you pass an entity to a function, BusinessEvents evaluates the rule every time the entity or its properties changes.

Order of Evaluation for Different Types of Rule Conditions

Using a rule's dependency set, BusinessEvents evaluates the following kinds of rule conditions in the order shown, to perform the evaluation efficiently.

1. Filters, that is, conditions that only involve one scope element (object).
2. Equivalent join conditions, that is, conditions that compare two expressions using == or != where each expression involves one (different) object from the scope.
3. Non-equivalent join conditions, that is, conditions involving two or more scope elements other than equivalent joins.

Class filters are the least expensive operations, in terms of processing cost, and non-equivalent joins are the most expensive. To optimize performance, ensure that as much filtering as possible is done first, to reduce the number of objects for which an expensive condition must be evaluated.

Working With Rule Sets, Rules, and Rule Functions

The Rule resource is only available within a Ruleset resource.

See Also

- [Working With the Rule Editor on page 136](#).
- [Working With Startup and Shutdown Rule Functions on page 142](#)
- [Working With Event Preprocessors on page 145](#)
- *TIBCO BusinessEvents Language Reference* for details about working with the rule language.

Creating Rule Sets and Rules

See [Ruleset Resource Reference on page 151](#) and [Rule Resource Reference on page 152](#) for details on the settings.

1. Navigate to the place in your folder structure where you want to add the rule set. Right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > RuleSet**.
2. In the Configuration panel, give the rule set a name and (optional) description.
3. With the rule set open, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > Rule**.
4. In the Configuration panel, give the rule a name and (optional) description.
5. If you want to control the order in which rules execute, set the Priority field accordingly. Highest priority is 1



The Priority setting is used by the runtime engine when determining the order in which rules are fired. Those with a number closer to one fire first.

When there is no reason to force rules to execute in a particular order, leave the Priority set to the default and let the runtime engine determine rule order.

6. Click **Apply** and save the project.
7. Double-click the rule icon to open the rule editor. See [Working With the Rule Editor on page 136](#) for details.

Creating Rule Functions

1. Navigate to the place in your folder structure where you want to add the rule function. Right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > RuleFunction**.
2. In the Configuration tab, give the rule function a name and (optional) description.
3. Select the Is Virtual check box if you are creating a virtual rule function for Decision Manager.
4. From the Validity column, select the desired scope for the rule function:
 - Action
 - Action and Condition
 - Action, Condition and Query
5. If the rule function returns a value, specify the Return Type, otherwise leave set to void.
6. Click **Apply** and save the project.
7. Double-click the rule icon to open the rule function editor.

Working with the Rule Function Editor

Working with the rule function editor is similar to working with the rule editor, except that instead of the Declarations panel, you use an Arguments panel to specify any arguments for the function.

In the Body area, use the BusinessEvents rule language to specify the behavior of the function (unless you are defining a virtual rule function, which has only a signature and no implementation).

See [Working With the Rule Editor on page 136](#) for details.

Using Rule Functions

One way to select a rule function for use in a rule is to click the Ontology Function tab, then expand folders as needed and select a rule function from the list. (The Ontology function tab is one of the tabs that appears on the right side of the TIBCO Designer window when you are using the rule editor.)

Alternatively if you know the function signature you can directly enter the details.

Working With the Rule Editor

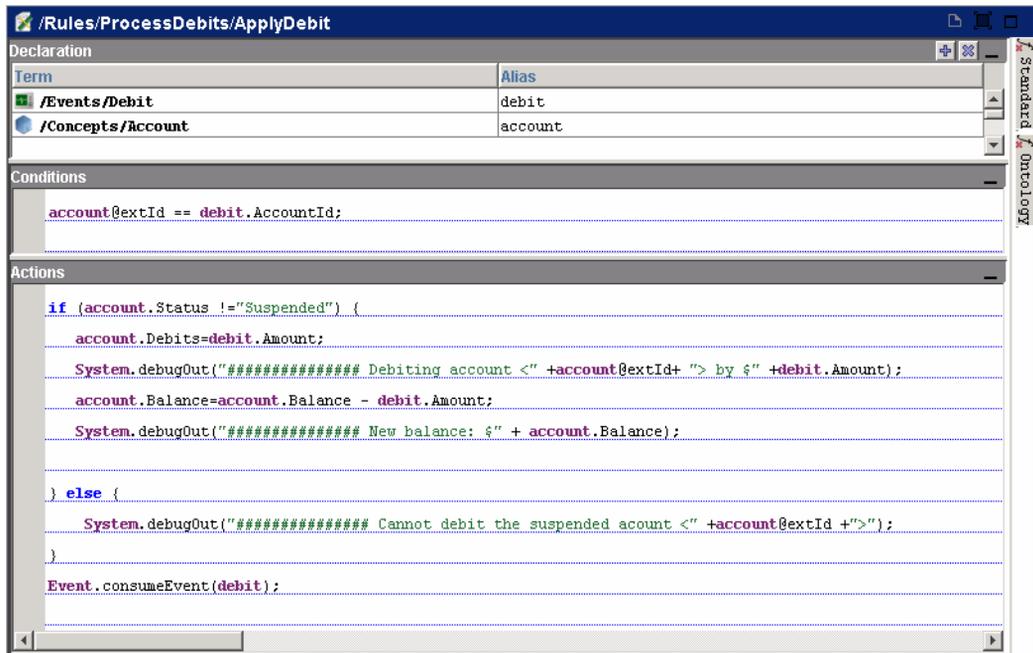
This section explains how to work in the rule editor user interface.

For details on working with the rule language see *TIBCO BusinessEvents Language Reference* for details about working with the rule language.

For details on creating new rules see [Working With Rule Sets, Rules, and Rule Functions on page 134](#).

Opening the Rule Editor

To open the rule editor select a rule in the project panel, or double-click a rule in an open rule set.



The rule editor includes three panels, the Declaration panel, the Conditions panel, and the Actions panel, explained next.

Adding Terms to the Declaration Panel

Drag and drop the resources that you will be using in your rule into this area, or click the plus sign and select resources. This defines the scope of the rule.

BusinessEvents assigns an alias to each resource. You can edit the alias.

Declaring multiple terms of the same type allows the rule to consider multiple instances of the corresponding Entity.



Scorecards Scorecards are like concepts except that there is only one instance of a scorecard (or more accurately, one instance per agent when multi-engine features are used). It is therefore not necessary to put scorecards in the declaration because a scorecard never requires an alias. You can use scorecard properties in conditions (just as you would concept properties).

Specifying Conditions in the Conditions Panel

Write condition statements in this area. Each line is a complete statement. Each condition must evaluate to a Boolean value. Each line is joined to the others with an implicit AND operator. All of a rule's conditions must evaluate to true for the conditions to be satisfied.



The order in which conditions are evaluated is determined internally by the engine (see [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#)). To enforce the order of evaluation between two or more conditions, put them on the same line (that is, in one statement ending in a semicolon) joined by the logical operator &&.

Be aware of some differences in execution when you combine conditions. For example, consider these two conditions:

```
concept.containedConcept != null;
concept.containedConcept.property == "test";
```

A null pointer exception may be thrown if `concept.containedConcept` is null, because the second condition can be checked before the first.

However, if you combine the conditions as follows:

```
concept.containedConcept != null &&
concept.containedConcept.property == "test";
```

A null pointer exception is not thrown if `concept.containedConcept` is null:

Specifying Actions in the Actions Panel

Write action statements in this area. BusinessEvents executes these actions when there is a set of objects in working memory that matches all the conditions, and the rule is at the top of the rule agenda.

Using Global Variables

To use a global variable in the rule editor, use one of the `System.getGlobalVariableAs*` functions. For example:

```
System.getGlobalVariableAsString("Hostname", "Localhost")
```

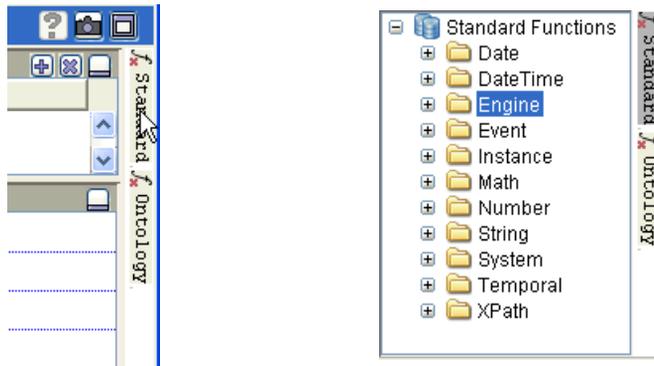
Where `Hostname` is the name of the variable and `Localhost` is an optional literal value to use if the variable is not found.

Do not use this syntax: `%%Global.Variable.Name%%`.

Using Functions

For information on additional procedures for using mapper functions (which are decorated with a small *m* when viewed in the function catalog) see [Using XSLT Mapper Functions on page 139](#).

1. In the Conditions or Actions area of the rule editor (as needed) do one of the following.
 - Type the function name, including catalog folders but not the catalog name, for example `Database.beginTrace()`.
 - In the function registry, click on the name of a catalog, for example, *fx Standard*. Drill down on categories within the catalog to expand to lists of functions and drag the desired function to the Condition or Actions area.



2. Provide parameter values as indicated by the tooltip. You can hover the cursor over a function to display a tooltip showing the function's syntax. You can also see the tooltip contents in the online reference, *TIBCO BusinessEvents Functions Reference*.

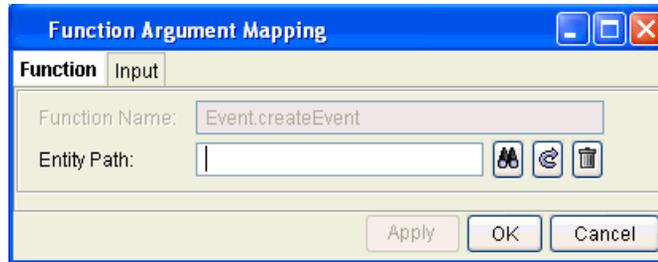
See [Understanding and Working With Functions on page 154](#) for descriptions of the various function catalogs, and an explanation of the decorations that appear on many function names.

Using XSLT Mapper Functions

When you drag a mapper function or type the name of a mapper function into the **Actions** or **Conditions** areas, then type an open parenthesis ("(") BusinessEvents displays `<<xslt-template>>`, which is a link to the mapper.

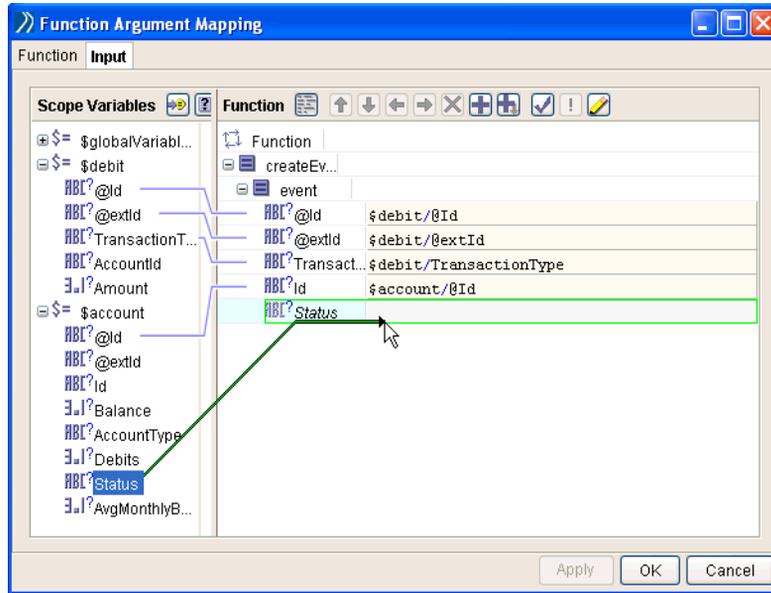
To use the mapper, do the following:

1. Click on the link (`<<xslt-template>>`) to open the mapper:



2. Click the browse icon () to the right of the **Entity Path** field to display a list of resources of the appropriate type for the function.
3. Select a resource and click **Apply**.
4. Select the **Input** tab. BusinessEvents displays a list of variables associated with the project on the left and a list of properties for the selected resource on the right.
5. Drill down on the lists on both sides to expose the variables and properties.

- Drag variables from the left to the appropriate property on the right, as shown:



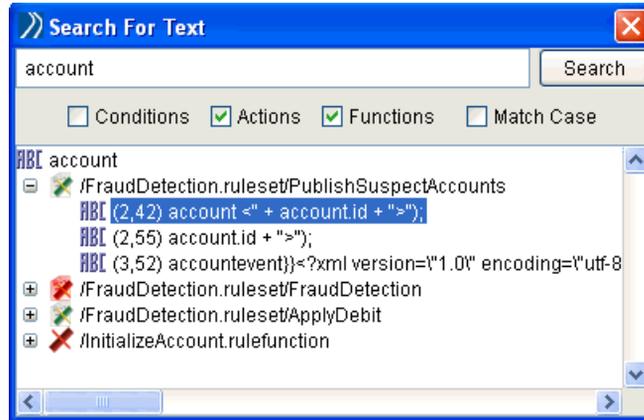
- Click **OK**.



Inside the mapper, indexes for property arrays start with one (1). In the BusinessEvents language, indexes for property arrays start with zero (0).

Working With the Rule Editor (and BAR) Search for Text Utility

The rule editor has its own search utility, which allows you to search for text within the Condition and Action areas. You can search for any text or limit your search to functions. When you select a rule or ruleset resource, a search button () appears on the main toolbar. Click the button to bring up the Search For Text dialog:



Drill down and then double-click on an item in the list of results to highlight that text in the rule editor.



This feature is also available in other BusinessEvents resources, for example, BAR resources. The behavior is the same.

Adding Advisory Events to Rules

You can add the AdvisoryEvent event type to a rule declaration. You might do this in order to track certain types of exceptions.

To add an advisory event to a rule, click the plus icon () in the Declarations panel and select Advisory Event from the list of resources.

See [Chapter 5, Working With Advisory Events, on page 71](#) for more details.

Working With Startup and Shutdown Rule Functions

Startup and shutdown rule functions are rule functions that are configured to execute during normal system startup and shutdown, respectively. There is more to say about startup rule functions and most of this section focuses on them.

Startup and shutdown rule functions take no argument and their Validity field (see [RuleFunction Resource Reference on page 148](#)) must be set to Action.



See [Engine Startup and Shutdown Sequence on page 401](#) for a useful reference that helps you understand what you can do in startup and shutdown actions.



Reminder, Cache Only cache mode If you are using the Cache Only cache mode (advanced option) for one or more entities, you must consider how to handle any cache-only entities used in the functions. See [Explicitly Loading Objects into Working Memory, With the Cache Only Mode on page 285](#).

Startup Rule Functions

Startup rule functions are optional and are used to initialize the system. For example they can provide initial values for scorecards. Startup rule functions can be used to perform more "expensive" operations so that the system is more efficient at runtime. For example, in a startup rule function for a query agent, you can create a query definition that you then reference in other rule functions. As another example, in BusinessEvents-ActiveMatrix BusinessWorks integration projects, you can call the `BusinessWorks.init()` function in a startup action.

Startup rule functions may trigger rule actions. However, note that BusinessEvents executes all startup rule functions before it begins the first RTC cycle, which completes when all rules eligible to execute have executed and no more actions remain.

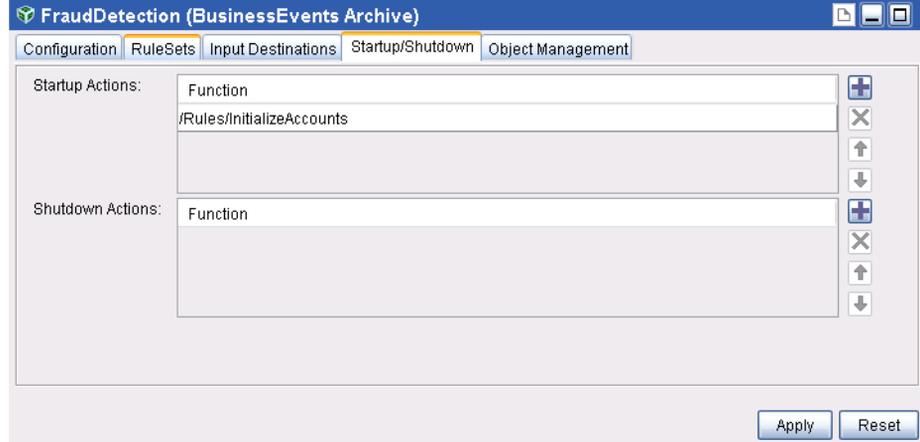
Shutdown Rule Functions

Shutdown rule functions are optional and are used to perform various actions during a normal shutdown, for example, they can send events to external systems.



BusinessEvents cannot consume an event in a shutdown action.

How Configured Add rule functions, in order of desired execution, to the respective sections of the Startup/Shutdown tab of the BusinessEvents archive (BAR) resource:



See [Startup and Shutdown on page 366](#) for more details.

When Startup Rule Functions Execute

Startup rule functions execute whenever an active node starts This includes failback to a failed node that has restarted. If you want to execute startup rule functions on only one node in a deployment, use programming logic to do so. For example, first perform a check that determines if the function has already executed using a `try catch` block.

Startup rule functions do not execute on failover When an inactive node becomes active, it does not execute startup rule functions (See [Inactive Agents and Fault Tolerance Behavior on page 307](#)).

Creating Entities With a Startup Action in a Multi-Engine Project

Startup (and shutdown) rule functions execute in all active agents. Startup rule functions can be used to create entities, for example, for static reference data. When multi-engine functionality is used, ensure that multiple agents do not attempt to create the same entity. For example, use a `try catch` block, as is done *BE_HOME/examples/FraudDetectionCache* example.

See [Designing With Multiple Active Inference Agents on page 274](#) for more information.

ActiveMatrix BusinessWorks Containers

In ActiveMatrix BusinessWorks integration projects, if ActiveMatrix BusinessWorks is running as the container, do not specify any startup actions that result in starting or invoking an ActiveMatrix BusinessWorks process. See [Design Considerations on page 195](#).

Note that after the ActiveMatrix BusinessWorks engine is initialized, processes that invoke BusinessEvents rule functions will fail if the BusinessEvents engine has not finished starting up. For example, an ActiveMatrix BusinessWorks process that listens to a JMS queue may attempt to invoke a BusinessEvents rule function before the BusinessEvents engine has started up.

Working With Event Preprocessors

Event preprocessors are rule functions with one argument of type simple event. They perform tasks after an incoming message is transformed into a simple event but before it is asserted into working memory.



You can modify and enrich events before they are asserted into working memory. Rule evaluation depends on event values at time of assertion, so they can be changed only before assertion.



Modifying concepts not permitted A preprocessor can assert and delete concepts, but cannot modify concepts (doing so could disrupt an RTC).

Locking If you are using multi-engine (engine concurrency) features, you must use locking as appropriate to coordinate access to objects. See [Locking and Synchronization Functions in Preprocessors on page 147](#).

Reminder, Cache Only cache mode If you are using the Cache Only cache mode (advanced option) for one or more entities, you must consider how to handle any cache-only entities used in the functions. See [Explicitly Loading Objects into Working Memory, With the Cache Only Mode on page 285](#).

How Configured

To configure a preprocessor you associate the desired rule function with a destination in the Input Destinations tab of the BusinessEvents Archive resource (see [Configuring a BAR File for Deployment on page 360](#)):

BusinessEvents Archive (BusinessEvents Archive)

Configuration | RuleSets | **Input Destinations** | Startup/Shutdown | Object Management

ListenerSet: Custom Defaults

URI	Default	Enable	Preprocessor	Workers	Queue Size
/Channels/Channel_A.channel/Destination_1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		Shared Queue and Threads	0
/Channels/Channel_A.channel/Destination_2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/Rules/Preprocessor_1	Shared Queue and Threads	0
/Channels/Channel_A.channel/Destination_3	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/Rules/Preprocessor_2	Caller's Thread	0
/Channels/Channel_A.channel/Destination_4	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/Rules/Preprocessor_2	2	0

Apply Reset

Worker Threading and Queue Options

Event preprocessing is multi-threaded for high performance. When you configure a preprocessor you also define thread settings. This section explains the advantages of each option, specifically considering JMS destinations. Each JMS Destination creates a separate JMS Session internally, and creates a JMS thread and a dedicated JMS Connection for itself.

Caller's Thread

Uses the thread (and queue size) provided by the channel resource client (the Rendezvous or Enterprise Message Service client, for example).

- | | |
|---------------|---|
| Advantages | <ul style="list-style-type: none"> • Less context switching The messaging library's thread does the message delivery, pre-processing and the Rete operations. • Self-throttling The messaging system cannot push events faster than the rate at which it can get consumed. |
| Disadvantages | <ul style="list-style-type: none"> • To scale up, many destinations have to be created in order to create that number of caller threads. • Because each destination creates a JMS session, a session might be under used. On some operating systems, sockets and sessions could be very under-used. |

Shared Queue and Threads

Uses the BusinessEvents system-wide shared queue and threads. (In this case there is no dedicated queue for each destination.) See [Shared Queue and Threads Properties on page 367](#) for related property settings.

- | | |
|---------------|--|
| Advantages | <ul style="list-style-type: none"> • Good for Multi-core machines which can make good use of a heavily threaded set-up. |
| Disadvantages | <ul style="list-style-type: none"> • Too many threads create context switching. • Use of multiple queues makes it hard to match inbound and outbound (or RTC) rates. It can be harder to tune performance. |

Dedicated Workers Option (Specified Number of Threads)

Specifies a number of threads. BusinessEvents creates this number of new worker threads for the input destination. (In this case there is a dedicated queue for each destination). When you choose this option, you must also specify the Queue Size.

This option is similar to the Shared Queue option except that the Destination has a dedicated thread pool and set of worker threads to process messages.

The advantages and disadvantages are similar to those for [Shared Queue and Threads](#).

Locking and Synchronization Functions in Preprocessors

Because the event preprocessor is multi-threaded, multiple preprocessor threads must coordinate access to objects through appropriate use of locking. To achieve locking, use the provided synchronization functions. These functions have a format similar to the Java synchronized keyword:

```
Coherence.C_Lock(String key, long timeout, boolean LocalOnly)
```

If you want to acquire the lock only within the agent, set *LocalOnly* to true. Set the *LocalOnly* parameter to false to acquire the lock cluster wide.

The thread that calls the `C_Lock()` function is the only thread that gets blocked, until the thread that was holding the lock earlier releases.

Note that it is advisable (but not necessary) to use the corresponding `C_Unlock()` function. All the locks acquired during event processing are released at the end of the RTC cycle, that is, after data is written to cache.

The format of the unlock function is as follows:

```
Coherence.C_UnLock(String key, boolean LocalOnly)
```



The `Coherence.C_Lock()` and `Coherence.C_UnLock()` functions are available only in event preprocessors.

The example `LockExample` (in `BE_HOME/examples`) demonstrates these points, showing use of locks to prevent race conditions.

Example Suppose an event preprocessor is associated with a JMS destination. The preprocessor function uses locking and unlocking. Here is what `BusinessEvents` does:

1. Dequeues the message from the queue
2. Deserializes the message to an event
3. Calls the preprocessor function
4. Gets a lock on the Rete network
 - a. Asserts all objects to the Rete network
 - b. Retracts objects that have been deleted
5. Releases the lock on the Rete network

RuleFunction Resource Reference



RuleFunction resources enable you to write rule functions that you can use in rules, as startup and shutdown actions, and as event preprocessors.

See also *TIBCO BusinessEvents Language Reference*.

Design Panel (Rule Function Editor)

Area	Property	Description
Arguments	Type	<p>The type of the argument (if the rule function uses arguments):</p> <p>Arguments and return type can be any of these:</p> <ul style="list-style-type: none"> • Primitive, that is any of: <code>String</code>, <code>int</code>, <code>long</code>, <code>double</code>, <code>boolean</code>, <code>DateTime</code>, <code>Object</code> • Concept • Event • Specific type of Concept • Specific type of Event <p>The Object data type is used to pass parameters between standard and user-defined functions and external Java sources.</p> <p>For more details, refer to the Datatypes chapter in <i>TIBCO BusinessEvents Language Reference</i>.</p>
	Identifier	<p>A name for the argument (if the rule function uses arguments). Names must be valid identifiers</p>
Body		<p>List of statements that will be executed when the rule function executes.</p>
Function Registry		<p>The function registry is on the top right side of the rule editor. As shipped, BusinessEvents includes two catalogs in the function registry: Standard and Ontology. See Understanding and Working With Functions on page 154.</p> <p>You can add more catalogs for custom functions. See <i>TIBCO BusinessEvents Language Reference</i> for details.</p>

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Is Virtual	No	If set to yes, the rule function is a virtual rule function, for use in Decision Manager (See <i>TIBCO BusinessEvents Decision Manager</i>). Virtual rule functions have arguments but no body. The Body panel is disabled and so is the Return Type field. See Virtual Rule Functions and Decision Manager on page 129 .
Validity	No	Specifies where the rule function can be used. Possible values are as follows: Action: Indicates that this rule function can be used only in the Action block of a rule. Note: Only Action rule functions can be used as start up actions and shutdown actions. Action and Condition: Indicates that this rule function can be used in the Action and Condition blocks of a rule. Action, Condition and Query: Indicates that this rule function can be used in the Action and Condition blocks of a rule, and can also be used in the text of a query (The query language features are available only in TIBCO BusinessEvents Enterprise Suite).
Return Type	No	If the rule function returns a value, specify the Return Type, otherwise leave set to void. See the list of valid types for the Argument area, above.

Extended Properties

The Extended Properties tab is used internally in this release and is reserved for future use.

Ruleset Resource Reference



Ruleset resources enable you to group rules for deployment to different BusinessEvents archives (BARs).

You can put only one kind of resource in a Ruleset: a Rule resource.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.

Rule Resource Reference



Rule resources enable you to write rules that are triggered when entities are asserted into working memory or are modified.

Rule resources exist within Ruleset resources. See [Ruleset Resource Reference on page 151](#) for details.

See also *TIBCO BusinessEvents Language Reference*.

Design Panel (Rule Editor)

Area	Property	Description
Declaration	Term	<p>A concept or event type in the project that you will use in your rules. Types you add to the declaration define the scope of the rule.</p> <p>You can use simple events, time events, and advisory events.</p> <p>It is not necessary to add scorecards in the declaration in order to use them in the rule.</p> <p>For example, a concept definition such as <code>/Concepts/Accounts/CheckAccount</code>.</p>
	Alias	<p>A name, valid only within the scope of the rule, for an instance of the type specified in the Term field. For example: the name <code>checkaccount</code>, or the letter <code>c</code>. Use aliases to refer to instances of the types in the rule conditions and actions. You can change the alias. Aliases must be valid identifiers</p>
Conditions		<p>Each line in the Conditions area is a single expression that evaluates to <code>true</code> or <code>false</code>. Each line is joined to the others with an implicit and operator.</p> <p>For the OR operator, use a double pipe (<code> </code>) on the same line.</p> <p>BusinessEvents evaluates single lines from left to right. BusinessEvents optimizes the evaluation of multiple lines.</p>
Actions		<p>List of statements that will be executed when the rule is fired.</p>

Area	Property	Description
Function Registry		<p>The function registry is on the top right side of the rule editor. As shipped, BusinessEvents includes two catalogs in the function registry: Standard and Ontology. See Understanding and Working With Functions on page 154.</p> <p>You can add more catalogs for custom functions. See <i>TIBCO BusinessEvents Language Reference</i> for details.</p>

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Priority	No	<p>Specify a value between 1 and 10, where 1 is the highest priority and 10 is the lowest priority. Rules with a higher priority execute before rules with a lower priority.</p> <p>Only set priority where there is a reason to control the order of rules. As a general practice, it is more efficient to let the engine determine the order of rule firing.</p>

Understanding and Working With Functions

The functions registry includes various catalogs of functions provided with the product, and each catalog organizes functions into various related categories. You can use functions in rule conditions and actions and in rule function bodies.

You can also create custom functions. Custom functions appear in a custom function catalog. For information about custom functions, see *TIBCO BusinessEvents Language Reference*.

All catalogs appear in tabs on the right side of the rule editor.

Function Catalogs

This section lists the main categories in each functions catalog (but not sub-categories). See *TIBCO BusinessEvents Functions Reference* for full details.

Standard Functions

The standard function catalog include the following categories:

- **BusinessWorks** functions are used in ActiveMatrix BusinessWorks integration projects. See [Chapter 12, In-Process ActiveMatrix BusinessWorks Integration, on page 191](#)
- **Channel** functions return information about destinations, and can resume and suspend a destination.
- **Cluster functions** help with multi-engine functionality
- **Coherence** functions are for use with Cache object management.
- **Date** functions allow you to compare two DateTime values using only the date portion of the value.
- **DateTime** functions allow you to perform these date/time related tasks and more: add units of time to a dateTime, compare, retrieve, and format dates and times.
- **Engine** functions allow you to retrieve information about the engine, for example, available memory or the number of rules fired.
- **Event** functions allow you to assert, create, and send simple events and perform other event-related tasks, for example, return the default destination URI of a simple event.
- The **Exception** function enables you to create an exception.
- **File** functions provide various useful functions used when working with files.

- **Instance** functions allow you to create and delete concept instances and perform other instance-related tasks, for example, return an instance given an internal ID.
- **Math** functions allow you to perform advanced mathematical operations.
- **Number** functions allow you to perform type conversions from and to numbers and return the maximum and minimum values for a numeric type.
- **String** functions allow you to perform comparisons, searches, conversions, and other operations with strings.
- **System** functions allow you to send messages to a debug log, retrieve global variables, retrieve system properties, and write data to a file.
 - **IO** functions allow the writing and closing of specific files.
- **Temporal** functions allow you to examine and perform calculations on values stored in a property's history. For information about using temporal functions, see [Temporal Functions on page 160](#).
- **VRF** functions (that is, Virtual Rule Function functions) allow you to work with decision tables imported from Decision Manager. See [VRF Functions on page 158](#) for details.
- **XPath** functions allow you to evaluate XPath expressions.

RDBMS Functions

Database functions are provided for working with database concepts. See [Chapter 7, Working With Database Concepts, on page 107](#) for more on database concepts.

CEP Query Functions

Query functions are used with the query language for querying data in the cache. See *TIBCO BusinessEvents Language Reference* for details.

RMS Functions

RMS (Rules Management Server) functions are not visible by default. They are used only to customize the default behavior of the rules management server.

To make this function catalog visible to TIBCO Designer, ensure that `cep-rms.jar` is in your classpath. This JAR file is located in `BE_HOME/rms/lib`.

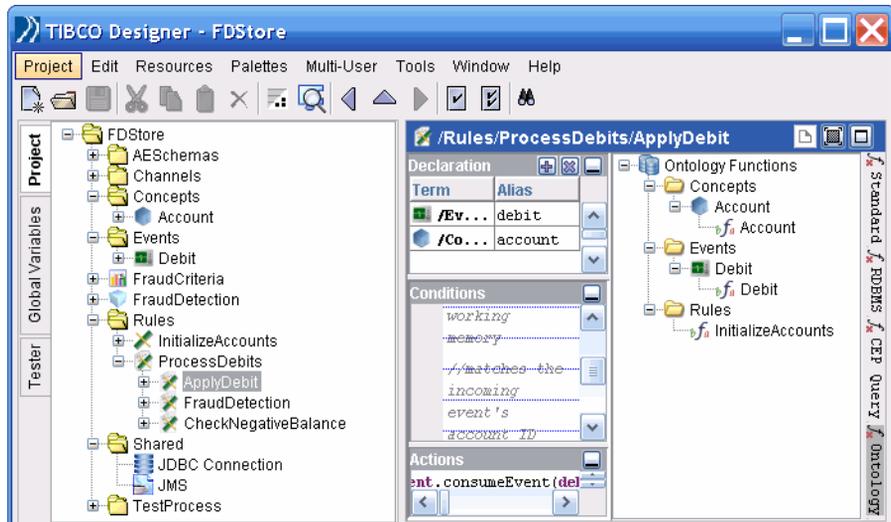
See *TIBCO BusinessEvents Decision Manager* for more details.

Ontology Functions

Ontology functions are generated by BusinessEvents based on the concepts, events, and rules in your project. There are three types of ontology functions:

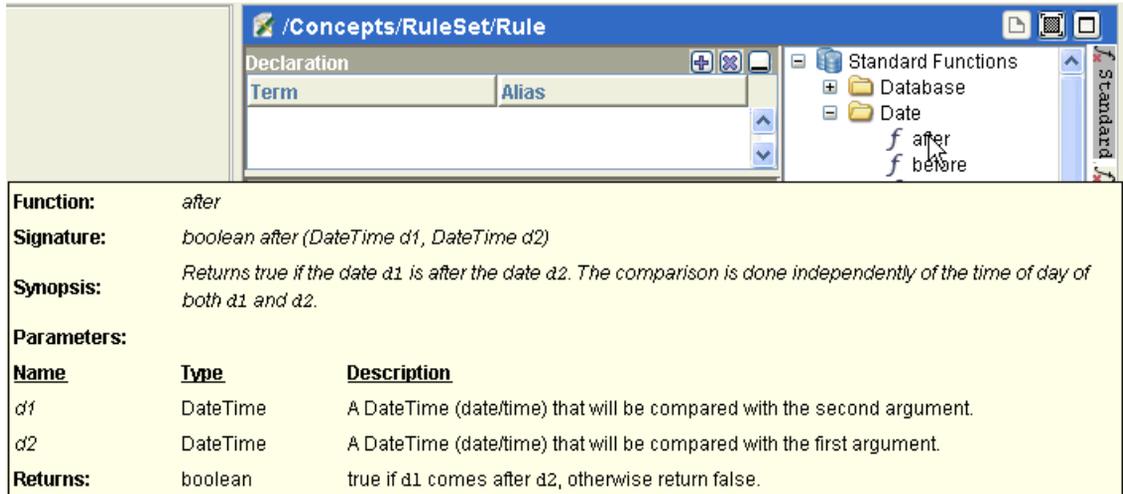
- Constructors — Allow you to create a simple event or concept instance.
- Time events — Allow you to create and schedule a time event. See [Chapter 4, Working With Time Events, on page 63](#).
- Rule functions — Allow you to invoke a rule function. See [Working With Rules and Functions on page 127](#).

The Ontology Functions area uses the same folder structure as the project (or rather, a subset of that structure). The example below shows the constructor functions on the right and the corresponding concept, event, rule function on the left.



Tool Tips

When you float the cursor over a function in the registry, BusinessEvents displays the description and syntax in a tool tip next to the cursor:



The screenshot shows a window titled "/Concepts/RuleSet/Rule" with a "Declaration" table and a "Standard Functions" tree. The tool tip for the "after" function is displayed below the tree.

Declaration	
Term	Alias

Function: *after*

Signature: *boolean after (DateTime d1, DateTime d2)*

Synopsis: *Returns true if the date d1 is after the date d2. The comparison is done independently of the time of day of both d1 and d2.*

Parameters:

Name	Type	Description
d1	DateTime	A DateTime (date/time) that will be compared with the second argument.
d2	DateTime	A DateTime (date/time) that will be compared with the first argument.

Returns: *boolean* true if d1 comes after d2, otherwise return false.



You can create your own tool tips for custom functions. See the *TIBCO BusinessEvents Language Reference* for detailed information.

Decorations Indicating Where Functions can be Called

Functions are decorated with small letters that indicate where you can call the function.

Action-Only Functions

These functions are only for use in actions. Some of these functions have side effects, for example they can change values. Other functions are limited to actions for other reasons. These action-only functions are identified by a small *a* at the bottom right of the *f*. For example: *f_a setDateTime* .

Mapper Functions

Functions that bring up the XSLT mapper and XPath builder are identified by a small *m* at the upper left of the *f*, for example: *^mf evalAsString* . Some functions are both action functions and mapper functions; they are identified with both the *m* and the *a*, for example: *^mf_a createInstance* . See [Using XSLT Mapper Functions on page 139](#).

Functions That Can Be Used in Decision Manager

Functions that can be used in Decision Manager are marked with a green *b*, *bf* `lastCheckpointDuration` for example. The *b* stands for BUI or business user interface, which is what Decision Manager provides for BusinessEvents.

Functions That Can Be Used in Queries

Functions that can be used in queries are marked with a blue *q*. Such functions are often also available for use in the BUI and so are also marked with a *b*, *bfq* `getMonth`, for example. You can call such functions in a query string. See *TIBCO BusinessEvents Language Reference* for details on use of the query language, available in TIBCO BusinessEvents Enterprise Suite only.

VRF Functions

The VRF category of functions (within the Standard Functions) provide flexibility when you are working with virtual rule function implementations. The functions are available in the Standard catalog and also in Decision Manager.

Virtual rule functions are used with Decision Manager. See [Decision Manager and Rules Management Server on page 8](#) and [Virtual Rule Functions and Decision Manager on page 129](#) for more information.

These functions enable you to do the following:

- Partition decision tables into smaller, more manageable decision tables.
- Define multiple decision tables for a given virtual rule function.
- Choose which decision tables to use, on invocation of a virtual rule function.

The functions are as follows:

```
getVRFImplByName()
getVRFImplNames()
getVRFImpls()
invokeAllVRFImpls()
invokeVRFImpl()
invokeVRFImplByName()
invokeVRFImpls()
```

Common arguments used in the above functions are described in [Table 14, Common Arguments for VRF Functions, on page 159](#).

VRF Function Arguments

The VRF functions use various subsets of the following common arguments:

Table 14 Common Arguments for VRF Functions

Name	Type	Notes
<code>vrfName</code>	String	The universal resource identifier (URI) for the virtual rule function. This is typically the full path to the virtual rule function within the project directory. For example, in the <code>CreditCardApplication</code> example, the virtual rule function <code>Person_VirtualRuleFunction()</code> has the following URI: <code>/Virtual_RF/Person_VirtualRuleFunction</code>
<code>vrfImpl</code>	Object	An object representing a virtual rule function implementation. This argument is required when invoking specific virtual rule function implementations.
<code>implName</code>	String	The name of a decision table (also known as a virtual rule function implementation). For example, in the <code>CreditCardApplication</code> example, the virtual rule function <code>BankUser_VirtualRuleFunction</code> has an implementation (decision table) called <code>bankUser</code> . The <code>implName</code> argument is used to retrieve a corresponding implementation object, or to execute an implementation.
<code>args</code>	Object array	The arguments to be passed to one or more virtual rule function implementations on invocation. These objects consist of the concepts, events, scorecards, and so on. that are needed by the implementation or implementations. For example, the <code>processApplication</code> implementation in the <code>CreditCardApplication</code> example project requires concepts of type <code>Application</code> , <code>BankUser</code> , and <code>CreditCardApplication</code> to be passed as arguments. In order to invoke the <code>processApplication</code> implementation, an instance of each concept type must be passed in the <code>args</code> array.
<code>returnValues</code>	Object array	This argument is used only for the <code>invokeVRFImpls</code> function. When invoking multiple implementations, the return value of each implementation is stored in this array. The array will contain a null entry for each implementation that does not return a value.

Temporal Functions

The set of standard functions that come with TIBCO BusinessEvents includes functions that allow you to perform calculations on numeric values sampled over time. These functions are called temporal functions and they work exclusively with concept properties that store numeric values. Temporal functions make use of the history ring buffer to sample a property's values over time.

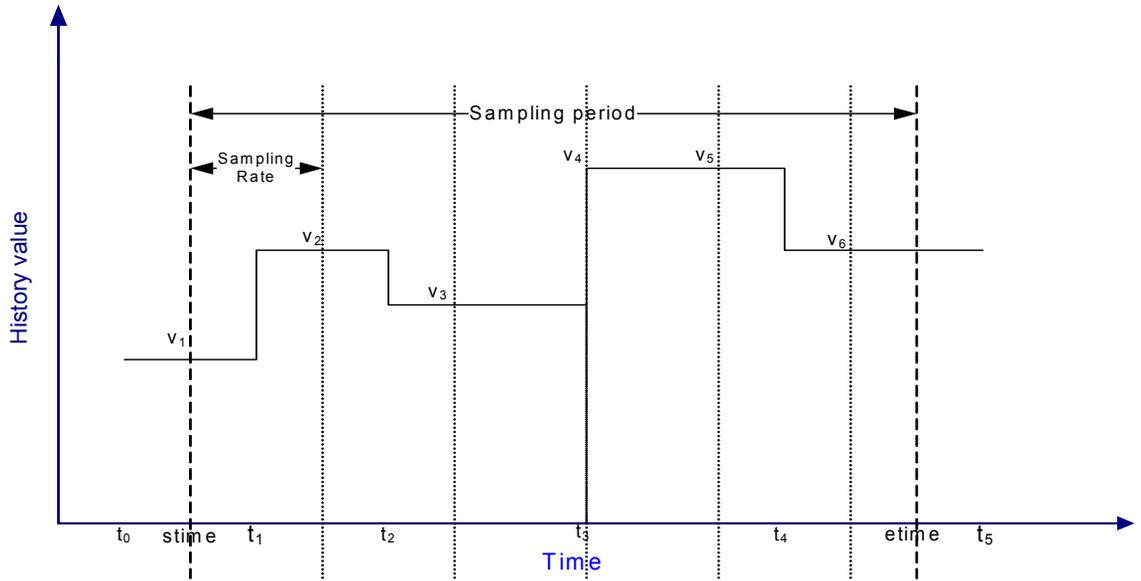


Use of a temporal function with a concept that has a history size of 0 may cause a runtime exception.

All temporal functions include these parameters, illustrated in [Figure 8](#):

- **property** — The name of the property for which you want to sample values.
- **stime** — The time from which you want to begin sampling values (the start time) measured in milliseconds since 00:00:00 UTC on January 1, 1970.
- **etime** — The time at which you want to stop sampling values (the end time) measured in milliseconds since 00:00:00 UTC on January 1, 1970.
- **sample_rate** — The number of milliseconds between samples.
- **bound_by_stime** — A flag indicating whether the start-time is flexible:
 - **True** indicates that if the start time you provide is earlier than the time stamp for the oldest available value, you want to perform the calculation starting with the oldest available value.
 - **False** indicates that if the start time you provide is earlier than the time stamp for the oldest available value, you want to abort the calculation.

Figure 8 Temporal Functions Parameters



This chapter explains how to use the state modeler features. The state modeler features are available only in the Enterprise Suite of TIBCO BusinessEvents.

Topics

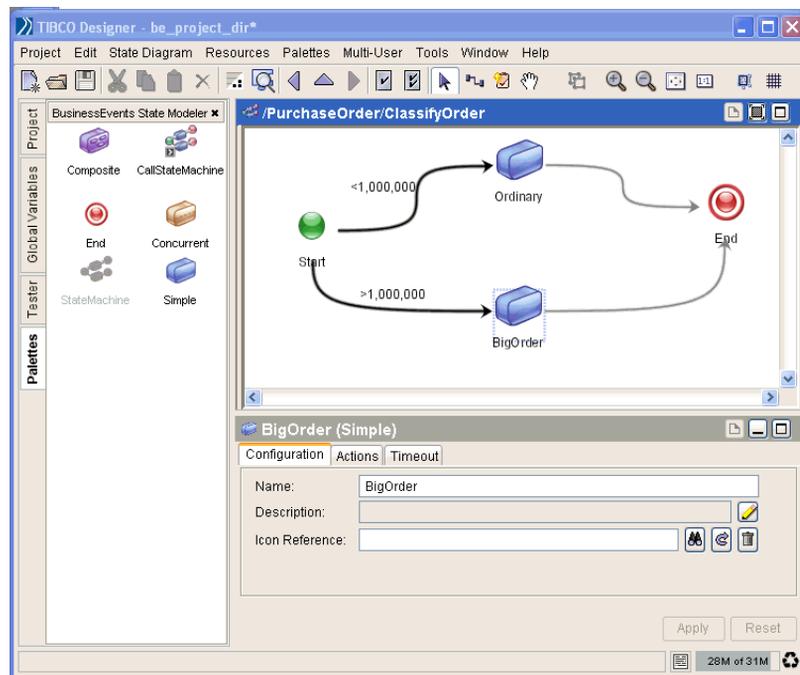
- [State Modeler Overview, page 164](#)
- [State Machine Resource Reference, page 166](#)
- [CallStateMachine Resource Reference, page 168](#)
- [State Machine States Reference, page 170](#)
- [Transitions, page 173](#)
- [Entry and Exit Actions, page 177](#)
- [BusinessEvents State Modeler Palette, page 178](#)
- [Exporting a State Model to SVG Format, page 180](#)

State Modeler Overview

The State Modeler is a UML-compliant application that allows you to model the life cycle of an instance — that is, for each instance of a given concept, you can define which states it will pass through and how it will transition from state to state based on rules that apply.

Each state model begins with a *start* state and ends with one or more *end* states. Between the start and end states may be *simple*, *composite*, and *concurrent* states connected by *transitions*. Figure 9 displays a simple state model that classifies orders based on their dollar values:

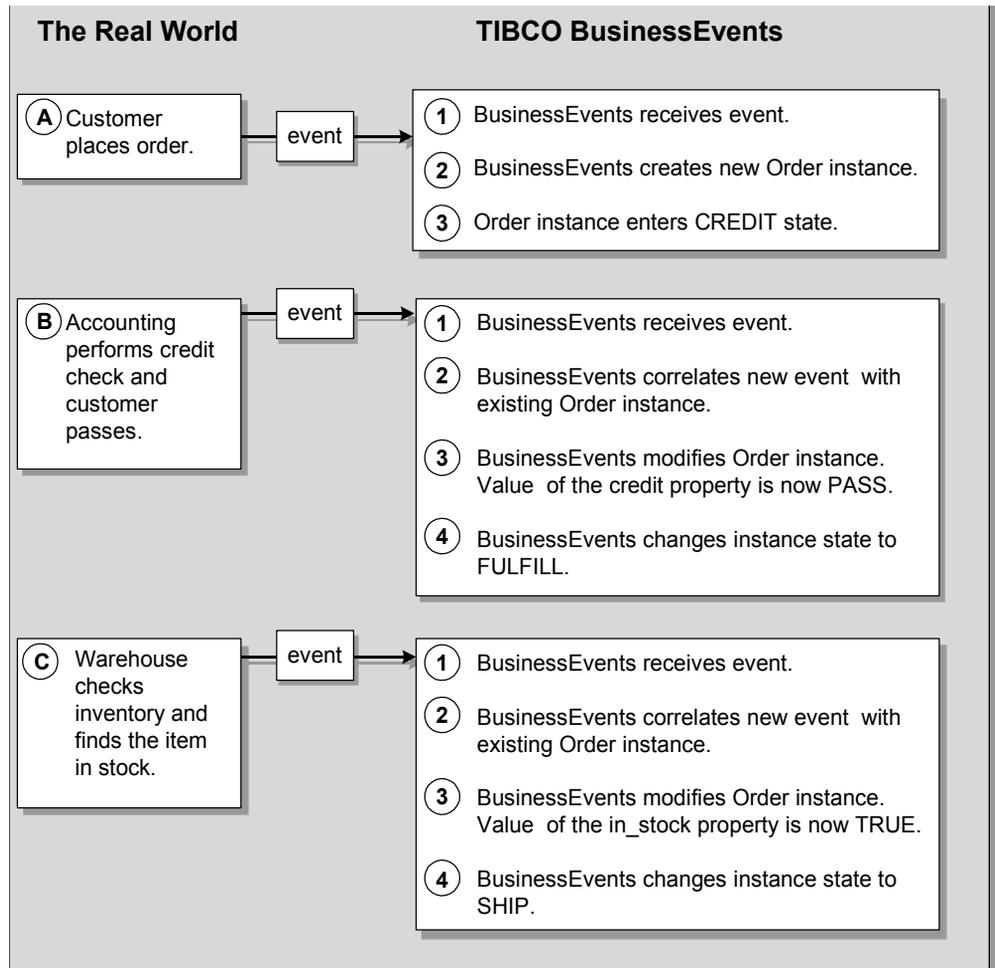
Figure 9 Simple State Model



For a slightly more complex example, consider another Order concept. When a customer places an order, BusinessEvents receives a simple event and creates a new instance of the Order concept. Through the life of the Order instance, it will pass through a credit-check state, an inventory-check state, a fulfillment state, and so on until the customer has received and paid for the order. As these order

activities happen in the real world, BusinessEvents receives new events, correlates the events to the existing instance, modifies one or more values within the instance, and changes the state of the instance. Figure 10 illustrates this order process:

Figure 10 Modeling an Order Process



BusinessEvents ships with a simple state model example, which you can open and examine. It also includes a document with instructions for creating the example yourself. This will help to familiarize you with the BusinessEvents State Modeler. This and other examples are stored in the `BE_HOME/examples` directory.

At runtime, the transitions are used in the rule session as state machine rules, and the concept states are stored as properties of the concept.

State Machine Resource Reference



To model the life cycle of an instance, use a state machine resource. Within a state machine resource you can configure the states and transitions. A state machine resource always exists within a concept.

Usage Notes: To access a state machine resource, double-click on the relevant concept resource. The BusinessEvents State Modeler palette appears on the **Palettes** tab and displays the state machine resource, shown here in the left margin.

This section describes the attributes of the resource. Later sections describe states and transitions.

Name

As with other resources, the name of the state machine resource must conform to Java rules for identifiers.



Length limitation The state machine name and path together must not exceed 255 characters.

Main State Machine

At design time, one concept may have multiple state machine definitions. At runtime, BusinessEvents creates an instance of the concept's main state machine. The main state machine usually makes calls to the other state machines associated with the concept. (See [CallStateMachine Resource Reference on page 168.](#))

Each concept is allowed at most one main state machine. A concept can inherit its main state machine. At runtime, BusinessEvents searches for a main state machine starting with the concept instance. If it does not find one, it moves up the inheritance chain until it finds a main state machine. It creates an instance of the first main state machine it locates.

If BusinessEvents cannot locate a main state machine in the concept instance or any concepts higher in the inheritance chain, it does not create an instance of any state machine for this concept.

Use the **Main State Machine** checkbox on the **Configuration** tab to designate the main state machine.

Timeouts

Both state machines and states have timeouts. For the state machine, the timeout value specifies how long the state machine should wait before it completes. The Timeout Expression field allows you to define the length of the timeout as an expression. If the state machine's timeout value is less than the sum of all of the states' timeouts, the state machine's timeout takes precedence.

To prevent the state machine from timing out, set the timeout value to 0.



BusinessEvents supports only positive integers or zero as timeout values. It does not support negative numbers or decimal values.

Functions

The State Modeler includes its own set of functions, which are located in the Standard function registry of the Rule Editor under `Instance > StateMachine`. For more information about the rule editor and general use of BusinessEvents functions, see [Chapter 9, Working With Rules and Functions, on page 127](#). The rule editor itself provides tool tip documentation. You may also want to refer to *TIBCO BusinessEvents Language Reference*.

Inheritance

A concept's state machines can call any state machine that belongs to an ancestor of the concept. The state machine of a concept cannot call the state machine of a concept that is lower in the inheritance chain.

CallStateMachine Resource Reference



The CallStateMachine resource allows you to call any state machine that is at the same level or higher in the inheritance chain. CallStateMachine resources exist only within state machines.



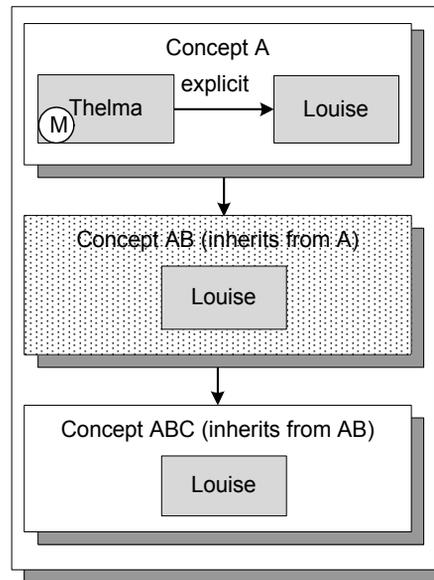
The state machine of a concept cannot call the state machine of a concept that is lower in the inheritance chain, and you cannot call a state machine recursively. That is, you cannot call a state machine from within itself either directly or indirectly.

Call Explicitly

At design time, you may have concepts in the same inheritance chain that include same-named state machines. The Call Explicitly checkbox allows you to select the state machine you need.

With the Call Explicitly checkbox deselected, BusinessEvents runs the state machine from the concept instance that contains the state machine which is making the call, if it exists. If the concept instance does not include a state machine with the specified name, it looks for the state machine in the concept that is directly above the concept instance in the inheritance chain. It continues its search up the inheritance chain, creating an instance of the first state machine it finds with the specified name. With the **Call Explicitly** checkbox deselected, the call is similar to a Java or C++ virtual function.

Figure 11 Explicit Calls



In [Figure 11](#), Concept AB represents the concept instance at runtime. Note that it has no main state machine. For each instance of AB, BusinessEvents creates an instance of Thelma, which is a main state machine. Because Thelma explicitly calls Louise from Concept A, BusinessEvents runs A.Louise. If the call to Louise had not been explicit, BusinessEvents would have created an instance of AB.Louise because AB is the concept instance. Under no circumstances would BusinessEvents create an instance of ABC.Louise because it is below the concept instance in the inheritance chain.

Configuring

To call a state machine, drag a CallStateMachine resource into an open state machine resource in the design panel. Then, in the **State Machine** field of the **Configuration** tab, browse for and select the desired state machine.

State Machine States Reference

A state model can include up to five types of states, which are described below.



Most of the state types described below only allow exclusive-or (XOR) transitions, which go to one state only. To allow a state to go from one state to multiple states, use a *concurrent* state.

This section also provides detailed information about state timeouts.

Start and End States



Start



End

Within each state machine is a start state and an end state. Each start state has an exit action, but no entry action. Each end state includes an entry action but no exit action.

Simple State



Simple

A simple state includes an entry action and an exit action.

Composite State



Composite

Composite states are like nested folders: they contain other states. For example, consider a state model for an order instance: the order instance may need to travel through complex credit check and fulfillment processes. You can group the complex credit-check process in one composite state and the order-fulfillment process in another composite state.

Use a composite state if you want to ensure that all states within a particular group of states process successfully before continuing. If one member state within a composite state fails, the composite state also fails.

Composite states can contain simple states, other composite states, and concurrent states.

For more information about composite states, see [Complex Transitions on page 173](#).

Concurrent State



A concurrent state allows multiple state flows to operate at one time. A concurrent state has multiple processing lanes, called regions. A state machine instance cannot exit a concurrent state until all its regions have finished processing, unless a timeout occurs. A region of a concurrent state can contain a composite state.

For more information about concurrent states, see [Complex Transitions on page 173](#).

Configuring a Start State



To configure the start state, perform these steps:

1. Select the **Start** state icon in the design panel.
2. Select the **Actions** tab.
3. Click the rules editor icon () to the right of the **Exit Action** field.
4. Create an exit action in the **Actions** panel.
5. Click **Apply** to save your work.

State Timeouts

Except start and end states, each state within a state machine has its own timeout setting and timeout action. Use the **Timeout** tab to specify the wait time, the timeout action, and the timeout state.

Timeout State

When a timeout occurs, in addition to the timeout action, BusinessEvents can do one of three things:

- Stay at the current state.
- Go to the next state.
- If there are multiple possible next states, continue in one of these ways:
 - Specify the state to which BusinessEvents goes next.
 - Go to all of the possible next states and wait until a simple event arrives that establishes the next state.

To configure the timeout behavior, select one of the following options for the **timeout State** field:

- **Specified:** — To specify a timeout state, select this option and browse for the desired state. You can specify any top-level end state and any state that has the same parent as the state you are configuring.
- **Current** — Select this option to remain at the current state.
- **All** — Select this option to have **BusinessEvents** prepare to go to any possible next state. When a simple event arrives that clarifies the correct next state, BusinessEvents goes to that state.



For regions within a concurrent state, select the **Current** state option.

If a state does not receive the expected event within the time period specified, the state times out and BusinessEvents executes the timeout rule.

The state machine as a whole has a time out that limits the time that BusinessEvents spends in a state machine instance. The state machine timeout takes precedence over the timeout of the current state.

Similarly, the timeout of composite states takes precedence over the timeouts of their member states.



Explicitly Delete Concept Instances when a State Machine Ends Unlike events, concept instances are not automatically deleted from working memory when a state machine ends. You must explicitly delete concept instances in your rules, using `Instance.deleteInstance()`.

Transitions

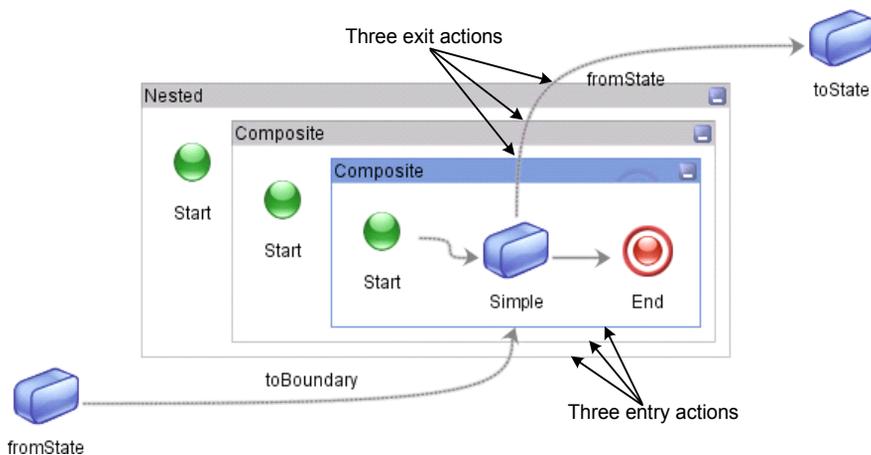
The states in a state model are connected by transitions. Both state and transition resources are containers for rules, which control how each instance changes states and property values.

Self Transitions

Typically, transitions take an instance from one state to another state. Self transitions connect a state to itself, thereby reprocessing the rule set until conditions are met to transition to a different state. Each time the instance loops back to the state, it triggers the entry action, and each time it leaves the state, it triggers the exit action.

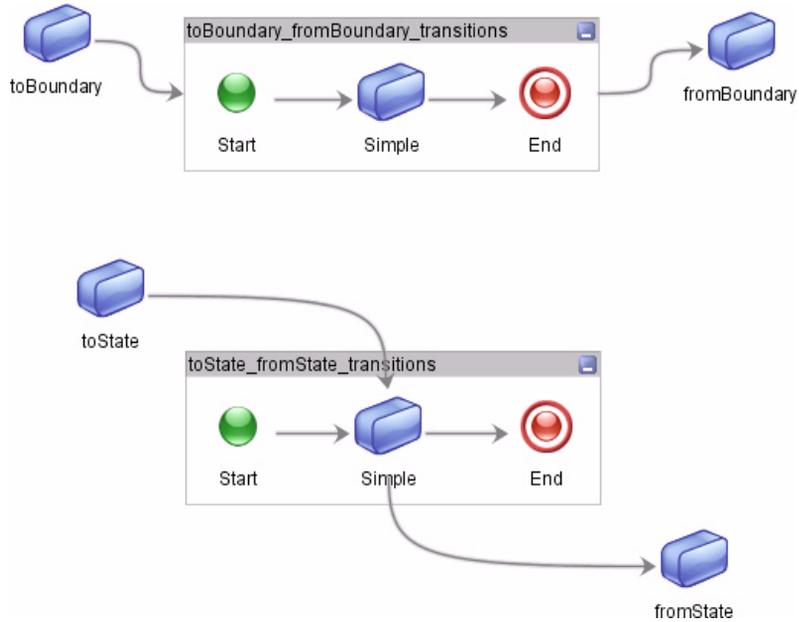
Complex Transitions

This subsection presents BusinessEvents behavior when transitioning into and out of composite and concurrent states. This discussion mentions entry and exit actions, which you can learn about in [Entry and Exit Actions on page 177](#).



Composite State Transitions

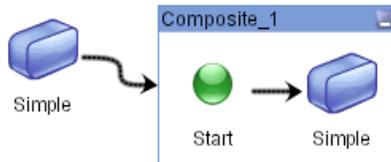
Composite states allow to-boundary transitions, to-state transitions, from-boundary transitions, and from-state transitions, as shown:



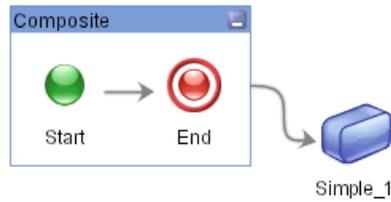
When a transition enters or exits a composite state, it triggers the composite state's entry or exit action. In the case of nested composite states, each time a transition passes the boundary of one of the composite states, it triggers the assigned action.

Composite and Concurrent State Transitions

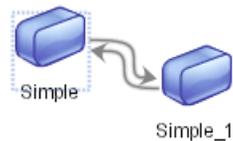
To enter a composite or concurrent state from the boundary, its start state must have a transition to the next state:



Similarly, to exit a composite state from the boundary, its end state must have a transition coming from the previous state:

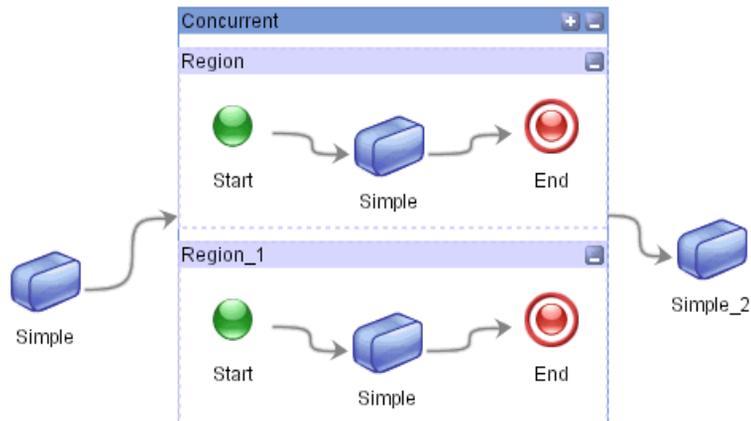


Loopbacks are allowed:



Transition Rules for Concurrent States

You can only transition into and out of a concurrent state through the boundaries. You cannot transition into or out of a state directly as you can with composite states.



As with nested composite states, to-boundary and from-boundary transitions trigger an entry or exit action each time they enter or exit a nested concurrent state.

Configuring a Transition

To configure a transition, perform these steps:

1. With the state engine resource open, click the transitions button: .
2. Click the state from which you want to transition.
3. Click the state to which you want to transition.
4. Click the select button: .
5. Select the transition.
6. Type a label for the transition if desired. Transition labels do not have to follow Java naming requirements; therefore you can, for example, label the transition with the condition: `a>b`.
7. Select the **No Condition** checkbox if you want to create a transition without a condition (that is, a *lambda* transition).
8. Click the rules editor icon () to the right of the **Rules** field. The rule editor appears with both the **Conditions** and **Actions** panels. See [Chapter 9, Working With Rules and Functions, on page 127](#), for more information about working with the rule editor.

Entry and Exit Actions

Each state has both an entry action and an exit action, and may include an internal transition, which causes BusinessEvents to re-evaluate the current values one or more times before transitioning to the next state. As you configure each state, you can use the rule editor to define the rules that are fired as the instance enters and exits each state.

As an instance enters a state, it triggers a rule defining an entry action. While it is within a state, it is subjected to one or more rules that determine if, when, and how it will exit the state. As it exits a state, it triggers a rule defining an exit action. As it transitions into the next state, another rule determines its fate. Each transition includes rules.

BusinessEvents State Modeler Palette

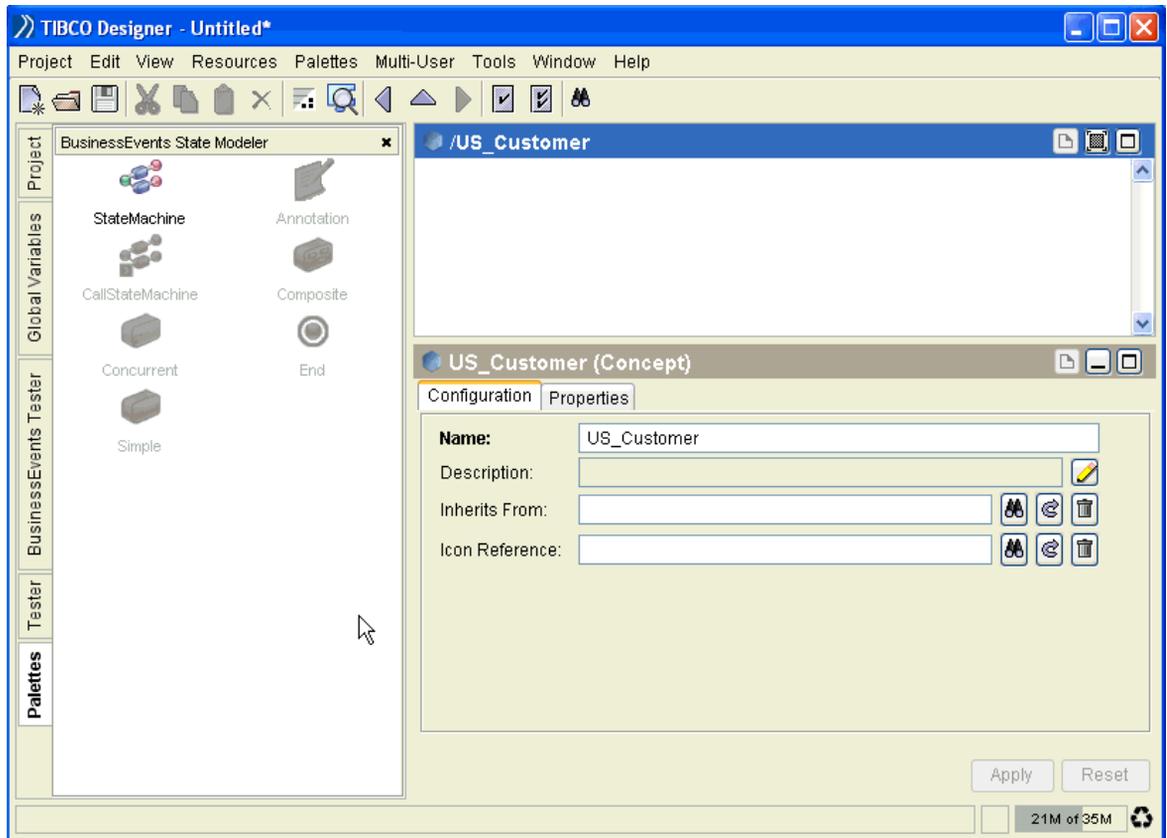
The Enterprise Suite of TIBCO BusinessEvents includes the BusinessEvents State Modeler palette. All of the State Modeler resources are located in the BusinessEvents State Modeler palette. This palette is only accessible when a concept resource is open in the design panel, and it is the only palette accessible under this condition.

To locate the BusinessEvents State Modeler Palette, perform these steps:

1. Locate the concept for which you want to build a state engine.
2. In the design panel, double-click the concept.
3. Select the **Palettes** tab. The BusinessEvents State Modeler palette appears.

[Figure 12, BusinessEvents State Modeler Palette](#), displays a screen capture of the palette, design panel, and configuration view.

Figure 12 BusinessEvents State Modeler Palette



Exporting a State Model to SVG Format

You can export a state model to SVG (scalable vector graphics) format for viewing within an SVG application. To export a state model to SVG format, perform these steps:

1. Double-click the state model resource to open it.
2. Select **State Diagram>Save SVG** or click the save SVG button . A save dialog appears.
3. Name the file and save it in a location of your choice.

Out-of-Process ActiveMatrix BusinessWorks Integration

This chapter explains how to use the ActiveMatrix BusinessWorks activities to communicate with ActiveMatrix BusinessWorks engines.

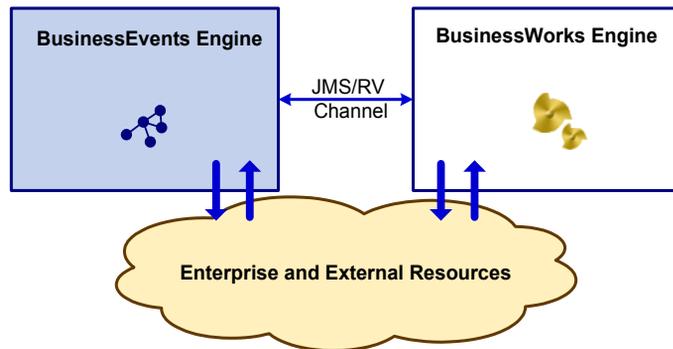
Topics

- *Understanding Out-of-Process ActiveMatrix BusinessWorks Integration, page 182*
- *Working With the BusinessEvents Activities, page 184*
- *Receive Event Resource Reference, page 185*
- *Send Event Resource Reference, page 187*
- *Wait for Event Resource Reference, page 188*

Understanding Out-of-Process ActiveMatrix BusinessWorks Integration

The activities documented in this chapter enable you to send and receive BusinessEvents events in an ActiveMatrix BusinessWorks engine. Both engines run in separate JVMs as shown below:

Figure 13 Out-of-Process Integration with ActiveMatrix BusinessWorks



The engines communicate through BusinessEvents channels. Both engines can communicate with enterprise resources.

For an in-process method of integrating BusinessEvents and ActiveMatrix BusinessWorks, see [Chapter 12, In-Process ActiveMatrix BusinessWorks Integration, on page 191](#).

Designtime Uses

You can use these activities to test your projects at design time, simulating messages that BusinessEvents would send to and receive from enterprise and external resources. An example of this kind of use is given in *TIBCO BusinessEvents Getting Started*. It is also extensively used to exercise the examples provided with BusinessEvents, in the `BE_HOME/Examples` directory.

Runtime Uses

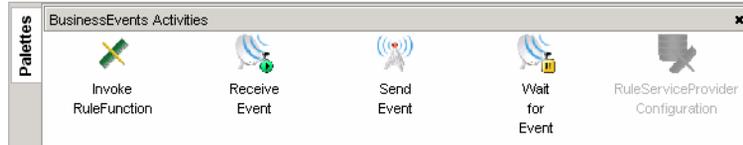
Runtime uses depend on your business needs. However, note the following restriction.



Production use of TIBCO ActiveMatrix BusinessWorks requires a fully licensed version of ActiveMatrix BusinessWorks (that is, a production license) The ActiveMatrix BusinessWorks software provided with BusinessEvents is for design-time and testing purposes only and is not for use at runtime.

Working With the BusinessEvents Activities

This guide assumes you know how to work with ActiveMatrix BusinessWorks processes. See *TIBCO BusinessWorks Process Design Guide* if you need detailed guidance.



TIBCO BusinessEvents includes the BusinessEvents Activities palette to allow you to connect BusinessEvents with TIBCO ActiveMatrix BusinessWorks. You can only access the BusinessEvents Activities palette through an ActiveMatrix BusinessWorks process definition.

The BusinessEvents Activities palette includes three activities used for out-of-process integration with ActiveMatrix BusinessWorks:

- [Receive Event Resource Reference](#)
- [Send Event Resource Reference](#)
- Wait for Event

Note that the palette also includes one activity and one resource for use with the in-process integration with ActiveMatrix BusinessWorks: Invoke RuleFunction Activity and RuleServiceProvider Configuration. See [Chapter 12, In-Process ActiveMatrix BusinessWorks Integration, on page 191](#) for details on setting up in-process integration projects.

To Work with the BusinessEvents Activities Palette

1. Open the folder in which you want to add a process definition. Right-click in the design panel, and select **Add Resource > Process > Process Definition**.
2. Double-click the Process Definition resource to open it. You see the Start and End activities.
3. Right-click in the design panel, and select **Add Resource > BusinessEvents Activities**. Select activities as needed to configure the process. See the following for guidelines:
 - [Receive Event Resource Reference on page 185](#)
 - [Send Event Resource Reference on page 187](#)
 - [Wait for Event Resource Reference on page 188](#)
4. Click **Apply** and save the project.

Receive Event Resource Reference



This process starter starts the process when the specified simple event arrives from its default destination channel, or from a destination specified in the configuration tab.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Event Reference	No	The simple event you want to receive.
Custom Destination	No	If you do not want to use the default destination for the specified simple event, identify an alternate destination in this field.

Misc

The Misc tab has the following fields.

Field	Description
Sequencing Key	This field can contain an XPath expression that specifies which processes should run in order. Process instances with sequencing keys that evaluate to the same value will be executed sequentially in the order the process instance was created. See <i>TIBCO BusinessWorks Process Design Guide</i> for more information about controlling the execution order of process instances and about XPath expressions.
Custom ID	This field can contain an XPath expression that specifies a custom ID for the process instance. This ID is displayed in the View Service dialog of TIBCO Administrator, and it is also available in the <code>\$_processContext</code> process variable.

Output

The Output tab has the following fields.

Output Item	Data Type	Description
BEReceiveEventOutput	complex	Expand to show the name, properties, and attributes of the event type selected in the Configuration tab.

Send Event Resource Reference



Data from the ActiveMatrix BusinessWorks process context is used to send an event of a specified type to its default destination or to a destination specified in the configuration tab.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Event Reference	No	The simple event you want to send.
Custom Destination	No	If you do not want to use the default destination for the specified simple event, identify an alternate destination in this field.

Input

The Input tab has the following fields.

Input Item	Data Type	Description
BESendEvent Input	complex	Expand to show the name, properties, and attributes of the event type selected in the Configuration tab.

Wait for Event Resource Reference



This activity listens to the default destination of a specified event type or to a destination specified in the configuration tab, and waits for a matched simple event.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Event Reference	No	The simple event you want to wait for.
Custom Destination	No	If you do not want to use the default destination for the specified simple event, identify an alternate destination in this field.

Event

The Event tab has the following fields.

Input Item	Global Var?	Description
Candidate Event Key	No	<p>Use to filter events. Only events whose candidate event key matches the key provided will trigger the activity.</p> <p>For example you can use the event's extID as the event key. You can also use any of the event's properties as a matching candidate event key. There can multiple keys, but only the first one that matches is considered.</p> <p>As an example use of this property, suppose you want "Wait for Event" to fire only for the 10th occurrence of an event. You would set the Candidate Event Key value to 10. In the input tab, you would map the input item key to a global variable that is incremented by 1 each time the event occurs.</p> <p>Refer to ActiveMatrix BusinessWorks documentation on JMS/RV Wait for Activities for more information.</p>
Event Timeout	No	<p>If the event is received before the process reaches this activity, the event waits for this number of milliseconds. If the timeout period ends before the process reaches this activity, the event is discarded.</p>

Input

The Input tab has the following fields.

Input Item	Data Type	Description
key	String	Candidate Event Key value (specified on the Event tab)
processTimeout	Integer	Event Timeout from Event tab

Output

The Output tab has the following fields.

Input Item	Data Type	Description
BEReceiveEventOutput	complex	Expand to show the name, properties, and attributes of the event type selected in the Configuration tab.

In-Process ActiveMatrix BusinessWorks Integration

This chapter explains how you can integrate ActiveMatrix BusinessWorks and BusinessEvents functionality in one JVM.



Production use of TIBCO ActiveMatrix BusinessWorks requires a fully licensed version of ActiveMatrix BusinessWorks The ActiveMatrix BusinessWorks software provided with BusinessEvents is for design-time and testing purposes only and is not for use at runtime. A production license is required for runtime uses.

Topics

- [Understanding In-Process TIBCO ActiveMatrix BusinessWorks Integration, page 192](#)
- [Design Considerations, page 195](#)
- [Configuring the Environment for BusinessEvents Containers, page 198](#)
- [Invoking a BusinessEvents Rule Function From ActiveMatrix BusinessWorks, page 204](#)
- [Specifying the Rule Service Provider for ActiveMatrix BusinessWorks Containers, page 206](#)
- [RuleServiceProvider Configuration Resource Reference, page 207](#)
- [Working With Invoke RuleFunction Activities, page 208](#)
- [Invoke RuleFunction Resource Reference, page 210](#)
- [Working With the BusinessWorks Functions, page 213](#)
- [Deploying the Integration Project, page 219](#)

Understanding In-Process TIBCO ActiveMatrix BusinessWorks Integration

Integration between BusinessEvents and ActiveMatrix BusinessWorks enables each product to take advantage of the strengths of the other product. ActiveMatrix BusinessWorks can use BusinessEvents as a light-weight rules engine, for example, and BusinessEvents can use transports available in ActiveMatrix BusinessWorks.



To see integration features in an example project, open the project in this directory: `BE_HOME/examples/FraudDetectionBEBW/`.

With the in-process method of integration, the integrated project runs in a single container, that is, a single JVM. The JVM can run as a BusinessEvents engine, or as an ActiveMatrix BusinessWorks engine.

The engine you start first functions as the container.

When BusinessEvents is the container, the ActiveMatrix BusinessWorks engine can communicate with resources outside of the BusinessEvents container. However, when ActiveMatrix BusinessWorks is the container, the BusinessEvents engine cannot communicate with resources outside of the ActiveMatrix BusinessWorks container.

Which engine you run as the container depends on what you want to achieve. Choice of container affects project design and runtime behavior, as explained in [Design Considerations on page 195](#).

Out-of-process model

You can also use the out-of-process integration model. It enables ActiveMatrix BusinessWorks to interact with BusinessEvents using Send Event and Receive Event activities, as explained in [Chapter 11, Out-of-Process ActiveMatrix BusinessWorks Integration, on page 181](#).

Managing a single container, however, simplifies maintenance and enables horizontal scaling. It involves less management overhead than that required by the out-of-process model of integration.

Feature Summary

ActiveMatrix BusinessWorks Features

To enable an ActiveMatrix BusinessWorks process to call a BusinessEvents rule function, you configure the following resources:

Invoke RuleFunction Activity Provided in the BusinessEvents Activities palette, an Invoke RuleFunction activity invokes a BusinessEvents rule function in a specified agent instance (rule session) and passes a concept or event or BusinessEvents primitive (except Object) to it. The Invoke RuleFunction activity can be combined with any process starter. Execution is synchronous. For ActiveMatrix BusinessWorks containers only, you also specify a RuleServiceProvider Configuration.

Note that BusinessEvents generates its own threads to execute the rule function that the ActiveMatrix BusinessWorks process calls through the Invoke RuleFunction activity. Then the ActiveMatrix BusinessWorks thread is released and the process is set to a pending state. When the rule function returns, the ActiveMatrix BusinessWorks process resumes its Ready state.

RuleServiceProvider Configuration Required for ActiveMatrix BusinessWorks containers only. Provided in the BusinessEvents Activities palette, a RuleServiceProvider Configuration resource is used to identify the location of the BusinessEvents application at runtime.

BusinessWorks Functions

The integration model provides a BusinessWorks category of functions. the functions are as follows:

- **BusinessWorks.invokeProcess():** Invokes an ActiveMatrix BusinessWorks process in synchronous mode and waits for completion of the process before returning to the rule or rule function. Starts the process engine if it is not already started. Returns an event, or null. Generates an advisory event if it times out.
- **BusinessWorks.startProcess():** Invokes an ActiveMatrix BusinessWorks process in asynchronous mode and returns the process ID (job ID). Upon completion, the ActiveMatrix BusinessWorks process passes an event to the rule function specified in an argument of `startProcess()`. Starts the process engine if it is not already started.
- **BusinessWorks.cancelProcess():** Cancels the specified ActiveMatrix BusinessWorks process. Useful for canceling a long running job. (Cancellation is not guaranteed because the process may complete before receiving the cancellation command.)

- **BusinessWorks.init():** For use with BusinessEvents containers only (but harmless if present and not needed). Initializes the ActiveMatrix BusinessWorks engine. Optional.
- **BusinessWorks.shutdown():** Shuts down the ActiveMatrix BusinessWorks process engine. Optional. You can use this function to shut down the ActiveMatrix BusinessWorks engine when it is no longer needed.

Design Considerations

This section presents some considerations to keep in mind when you are designing your integration project.

Integration Scope

You can only use BusinessEvents rule sessions and rule functions and ActiveMatrix BusinessWorks processes that are configured in the TIBCO Designer integration project, and they must also be included in the relevant EAR files deployed for the integration project.

You can use all the integration features regardless of which product's engine is used as the container, ActiveMatrix BusinessWorks or BusinessEvents, within the guidelines presented in this chapter.

Avoiding Threading Issues

It is possible to configure a complex execution path in your project, for example, one in which BusinessEvents invokes an ActiveMatrix BusinessWorks process, which in turn invokes a BusinessEvents rule function, and so on.

The `BusinessWorks.InvokeProcess()` rule function and the `InvokeRuleFunction` activity both operate synchronously. Both attempt to acquire a lock on BusinessEvents working memory.

Take care to avoid threading issues. Check your execution path carefully to ensure that there are no threading issues leading to deadlock. In general, make sure that no action in the entire execution path attempts to use the same working memory that is already locked.

For example, if you execute `BusinessWorks.InvokeProcess()` function in BusinessEvents, then in the ActiveMatrix BusinessWorks process it calls, you cannot use an `InvokeRuleFunction` activity to invoke a rule function in the same rule session. You could invoke a rule function running in a different rule session, however.

Use of Global Variables and Environment Variables

All of the variables and properties set in TIBCO Designer and TIBCO Administrator and in the container's TRA file are available to the contained engine at runtime.

Design Considerations Related to Container

The engine used as the container is responsible for state and object management, fault tolerance, and logging. The contained engine delegates management of these features to the container. The main points to keep in mind are listed below.

Table 15 *Design Considerations Related to Container (Integration with ActiveMatrix BusinessWorks)*

Item	BusinessEvents Containers	ActiveMatrix BusinessWorks Containers
State and OM	BusinessEvents manages state. See also Fault Tolerance below.	ActiveMatrix BusinessWorks manages state. Use In Memory object management in BusinessEvents. Other OM methods are not supported.
Engine TRA File	The ActiveMatrix BusinessWorks engine TRA file is not used. Provide any properties needed, such as classpath and palette path, in the BusinessEvents engine TRA file or in a property file, using the -p option. (See Configuring the Environment for BusinessEvents Containers on page 198).	The BusinessEvents engine TRA file is not used. Provide any properties needed for BusinessEvents in the ActiveMatrix BusinessWorks property file.
TIBCO Hawk microagent	With configuration of one extra property in the BusinessEvents TRA file, you can use TIBCO Hawk methods for BusinessEvents and ActiveMatrix BusinessWorks. See the procedure in the section Configuring the Environment for BusinessEvents Containers on page 198 for details.	Only the TIBCO Hawk methods for ActiveMatrix BusinessWorks are used.
Fault Tolerance	BusinessEvents fault tolerance is used. See Fault Tolerance With a BusinessEvents Container on page 197 .	ActiveMatrix BusinessWorks fault tolerance is used. Do not configure BusinessEvents in fault tolerant mode.
Logging	BusinessEvents manages logging.	ActiveMatrix BusinessWorks manages logging

Fault Tolerance With a BusinessEvents Container

When BusinessEvents is the container, ActiveMatrix BusinessWorks checkpointing can be used in a limited way. Do not use ActiveMatrix BusinessWorks checkpointing in any process that is called by `startProcess()` or `invokeProcess()`, or in any process called by such processes.

When a secondary BusinessEvents engine takes over as primary, it starts the ActiveMatrix BusinessWorks engine (if it was running in the primary at time of failure). On failback to the primary engine, the secondary then stops the ActiveMatrix BusinessWorks engine (and therefore any ActiveMatrix BusinessWorks channels also).

Tips for Working With ActiveMatrix BusinessWorks Containers

Keep the following points in mind when designing your project.

- The BusinessEvents hot deployment feature is not available.
- BusinessEvents communicates only with the ActiveMatrix BusinessWorks process that invokes it. BusinessEvents channels are not available.
- Do not execute any action in a BusinessEvents startup rule function that results in use of `BusinessWorks.invokeProcess()` or `BusinessWorks.startProcess()` functions. The ActiveMatrix BusinessWorks engine may not be fully initialized when these functions are executed in a startup rule function (or in any rule in the RTC cycle of a startup rule function).
- Before deploying the ActiveMatrix BusinessWorks instance using TIBCO Administrator, first disable the BAR (BusinessEvents Archive).

Tips for Working With BusinessEvents Containers

- Add both the PAR and BAR or BARs to the same EAR. This avoids the necessity of identifying the ActiveMatrix BusinessWorks processes to the BusinessEvents engine at runtime.
- Before deploying the ActiveMatrix BusinessWorks instance using TIBCO Administrator, first disable the PAR (BusinessEvents Archive).

Configuring the Environment for BusinessEvents Containers

Before you begin configuration work, read [Design Considerations on page 195](#) and ensure your project meets the requirements and guidelines.

In order to use the integration features when running in a BusinessEvents container, you must configure the environment as explained in this section.

You must provide all the required engine and environment properties for ActiveMatrix BusinessWorks in the BusinessEvents TRA file, or in a property file used with the `-p` option.



The values shown in this section work with various test projects. Depending on the ActiveMatrix BusinessWorks services used in your project, you may have to make additional changes.

Task A Modify the BusinessEvents Engine Property File



BusinessWorks version 5.3.3 If you use BusinessWorks 5.3.3, ensure that you have applied hot fix 8 before performing this procedure.

1. Open `BE_HOME\bin\be-engine.tra` for editing.
2. Ensure that the property `tibco.env.BW_HOME` is present and uncommented.
3. Add the following properties and their values, taken from the `bwengine.tra` file, if they are present there:

```
tibco.env.BW_MIGRATION_APPEND_VERSION
tibco.env.BW_PLUGINS_HOME_OLD
tibco.env.BW_PLUGINS_HOME
```

4. Add the classpath to the ActiveMatrix BusinessWorks libraries to the `tibco.env.CUSTOM_EXT_PREPEND_CP` property. Below is an example property:

```
tibco.env.CUSTOM_EXT_PREPEND_CP=%BW_HOME%/hotfix/lib%PSP%BW_HOME%/lib%PSP%BW_HOME%/lib/palettes%PSP%TRA_HOME%/hotfix/lib/palettes%PSP%TRA_HOME%/lib/palettes%PSP%TRA_HOME%/hotfix/icjava/6.2/lib%PSP%TRA_HOME%/icjava/6.2/lib
```

5. Copy the `tibco.env.CUSTOM_EXT_APPEND_CP` property from `bwengine.tra` to `be-engine.tra`. For example:

```
tibco.env.CUSTOM_EXT_APPEND_CP=%TPCL_HOME%/tomcat/5.5/compatible/libtibco.env.CUSTOM_EXT_APPEND_CP=%TPCL_HOME%/tomcat/5.5/compatible/lib
```

Your property values will depend on what products you have installed. For example, TIBCO BusinessFactor prepends the classpath with the classpath to its libraries.

6. Add the `java.property.palettePath` property and configure it as needed to include the path to all palettes used in the integration. For example:

```
java.property.palettePath
%BW_HOME%/lib/palettes%PSP%%BW_PLUGINS_HOME%/lib/palettes%PSP%%TRA_HOME%/lib/palett
es
```

7. For startup outside of a TIBCO Administrator domain (simple command-line startup), or from BusinessEvents debugger, add the following property:

```
tibco.bwrepourl = path-to-TIBCO Designer-project
```

Use forward slashes.

Task B Add ActiveMatrix BusinessWorks Engine Properties

Add to the `be-engine.tra` file any ActiveMatrix BusinessWorks engine properties that must be set for deployment. As with any engine property that you want to change at deploy time, you can enable the properties to be configured in TIBCO Administrator at deploy time (see [Customizing the List of Properties on the Advanced Tab on page 374](#) for details).

Task C Configure for Non-Default TIBCO Hawk Microagent Names

When BusinessEvents is the container, the ActiveMatrix BusinessWorks TIBCO Hawk microagent (HMA) name is the same as the BusinessEvents HMA name, appended with `-bw`. The name is defined using the `Hawk.AMI.DisplayName` property (see [Table 16, ActiveMatrix BusinessWorks integration Properties for BusinessEvents Containers, on page 201](#) for details).

- If you want to use default HMA names, skip this task.
- If you want to use the same non-default names, except that the ActiveMatrix BusinessWorks HMA is appended with `-bw`, then define the non-default name using the `Hawk.AMI.DisplayName` property
- If you want to use different names for the ActiveMatrix BusinessWorks and BusinessEvents HMAs, do the following:
 - Define the name to be used by the ActiveMatrix BusinessWorks HMA using `Hawk.AMI.DisplayName` property. Note that whatever name you specify will be automatically appended with `-bw`.

- Add a property called `be.hawk.microagent.name` and define its value as the desired name of the BusinessEvents HMA.

Task D If PAR and BAR are Deployed in Separate EAR Files

When the following conditions are true:

- BusinessEvents is the container
- And BusinessEvents and ActiveMatrix BusinessWorks are deployed using separate EAR files
- And you will deploy to a TIBCO Administrator domain

Then you must identify the ActiveMatrix BusinessWorks processes to the BusinessEvents runtime engine, using the `tibco.bwrepourl` property.

To configure this property, provide the repo URL for the deployed ActiveMatrix BusinessWorks application (and not the EAR file or TIBCO Designer project location). See [Example Repo URL Values on page 200](#) for accepted formats.

(Note that for startup outside of a TIBCO Administrator domain, you must always add the `tibco.bwrepourl` property and its value is the path to the TIBCO Designer project, as explained in [Task A](#)).

As for other properties, you can configure the `tibco.bwrepourl` property in the `be-engine.tra` before deploying, or on the TIBCO Administrator Advanced tab for the application (the property is available on the Advanced tab by default).



Recommendation: Deploy PAR and BAR in One EAR File Add the PAR (process archive) resource to the EAR resource that contains the BAR (BusinessEvents archive) resource. Configure the PAR resource to include all processes used in the integration.

Note: Before deploying the ActiveMatrix BusinessWorks instance using TIBCO Administrator, first *disable* the PAR. The PAR must not start at runtime.

Example Repo URL Values

The repo URL format depends on the deployment transport used. Supported formats for the URL are `tibcr`, `http`, `https`, and `file`.

Rendezvous transport format:

```
tibco.bwrepourl=tibcr@domain_name-deployment_name:service=repo roService:daemon=repo
roDaemon:userName=uid:server=domain_name:password=encrypted_password
```

Local transport (also known as file transport) format:

```
tibco.bwrepourl=domain_home>/domain_name/datafiles/deployment_name_root
```

HTTP transport format:

```
tibco.bwrepourl=http://machine_name:domain_http_port?domain_name-deployment-name&server=domain_name&timeout\=600&userName=uid&password\=encrypted_password
```

Task E Configure for ActiveMatrix BusinessWorks Checkpointing (if Used)

See [Design Considerations Related to Container on page 196](#) for advice on using checkpointing in ActiveMatrix BusinessWorks when BusinessEvents is the container.

By default, an ActiveMatrix BusinessWorks instance running inside BusinessEvents has the same name as the BusinessEvents instance. If you will use ActiveMatrix BusinessWorks checkpointing, and if BusinessEvents engine instances are deployed with different names, you must add a property that specifies the same ActiveMatrix BusinessWorks instance name in all `be-engine.tra` files:

```
tibco.bwengine.name all-same-engine-name
```

You can specify the property and its value in the `be-engine.tra` files, or in the TIBCO Administrator Advanced tab at deploytime.

Table 16 ActiveMatrix BusinessWorks integration Properties for BusinessEvents Containers (Sheet 1 of 3)

Property	Notes
<code>tibco.bwengine.name</code>	<p>The name of the ActiveMatrix BusinessWorks engine. Used for ActiveMatrix BusinessWorks integration projects where BusinessEvents is the container, and used only if the following is true:</p> <ul style="list-style-type: none"> You will use ActiveMatrix BusinessWorks checkpointing BusinessEvents engine instances are deployed with different names. <p>Use this property to ensure that the same ActiveMatrix BusinessWorks engine name is specified in all nodes.</p> <p>Available in TIBCO Administrator by default. No default value.</p>

Table 16 ActiveMatrix BusinessWorks integration Properties for BusinessEvents Containers (Sheet 2 of 3)

Property	Notes
tibco.bwreporurl	<p>Used in certain circumstances for ActiveMatrix BusinessWorks integration projects where BusinessEvents is the container.</p> <p>Required when deploying to a TIBCO Administrator domain, and BAR and PAR are in different EAR files. In this case the value is the repo URL for the deployed ActiveMatrix BusinessWorks application repository. Supported formats for the ActiveMatrix BusinessWorks repository URL are <code>tibco</code>, <code>http</code>, <code>https</code>, and <code>file (local)</code>. See If PAR and BAR are Deployed in Separate EAR Files on page 200 for details about using this property.</p> <p>For startup outside of a TIBCO Administrator domain, you must always add the <code>tibco.bwreporurl</code> property and its value is the path to the TIBCO Designer project (see Task A).</p> <p>Available in TIBCO Administrator by default. No default value.</p>
Hawk.AMI.DisplayName	<p>The name of the TIBCO Hawk microagent (HMA) instance used for BusinessEvents. You can change the name as desired. The default name uses this format:</p> <pre>com.tibco.Adapter.{be-engine bw-engine}.DOMAIN.DEPLOYMENT.COMPONENT_INSTANCE</pre> <p>For in-process BusinessEvents-ActiveMatrix BusinessWorks integration projects where BusinessEvents is the container, BusinessEvents internally appends <code>-bw</code> to this name and uses it for the ActiveMatrix BusinessWorks HMA name. However, if <code>be.hawk.microagent.name</code> is used the behavior is different.</p> <p>See <code>be.Hawk.microagent.name</code> for related information.</p>

Table 16 *ActiveMatrix BusinessWorks integration Properties for BusinessEvents Containers (Sheet 3 of 3)*

Property	Notes
<code>be.hawk.microagent.name</code>	<p>This property is used only for in-process BusinessEvents-ActiveMatrix BusinessWorks integration projects where BusinessEvents is the container. This property is not present by default. You must add it if you want to use it.</p> <p>If you want to use different names for the ActiveMatrix BusinessWorks and BusinessEvents TIBCO Hawk microagents (HMAs), but you don't want to use the <code>Hawk.AMI.DisplayName</code> naming scheme (where both HMAs have the same name, except that the ActiveMatrix BusinessWorks name is appended with <code>-bw</code>), then do the following:</p> <ul style="list-style-type: none"> • Define the name of the BusinessEvents HMA using <code>be.hawk.microagent.name</code>. • Define the name of the ActiveMatrix BusinessWorks HMA using <code>Hawk.AMI.DisplayName</code> (or just use the default).

Invoking a BusinessEvents Rule Function From ActiveMatrix BusinessWorks

To enable an ActiveMatrix BusinessWorks process to call a BusinessEvents rule function, you configure one or more Invoke RuleFunction activities according to your needs. For ActiveMatrix BusinessWorks containers only, you also configure one RuleServiceProvider Configuration resource.

See [Feature Summary on page 193](#) for an introduction to the integration features.



- See [Avoiding Threading Issues on page 195](#) for important information about avoiding deadlock.
- Be careful in using the `ThreadLocal` variable because the thread of execution may not be same as that in the prior activity.
- Concepts passed to the rule function are not automatically asserted into working memory. Similarly, events passed to the rule function are not automatically asserted into working memory. You must assert them explicitly.
- `Object` support is not provided. You can't set `Object` in the scope of the rule function or as its return value. You must specify a specific type of BusinessEvents object or a primitive (other than `Object`).

Specifying Input Arguments

All BusinessEvents primitives (except `Object`) are supported. Appropriate boxing and unboxing of XML primitive types to BusinessEvents primitive types is handled. For more details, refer to the Datatypes chapter in *TIBCO BusinessEvents Language Reference*.

You can pass null values. Select the argument in the Activity Input panel of the Input tab. Click the Edit Statement (exclamation point) button. In the Edit Statement dialog, Content tab, check the Set Explicit Nil checkbox. If `xsi:nil==true`, then a null is passed as the argument.

Base Concept and Base Event are supported as argument types. You must map a specific type to the base concept or base event. Open the Edit Statement dialog, and in the Type tab, check the Type Substitution check box. In the Type field, select XML Type Reference. Select a Schema and a Type.

Using Synchronous Invocation

Execution is synchronous. The process waits for the rule function to return a value. The rule function acquires a lock on BusinessEvents working memory until the RTC cycle completes and then returns a value to the process.

You can use more than one Invoke RuleFunction activity in a process only if each references a different rule session. See [Avoiding Threading Issues on page 195](#) for more details.

Using the lockWM Parameter

By default the working memory is locked during the invocation (that is, the value is true). You can change this behavior by setting the `lockWM` parameter to false.

Setting the value to false results in the following behavior:

- The rule function is executed outside the context of the working memory.
- The results of the rule function execution are applied to working memory after acquiring a lock on working memory.

The results must not modify concept instances or scorecards (as in the case of a preprocessor).



Set the `lockWM` parameter to false for optimization purposes only and use with care. Set it to false only if the call does not modify the working memory in any way. For example, use it to perform stateless calls or calculations, or to create new concept instances or events, or for lookups and so on.

Overriding the Rule Function at Runtime

To override the rule function at runtime, enter a global variable as the value for `rulefunction` in the input tab. At runtime, you can specify a rule function with a different name but the same signature as the one specified in the activity's Configuration tab. You can only use rule functions that are in the TIBCO Designer project and in the BusinessEvents EAR file deployed for the integration project.

Specifying the Rule Service Provider for ActiveMatrix BusinessWorks Containers

You reference a RuleServiceProvider Configuration resource in the Invoke RuleFunction activity, so ActiveMatrix BusinessWorks containers know which BusinessEvents deployment to use as the rule service provider. The configured resource is automatically added to the Shared Archive (SAR) for deployment.

To Configure a RuleServiceProvider Configuration Resource

1. Open your project in TIBCO Designer and open the project folder where you want to place the resource (for example, a folder called Resources).
2. Right-click in the design panel and select **Add Resource > BusinessEvents Activities > RuleServiceProvider Configuration**.
3. Enter values in the Configuration tab, following guidelines in [RuleServiceProvider Configuration Resource Reference on page 207](#).
4. **Click Apply** and save the project.



Because the actual value is generally not known until deploytime, it is recommended that you enter a global variable for the value of the RepoURL/EAR Path field, and set the value at deploytime. (See [Deploying the Integration Project on page 219](#).)

RuleServiceProvider Configuration Resource Reference



When ActiveMatrix BusinessWorks is the container, you must specify which BusinessEvents deployment is the rule service provider for the integration by configuring a RuleServiceProvider resource. See [Specifying the Rule Service Provider for ActiveMatrix BusinessWorks Containers on page 206](#).

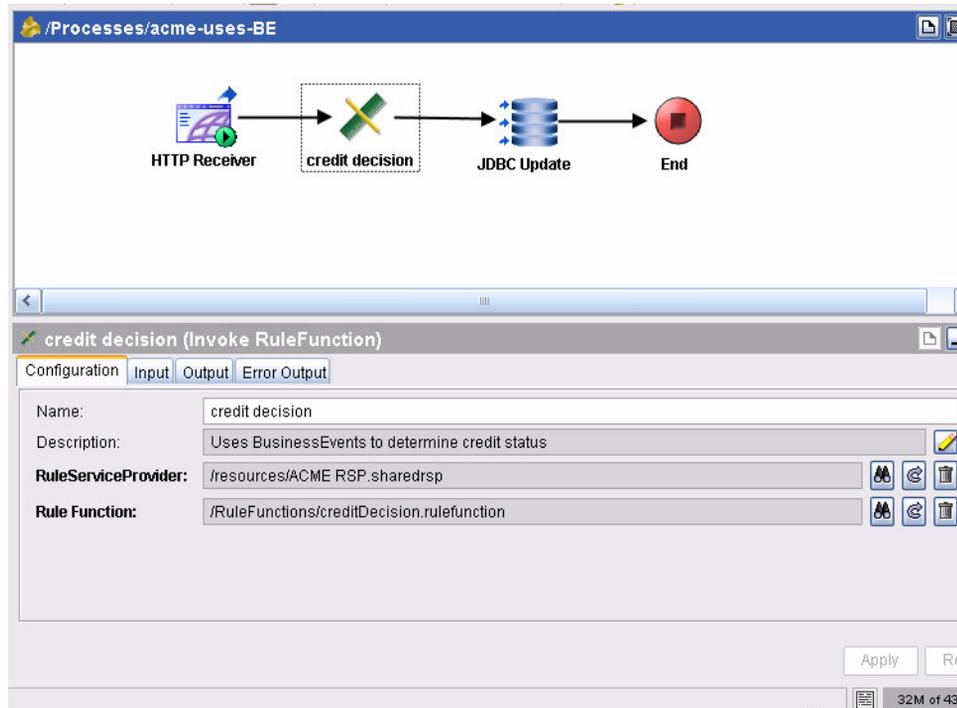
Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See <i>Identifiers (Names) in TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
Repo/URL	Yes	Path to and name of the BusinessEvents EAR file, or server-based repository URL for BusinessEvents projects deployed to a TIBCO Administrator domain. See Determining the TIBCO Repo URL for BusinessEvents on page 388 for a way to determine the repo URL.

Working With Invoke RuleFunction Activities

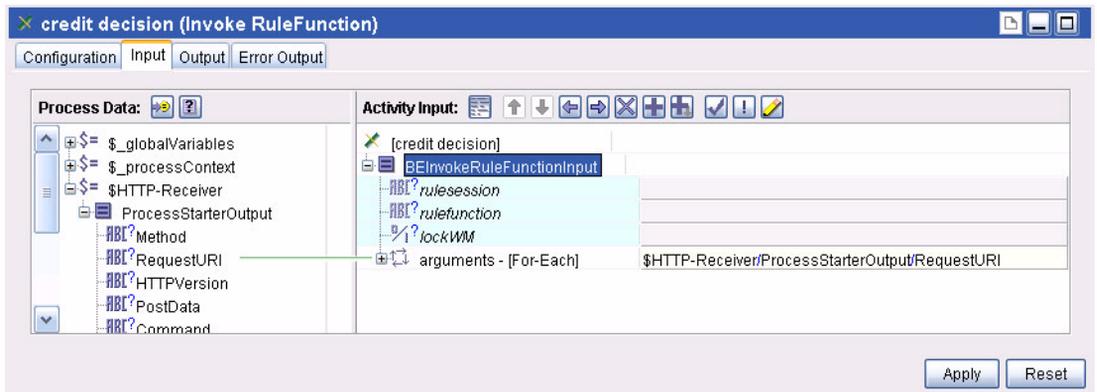
See [Invoking a BusinessEvents Rule Function From ActiveMatrix BusinessWorks on page 204](#) for important information about using the Invoke RuleFunction activity in your ActiveMatrix BusinessWorks process.



To Configure an Invoke RuleFunction Activity

1. Open your project in TIBCO Designer and open (or add) the process in which you want to use an Invoke RuleFunction activity.
2. Right-click in the design panel and select **Add Resource > BusinessEvents Activities > Invoke RuleFunction**. Link the activity to other activities as appropriate.

3. Complete the Configuration tab as follows, using guidelines provided in [Invoke RuleFunction Resource Reference on page 210](#).
 - Name the activity and give it a description as desired.
 - If the container is ActiveMatrix BusinessWorks, browse to and select the RuleServiceProvider that references the BusinessEvents deployment used for the integration. For BusinessEvents containers, this field is ignored.
 - Browse to and select the BusinessEvents rule function you want to invoke.
4. Select the **Input** tab. Expand each item to see all fields. Enter values or map values from process data. Follow guidelines provided in [Invoke RuleFunction Resource Reference on page 210](#).



5. Click **Apply** and save the project.

Invoke RuleFunction Resource Reference



To enable an ActiveMatrix BusinessWorks process to call a BusinessEvents rule function, you configure one or more Invoke RuleFunction activities according to your needs. Execution is synchronous. See [Invoking a BusinessEvents Rule Function From ActiveMatrix BusinessWorks on page 204](#) and [Working With Invoke RuleFunction Activities on page 208](#) for more details.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. Names follow Java variable naming restrictions. Do not use any reserved words. Names must be unique within a folder. See Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .
Description	No	Short description of the resource.
RuleService Provider	No	The RuleServiceProvider identifies the BusinessEvents deployment to use for the integration. The deployment must be available to the ActiveMatrix BusinessWorks engine at runtime. Required for ActiveMatrix BusinessWorks containers only. If BusinessEvents is the container, any value present is ignored.
Rule Function	No	Select the desired rule function from the BusinessEvents project.

Input

The Input tab has the following fields.

Field	Global Var?	Description
<code>rulesession</code>	Yes	<p>Specifies the rule session where the rule function is to execute.</p> <p>Optional if the project has only one rule session (BAR).</p> <p>To specify a value, enter the BAR resource name, followed by <code>.0</code>. For example:</p> <pre>mybar.0</pre>
<code>rulefunction</code>	Yes	<p>Optional. Allows you to override the rule function specified in the Configuration tab.</p> <p>Enter a global variable that is used to specify the name of the rule function at runtime. Do not enter the <code>.rulefunction</code> extension. The rule function you specify must have the same signature as the function specified in the Configuration tab (and a different name). See Overriding the Rule Function at Runtime on page 205</p>
<code>lockWM</code>	Yes	<p>Default value is true. The working memory is locked during the invocation.</p> <p>You can set the value to false only if conditions explained in the section Using the lockWM Parameter on page 205 are met.</p>
<code>arguments</code>	No	<p>Displays the arguments for the rule function, if it has any.</p> <p>Map the available process data to the activity input, or enter values in the fields as appropriate for the datatype of each argument.</p>

Output

The Output tab has the following fields.

Output Item	Data Type	Description
Return	(Varies)	The return type for the specified rule function. Return type can be any of the following: String, integer, long, double, Boolean, datetime, concept, event, Void.

Working With the BusinessWorks Functions

The integration model provides a BusinessWorks category of functions. This category is located in the General Functions library. Use and configuration of the functions in the BusinessWorks category are described in this section. See [Feature Summary on page 193](#) for a brief introduction to the integration features.

Configuring and Using `invokeProcess()`

Purpose Use this function to take advantage of ActiveMatrix BusinessWorks features. For example, you could use `invokeProcess()` to send customer information to an ActiveMatrix BusinessWorks process which gets the discount information from a database and returns it to BusinessEvents.

Runtime Behavior If the ActiveMatrix BusinessWorks engine is not already started, BusinessEvents will start it before `invokeProcess()` executes. See [Configuring and Using `init\(\)` on page 217](#) for another way to start the engine.

When a rule containing or calling `invokeProcess()` is triggered, `invokeProcess()` calls the specified ActiveMatrix BusinessWorks process and passes it an event. The `invokeProcess()` function executes synchronously, so the BusinessEvents rule engine waits. The ActiveMatrix BusinessWorks process performs its work and returns an event or null at completion of the process, or it times out if you set a timeout that is exceeded. It generates an advisory event if it times out. Events received by the rule are not automatically asserted.

Error Handling and Advisory Events BusinessEvents asserts an advisory event and returns null if the ActiveMatrix BusinessWorks process fails, or if the invocation times out. The advisory event category is `Engine`, and the type is `INVOKE BW PROCESS`. The message contains the error message from the failed ActiveMatrix BusinessWorks process, or the timeout message. The advisory event is also created if the process is cancelled using `cancelProcess()`, because the system can't differentiate between different causes for the process stopping before completion.

Because the `invokeProcess()` function returns null when an error occurs, you must handle the possibility of a null return in rules that use any property or attribute of the returned event.

See [Understanding and Working With Advisory Events on page 72](#) for more information about advisory events.



Do not use `invokeProcess()` more than once in the same thread of execution. See [Avoiding Threading Issues on page 195](#).

Configuring the Function

When configuring the `invokeProcess()` function in a rule or rule function, specify the parameters as follows:

- The ActiveMatrix BusinessWorks process that the function invokes. The specified process must not contain a process starter.
- The event to pass to the process (or specify null if you don't want to pass an event to the process).
- A timeout (or specify zero if you don't want to use a timeout).

For example, you might send customer information from an event (alias `neworder`) to an ActiveMatrix BusinessWorks process, which returns an event with the discount level to offer:

```
Events/Discountlevel Discount;
Discount = ActiveMatrix
BusinessWorks.invokeProcess("/Processes/CustInfo", neworder, 0);
```



Timeouts If you set a timeout period and a timeout occurs, the rule or rule function containing `InvokeProcess()` continues without waiting for the ActiveMatrix BusinessWorks process to complete. If you use a timeout, set it to a period long enough for the ActiveMatrix BusinessWorks process to complete. Include logic to handle the case that the `invokeProcess()` function does time out.

Configuring the Process

Configure the Start activity of the specified ActiveMatrix BusinessWorks process to accept the event passed by the `invokeProcess()` function, if an event is passed. In the Input Editor, specify the event type. Then in the Input tab, map the event properties to the process properties. If you specify null, of course, this step is not required.

Configure the rest of the activities in the process to carry out whatever processing is desired using the data passed into it by `invokeProcess()`.

Similarly, In the Output Editor of the End activity, specify the event type to return to the `invokeProcess()` process. In the Output tab, map the process data to the event properties.

The returned event is then used as needed by the logic of the rule or rule function.



Events returned are not asserted. You must explicitly assert them as needed.

Configuring and Using `startProcess()`

Purpose You would use the `startProcess()` function when you want to invoke an ActiveMatrix BusinessWorks process that performs work that can be completed asynchronously. The `startProcess()` function invokes an ActiveMatrix BusinessWorks process in asynchronous mode and immediately returns the job ID of the process. Rule processing continues. When the ActiveMatrix BusinessWorks process completes, it passes an event to a callback rule function that is specified in a `startProcess()` argument. For example, you send order information to an ActiveMatrix BusinessWorks order fulfillment process. When the order ships, notification is returned to BusinessEvents, which updates a customer concept instance.

Runtime Behavior If the ActiveMatrix BusinessWorks engine is not already started, BusinessEvents will start it before `startProcess()` executes. See [Configuring and Using `init\(\)` on page 217](#) for another way to start the engine.

When a rule containing or calling `startProcess()` is triggered, `startProcess()` calls the specified ActiveMatrix BusinessWorks process and passes it an event. `startProcess()` returns the `jobID` of the ActiveMatrix BusinessWorks process. (You can use the returned job ID, for example, in the `cancelProcess()` rule function.) The function executes asynchronously, so the BusinessEvents rule engine continues while at the same time the ActiveMatrix BusinessWorks process executes. The ActiveMatrix BusinessWorks process performs its work and passes an event to the callback (`ruleFnURI`) rule function specified in the `startProcess()` function arguments. The `ruleFnURI` rule function performs its work, for example, creating and asserting the event into the rule engine.



The `ruleFnURI` rule function must not modify concept instances or scorecards.

Configuring the Function

When configuring the `startProcess()` function in a rule or rule function, specify the following in the parameters:

- The ActiveMatrix BusinessWorks process that the function invokes. The specified process must not contain a process starter.
- The event to pass to the process (or specify null if you don't want to pass an event to the process).
- The BusinessEvents rule function that the process calls on completion (known as a callback rule function). The required signature for the rule function that is called when the process completes is shown below.

Configuring the ruleFnURI Rule Function

The rule function specified in the `startProcess()` `ruleFnURI` argument is called when the ActiveMatrix BusinessWorks process completes.

For short, this rule function is referred to as the `ruleFnURI` rule function in this guide. The `ruleFnURI` rule function must have the following signature, and the Validity field (in the Configuration tab) must be set to Action:

```
void ruleFn(long jobID, int status, Event outputEvent, Object closure)
```

- **jobID** The job id can be used to correlate the information passed to `ruleFnURI` with related information in BusinessEvents.
- **status** Returns 0 (zero) if the process completed successfully, and -1 if the process did not complete successfully (for example because `cancelProcess()` was called).
- **outputEvent** An event passed to it by the ActiveMatrix BusinessWorks process. It can be created by the process, or it can be an existing event.
- **closure** A closure object could be, for example, a value from the original context that has to be passed back to the BusinessEvents engine, for example, a loan rate that has been promised. (Note that type is Object, so you can't pass an event or concept.)

Add the `ruleFnURI` rule function to the BusinessEvents project.

Configuring the Process

Configure the Start activity of the specified ActiveMatrix BusinessWorks process to accept the event passed by the `startProcess()` function, if an event is passed. In the Input Editor, specify the event type. Then in the Input tab, map the event properties to the process properties. If you specify null, of course, this step is not required.

Configure the rest of the activities in the process to carry out whatever processing is desired using the data passed into it by `startProcess()`.

In the Input Editor of the End activity, specify the event type that is to be passed to the specified `ruleFnURI` rule function. In the Input tab, map the process data to the event properties.

The returned event is then used as needed by the logic of the rule or rule function. Note that if you want to assert the event that is returned, you must explicitly assert it.

Configuring and Using `cancelProcess()`

Purpose	<p>Use the <code>cancelProcess()</code> function to cancel a long running process, specified by <code>jobID</code>.</p> <p>Cancellation may fail if the process has already completed before receiving the cancellation command. In this case, the following exception is thrown:</p> <pre>java.lang.Exception: Job <i>JobId</i> not found</pre>
Configuration	The job ID for the process you want to cancel is provided in the return value of <code>startProcess()</code> .

Configuring and Using `init()`

Purpose	<p>The <code>init()</code> function initializes the ActiveMatrix BusinessWorks engine if it is not already running. Use of <code>init()</code> when the engine is already running is harmless.</p> <p>Use of <code>init()</code> is optional. It is provided as a convenience. For example you can use it in a BusinessEvents startup rule function to initialize the ActiveMatrix BusinessWorks engine at startup, so as to save valuable time later.</p> <p>If <code>invokeProcess()</code> or <code>startProcess()</code> are executed when the ActiveMatrix BusinessWorks engine is not already started, they will start the engine at that time.</p> <p>When ActiveMatrix BusinessWorks is the container, <code>init()</code> is not required. However its use is harmless.</p>
Configuration	<p>A general good practice is to call <code>init()</code> to start the ActiveMatrix BusinessWorks engine in a startup rule function (specified in the Startup/Shutdown tab of the BAR). The <code>init()</code> function takes no parameters. The rule function might simply contain this code:</p> <pre>BusinessWorks.init();</pre>

Configuring and Using `shutdown()`

Purpose	<p>The <code>shutdown()</code> function shuts down the ActiveMatrix BusinessWorks engine if it is running. Use of <code>shutdown()</code> when the engine is already shut down is harmless.</p> <p>When ActiveMatrix BusinessWorks is the container, <code>shutdown()</code> is ignored. Its presence is harmless.</p>
---------	--

Use of `shutdown()` is optional. It is recommended that you use it only when you have finished using the ActiveMatrix BusinessWorks engine, and won't need to start it again. Stopping and restarting the engine is not necessary and can affect performance

Configuration Simply place the shutdown command so that it is executed when needed. The `shutdown()` function takes no parameters. The rule function might simply contain this code:

```
BusinessWorks.shutdown();
```

Deploying the Integration Project

These summary instructions assume you are familiar with deployment procedures in BusinessEvents and in ActiveMatrix BusinessWorks.

You must include all ActiveMatrix BusinessWorks processes used for the integration project when you build the PAR, not just those that call a rule function or that are called by a rule function. For example a process may itself contain a process call to perform some related action. You must include that called process too.

When ActiveMatrix BusinessWorks is the Container

Build two EAR files and deploy them as follows

1. Add a an EAR resource to your project and configure a BusinessEvents BAR resource within it, in the usual way.
2. Add another EAR resource to your project and configure an ActiveMatrix BusinessWorks PAR resource in the usual way.
3. Deploy the BusinessEvents EAR as desired.
4. Upload the ActiveMatrix BusinessWorks EAR into TIBCO Administrator. Before deploying, provide the value for the global variable you defined to identify the location of the deployed BusinessEvents application (see [Specifying the Rule Service Provider for ActiveMatrix BusinessWorks Containers on page 206](#)).
 - If the BusinessEvents application is deployed to a TIBCO Administrator domain, use the server-based repository URL (this URL is the value of the `TIBCO.repourl` property in the generated TRA file for the application). TIBCO-supported protocols are `tibcr`, `HTTP`, `HTTPS`, or `file (Local)`.
 - If the BusinessEvents application is not deployed to a TIBCO Administrator domain, provide the path to the BusinessEvents EAR file, and the EAR file name itself.
5. Deploy the ActiveMatrix BusinessWorks EAR as desired.

When BusinessEvents is the Container

Deploy the EAR file in the usual way.



Disabling the PAR When you start the BusinessEvents application using TIBCO Administrator and you have specified the location of the ActiveMatrix BusinessWorks process engine using a PAR, *disable the PAR*.

To disable the PAR, uncheck the Enable Service checkbox, or select the Advanced tab for the deployment and make sure that the "start deployed instance after deployment" is unchecked before deploying.

See [If PAR and BAR are Deployed in Separate EAR Files on page 200](#) for more details.

Chapter 13 **BusinessEvents Performance Profiler**

This chapter explains how you can run a profiler utility to gather statistics about activities that occur during each RTC cycle. This information helps to identify bottlenecks in the project, which can often be addressed by redesigning rules or other aspects of a project.

Topics

- [Overview of Profiler, page 222](#)
- [Working With the Profiler, page 223](#)
- [Profiler Reference, page 228](#)

Overview of Profiler

The profiler utility collects statistics relating to the run to completion (RTC) rule evaluation cycle in an inference agent.

The utility does not collect data about object management. It also does not collect data for query agent.

The profiler records time spent during each RTC on activities such as number of times each condition or action is performed, and total time spent on each condition and action. A complete RTC includes conditions and actions, although any individual RTC might contain only conditions or only actions.

Statistics are collected for each completed RTC. When the profiler is directed to stop during an RTC, it continues to collect data for the current RTC until that RTC is completed.

When the profiler is turned off, it continues to write statistics for the current session until that session is completed. So the RTC in progress is always completed, even if the profiler is directed to stop during an RTC.

After the profiler finishes, statistics data is written to the specified (or default) file and cleared from memory.

You can execute the profiler and turn it off in three ways:

Using engine properties Profiler turns on when the rule session initializes at system startup. Used to profile RTC time, including startup rule functions.

Using BusinessEvents catalog functions Profiler turns at the beginning of the next RTC after the function call, if it has not already been enabled. (There is no effect if the profiler is already on). Used to turn the profiler inside on and off inside a rule or rule function.)

Using a TIBCO Hawk method Profiler turns on by invoking a BusinessEvents microagent Hawk method. Profiler is turned on at the beginning of the next RTC after the method call, if it has not already been enabled. (There is no effect if the profiler is already on). Used to dynamically turn the profiler on and off.

See [Working With the Profiler on page 223](#) for details.

See [Profiler Reference on page 228](#) for a reference to all output data file column headings.

Working With the Profiler

This section explains the different ways you can turn the profiler on and off.

To Turn Profiler On and Off Using Engine Properties

Set the following properties in the `be-engine.tra` file of the node or nodes of inference agent or agents whose RTC performance you want to profile.

Table 17 Profiler Configuration Properties (Sheet 1 of 3)

Property	Notes
<code>be.engine.profile. BAR_Name.enable</code>	<p>If set to true, enables profiler for the specified rule session (<i>BAR_Name</i>) when the rule session initializes. A rule session is also known as an inference agent.</p> <p>Default is false.</p>
<code>be.engine.profile. *.enable</code>	<p>If set to true, enables the profiler for all rule sessions when each rule session initializes, even when a specified rule session profiler is disabled.</p> <p>Default is false.</p>
<code>be.engine.profile. BAR_Name.file</code>	<p>The name of output file that the profiler writes to, for the specified rule session (<i>BAR_Name</i>).</p> <p>Optional, if <code>be.engine.profile.*.file</code> is specified.</p> <p>If <code>be.engine.profile.*.file</code> is specified and <code>be.engine.profile.BAR_Name.file</code> is not specified, then the file name is the value of <code>be.engine.profile.*.file</code>, with the <i>BAR_Name</i> appended.</p> <p>If the properties <code>be.engine.profile.*.file</code> and <code>be.engine.profile.BAR_Name.files</code> are not specified the following occurs:</p> <ul style="list-style-type: none"> • The file name is <code>be-profile.csv</code>, followed by an underbar, followed by the <i>BAR_Name</i>: <code>be-profile_BAR_Name.csv</code> • Spaces in a BAR name are replaced in the file name by the underbar character. For example <code>my bar</code> becomes <code>my_bar</code>.

Table 17 Profiler Configuration Properties (Sheet 2 of 3)

Property	Notes
<code>be.engine.profile.*.file</code>	<p>The default (prefix for the) name of the output file that the profiler writes to. In all cases, the appropriate <i>BAR_Name</i> is appended.</p> <p>Default name is <code>be-profile.csv</code> and it is located under the current working directory, if file name is not specified.</p>
<code>be.engine.profile.BAR_Name.duration</code>	<p>Specifies the duration of profile data collection in seconds, for the specified rule session (<i>BAR_Name</i>).</p> <p>When the duration period ends, the profiler continues to collect statistics for the current RTC until the RTC is completed, then outputs data and stops. So the RTC in progress is always completed, even if the profiler is directed to stop during an RTC.</p> <p>If you set duration to a value of zero or less (≤ 0), then profiling continues until rule session ends or profiler is explicitly turned of using a function or Hawk method.</p> <p>Default is -1.</p>
<code>be.engine.profile.*.duration</code>	<p>Specifies the duration of profile data collection in seconds, for all rule sessions (BAR files).</p> <p>When the duration period ends, the profiler continues to collect statistics for the current RTC until the RTC is completed, then outputs data and stops. So the RTC in progress is always completed, even if the profiler is directed to stop during an RTC.</p> <p>If you set duration to a value of zero or less (≤ 0), then profiling continues until the rule sessions end or profiler is explicitly turned of using a function or Hawk method.</p> <p>When <code>be.engine.profile.BAR_Name.duration</code> and <code>be.engine.profile.*.duration</code> are both present, the duration specified in <code>be.engine.profile.BAR_Name.duration</code> takes precedence.</p> <p>Default is -1.</p>

Table 17 Profiler Configuration Properties (Sheet 3 of 3)

Property	Notes
<code>be.engine.profile. BAR_Name.level</code>	<p>Level of depth that profile data will be collected for the specified rule session (<i>BAR_Name</i>):</p> <ul style="list-style-type: none"> -1: all levels of profile data are collected, including RTC level and conditions and actions within the RTC. 1: Only RTC level of profile data will be collected (and no condition and action data). <p>Default is -1.</p>
<code>be.engine.profile. *.level</code>	<p>Level of depth that profile data will be collected for all rule sessions:</p> <ul style="list-style-type: none"> -1: all levels of profile data are collected, including RTC level and conditions and actions within the RTC. 1: Only RTC level of profile data will be collected (and no condition and action data). <p>When <code>be.engine.profile.BAR_Name.level</code> and <code>be.engine.profile.*.level</code> are both present, the duration specified in <code>be.engine.profile.BAR_Name.level</code> takes precedence.</p> <p>Default is -1.</p>

To Turn Profiler On and Off Using Functions

This section assumes you understand how to use BusinessEvents functions. It tells you which functions to use and the effect of each function.



You can turn the profiler on using the engine properties, and turn it off using a function or Hawk method, as desired. See notes for `be.engine.BAR_Name.profile.duration` and `be.engine.*.profile.duration` in [Table 17, Profiler Configuration Properties, on page 223](#).

To turn the profiler on In your rule or rule function, use the following function to turn on the profiler:

```
Engine.Profiler.startCollectingToFile(String fileName, int level, long duration)
```

The above function turns on the BusinessEvents Profiler and starts collecting data for the specified duration for the rule session (inference agent) in which the rule or rule function that calls this function is executed. The profiler starts collecting data at the beginning of next RTC.

Profile data is output to the specified file in comma-separated value format at the end of the duration period, unless the profiler is turned off before the end of the duration, in which case it is output at the end of the RTC that completes after the profiler is turned off.

Input arguments are the same as the engine properties show in [Table 17, Profiler Configuration Properties, on page 223](#):

String *fileName*: See `be.engine.profile.BAR_Name.file`

int *level*: See `be.engine.profile.BAR_Name.level`

long *duration*: See `be.engine.profile.BAR_Name.duration` (Here *BAR_Name* is the current rule session in which the rule or rule function that calls this function is executed.)

To turn the profiler off In your rule or rule function, use the following function to turn off the profiler:

```
Engine.Profiler.stopCollecting()
```

The above function turns off the BusinessEvents profiler and writes the profile data to a file for the rule session in which the rule or rule function that calls this function is included (the file is output at the end of the RTC that completes after the profiler is turned off). There is no effect if the profiler is not on.

To Turn Profiler On and Off Using TIBCO Hawk Methods

This section assumes you understand how to use TIBCO Hawk methods. It tells you which methods to use and the effect of each method



You can turn the profiler on using the engine properties, and turn it off using a function or Hawk method, as desired. See notes for `be.engine.profile.duration` in [Table 17, Profiler Configuration Properties, on page 223](#).

Before you Begin Ensure that the property `hawk.enabled` is set to true in `be-engine.tra` or your deployed application TRA file before the BusinessEvents engine starts.

To turn the profiler on Use the following method to turn on the profiler:

```
StartFileBasedProfiler(String session, String fileName, int level, long duration)
```

The above method turns on the BusinessEvents profiler for the specified rule session (inference agent). The profiler starts collecting data at the beginning of next RTC for the specified duration.

This method works the same way as the `Engine.Profiler.startCollectingToFile()` function (see [To Turn Profiler On and Off Using Functions on page 225](#)), except that it requires you to specify a rule session (inference agent).

Input arguments are the same as the engine properties shown in [Table 17, Profiler Configuration Properties, on page 223](#):

`String session`: If you want to monitor multiple rule sessions (inference agents) execute the method once for each, specifying the BAR file name of the rule session in each case. If there is only one rule session, the session parameter is optional.

`String fileName`: See `be.engine.profile.BAR_Name.file`

`int level`: See `be.engine.profile.BAR_Name.level`

`long duration`: See `be.engine.profile.BAR_Name.duration`

If you attempt to turn on the profiler when it is already running, an error is returned, but the running profiler is not affected.

To turn the profiler off In your rule or rule function, use the following function to turn off the profiler:

```
StopFileBasedProfiler(String session)
```

The above method turns off the BusinessEvents profiler and writes the profile data into a file for the specified rule session when the current RTC has completed. You must execute the method once for each session, as needed.

If you attempt to turn off the profiler when it is already off, an error is returned, but there is no effect on the profiler.

Profiler Reference

The table in this section explains each of the columns in the profiler report. Data is grouped by `RTC_Stats_Type` and `Description`. (`Description` contains information about the specific RTC.) All data collected for conditions and actions performed during each RTC is listed within each RTC grouping.

Three rows of column headers for the RTC, condition and action are listed at the beginning of the file:

- One for statistics relating to the overall RTC
- One for statistics relating to conditions
- One for statistics relating to actions.

The first column of each data line is always the statistic type, which begins with one of `RTC-`, `CONDITION-`, or `ACTION-`.

Data is also grouped, one row for the overall RTC, and zero or more rows for different conditions or actions or both, as appropriate.

Table 18 Profiler Column Heading Reference (Sheet 1 of 5)

Column Heading	Notes
Statistics Relating to the Overall RTC	
<code>RTC_Stats_Type</code>	Type of rule evaluation cycle (RTC) There are 10 different types: RTC: <code>RTC-Object-Asserted</code> <code>RTC-Object-Modified</code> <code>RTC-Object-Deleted</code> <code>RTC-Event-Expired</code> <code>RTC-Execute-Rule</code> <code>RTC-Invoke-Action</code> <code>RTC-Invoke-Function</code> <code>RTC-Post-Process</code> <code>RTC-Repeat-TimeEvent</code> <code>RTC-Reevaluate-Element</code>
<code>Timestamp</code>	The time at which the first RTC begins.
<code>Description</code>	Information relating to the current <code>RTC_Stats_Type</code> . For example, the description of type <code>RTC-Object-Asserted</code> is the name of the object being asserted.
<code>NumExecuted</code>	Total number of times the same RTC has been executed.

Table 18 Profiler Column Heading Reference (Sheet 2 of 5)

Column Heading	Notes
TotalRtcTime	Total time in milliseconds spent on the total number of executions of the same RTC.
AvgRtcTime	$\text{TotalRtcTime} / \text{NumExecuted}$
MaxRtcTime	The maximum time in milliseconds spent on a single RTC.
MinRtcTime	The minimum time in milliseconds spent on a single RTC.
MaxResolvedTime	The maximum time in milliseconds spent to resolve a single RTC, including condition evaluation and action execution, but excluding operations related to object management (OM).
MinResolvedTime	The minimum time in milliseconds spent to resolve a single RTC, including condition evaluation and action execution, but excluding operations related to object management (OM).
Statistics Relating to Conditions	
CONDITION_Stats_Type	Type of rule Condition. One of the following: CONDITION-Filter Condition-Join
Timestamp	The timestamp of the first time the RTC begins.
RuleDescription	Name of the rule containing the condition, or name of state machine transition rule containing the condition.
ConditionDescription	Condition statement of a rule or a state machine transition rule for user-defined condition, or predefined condition name for internal conditions. When a user-defined rule condition has a commented-out line, the ConditionDescription of the next condition is <code>//... Only applies to CONDITION_Stats_Type</code>
NumEvaluated	Total number of times this condition is evaluated in the same RTC.

Table 18 Profiler Column Heading Reference (Sheet 3 of 5)

Column Heading	Notes
NumSuccess	Total number of times this condition is evaluated to true. The total number of times this condition being evaluated to false should be NumEvaluated - NumSuccess. Only applies to CONDITION_Stats_Type.
TotalTime	Total time in milliseconds spent on the total number of condition evaluations.
AvgTime	$TotalTime / NumEvaluated$.
MaxTime	The maximum time in milliseconds spent on a single condition evaluation.
MinTime	The minimum time in milliseconds spent on a single condition evaluation.
TotalNumLeftSearch	Total number of objects that have been searched in all left-side searches of this condition. Only applies to Condition-Join type.
NumSuccessLeftSearch	Total number of objects that have been found in all left-side searches of this condition. Only applies to CONDITION-Join type.
AvgSuccessLeftSearchRate	$TotalNumLeftSearch / NumSuccessLeftSearch$ Only applies to CONDITION-Join type.
MaxNumLeftSuccessSearch	The maximum number of objects found in a successful left-side search. Only applies to CONDITION-Join type.
MinNumLeftSuccessSearch	The minimum number of objects found in a successful left-side search. Only applies to CONDITION-Join type.
TotalNumRightSearch	Total number of objects that have been found in all right-side searches of this condition. Only applies to CONDITION-Join type.

Table 18 Profiler Column Heading Reference (Sheet 4 of 5)

Column Heading	Notes
NumSuccessRightSearch	Total number of objects that have been found in all right-side searches of this condition. Only applies to CONDITION-Join type.
AvgSuccessRightSearchRate	$\text{TotalNumLeftSearch} / \text{NumSuccessLeftSearch}$ Only applies to CONDITION-Join type.
MaxNumRightSuccessSearch	The maximum number of objects found in a successful right-side search. Only applies to CONDITION-Join type.
MinNumRightSuccessSearch	The minimum number of objects found in a successful right-side search. Only applies to CONDITION-Join type.
Statistics Relating to Actions	
ACTION_Stats_Type	Type of Actions. There are 10 RTC types and four action types. The four action types are: ACTION-Rule-Action ACTION-Event-Expiry ACTION-Invoke-Action ACTION-Invoke-Function
Timestamp	The timestamp of first time the action execution begins.
Description	Information about the action corresponding to current action type. For example, description of type ACTION-Rule-Action is the name of the rule.
NumExecuted	Total number of times the same action has been executed. A complete action has two phases, action execution and operation.
TotalActionTime	Total time in milliseconds spent on the total number of actions. $\text{TotalActionTime} = \text{TotalExecutionTime} + \text{TotalOperationTime}$

Table 18 Profiler Column Heading Reference (Sheet 5 of 5)

Column Heading	Notes
AvgActionTime	TotalActionTime / NumExecuted.
MaxActionTime	The maximum time in milliseconds spent on a single action.
MinActionTime	The minimum time in milliseconds spent on a single action.
TotalExecutionTime	Total time in milliseconds spent on the total number of action execution phases. Action execution time is the time Rete network spends on executing the action, for example, time spent in creating new objects, deleting existing objects, and so on.
AvgExecutionTime	TotalExecutionTime / NumExecuted.
MaxExecutionTime	The maximum time in milliseconds spent on a single action execution.
MinExecutionTime	The minimum time in milliseconds spent on a single action execution.
TotalOperationTime	Total time in milliseconds spent on the total number of action operation phases. Action operation time is the time the Rete network spends on applying changes to the working memory, for example, time spent asserting newly created objects, or retracting deleted objects.
AvgOperationTime	TotalOperationTime / NumExecuted.
MaxOperationTime	The maximum time in milliseconds spent on a single action operation.
MinOperationTime	The minimum time in milliseconds spent on a single action operation.
MaxAgenda	The maximum size of rule agenda as a result of all action operations.
MinAgenda	The minimum size of rule agenda as a result of all action operations.

Chapter 14 **Understanding Object Management and Fault Tolerance**

This chapter introduces object management and fault tolerance options in BusinessEvents so you can select the appropriate options for your needs.

Topics

- [Object Management \(OM\) and Fault Tolerance Overview, page 234](#)
- [Object Management and Fault Tolerance Scenarios, page 239](#)

Object Management (OM) and Fault Tolerance Overview

Object management refers to various ways that BusinessEvents can manage the state of ontology object instances created by each Rete network (inference agent).

Fault tolerance refers to high availability of the BusinessEvents engine (or agent) process. The fault tolerance features available depend on which object management option you choose.

By reading the topics in this chapter, you will be able to decide which object management option is best for your needs.

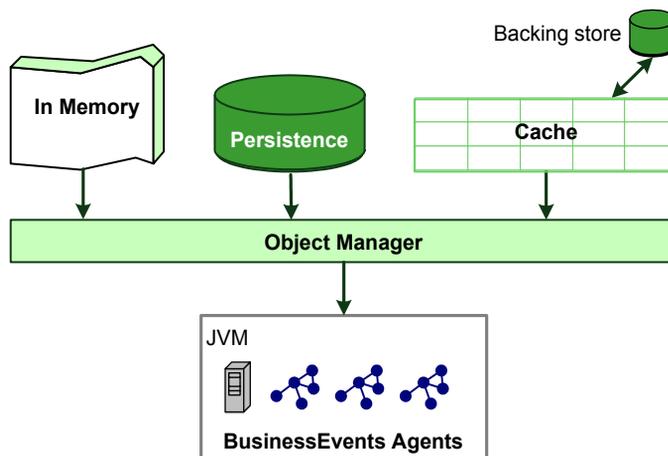
Object Management

As mentioned in [Runtime Architecture and Flow on page 18](#), BusinessEvents has three layers of functionality:

- Rules evaluation and execution
- Lifecycle management of objects and events
- Facilities to query cached data.

This section presents the options for lifecycle management of objects and events, known as object management (OM).

Figure 14 Three Main Options for Object Management



The goal of object management is to manage the state of concepts, state machines, simple events and time events that are created and used by each Rete network. Specific aspects of that goal are as follows:

- **Object Persistence** Enables objects to be available for reuse, either in memory caches or in databases.
- **Data Recovery** Ability to survive failures without loss of data.
- **Object Partitioning** The ability to partition the objects among multiple JVMs, and to handle notifications of object additions, deletions, and changes to all the agents, enabling them to remain synchronized
- **Object Clustering** The ability to maintain multiple copies of each object in different nodes such that if one node fails, another node can take over (backup-count).

Additionally, the choice of object management options affects the fault tolerance options available.

Fault Tolerance

Fault tolerance is provided using different mechanisms for each of the three main OM options:

In Memory Fault tolerance is provided at the engine level. Configuration uses various engine properties to define primary and secondary engines. See [Configuring Fault Tolerance for In Memory OM Systems on page 246](#).

Persistence Fault tolerance requires a custom solution, not documented in this guide.

Cache Fault tolerance is provided at the inference agent level. With multi-engine features enabled, fault tolerance and load balancing are provided by the same set of features. See [Load Balancing and Fault Tolerance Between Inference Agents on page 272](#) for more details.

Summary of Object Management Options



Message Acknowledgment See [Message Acknowledgment Timing for Each Object Management Type on page 26](#) for information on the way each object management option handles message acknowledgment.

In Memory

Objects are managed in memory by standard JVM features. This basic option does not provide data recovery in case of system failure. The working memory on each system is not synchronized. However, fault tolerance of the engine process is available.

The default option for object management, In Memory object management is useful for agents that use only transient objects, such as objects created at startup. It is also useful for test or demonstration purposes.

Fault tolerance and other object management topics relating to In Memory OM are documented in [Chapter 15, Configuring In Memory Object Management, on page 243](#).

Persistence



Do not confuse persistence object management with cache plus backing store.

Object data is periodically written to a data store on disk. Each agent (BAR) has its own data store. This option enables recovery of objects from the persisted state, but does not support built-in fault tolerance mechanisms. Custom means can be used to provide fault tolerance.

See [Chapter 16, Configuring Persistence Object Management, on page 251](#) for more details.

Cache

Using cache clustering technology, object data is kept in memory caches, with redundant storage of each object for reliability and high availability. Within a cache cluster, nodes deployed as cache servers manage the data and handle recovery. Cache data is shared by all agents in the cluster.

Recovery from total failure is available if you implement a persistent backing store. Recovery from failure of individual nodes is available without a backing store. Optional advanced cache management features (cache modes) provide fine-grained controls for managing the memory footprint.

Cache-based object management is generally the best choice for CEP systems. It offers richer functionality, and is the method that receives most focus in these chapters.



Cache-Related Features Cache-based object management enables product functionality in addition to object management, specifically features for querying the cache (Enterprise Suite only), and concurrent active engine features for scaling, fault tolerance and load balancing.

See the following chapters for more details on cache OM:

- [Chapter 17, Understanding Cache OM and Multi-Engine Features, on page 263](#).

- [Chapter 18, Understanding and Working With Cache Modes, on page 281](#)
- [Chapter 19, Configuring Cache Cluster Discovery, on page 287](#)
- [Chapter 20, Configuring Cache Cluster Settings, on page 295](#)
- [Chapter 21, Configuring Inference Agents \(Cache OM\), on page 305](#)
- [Chapter 22, Configuring Cache Servers \(Cache OM\), on page 319](#)
- [Chapter 23, Configuring Query Agents \(Cache OM\), on page 325](#)
- [Chapter 24, Setting up a Backing Store Database, on page 329](#)
- [Chapter 25, Project Configuration for Backing Store, on page 341](#)

Summary of Object Management and Fault Tolerance Features

The following table illustrates which of the object management and fault tolerance features are supported for each object management option (see [Summary of Object Management Options on page 235](#)).

OM Option	Persistence	Data Recovery	Partitioning	Clustering	Fault Tolerance
In Memory	No	No	No	No	Yes (at node level)
Persistence	Yes	Yes (snapshot)	No	No	(Custom)
Cache	Yes	Yes	Yes	Yes	Yes (at agent level)

Mixing Object Management Options

You configure object management options separately for each agent, that is, for each BAR resource in a TIBCO Designer project. Depending on your application logic and data requirements, you can configure In Memory OM and Persistence, or In Memory and Cache OM in one deployment.

For example, inference agent A acts as an event router. It takes all incoming events and directs them to agents B or C as appropriate. Agent A uses In Memory object management, because it does not generate any ontology object instances, and agents B and C use Cache OM to manage the ontology objects.

Migrating to a Different Object Management Method

You can use In Memory object management in early development phases of a project, and move to the option that will be used in production later.

After a project goes into production, you can move from In Memory to Persistence or Cache. You can also migrate a production system from Persistence to Cache. See *TIBCO BusinessEvents Installation*, Chapter 7, Migrating Persistence Data to Backing Store.

Object Management and Fault Tolerance Scenarios

The tables in this section help you understand how fault tolerance and object management options work in various deployment scenarios to maintain data integrity. The tables explain what is possible in each type of object management given the following conditions:

Nodes One or multiple nodes, where a node is a JVM containing one BusinessEvents server.

Agents One or multiple inference agents. Each inference agent is configured by one BAR resource in a project. An inference agent has a Rete network. See [Designing With Multiple Active Inference Agents on page 274](#) for related details.

When implementing a recovery strategy for a rule engine product such as BusinessEvents, you must take care to maintain the integrity of stateful objects. Concepts and scorecards are stateful objects and must maintain state across inference agents. Not all options provide that option.

In Memory and Persistence OM—Behavior in Multiple-Agent Nodes

When multiple agents in a node use In Memory or Persistence object management options, concept instances and scorecards are *not* shared between them. For behavior of multiple agents in a node with In Memory OM see [Local Channels on page 25](#).

For behavior of multiple concurrent agents in Cache OM deployments see [Designing With Multiple Active Inference Agents on page 274](#)

In Memory Object Management and Fault Tolerance Scenarios

Table 19 In Memory OM and Fault Tolerance Scenarios

# Nodes # Agents	With Fault Tolerance Configuration	No Fault Tolerance Configuration
1 Node 1 Agent	(N/A)	Data is isolated to a single node JVM. No recovery.
1 Node <i>n</i> Agents	(N/A)	No recovery.

Table 19 In Memory OM and Fault Tolerance Scenarios (Cont'd)

# Nodes # Agents	With Fault Tolerance Configuration	No Fault Tolerance Configuration
n Nodes 1 Agent	Data is isolated in each node JVM. Failover and failback are allowed. Object state is not preserved or transferred. Recommended only for stateless operations.	Data is isolated to each node JVM. No recovery.
n Nodes n Agents	Data is isolated in each multi-agent node. Object state is not maintained during failover and failback. Recommended only for stateless operations.	No recovery.

Persistence Object Management and Fault Tolerance Scenarios



Fault Tolerance with Persistence-Based Object Management As explained in the table below, the BusinessEvents built-in fault tolerance feature is not supported for use with persistence-based object management. You can implement a custom solution, however.

Table 20 Persistence and Fault Tolerance Scenarios

# Nodes # Agents	With Fault Tolerance Configuration	No Fault Tolerance Configuration
1 Node 1 Agent	(N/A)	Data is isolated in a single persistence database. On recovery, object state is recovered to the last checkpoint.

Table 20 Persistence and Fault Tolerance Scenarios (Cont'd)

# Nodes # Agents	With Fault Tolerance Configuration	No Fault Tolerance Configuration
1 Node n Agents	(N/A)	In all deployment scenarios, each agent's data is isolated in a separate persistence database. On recovery, object state is recovered to the last checkpoint of the appropriate database.
n Nodes 1 Agent	Not supported with BusinessEvents built-in fault tolerance. Automatic failover and failback is not possible due to presence of lock files. Use a custom solution.	
n Nodes n Agents	Not supported with BusinessEvents built-in fault tolerance. Automatic failover and failback is not possible due to presence of lock files. Multiple write operations by agents on the primary node could lead to data inconsistency. Use a custom solution.	

Cache Object Management and Fault Tolerance Scenarios

In all cases it is assumed that dedicated cache servers are also running. Fault tolerance of the engine process refers to inference agents only. See [Distributed Cache and Multi-Engine Architecture and Terms on page 269](#).

If you use multi-engine features, fault tolerance is implicit. When all agents in an agent group are active, if any active agent fails, remaining agents in the group automatically handle the work load.

In all cases, in the event of total system failure, use of a backing store ensures recovery of data written to the backing store.

Table 21 Cache and Fault Tolerance Scenarios

# Nodes # Agents	With Fault Tolerance Configuration	No Fault Tolerance Configuration
1 Node 1 Agent	(N/A)	(N/A)
1 Node n Agents	(N/A) Each agent in the same node is a different agent, not part of the same agent group.	(N/A)

Table 21 Cache and Fault Tolerance Scenarios (Cont'd)

# Nodes # Agents	With Fault Tolerance Configuration	No Fault Tolerance Configuration
n Nodes 1 Agent	<p>Fault tolerance is at the agent level.</p> <p>Multi-engine mode: If one or more agents in a group fails, the load is distributed among remaining agents in that group. All agents can be active or some can be inactive. Configuration uses a <code>MaxActive</code> property and a <code>Priority</code> property.</p> <p>Single-engine mode: Priority setting determines which agent in an agent group is active, as well as the failover and failback order.</p> <p>Cluster data is shared between agents in all groups across all nodes, using the cache cluster.</p> <p>If the number of cache object backups is one, one cache server (at a time) can fail with no data loss. If the number of backups is two, two servers can fail, and so on.</p> <p>Because caches exist in memory only, recovery is not available in the case of a total system failure. All data in each JVM memory is lost in a total system failure.</p> <p>In the event of total system failure, use of a backing store ensures recovery of data written to the backing store.</p>	<p>Multi-engine mode: N/A. Fault tolerance is implicit.</p> <p>Single-engine mode: N/A</p>
n Nodes n Agents	<p>Same as n Nodes 1 agent. Each of the agents in one node is fault tolerant with the agents in the same agent group, which are deployed in other nodes.</p>	<p>Multi-engine mode: N/A. Fault tolerance is implicit.</p>

Chapter 15 **Configuring In Memory Object Management**

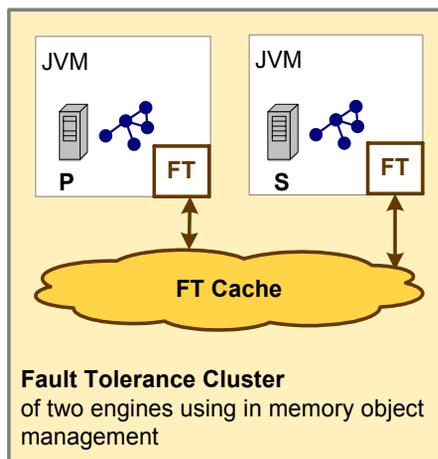
This chapter explains how to configure for In Memory object management. See [Chapter 14, Understanding Object Management and Fault Tolerance, on page 233](#) for an overview of all object management options.

Topics

- [Understanding In Memory-Based Object Management, page 244](#)
- [Configuring In Memory Object Management, page 245](#)
- [Configuring Fault Tolerance for In Memory OM Systems, page 246](#)

Understanding In Memory-Based Object Management

In memory object management does not persist ontology object instances. They are maintained in local JVM memory only.



Legend



BusinessEvents Server
(Engine)

P

Primary Server

S

Secondary Server



Fault Tolerance Cache
Memory



Rete Network
(Working Memory)

When you use the In Memory option, no recovery is possible in the case of total system failure. Object state is not maintained. At startup after a failure, object state is initialized to the application's starting state.

You can use a fault tolerance cluster to provide reliability between two or more BusinessEvents instances that use In Memory object management. However, data is not shared between the instances and is lost during failover and failback.

The In Memory option is a good choice for development and testing environments. In production environments, the In Memory option is best used for stateless operations. It can be useful in a multi-agent (rule session) system, where one agent acts as an event router, directing events to other agents for processing.

Configuring In Memory Object Management

To configure the In Memory type of object management, you select the In Memory option (which is the default option) when configuring each rule session (BAR file) in the BusinessEvents project. No other configuration is required.

Before You Begin

You must add an Enterprise Archive (EAR) resource and one or more BusinessEvents Archive (BAR) resources to your project before you begin. See [BusinessEvents Archive Resource Reference on page 362](#) for an overview and also see *TIBCO BusinessEvents Getting Started* for tutorial instructions.

To Configure In Memory Object Management

1. Open your project in TIBCO Designer.
2. From the project folders select the BusinessEvents Archive resource you want to configure and then select the **Object Management** tab.
3. From the Type drop-down list, select **In Memory**.
4. Click **Apply**.
5. If you are ready to continue to deployment, or to configuring fault tolerance, save the project and build the EAR file.

For fault tolerance configuration see [Configuring Fault Tolerance for In Memory OM Systems on page 246](#).

Configuring Fault Tolerance for In Memory OM Systems

With In Memory object management, data is transient and does not survive total system failure. However, for systems using In Memory object management, BusinessEvents offers priority-based fault tolerance to provide high availability of the BusinessEvents engine process.

Fault tolerance provides transitioning between inactive and active states, and provides state management for channels, but not Rete networks or working memories.

With In Memory object management, the failover behavior in a fault tolerance group is: When a node fails, the node with the next highest priority assumes responsibility for that node's work. So, when a primary engine fails, a secondary engine takes over as primary.

In cases where the primary and secondary are on different machines on the network, the secondary takes over if the primary loses connection to the network.

With In Memory object management, the failback behavior is: When a node restarts (or reconnects to the network in the case of a lost connection), it assumes responsibility from the node with the next lowest priority. So, when a primary engine recovers, it takes over from the secondary that had been functioning as primary.

Additionally a join time is recorded internally. The priority setting and join time are used to determine the failover and failback order in larger fault tolerance groups. If two servers have the same priority setting, then the server that joined the group first takes priority in determining the failover and failback order.



Same Weight (Priority): If two BusinessEvents servers have the same weight setting, then the server that joined the fault tolerance cluster first has the higher priority.

Fault Tolerance Configuration for In Memory OM—Using TRA Files

To Configure Fault Tolerance for In Memory OM Using TRA files

For each node, set the engine properties as follows:

1. Open `be-engine.tra` in a text editor.

`BE_HOME\bin\be-engine.tra`

(Or open the alternative property file you want to use.)

2. Configure the following properties to the same values in all engines that participate in the fault tolerance cluster:

— `Engine.FT.UseFT true`

— `Engine.FT.GroupName group_name`

Ensure that you enter the same fault tolerance group name for all members of the same fault tolerance cluster and that it is unique across fault tolerance cluster names.



If you deploy using TIBCO Administrator the value of `Engine.FT.GroupName` is generated. See [Fault Tolerance Configuration for In Memory OM—Using TIBCO Administrator on page 249](#)

3. In each engine property file, provide a short, unique name for the engine using the property `be.ft.nodename`. This name is used by the fault tolerance cluster only. The name must not be more than 30 characters long.

Configure Weight.

4. Configure the weight property to define the priorities among the servers. The primary engine has the highest priority (that is, the highest value numerically). Secondary engines have lower priorities (and lower numbers).

`Engine.FT.Weight integer`



If you give two or more engines the same weight value, then the actual priority is determined by join (server startup) time. The engine that joined the cluster first has the highest priority.

5. Save and close the file.

Table 22 *BusinessEvents Engine Properties for In Memory OM Fault Tolerance*

Property	Notes
<code>be.ft.nodename</code>	<p>Sets a stable node name used only in the fault tolerance cluster. The name is used during recovery from situations such as network disconnections.</p> <p>A node with the same name as an existing node cannot join the cluster.</p> <p>If you deploy to a TIBCO Administrator domain, and if this property is not present, then the last 30 characters of the generated engine name are used to ensure a unique node name.</p> <p>If you deploy outside a TIBCO Administrator domain and if this property is not present, then the engine name is used. See Determining the Engine (Node) Name on page 388 for various ways the engine name can be set.</p> <p>Note: The node name must not exceed 30 characters.</p>
<code>Engine.FT.GroupName</code>	<p>Defines the fault tolerance cluster (group) name in the <code>be-engine.tra</code> file, for command-line startup.</p>
<code>Engine.FT.UseFT</code>	<p>Enables or disables fault tolerance mode in the <code>be-engine.tra</code> file, for command-line startup.</p>
<code>Engine.FT.Weight</code>	<p>Sets the priority for a server in a fault tolerance group in the <code>be-engine.tra</code> file, for command-line startup. The <i>higher</i> the number, the higher the priority. (Note that the TIBCO Administrator property works in the same way: the higher the number, the higher the priority.)</p>

Fault Tolerance Configuration for In Memory OM—Using TIBCO Administrator



Unlike some other TIBCO products with which you may be familiar, BusinessEvents does not use Rendezvous for fault tolerance. When you configure fault tolerance for a BusinessEvents system in TIBCO Administrator, you use some of the same settings you would use for Rendezvous-based fault tolerance. However, the settings are used internally by BusinessEvents, not in their usual way.

BusinessEvents uses the following TIBCO Administrator settings for fault tolerance when In Memory object management is used. The property names shown in parentheses appear in the engine property file generated by TIBCO Administrator.

- **FT Weight** (`Engine.FT.Weight`)—This property is used to define the different engines as primary and secondary. The *higher* the number, the higher the priority.
- **Run Fault Tolerant** (`Engine.FT.UseFT`)—This property is used to enable or disable fault tolerance mode.
- **(No UI setting) Engine.FT.GroupName**—This property defines the members of the fault tolerance group. The value of this property is generated at deploytime, as explained next.

The value of `Engine.FT.GroupName` is generated as follows:

```
domain_name.domain_name-deployment_name.BAR_name_prefix2Ebar
```

For example, if the domain name is `acme`, the deployment name is `test`, and the bar name is `my.bar`, then the generated name would be:
`acme.acme-test.my2Ebar`.

To Configure Fault Tolerance for In Memory OM Using TIBCO Administrator

This procedure assumes that you have already configured object management options, generated the project EAR file, and uploaded it into TIBCO Administrator.

1. In TIBCO Administrator, click **Application Management**.
2. Select the BusinessEvents application and expand it.
3. In the Configuration Builder pane, click the service name (the merged bar file, which uses one of the bar file names).
4. On the General tab, in the Target Machines pane, click **Add to Additional Machines**.
5. Select a machine, and then click OK.

6. Continue to select machines according to your plan for fault tolerance.

Fault tolerance features appear in the Target Machines pane.

7. For each machine, define a fault tolerance weight in the FT Weight column

The *higher* the number, the higher priority of the server. The primary engine has the highest priority (that is, the number with the largest value). Secondary engines have lower priorities (and values closer to 1).

8. In the FT Group Settings pane, check the **Run Fault Tolerant** checkbox.

9. Click **Save**.

See [Deploying a Project in a TIBCO Administrator Domain on page 383](#) for next steps.

Configuring Persistence Object Management

This chapter explains how to configure Persistence object management options.

See [Chapter 14, Understanding Object Management and Fault Tolerance](#), on page 233 for an overview of all object management options.

See *TIBCO BusinessEvents Installation*, Chapter 7, Migrating Persistence Data to Backing Store for a procedure for migrating persistence data to a cache backing store. The procedure is used if you are changing your OM option from persistence to cache with backing store.

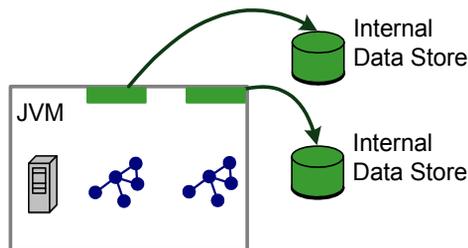
Topics

- [Understanding Persistence-Based Object Management](#), page 252
- [Configuring Persistence Object Management](#), page 254
- [BusinessEvents Archive Resource Object Management Tab—Persistence Settings Reference](#), page 255
- [Engine Property File Settings for Persistence Object Management](#), page 260

Understanding Persistence-Based Object Management

As illustrated in the figure on this page, the Persistence object management option persists a snapshot of the working memory for each inference agent (rule session) in the deployed system. The data for each inference agent is persisted to a data store at specified intervals.

A small cache for each inference agent ensures that currently used objects are available in memory for improved performance. You can control the size of the cache (see [Property Cache Size on page 256](#)).



Legend

-  BusinessEvents Server (Engine)
-  Rete Network (Working Memory - Rule Session)
-  Persistence Data Cache

The persistence data store is provided and managed by BusinessEvents.

Persistence-based object management provides data recovery in the case of a complete system failure. When the system comes up after a system failure, BusinessEvents restores the working memory (or memories) to the last checkpoint state. It also receives all of the previously unacknowledged messages (see [Message Acknowledgment Timing for Each Object Management Type on page 26](#)).

Data in memory at time of failure and not yet written to disk is lost.

Use of the Persistence option affects performance, due to the disk writes required. BusinessEvents provides parameters — checkpoint interval and property cache size — to help you tune performance. You can also determine how many objects to keep in the data cache, in order to manage JVM memory usage for the application for better performance.

You can fine tune the data cache usage at the rule session level, as described in [Defining the Database Directories for Each Rule Session \(Inference Agent\)](#) on page 258.



Memory Usage Tips

- The persistence database can be used purely as virtual memory. You can disable recovery features if you don't need them (see [Do Not Recover on Restart](#) on page 257).
- You can tune memory usage by setting the number of properties and number of events to keep in JVM memory. You can also set aside a percentage of JVM memory for use by the persistence layer. See [Engine Property File Settings for Persistence Object Management](#) on page 260 for details.
- For services that need a lot of memory, consider running each rule session (inference agent) in a separate engine.

Fault Tolerance With Persistence Object Management

Fault tolerance features for Persistence object management are not provided by BusinessEvents. You can, however, implement a custom fault tolerance solution using TIBCO Rendezvous and TIBCO Hawk or third-party fault-tolerance tools. For example, you could set up two servers that each point to the same persistence data store, and you could write rules in your fault-tolerance tool to detect failure and take appropriate steps (for example, removing any lock files) when failing over to the secondary server.



If you will provide a custom fault tolerance solution, do not enable any built-in BusinessEvents fault tolerance features. They are not used with your custom solution.

Data Migration to Cache Object Management With Backing Store

If you decide to change to a Cache OM option, follow procedures in *TIBCO BusinessEvents Installation*, to migrate data. You must of course configure the other features of Cache OM as well.

Configuring Persistence Object Management

To configure the Persistence type of object management, you select Persistence when configuring a rule session (BAR file) in the BusinessEvents project and define settings in the Object Management tab of the BAR resource, and also in the engine property file.

Before You Begin

You must add an Enterprise Archive (EAR) resource and one or more BusinessEvents Archive (BAR) resources to your project before you begin. See [BusinessEvents Archive Resource Reference on page 362](#) for an overview and also see *TIBCO BusinessEvents Getting Started* for tutorial instructions.

You must also decide on a location for the persisted data.

To Configure Persistence Object Management

1. Open your project in TIBCO Designer.
2. From the project folders select the BusinessEvents Archive resource you want to configure and select the **Object Management** tab.
3. From the Type drop-down list, select **Persistence**.
4. Configure the rest of the settings as explained in [BusinessEvents Archive Resource Object Management Tab—Persistence Settings Reference on page 255](#).
5. Click **Apply**.
6. If you are ready to continue to deployment save the project and build the EAR.file.
7. As needed, set the additional deploytime settings available in the BusinessEvents engine property file. These settings are for managing the persistence memory cache. See [Engine Property File Settings for Persistence Object Management on page 260](#).

For deployment details, [Chapter 28, Deploying a TIBCO BusinessEvents Project, on page 379](#).

BusinessEvents Archive Resource Object Management Tab— Persistence Settings Reference

Other tabs of the BusinessEvents Archive Resource are documented in [Chapter 26, Deploytime Configuration, on page 353](#).

Field	Global Var?	Description
Type	No	One of In Memory, Persistence, or Cache. Default is In Memory. See Chapter 14, Understanding Object Management and Fault Tolerance, on page 233 for information on each option. When you select Persistence, persistence configuration settings appear. Configure the settings as explained in this section.
Checkpoint Interval	No	<p>Default value: 30 seconds.</p> <p>A checkpoint is the point in time at which working memory data is written to disk. The checkpoint interval is the time, in seconds, between writes to disk.</p> <p>The term checkpoint also encompasses all the activities involved in writing the data to disk.</p> <p>Note: No changes can occur in the rule session during a checkpoint.</p> <p>It is recommended that you schedule checkpoints based on both the Checkpoint Interval and Max Outstanding Database Operations (see Checkpoint Interval and Outstanding Database Operations).</p> <p>If you want to use only the Outstanding Database Operations setting, set Checkpoint Interval to zero (0).</p>

Field	Global Var?	Description
Schedule a checkpoint if outstanding DB ops greater than	No	<p>Default value: 1000.</p> <p>Database operations include object creations, updates, and deletions. An outstanding database operation is one that is held in working memory only (it has not yet been written to disk). When the number of outstanding database operations exceeds the Max Outstanding Database Operations value, a checkpoint occurs.</p> <p>It is recommended that you schedule checkpoints based on both the Checkpoint Interval and Max Outstanding Database Operations (see Checkpoint Interval and Outstanding Database Operations).</p> <p>If you want to use only Checkpoint Interval, set Outstanding Database Operations to zero (0).</p>
Property Cache Size	No	<p>Default value: 10000.</p> <p>Defines the maximum number of concept properties that are kept in JVM memory for this rule session (BAR).</p> <p>When the persistence layer performs cleanup, the least recently used (LRU) properties are moved to the persistence store, to reduce the number of properties in memory to the specified number.</p> <p>See Caches Used for Persistence-Based Object Management on page 258.</p>
Delete Retracted Objects from Database	No	<p>Default value: checked.</p> <p>When objects are retracted (deleted) from the working memory, they are marked with a retraction flag. You can delete them from the persistence database, or you can leave them in the database (flagged with the retraction flag).</p> <p>To delete retracted objects from the database, check the Delete Retracted Objects from Database checkbox.</p> <p>It is recommended that you delete retracted objects to avoid accumulating large numbers of retracted objects in the database. However, you may want to keep retracted objects in the database for reporting or data mining purposes.</p>

Field	Global Var?	Description
Do Not Recover on Restart	No	<p>Default value: unchecked.</p> <p>Uncheck this field to use the persistence database to recover from unplanned system shutdown.</p> <p>Check this field to disable recovery features. In this case, the persistence database is used as virtual memory only.</p> <p>When recovery features are disabled, performance improves because the processing required to support the recovery features is not done.</p> <p> When you check the Do Not Recover on Restart checkbox, data is lost in the event of a system failure.</p>
Database Environment Directory	Yes	<p>Persistence files for the rule session are stored under the Database Environment directory on the target machine when the project is deployed. Enter the file path from the deployed engine to the directory.</p> <p>Each BAR must have its own database environment directory. See Defining the Database Directories for Each Rule Session (Inference Agent) on page 258.</p> <p>By default (if you do not specify a directory), persistence files are located under the BusinessEvents engine's working directory, in directories named <code>db/session_name</code>. (See Default Location of Log Files and Other Files and Directories on page 389 for details.)</p> <p>Tip: If you can't determine the location of a deployed application's persistence files, search for their filenames. The persistence file directory contains one file called <code>je.lock</code> and one or more files called <code>00000000.jdb</code>, <code>00000001.jdb</code>, and so on.</p>

Checkpoint Interval and Outstanding Database Operations

You can schedule checkpoints based on the Checkpoint Interval only, or on the Max Outstanding Database Operations only, or on both settings. It is recommended that you use both settings. When you do so, data is written to disk as follows:

- When the Checkpoint Interval passes (even if fewer than the Max Outstanding Database Operations have occurred).
- When the Max Outstanding Database Operations value is exceeded within the Checkpoint Interval. BusinessEvents then resets the Checkpoint Interval timer.

For example, assume the checkpoint interval is thirty seconds and the number of outstanding database operations is defined as five. Thirty seconds passes with only three outstanding database operations, so BusinessEvents performs a checkpoint. Then ten seconds passes and six database operations occur, so again, BusinessEvents performs a checkpoint. BusinessEvents also resets the checkpoint interval timer.

Caches Used for Persistence-Based Object Management

If your system has sufficient memory, you can improve performance by increasing the number of concept properties kept in memory for this rule session. When determining how many concept properties to keep in memory, consider the size of the properties — some may be quite large. Also consider your other requirements for memory.



Two caches are used with the Persistence option: a concept property cache and an event cache. The Property Cache size controls how many concept properties are kept in JVM memory. You define similar settings for the event cache in the `be-engine.tra` file.

Additional memory management settings also are available in the `be-engine.tra` file. They enable you to control the percentage of JVM memory that is set aside for use by the persistence layer's internal cache. See [Engine Property File Settings for Persistence Object Management on page 260](#).

Defining the Database Directories for Each Rule Session (Inference Agent)

If your project has multiple rule sessions (or if you will deploy the same application multiple times on the same machine) you must ensure that each rule session has its own database environment directory. Use one of the following options to set the values at deploy time:

- Create a different global variable for each BAR file, and use the appropriate variable in each BAR's Database Environment Directory field. At deploy time, set values for the rule sessions in each instance.
- Use the `be.engine.om.berkeleydb.dbenv` engine property (in `be-engine.tra`). At deploy time, directories are automatically created for each rule session under the directory you specify. Then do one of the following:
 - For TIBCO Administrator deployment, add the property to the Advanced tab and define the directory differently before deploying the project each

time. See [Customizing the List of Properties on the Advanced Tab on page 374](#) for more details.

- For command line deployment, define the property in override property files and specify the appropriate file using `-p` option at deploy time.

Engine Property File Settings for Persistence Object Management

When you define Persistence object management settings for a project, especially one that has multiple rule sessions (inference agents), the following tuning controls are available in the engine property file (`be-engine.tra`). These settings enable you to control how JVM memory is used for the persistence data cache.

Table 23 *BusinessEvents Engine Properties for Configuring Persistence Object Management*

Property	Notes
<code>be.engine.om.berkeleydb.internalcachepersent</code>	<p>Percentage of JVM memory to set aside for use by the persistence layer's internal cache. This memory is set aside when the engine starts up.</p> <p>Default value: 20.</p> <p>For projects with multiple rule sessions, you can also set <code>be.engine.om.berkeleydb.cacheweight.rule_session</code></p>
<code>be.engine.om.berkeleydb.cacheweight.RuleSession</code>	<p>For projects with multiple rule sessions, you can provide a weight for one or more rule sessions. The weight enables the system to calculate what percentage of the memory set aside using <code>be.engine.om.berkeleydb.internalcachepersent</code> to allocate to each rule session.</p> <p>Default value: 1</p> <p>The formula is as follows:</p> $\text{Session cache percent} = \frac{\text{internalcachepersent} * (\text{cache.weight.RuleSession} / \text{total of all session cacheweight values})}{1}$ <p>This value is an integer. Any fractional part resulting from the formula is truncated.</p> <p>For example, if you want to provide a lot of the allocated memory to a certain rule session, you can add a line in your <code>be-engine.tra</code> file for that rule session providing a higher weight value, and the rest of the rule sessions will be assigned the default weight.</p>

Table 23 *BusinessEvents Engine Properties for Configuring Persistence Object Management (Cont'd)*

Property	Notes
<code>be.engine.om.eventcache.defaultmaxsize</code>	<p>Defines the maximum number of events that are kept in JVM memory for a rule session.</p> <p>Sets the default maximum event cache size. This default is used if <code>be.engine.om.eventcache.maxsize.rule_session</code> is not specified for a rule session.</p> <p>Default value: -1 (which means, do not use this setting. See below for default behavior).</p> <p>If you do not set either of the event cache maximum size properties (<code>be.engine.om.eventcache.defaultmaxsize</code> or <code>be.engine.om.eventcache.maxsize.rule_session</code>) then the value of the Object Management tab setting Property Cache Size on page 256 is used.</p> <p>Note, however, that the Property Cache Size applies to the number of concept <i>properties</i>. Events store their properties inside the event. The event cache maximum size settings refer to the entire event, not its individual properties.</p>
<code>be.engine.om.eventcache.maxsize.RuleSession</code>	<p>Defines the maximum number of events that are kept in JVM memory for the specified rule session.</p> <p>The default value is provided by the <code>be.engine.om.eventcache.defaultmaxsize</code> property.</p> <p>When the persistence layer performs cleanup, the least recently used (LRU) events are moved to the persistence store, to reduce the number of events in memory to the specified number.</p> <p>If your system has sufficient memory, you can improve performance by increasing the number of events kept in memory. When determining how many events to keep in memory, consider the size of the events — some may be quite large. Also consider your other requirements for memory.</p>
<code>be.engine.om.berkeleydb.dbenv</code>	<p>At deploy time, directories are created for each rule session (using rule session names) under the directory you specify.</p> <p>Default value: <code>./db</code></p> <p>See Default Location of Log Files and Other Files and Directories on page 389 for more on how working directory (indicated by the dot in the default value) is determined.</p>

Understanding Cache OM and Multi-Engine Features

Cache object management (OM) provides richer functionality than the other OM options and is the standard choice for most BusinessEvents deployments.

This chapter provides a high-level overview of Cache OM and multi-engine features. More detailed information is provided in the chapters following.

Topics

- [Cache Object Management and Multi-Engine Feature Overview, page 264](#)
- [Characteristics of Distributed Caching Schemes, page 267](#)
- [Distributed Cache and Multi-Engine Architecture and Terms, page 269](#)
- [Load Balancing and Fault Tolerance Between Inference Agents, page 272](#)
- [Designing With Multiple Active Inference Agents, page 274](#)
- [Tuning the Threading Options, page 278](#)
- [Cache OM Configuration Task Summary, page 279](#)

Cache Object Management and Multi-Engine Feature Overview

Cache-based object management is generally the best choice for a CEP system, and within the various caching schemes available, (such as replicated cache and near cache) it has been determined that distributed cache is generally the most appropriate, especially when used with a backing store (database).

All the provided caching schemes use a distributed cache and are configured for production as shipped. For configuration of other caching schemes, and for advanced configuration of the provided schemes, consult the *TIBCO BusinessEvents Cache Configuration Guide* online reference.

With Cache OM, multi-engine features (engine concurrency) are also available. These features provide performance improvements.

Distributed Cache Characteristics

In a distributed cache, cached object data is partitioned between the nodes (JVMs) in the cache cluster for efficient use of memory. This means that no two nodes are responsible for the same item of data. By default one backup of each item of data is maintained, on a different node. You can configure more backups of each object to be kept on different nodes to provide more reliability as desired (and you can disable maintenance of backups).

Distributed caching offers a good balance between memory management, performance, high availability and reliability. It also offers excellent system scaling as data needs grow. See [Characteristics of Distributed Caching Schemes on page 267](#) for more details.

Scaling the System

You can scale the system's capacity to handle more data by providing more cache servers, which are nodes specialized to handle cache data only (see [Cache Server Nodes \(Storage Nodes\) on page 271](#)).

You can also scale the engine processes by adding more inference agents (see [Inference Agents on page 270](#)) to handle an increased load.

Reliability of Cache Object Management

When you use Cache object management without adding a backing store, objects are persisted in memory only, and reliability comes from maintaining backup copies of cached objects in memory caches.

Cache OM uses cache clustering to provide data failover and failback. The default backup count is one, meaning each object is backed up in one cache server. You can configure a larger backup count to determine how many backups of cache data you want the object manager to keep, each on different nodes.

Because they store data in memory, cache-based systems are reliable only to the extent that enough cache servers with sufficient memory are available to hold the objects. If one cache server fails, objects are redistributed to the remaining cache servers, if they have enough memory. You can safely say that if backup count is one, then one cache server can fail without risk of data loss. In the case of a total system failure, the cache is lost. To provide increased reliability in the case of a total system failure, add a backing store.

See [Characteristics of Distributed Caching Schemes on page 267](#) for more details.

Backing Store for Data Persistence

You can implement a backing store for use with Cache OM to provide data persistence. During regular operation, cache data is written to the backing store. On system restart, data in the backing store is restored to the cache cluster. For Cache Only objects, you can specify what gets preloaded at startup: all objects, only specified objects, or all objects except those specified.

Limiting Cache Size

In addition, when you use a backing store, you can limit the size of the cache by specifying the cache size. This is helpful for large volumes of data that the available memory cannot hold. When the number of objects in cache reaches the cache size limit, some of the objects are automatically evicted from cache (they are still in the backing store). When the system demands an object that exists in backing store but not in cache, the object is automatically loaded from the backing store into the cache.

To implement a backing store, you need a supported database product. You perform database setup using provided scripts, and perform various configuration tasks to enable writes to the database and to handle system startup behavior.

See [Chapter 24, Setting up a Backing Store Database, on page 329](#) and [Chapter 25, Project Configuration for Backing Store, on page 341](#).



If you change your project ontology, you must update the backing store schema to reflect the changes. See [Updating an Existing Backing Store Database Schema on page 338](#).

Multi-Engine Features

Multi-engine features require use of Cache OM.

Multiple inference agents can run concurrently in either of two ways. In both cases the agents share the same ontology and same cache cluster:

- Multiple instances of the same inference agent, each running on different nodes, form an *agent group* that provides load balancing of messages arriving from queues, as well as fault tolerance.
- Different agents can work concurrently to distribute the load on the JVM processes, leading to quicker conflict resolution and the ability to handle a heavy incoming message load. For example, Agent X connects to Agents Y and Z to create rule chaining across a set of nodes. Each agent uses different rule sets, such as rule sets for fraud, upsell and cross-sell, and all agents operate against the same cluster and share the same ontology. The output from one agent may trigger rules deployed in another agent, causing forward chaining of the work load.

See [Designing With Multiple Active Inference Agents on page 274](#) for more details.



If different agents can update the same concept property, you must use care to avoid race conditions. See [Understanding Concurrency and Locking Issues on page 276](#) for details.

Cache Modes for Individual Entity Types

When you use Cache object management, you can fine tune performance and memory management using various cache modes. These modes allow you to define how to manage the instances of each object type. For example, very large and infrequently used concept instances can be kept in the cache and retrieved into JVM memory on demand, freeing up the BusinessEvents engine's JVM memory. Small, frequently used, stateless entities can be kept in JVM memory only, for improved performance. See [Working With Cache Modes on page 282](#) for details.



Cache Modes and Project Design Use of some advanced cache modes can affect project design. See [Design Constraints With Cache Only Cache Mode on page 285](#) for details.

Characteristics of Distributed Caching Schemes

The cache characteristics are defined by a caching scheme. The provided caching schemes are all distributed caching schemes (see [Provided Caching Schemes on page 410](#)). This section explains why a distributed scheme is the best option.

In a distributed cache, cached object data is partitioned between the storage nodes in the cache cluster for efficient use of memory. This means that no two storage nodes are responsible for the same item of data. A distributed caching scheme has the following characteristics:

- Data is written to the cache and to one backup on a different JVM (or to more than one backup copy, depending on configuration). Therefore, memory usage and write performance are better than in replicated cache. There is a slight performance penalty because modifications to the cache are not considered complete until all backups have acknowledged receipt of the modification. The benefit is that data consistency is assured.



Each piece of data is managed by only one cluster node, so data access over the network is a "single-hop" operation. This type of access is extremely scalable, because it can use point-to-point communication and take advantage of a switched network.

- Read access is slightly slower than with replicated cache because data is not local. The cache is distributed between the nodes.
- Data is distributed evenly across the JVMs, so the responsibility for managing the data is automatically load-balanced across the cluster. The physical location of each cache is transparent to services (so, for example, API developers don't need to be concerned about cache location).
- You can add more cache servers as needed.
- The system can scale in a linear manner. No two servers (nodes) are responsible for the same piece of cached data, so the size of the cache and the processing power associated with the management of the cache can grow linearly as the cluster grows.

Overall, the distributed cache system is the best option for systems with a large data footprint in memory.

Failover and Failback of Distributed Cache Data

The object manager handles failover of the cache data on a failed storage node and it handles failback when the node recovers.

When a storage node (that is a node hosting a data cache) fails, the object manager redistributes objects among the remaining storage nodes using backup copies, if the remaining number of cache nodes are sufficient to provide the number of backups, and if they have sufficient memory to handle the additional load.

However, note that if one node fails, and then another cache node fails before the data can be redistributed, data loss may occur.

If redistribution is successful, the complete cache of all objects, plus the specified number of backups, is restored. When the failed node starts again, the object management layer again redistributes cache data.

Specifically, when a node fails, the node that maintains the backup of failed node's cache data takes over primary responsibility for that data. If two backup copies are specified, then the node responsible for the second backup copy is promoted to primary backup node. Additional backup copies are made according to the configuration requirements. When a new cache node comes up, data is again redistributed across the cluster to make use of this new node.

Limited and Unlimited Schemes

The size of a cache can be unlimited or limited. The limit is specified as a number of cache entries. By default, the cache is of unlimited size, but it can be limited using configuration options.

Performance is best when all the data is in cache. But if the amount of data exceeds the amount of memory available in the cache machines, you must limit the cache size and use a backing store to store additional data. With a limited cache, objects are evicted from the cache when the number of entries exceeds the limit. The evicted objects are transparently loaded from the backing store when needed by agents.



Only use an unlimited caching scheme if you deploy enough cache servers to handle the data. Otherwise out of memory errors may occur.

Only use a limited caching scheme if you have configured a backing store. Evicted objects are lost if there is no backing store.

See [Enabling Backing Store and Setting Cache Characteristics on page 348](#).

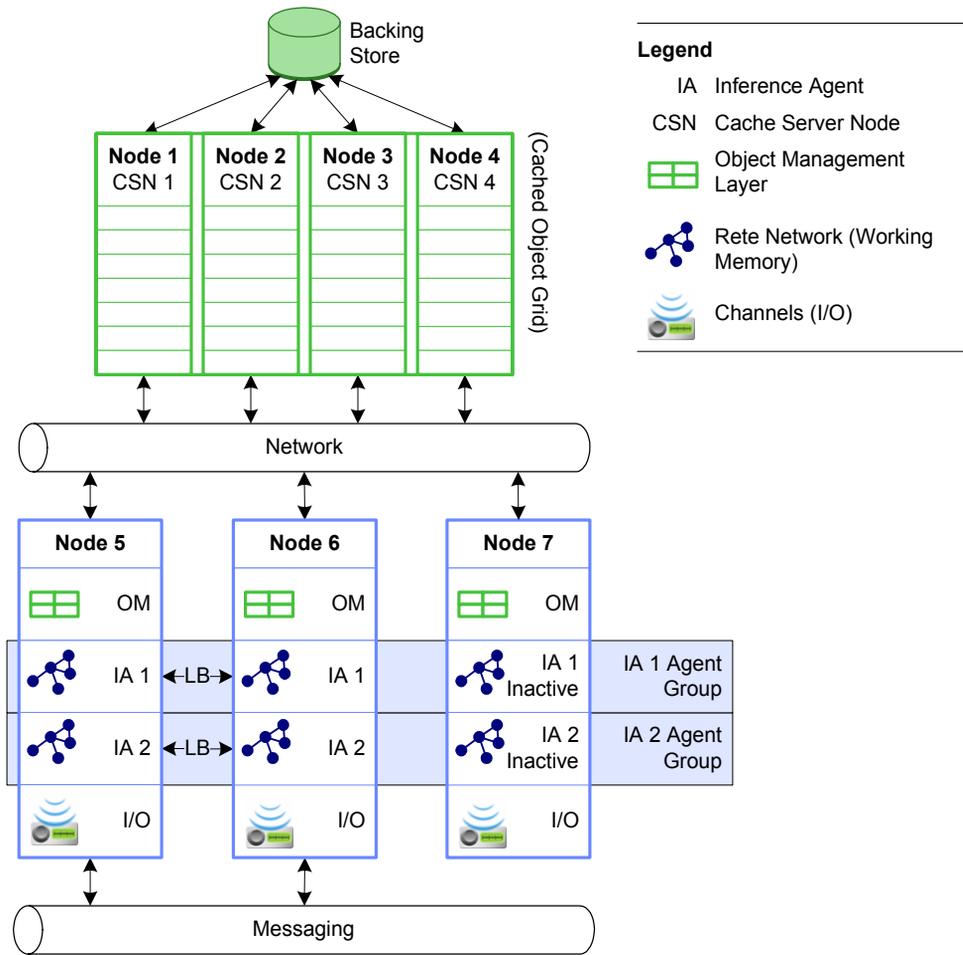
Distributed Cache and Multi-Engine Architecture and Terms

The drawing below illustrates the architecture introduced in [Cache Object Management and Multi-Engine Feature Overview on page 264](#) and provides more detail.

The drawing shows two agent groups, each with a load balanced pair of agents and an inactive agent for fault tolerance of the engine processes.

Also see [Inference Agent Runtime Architecture on page 308](#).

Figure 15 Cache Object Management and Fault Tolerance Architecture



Different BusinessEvents nodes and agents within the nodes have specialized roles in a Cache OM architecture. Sections below explain these terms.

Cache Clusters

A cache cluster is a logical entity that provides the following services:

- Cache Management: Partitioning, replication, distribution and failure recovery (see [Reliability of Cache Object Management on page 264](#)).
- Fault Tolerance (of data): Notifications to inference agents so that the state of each agent's working memory remains synchronized with the others, so any agent in the cluster can take over in event of node failure.

Members of a cache cluster are defined at the node level. Any and all agents deployed to those nodes participate in the cluster, according to their function.

Cache Cluster Nodes

A cache cluster node is a physical entity. It is an instance of a Java virtual machine (JVM) hosting one or more BusinessEvents agents.

To define the members of a cache cluster you configure a few properties in each cluster node's engine property file. See [Configure the Cache Cluster Discovery Settings on page 279](#).

A node can be configured to contain inference agents, or query agents, or both. When a node is configured to be a cache server, however, it should be used for cache data storage only. A cache server node can also contain inference agents and query agents. However for production systems it is recommended that cache server nodes do no other work.

The following sections provide more details about cache server nodes, inference agents, and query agents.

Inference Agents

An *inference agent* can be described as a rule session. Inference agents are supported in all object management options, but in Cache OM systems, inference agents are connected to the cache cluster, enabling fault tolerance of engine processes and cache data, as well as load balancing.

In TIBCO Designer, one BAR resource (added within an EAR resource) is used to configure an inference agent. For example, you can configure an inference agent with a selection of rule sets to provide business logic on the underlying objects in the cluster.

You also configure what cache modes to use for individual types of objects created and used by the agent (see [Working With Cache Modes on page 282](#)).

Active inference agents have local working memory that maintains the Rete network of past matches.

An *inference agent group* is a group of identical agents running in different engines concurrently, to enable load balancing and fault tolerance (see [Load Balancing and Fault Tolerance Between Inference Agents on page 272](#)). All agents within a group have the same rules deployed and service the same destinations.

Each agent in an inference agent group uses the same agent group name (and also has an internal ID used to identify each agent instance).

See [Chapter 21, Configuring Inference Agents \(Cache OM\), on page 305](#).

Cache Server Nodes (Storage Nodes)

The purpose of cache servers is to store and serve cache data for the cluster. A dedicated cache server node is a non-reasoning agent used as a storage node only. It does not instantiate rule sets or query sets. Cache servers are responsible for object management. They participate in distribution, partitioning and storage of the objects in the cluster.



Agent Nodes Functioning as Cache Servers It is possible, but not generally recommended, to enable inference and query agent nodes to act as cache servers in addition to their other functions. Using dedicated cache server nodes for data storage is more efficient and more scalable for production scenarios.

See [Chapter 22, Configuring Cache Servers \(Cache OM\), on page 319](#).

Query Agents

Query agents are available only in TIBCO BusinessEvents Enterprise Suite. They are used only with Cache object management. Query agents enable you to query the objects in the cache using an SQL-like syntax.

A query agent is a non-reasoning agent that has read-only access to the underlying objects in the cache cluster. A query agent has no Rete network.

You can mix query agents and inference agents within one node as desired.

See [Chapter 23, Configuring Query Agents \(Cache OM\), on page 325](#).

Load Balancing and Fault Tolerance Between Inference Agents

When multi-engine features are enabled, load balancing and fault tolerance between agents in an agent group are available. See [Designing With Multiple Active Inference Agents on page 274](#) for more on multi-engine features.

In single-engine mode, fault tolerance between agents in an agent group is available, but load balancing is not.

For configuration details, see [Configuring Fault Tolerance and Load Balancing for an Inference Agent Group on page 314](#).

Load Balancing of Inference Agents in a Group (Multi-Engine Mode Only)

Load balancing enables horizontal and vertical scaling. The underlying cluster behaves like a database for all the agents connected to the cluster. Concepts are shared between all agents and the cluster uses notifications to the different agents to keep the Rete networks synchronized.

Load balancing makes use of point-to-point messaging, such as JMS queues. With point-to-point communication, messages are automatically distributed among the members of the group. You can also use different agents to listen to different queues.

Every JMS input destination runs in its own JMS Session. This provides good throughput for processing, and less connections (see [Each JMS Input Destination Runs a Session on page 28](#)). For example, multiple instances of an inference agent listen to a JMS queue.

Certain aspects of the design have to be managed by the application. See [Designing With Multiple Active Inference Agents on page 274](#) for related information.

Fault Tolerance Between Inference Agents in a Group

In multi-engine mode, all agents in an inference agent group automatically behave in a fault tolerant manner. All load is distributed equally within all active agents. If any agents fail, the other agents automatically distribute the load between the remaining active agents.

You can optionally start a certain number of agents and out of these specify that a certain number remain inactive. If an active agents fails, an inactive agent is automatically activated.

For many situations, there is no need to maintain inactive nodes.

In single-engine mode, only one agent in a group is active at a time (because each agent instance is in a different engine). A priority property determines the startup order and the failover and fallback order.



Fault Tolerance Limitation

Entities that use In Memory Only cache mode are not recoverable in failover or fallback situations. (See [Working With Cache Modes on page 282](#).)

Behavior of Inactive Agents

Inactive agents maintain a passive Rete network. They do not listen to events from channels, do not update working memory, and do not do read or write operations on the cache.

On failover to an inactive agent, startup rule functions do not execute when the agent becomes active.

Fault Tolerance of Cache Data

Note that fault tolerance of cache servers is handled transparently by the object management layer, and that query agents do not require fault tolerance. For fault tolerance of agents, cache data, the only configuration task is to define the number of backups you want to keep, and to provide sufficient storage capacity (see [Reliability of Cache Object Management on page 264](#)).

Designing With Multiple Active Inference Agents

You can use multiple active inference agents to achieve load balancing, scaling, and performance. These are known as multi-engine features. You must be aware of some design considerations when designing project that take advantage of these multi-engine features.

Multi-engine features can be used in two ways:

- Deployment of multiple instances of the *same* agent, each in different nodes, for load balancing and fault tolerance.
- Deployment of instances of *different* agents, to achieve rule-chaining for high distribution of the work load and high performance.

In both multi-engine cases, the agents are in the same cache cluster and work on the same ontology objects.

Concepts are Shared Across Agents Asynchronously

All concept objects are shared between agents in the cluster in an asynchronous manner.

For instance, an Agent X receives an event E, fires a rule R1 that creates a concept C1. An agent Z receives an event E2, fires a rule R2 that joins concept C1 and event E2.

Therefore, there is inherent latency between an object change in an agent to the other agents in the cluster receiving the notification.



Because of the asynchronous sharing of objects between agents, give events an infinite time to live setting and explicitly consume them.

Use events with TTL=0 for queries only.

See the `StateMachineMultiEngineExample` example to see a demonstration of this point. (Requires TIBCO BusinessEvents Enterprise Suite).

Scorecards are Local to the Agent

Scorecards are not shared between agents. (This is true in all OM types.) Each inference agent maintains its own set of scorecards and the values in each agent can differ. This enables scorecards to be used for local purposes and minimizes contention between the agents.

As an analogy consider a bank ATM scenario. Money can be drawn from the same account using different ATMs. However, each ATM maintains a "scorecard" indicating only how much money it dispenses.

An agent key property (`Agent.AgentGroupName.key`) is available for tracking scorecards. See [Defining a Unique Key for Each Agent on page 313](#).



Do not use scorecards as a mechanism to share data between multiple agents.

Events are Clustered but not Shared Across Agents in a Group

In a load balanced and (optionally) fault tolerant group of agent instances, event instances are *clustered* between agents in an agent group—they are not shared. That is, each event instance is present on only one agent in the cluster.

For instance, when an agent X receives an Event E1, agent B does not see the event.

Cache cluster services provide for reassignment of ownership of these events to other agents in the same group, in the event of node failure. The events owned by this agent are redistributed to the remaining agents. (Therefore there is no single point of failure.)

Note that this can happen only if the event's time-to-live (TTL) is long enough for other agents to receive cache notification of the event.

See the `Events-MultiEngineExample` example for a demonstration of this point.

Event Related Constraints

Repeating Time Events Not Supported

Time events configured to repeat at intervals are not supported in multiple-agent (multi-engine) configurations. Rule-based time events, however, are supported.

State Machine Timeouts

State machines can be configured to have state timeouts. The objects are shared across all agents, and the agents in the cluster collaborate to take ownership of management of the state machines, thereby providing automatic fault tolerance.

Understanding Concurrency and Locking Issues

Multiple agents can read and write to the same cache cluster and at times operate on the same set of objects. You must therefore deal with the possibility of concurrent modifications. Use locks to deal avoid race conditions.

See [Locking and Synchronization Functions in Preprocessors on page 147](#) for details on managing access to objects in preprocessors, which are multi-threaded.

Updates are Done at the Property Level

Multiple agents can work on different properties of the same concept without issues. The object itself does not need to be explicitly locked. Agents update the cluster based on the actual properties being modified.

Multi-Engine Example

The following example shows how concepts are shared and events are clustered in a load balancing agent group.

Rule set A has the following rules:

```
Rule A: Scope: Event E1
        Condition: None
        Action: Create a concept C.1
```

```
Rule B: Scope: Event E2
        Concept C1
        Condition: E2.x == C1.x;
        Action: Send an event E3
```

You can start multiple instances of an agent. Suppose agent A1 and A2 are instances of an agent. Each contains rule set R.

Each agent instance has an internal ID to keep it distinct from other agents, so assume that the ID is 1 and 2 respectively.

Both agents on startup are activated based on the `maxActive` setting and therefore they both listen to the destinations on which events (E1 and E2) arrive.

The following scenario describes the behavior:

1. Agent A1(id=1) receives an instance of Event E1:

1. Rule R1 is fired and an instance of concept C1 is created.
2. The instance of C1 is replicated through the cluster to Agent A2.
3. Agent A2 gets the notification and loads the instance of concept C1 in its RETE network.

The events are clustered but are not shared. For example Agent A1 receives the event E1. If the Time to Live of the event is > 0 , then the event is acknowledged in the channel and moved to the cluster.

2. Agent A2 receives an instance of Event E2:

1. Rule R2 is fired because this agent is aware of the instance of C2.
2. Event E3 is sent out from this agent.

In the case of failure of the node containing Agent A1, Agent A2 will move the pending events to its Rete network.

Tuning the Threading Options

This section contains some tips about improving performance using the various threading options.

- Event preprocessors offer multi-threading and various thread and queue options. In the section [Working With Event Preprocessors](#), see [Worker Threading and Queue Options on page 146](#).
- If you are using shared queue and threads, choose the `com.tibco.cep.runtime.scheduler.queueSize` and `com.tibco.cep.runtime.scheduler.default.numThreads` with care (see [Shared Queue and Threads Properties on page 367](#)).
- The following properties specify the number of threads used by the distributed cache service and inference agent L1 cache respectively. As a starting point, set these properties to the number of processors available to the JVM:
 - `java.property.tangosol.coherence.distributed.threads` (see [Table 27, Cache Cluster Properties, on page 297](#))
 - `Agent.AgentGroupName.threadcount` (see [Table 29, Inference Agent Engine Properties, on page 315](#))
- If the above tuning measures don't provide the desired performance, and if the inbound JMS message rate is too high, try reducing the queue-depth size in the JMS server
- Watch the server logs for JVM heap size warnings.

Cache OM Configuration Task Summary

This section explains the overall procedure of configuring Cache object management, with references to other sections with more details. The ordering of tasks shown below is a suggestion only and is not a requirement.

Before You Begin

- Work out a detailed plan for your project architecture. It's good to begin with a simplified form of your final plan, for example one that does not use multiple instances of inference agents, and test before adding complexity.
- Configure your BusinessEvents project ontology.



For AIX only You must add the following property to all engine property files (be-engine.tra) files:

```
java.property.java.net.preferIPv4Stack true
```

If you do not add the above property, you see the following exception:

```
java.net.SocketException: The socket name is not available on this system
```

Task A Configure the Cache Cluster Discovery Settings

If all defaults are appropriate for your environment, you need only specify the cluster name. Default multicast property values handle cluster member discovery. However, other options are available for use in different circumstances.

You add the same set of cache cluster discovery settings to property files of all nodes in the cache cluster.

See [Chapter 19, Configuring Cache Cluster Discovery, on page 287](#).

Task B Configure the Cache Cluster Settings

In this step you configure certain cluster characteristics such as what caching scheme to use, whether a backing store is used, and the number of cache servers that must be running before the rest of startup actions occur, and cache logging properties. If Decision Manager is used, you also define some properties used for deploying virtual rule function implementations to BusinessEvents

With a few exceptions, you add the same set of cluster-wide properties with the same values to property files of all nodes in the cache cluster.

See [Chapter 20, Configuring Cache Cluster Settings, on page 295](#)

Task C Configure Agents and Cache Servers

- [Chapter 21, Configuring Inference Agents \(Cache OM\)](#), on page 305
- [Chapter 22, Configuring Cache Servers \(Cache OM\)](#), on page 319
- [Chapter 23, Configuring Query Agents \(Cache OM\)](#), on page 325

Also see [Chapter 17, Understanding Cache OM and Multi-Engine Features](#), on page 263 and [Chapter 18, Understanding and Working With Cache Modes](#), on page 281, especially [Design Constraints With Cache Only Cache Mode](#) on page 285.

After You Finish

After you finish configuring and testing your caching solution, you may want to add a backing store for data persistence. See [Chapter 24, Setting up a Backing Store Database](#), on page 329.

Understanding and Working With Cache Modes

This chapter explains some Cache object management settings that have an impact on system design and behavior. It is important to be aware of these features during planning and design phases. This chapter describes the different cache modes you can use for each entity in a project, and explains the consequences of using them.

Topics

- [Working With Cache Modes, page 282](#)
- [Design Constraints With Cache Only Cache Mode, page 285](#)

Working With Cache Modes

This section describes the cache modes available and explains how to use them appropriately. These options are set when you configure an inference agent that operates within a Cache object management configuration.

The Rete network consumes a large amount of the available memory in the JVM. You can use cache modes to tune the performance of your application, and reduce its footprint in memory.



See [Design Constraints With Cache Only Cache Mode on page 285](#) for important information about how use of the cache only mode affects your project design.

Cache Modes are Set on Individual Entities to Tune Performance

You set cache modes at the level of individual entity types in your project. This fine granularity allows you to tune performance and memory usage based on the size and usage of the concepts, scorecards, and events in your project ontology.

For example, you can use the In Memory Only cache mode to that frequently used stateless entities are kept in memory (and are not cached). Objects kept in memory are highly available to the application.

On the other hand, using Cache Only mode for large and infrequently used entities reduces the memory footprint. However, you must explicitly load them (in rules or rule functions) so they are available to the Rete network.



Cache Modes and Related Concepts

All related concepts must use the same cache mode Concepts can be related to each other concept through inheritance, containment or reference. If, for example, the parent is Cache plus Memory and the child is Memory Only, inconsistent object graphs will result on recovery. Ensure all use the same cache mode.

See [Working With Concepts and Concept Relationships on page 90](#) for more details about the relationships between concepts.

Cache Plus Memory—The Default Cache Mode

When you use Cache object management, by default all the persistent object types use the cache plus memory setting (Cache + Memory).

With this mode, the entity objects are serialized and are always available in cache. There is one object in the cache (in a logical sense), and any number of references (handles) to that object in each JVM. References to the entities are tracked in the working memory so they can be deserialized and retrieved from the cache when needed by the engine.

The agent's working memory is used to store the Rete network for the loaded rule sets. The inference agent does not store the actual object. It relies on the object management layer to load the objects on demand from the backing store. For example, take a rule of the form:

Declaration (Scope): Customer

Condition: Customer.age > 20

If the cache mode is cache plus memory, (Cache + Memory) then working memory stores handles to all customers—including those whose age is greater than 20. The customer objects for customers whose age is less than 20 are deserialized and retrieved from cache when the rule condition is evaluated, in order to process the rule.

Because a Rete network is so large, the references themselves can consume large amounts of memory. So if you want to reduce the memory footprint, you can use the Cache Only mode for selected entity types.

In Memory Only—Useful for Stateless Entities

When you select In Memory Only mode for an entity type, instances of that entity are available only in the engine's local JVM memory only. These entities and their properties are not recoverable, or clustered or shared. For this reason, it is recommended that you use this mode for stateless entities only.

This mode is typically used for static reference data that can be created in the rule functions on startup.

In Memory Only mode can also be used for transient utility entities that created and deleted within a single processing, and are not needed across RTC cycles.

Entities configured in this mode do not persist objects to the cluster and correspondingly the objects are not recovered from the cluster.

This cache mode works the same as the In Memory object management option. See [Chapter 15, Configuring In Memory Object Management, on page 243](#) for more details.



Fault Tolerance Limitation

Entities that use In Memory Only cache mode are not recoverable in fault tolerance failover or failback situations.

Cache Only Mode

As with the default cache mode (Cache plus Memory), when you choose the Cache Only mode for selected entities, the entity objects are serialized and are always available in cache.

However, with the Cache Only mode, the references (handles) for the Rete network must be loaded into memory, as well as the deserialized objects themselves, whenever the objects are needed for rule processing. Therefore when a cache-only object is required for rule processing, it must be explicitly loaded in the rule or rule function (see [Design Constraints With Cache Only Cache Mode on page 285](#)). For example you can put such a rule function in an event preprocessor.

When the rules have run to completion and the entities are no longer needed, the objects and references are retracted, that is, removed from working memory, to free memory. The entity instances are written to the cache or deleted (as needed).

The Cache Only mode uses less memory but adds CPU overhead. Because an active Rete network is not maintained for the cached entities, the entities must be explicitly asserted when needed. Therefore the engine re-evaluates all rules relating to the instance (as it does for any newly asserted instance, for example, arriving through a destination). However, the portion of the working memory that would be used by the object and its references is reduced significantly. It's up to you to balance the benefit of reducing the memory footprint against the cost of the increased load on the CPU in any given situation.

If you use a backing store you can configure preloading options for Cache Only objects. See [Configuring How Backing Store Data is Loaded at Startup on page 349](#).

Design Constraints With Cache Only Cache Mode

It is important during planning phases to understand the effect of using cache-only cache mode at runtime, so that you can design your projects accordingly.

Explicitly Loading Objects into Working Memory, With the Cache Only Mode

When you use the cache only mode for an entity type, entities of that type are asserted into working memory in the usual way when created. However when they are no longer needed (at the end of an RTC), they are removed from working memory. You must explicitly load the instances back into working memory when they are needed, so that they are available to the rule engine.

Using Cache Load Functions to Explicitly Load Objects into Working Memory

The following functions are available for the purpose of loading cache-only objects into working memory. The functions are in the Coherence function category and they are as follows:

```
C_CacheLoadConceptByExtId()
C_CacheLoadConceptById()
C_CacheLoadEventByExtId()
C_CacheLoadEventById()
C_CacheLoadParent()
```

The functions that load concepts by ExtID or ID have a parameter to control whether contained concepts are also loaded. The function for loading a given concept's parents has the option to return all parents or the immediate parent (that is, concept or concepts related to the given concept by a containment relationship).

A general best practice is to use these functions in an event preprocessor. If you do so then the rules themselves do not have to take account of the need to load the objects during execution. For example, in the preprocessor, you could preload an order concept using an ExtID available in the event as follows:

```
Concepts.Order order =
Coherence.C_CacheLoadConceptByExtId(orderevent.Order_Id, false);
```

Working with Objects Without Asserting Them Into Working Memory

It is not always necessary to explicitly assert an object into working memory in order to work with it. For example, you can use `getbyextId()` to retrieve a local copy of the object from cache and use information in the object for calculations (or other uses) in a rule.

Modifying a Concept Instance in a Rule Implicitly Asserts Objects Into Working Memory

If you modify a concept instance `BusinessEvents` asserts it into working memory.

```
Concepts.Order o = Instance.getByExtId(orderevent.Customer_Id);
if(o.status == "open")
    o.Price = orderevent.Price;
Event.consumeEvent(orderevent);
```

If the entity instance is not found in working memory or cache, your application logic determines what to do next, for example, create a new instance of that entity type.

Objects are Removed From Working Memory at End of RTC

After the object is asserted into working memory from the cache, other rules in the same RTC that depend on that object can then execute. At the end of the RTC, all cache-only objects are removed from working memory.

The object is not available for a future RTC unless you do one of the following:

- Modify the object, and a rule has a dependency on that modification.
- Assert the object into working memory using the cache load functions, as explained in [Explicitly Loading Objects into Working Memory, With the Cache Only Mode on page 285](#).

Chapter 19 **Configuring Cache Cluster Discovery**

This chapter is the first of a group of chapters that explain how to configure Cache OM systems. Use these chapters sequentially to configure your system. The first step is to set cluster-wide configuration settings. Later chapters explain other cluster-wide settings, and how to configure agents and nodes to perform specific roles within the cache cluster.

The output of cluster-level configuration is a set of engine properties. With a few exceptions, the same set cluster-wide properties with the same values are used when deploying all nodes in the cluster.

Topics

- [*Cache Cluster Discovery Overview, page 288*](#)
- [*Discovering Cache Cluster Members Using Multicast, page 290*](#)
- [*Discovering Cache Cluster Members Using Well-Known-Addresses, page 292*](#)
- [*Discovery When Hosts Have Multiple Network Cards, page 294*](#)

Cache Cluster Discovery Overview

This chapter explains how the members of the cache cluster are defined. In many cases, the default multicast values in the provided engine property files work out of the box.

The chapter provides details about how you can specify members in different ways if the default multicast configuration is not appropriate for your environment.



For AIX only You must add the following property to all engine property files (be-engine.tra) files:

```
java.property.java.net.preferIPv4Stack true
```

If you do not add the above property, you see the following exception:

```
java.net.SocketException: The socket name is not available on this system
```

Cluster Member Discovery Using Multicast Discovery

Multicast discovery is used by default. A set of default values mean you may not have to configure any discovery-related properties other than cluster name. However, if you deploy multiple projects, you must specify different multicast address and port settings for each.

You can use multicast when instances of an application are deployed to hosts in the same subnet, or across subnets when broadcast is enabled between the subnets.

Multicast requires less maintenance than the well-known-address method of cluster configuration. For example, with multicast you don't have to update the property files after moving a deployed node (engine) to another machine.

As an alternative you can use well-known addresses for discovery. See [Discovering Cache Cluster Members Using Well-Known-Addresses on page 292](#).

Also see [Discovery When Hosts Have Multiple Network Cards on page 294](#).

Network Traffic

When multicast discovery is used, network traffic is generated by the following activities:

Cluster heartbeat The most senior member in the cluster issues a periodic heartbeat using multicast. The rate is configurable and defaults to one per second.

Message delivery Messages intended for multiple cluster members are often sent using multicast, instead of unicasting the message one time to each member.

For peer-to-peer communications and data transfer, the object management layer then uses unicast as needed. Peer-to-peer connections are automatically established with other servers listening on the multicast address.

Cluster Member Discovery Using Well-Known-Addresses

Well-known-address configuration uses direct member-to-member (point-to-point) communication, including messages, asynchronous acknowledgements (ACKs), asynchronous negative acknowledgements (NACKs) and peer-to-peer heartbeats. The addresses are "well known" to the extent that they are known to each server in the cluster.

Use well-known-addresses instead of multicast to define the cluster participants when multicast is undesirable or unavailable, for example, if nodes are deployed to different subnets and broadcast between the subnets is not enabled.

Also see [Discovery When Hosts Have Multiple Network Cards on page 294](#).

Overriding and Adding Cluster Configuration Properties

Operational settings for the cache cluster are provided in the operational descriptor, `tangosol-coherence.xml`, provided in `BE_HOME/lib/ext/coherence.jar`.

For details on all the settings in this file, see the online reference, [TIBCO BusinessEvents Cache Configuration Guide](#), which is available from the HTML documentation interface.

If you need to use different values than are provided in the operational descriptor, you do not make any changes in the `tangosol-coherence.xml` file itself. Engine properties and (as needed) override files enable you to override and extend the settings in the operational descriptor file.

See [Overriding and Extending the Operational Deployment Descriptor on page 411](#) for information about using `system-property` attributes in the operational descriptor to create new engine properties.

See [Specifying Operational Override File Locations on page 413](#) for details about using override files.

Discovering Cache Cluster Members Using Multicast

With multicast cache cluster configuration, the cluster membership is established using the multicast IP address and port. When a BusinessEvents node subscribes to this multicast IP address it broadcasts information about its presence to the address.

If the default values for the multicast properties values work for your environment, then you don't have to configure any multicast properties. You need only define a cluster name in the property file for each engine (node) of the cluster.

Below is an example showing the properties with their defaults:

```
java.property.tangosol.coherence.clusteraddress=224.3.3.1
java.property.tangosol.coherence.clusterport=33389
java.property.tangosol.coherence.ttl=0
```



If multicast discovery is not possible in your environment, see [Discovering Cache Cluster Members Using Well-Known-Addresses on page 292](#) for another technique.

Adding, Removing, and Moving Nodes

The object management layer uses multicast to discover new nodes and add them to the cache cluster. Similarly when nodes are removed or moved to a different server, the multicast protocol ensures that members are kept current without any additional configuration.

To Configure Multicast Settings

The properties for each node in the cluster must be set to the same values.

1. For each node, add the following property to the engine property file and provide the same value for *cluster_name*:

```
java.property.tangosol.coherence.cluster=cluster_name
```

2. If the default multicast configuration address and port does not work for your environment, add the following properties, specifying the appropriate value:

```
java.property.tangosol.coherence.clusteraddress=cluster_address
```

```
java.property.tangosol.coherence.clusterport=cluster_port
```

Table 24 Cache Cluster Network Settings for Multicast Protocol

Property	Notes
<code>java.property.tangosol.coherence.cluster</code>	<p>Specifies the name of the cache cluster. Use the same name for all nodes in a cache cluster.</p> <p>Required. No default value.</p>
<code>java.property.tangosol.coherence.clusteraddresses</code>	<p>Specifies the multicast IP address that the socket will listen to or publish on.</p> <p>Required for multicast configuration.</p> <p>Possible values are addresses between (and including) 224.0.0.0 and 239.255.255.255.</p> <p>Default value is 224.3.3.1</p>
<code>java.property.tangosol.coherence.clusterport</code>	<p>Specifies the port that the socket will listen to or publish on.</p> <p>Required for multicast configuration.</p> <p>Possible values are integers between 1 and 65535.</p> <p>Default value is 33389</p>
<code>java.property.tangosol.coherence.ttl</code>	<p>Specifies the time-to-live setting for the multicast, that is, the maximum number of "hops" a packet can traverse. A hop is defined as a traversal from one network segment to another via a router.</p> <p>For production use, set this value to the lowest integer value that works.</p> <p>On a single-host cluster, set to zero (0). On a simple switched backbone, set to 1. On an advanced backbone with intelligent switching, it may require a value of 2 or more.</p> <p>Setting the value too high can use unnecessary bandwidth on other LAN segments and can even cause the operating system or network devices to disable multicast traffic.</p> <p>Required for multicast configuration.</p> <p>Possible values are integers between 0 and 255.</p> <p>Default value is 4</p> <p>Note: While a value of 0 is intended to keep packets from leaving the originating machine, some operating systems do not implement this correctly, and the packets may in fact be transmitted on the network.</p>

Discovering Cache Cluster Members Using Well-Known-Addresses

See [Cache Cluster Discovery Overview on page 288](#) for an introduction to the two methods of configuring a cache cluster, multicast and well-known-addresses.



Specifying one or more well-known addresses disables all multicast communication.

Adding, Removing, and Moving Nodes

You must ensure that the well known address information is updated on all machines to account for system changes, such as adding, removing, and moving nodes to different machines.

If a machine changes name or IP address, the cluster continues to work correctly until the next time you restart that machine.

To Configure Well-Known-Address Settings

Enter the same set of cache cluster properties with the same values in the engine property files for all nodes in the cluster. The properties are shown below, and are described in [Table 25, Cache Cluster Network Settings for Well-Known Address Protocol, on page 293](#).

```
java.property.tangosol.coherence.cluster cluster_name

java.property.tangosol.coherence.wka1 host1
java.property.tangosol.coherence.wka1.port 8089

java.property.tangosol.coherence.wka2 host2
java.property.tangosol.coherence.wka2.port 8089
```

If two nodes are deployed on the same host, assign them different ports.



Provision is made for up to six well known addresses in the provided operational descriptor, using numbers 1–6. If you need more than six, see [Overriding and Extending the Operational Deployment Descriptor on page 411](#) for information about adding more well-known addresses.

Table 25 Cache Cluster Network Settings for Well-Known Address Protocol

Property	Notes
<code>java.property.tangosol .coherence.cluster</code>	<p>Specifies the name of the cache cluster. Use the same name for all nodes in a cache cluster.</p> <p>Required. No default value.</p> <p>Note Do not use the name <code>\$cluster</code>. It is a reserved name.</p>
<code>java.property.tangosol .coherence.wkan</code>	<p>Each "well-known-address," specifies the IP address (the value of the <code>wkan</code> property) and port that the socket will listen to or publish on.</p>
<code>java.property.tangosol .coherence.wkan.port</code>	<p>For port, enter a value between 1 and 65535. If multiple instances run on the same machine, use a different port number for each.</p> <p>For example:</p> <pre>java.property.tangosol.coherence.wka1 host1 java.property.tangosol.coherence.wka1.port 8088 java.property.tangosol.coherence.wka2 host2 java.property.tangosol.coherence.wka2.port 8088</pre>

Discovery When Hosts Have Multiple Network Cards

If a host machine has multiple network cards, you must specify which IP address and port to bind to. To do this you use `localhost` and `localport` properties. Add the `localhost` and `localport` settings to the property files of all nodes in the cluster that have multiple network cards.

Do this whether you are using multicast or well-known-addresses to define the cache cluster.

Table 26 Cache Cluster Network Settings for Multiple Network Cards

Property	Notes
<code>java.property.tangosol.coherence.localhost</code>	<p>Specifies the IP address that the socket will listen to or publish on.</p> <p>Required when a host has multiple network cards, to specify which card's IP address and port to use.</p> <p>You can generally set the value of the <code>localhost</code> property to the value "localhost." However, if <code>localhost</code> is used as the loop back address (127.0.0.1) you must enter a machine name or IP address.</p> <p>Default value is <code>localhost</code>.</p>
<code>java.property.tangosol.coherence.localport</code>	<p>Specifies the port that the socket will listen to or publish on.</p> <p>Required for well-known address configuration.</p> <p>Possible values are 1 to 65535.</p> <p>Default value is 8088.</p> <p>Note If a specified port is not available, the object management layer (by default) increments the port number until it finds an available port. Avoid potential conflicts by choosing a number that is not close to a port used by other software. For example, TIBCO Administrator uses port 8080 by default, so it's a good idea to set the <code>localport</code> property to a higher value, such as 8090.</p> <p>Alternatively you can turn off the auto-incrementing feature. To turn off auto-incrementing, set the value of the following override option to false:</p> <p><code>tangosol.coherence.localport.adjust</code></p>

Configuring Cache Cluster Settings

This chapter describes cluster-wide configuration settings, other than cluster discovery settings (see [Chapter 19, Configuring Cache Cluster Discovery, on page 287](#).)

The output of cluster-level configuration is a set of engine properties. With a few exceptions, the same set cluster-wide properties with the same values are used when deploying all nodes in the cluster.

Later chapters explain how to configure agents and nodes to perform specific roles within the cache cluster.

Topics

- [Configuring Caching Scheme, Multi-Engine, and Cluster Properties, page 296](#)
- [Configuring Cache Cluster Logging Properties, page 301](#)
- [Example Cluster Configuration Properties, page 303](#)

Configuring Caching Scheme, Multi-Engine, and Cluster Properties

These properties enable you to select a provided caching scheme, and configure various aspects of the cache cluster. Some properties are not directly related to the cache cluster but are set at the level of a node or above. See [Chapter 17, Understanding Cache OM and Multi-Engine Features, on page 263](#) for background information.



Except where noted, these settings must be identical on all nodes.

For rare situations requiring customization, contact TIBCO support for assistance.

Cluster Level Options, Summarized by Task

To do this...	Use this property
Specify whether to use a backing store (or not)	<code>be.engine.cluster.hasBackingStore</code>
If using a backing store, specify whether to use a limited cache size or not, and set the size if so (if the default is not acceptable)	<code>be.engine.cluster.isCacheLimited</code> <code>java.property.be.engine.limited.cache.back.size.limit</code>
To use multi-engine features (or not)	<code>be.engine.cluster.multiEngineOn</code>
To set cache data backup count	<code>java.property.tangosol.coherence.distributed.backupcount</code>
To set the number of cache servers to start before activating all engines	<code>be.engine.cluster.minCacheServers</code>
To set the number of threads	<code>java.property.tangosol.coherence.distributed.threads</code>
To set the path to Decision Manager external classes (rule function implementations) and select a class loader.	<code>be.engine.cluster.externalClasses.path</code> <code>be.engine.cluster.externalClasses.classLoader</code>

Cluster Level Option Details

Table 27 Cache Cluster Properties (Sheet 1 of 3)

Property	Notes
<code>be.engine.cluster.hasBackingStore</code>	<p>If true, the cluster is configured with a backing store.</p> <p>See Chapter 24, Setting up a Backing Store Database, on page 329 and Chapter 25, Project Configuration for Backing Store, on page 341 for details on configuring the backing store and its database.</p> <p>Add this property with the same value in the property files for all nodes in the cluster.</p> <p>Possible values: true or false.</p> <p>Default is false.</p>
<code>be.engine.cluster.isCacheLimited</code>	<p>If set to true the cache size is limited. Limited caches are used only when a backing store is used to store entries in excess of the limit.</p> <p>Also see notes for <code>java.property.be.engine.limited.cache.back.size.limit</code>.</p> <p>Possible values: true or false.</p> <p>Default is false.</p>
<code>java.property.be.engine.limited.cache.back.size.limit</code>	<p>Specifies the size of the limited cache, in number of cache entries in each cache server (that is, each node where local storage is enabled).</p> <p>Used only if <code>be.engine.cluster.isCacheLimited</code> is set to true and you want to use a non-default value.</p> <p>Default is 10000.</p>
<code>be.engine.cluster.multiEngineOn</code>	<p>Set to true to enable multiple engines to be active at the same time. Used only with cache OM.</p> <p>Load balancing and fault tolerance features are available only when this property is set to true.</p> <p>If set to false, only one engine is active at any time (as in earlier versions). This is appropriate for In Memory OM and Persistence OM.</p> <p>Possible values: true or false.</p> <p>Default is false.</p>

Table 27 Cache Cluster Properties (Sheet 2 of 3)

Property	Notes
<code>java.property.tangosol .coherence.distributed .backupcount</code>	<p>The backup count specifies the number of members of the distributed cache service that hold the backup data for each unit of storage in the cache. Recommended values are 0, 1, or 2.</p> <p>Value of 0 means that in the case of abnormal termination, some portion of the data in the cache will be lost. Value of N means that if up to N cluster nodes terminate at once, the cache data will be preserved.</p> <p>A backup count of 1 means one server plus one backup, that is, two cache servers (or storage enabled nodes if cache servers are not used).</p> <p>To maintain the partitioned cache of size M, the total memory usage in the cluster does not depend on the number of cluster nodes and will be in the order of $M*(N+1)$.</p> <p>See Failover and Failback of Distributed Cache Data on page 268.</p> <p>Default is 1.</p>
<code>be.engine.cluster.minC acheServers</code>	<p>Specifies the minimum number of storage-enabled nodes that must be active in the cluster when the system starts up before the following occur:</p> <ul style="list-style-type: none"> • Data is loaded from the backing store, if a backing store is configured (see Chapter 25, Project Configuration for Backing Store, page 341). • The other agents in the cluster become fully active. <p>Add this property with the same value in the property files for all nodes in the cluster.</p> <p>The property does not affect the running of the deployed application after startup (though a message is written to the log file if the number of cache servers running falls below the number specified in this property).</p> <p>As a guideline, set to the number of cache servers configured.</p> <p>Default is 1.</p>

Table 27 Cache Cluster Properties (Sheet 3 of 3)

Property	Notes
<code>java.property.tangosol.coherence.distributed.threads</code>	<p>Specifies the number of Coherence daemon threads used by the distributed cache service.</p> <p>If zero, all relevant tasks are performed on the service thread.</p> <p>As a guideline, set this value to the same number as there are processors available to the JVM. See Configuring Inference Agents—Engine Properties on page 313 for details on the <code>Agent.AgentGroupName.threadcount</code> property.</p> <p>Legal values are from positive integers or zero.</p> <p>Default is 0.</p>
<code>be.engine.cluster.externalClasses.path</code>	<p>Specifies the filepath used by the cluster to load external rule classes created in Decision Manager to all BusinessEvents cluster nodes.</p> <p>Set on one or more nodes as needed. Set this property in nodes where you also set <code>be.engine.cluster.externalClasses.classLoader</code> to true. Ensure these nodes can access the filepath.</p> <p>Note: Cache OM is required for this feature.</p> <p>Tip: Configure this property with the path to an RMS project deployment subdirectory to enable a seamless connection between the RMS approval process and the BusinessEvents loading process. This tip works for one RMS project only.</p> <p>See Virtual Rule Functions and Decision Manager on page 129 for more details.</p> <p>Required only if Decision Manager is used.</p>
<code>be.engine.cluster.externalClasses.classLoader</code>	<p>If true, this node has the ability to load external rule classes to all cluster nodes.</p> <p>Set on one or more nodes as needed.</p> <p>For related details, see notes for <code>be.engine.cluster.externalClasses.path</code>.</p> <p>Note: Do not use nodes that contain only query agents.</p> <p>Possible values: true or false.</p> <p>Default is true.</p> <p>Required only if Decision Manager is used.</p>

Configuring Cache Cluster Logging Properties

Use these settings as needed to tailor the cache logging behavior for your needs.

Table 28 Cache Cluster Logging Properties

Property	Notes
<code>java.property.tang osol.coherence.log .level</code>	<p>Specifies which logged messages are output to the log destination.</p> <p>Optional.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> • 0: Only output without a logging severity level specified will be logged • 1: All the above plus errors • 2: All the above plus warnings • 3: All the above plus informational messages • 4-9: All the above plus internal debugging messages (the higher the number, the more the messages) • -1: No messages <p>Default is 5.</p>
<code>java.property.tang osol.coherence.log</code>	<p>Specifies the output device used by the logging system.</p> <p>Optional.</p> <p>Possible values are:</p> <ul style="list-style-type: none"> • <code>stdout</code> • <code>stderr</code> • <code>jdk</code> (Requires JDK 1.4 or later) • <code>log4j</code> (Requires log4j libraries to be in the classpath) • A file name <p>If you specify <code>jdk</code> or <code>log4j</code> you must also perform appropriate configuration of the JDK or Apache <code>log4J</code> logging libraries.</p> <p>Default is <code>stderr</code>.</p>

Table 28 Cache Cluster Logging Properties (Cont'd)

Property	Notes
<code>java.property.tang osol.coherence.log .limit</code>	<p data-bbox="405 244 1282 340">Specifies the maximum number of characters that the logger daemon processes from the message queue before discarding all remaining messages in the queue.</p> <p data-bbox="405 357 1282 418">The message that causes the total number of characters to exceed the maximum is not truncated.</p> <p data-bbox="405 435 1282 565">All discarded messages are summarized by the logging system with a single log entry detailing the number of discarded messages and their total size. When the queue empties, the logger is reset and subsequent messages are again logged.</p> <p data-bbox="405 583 1282 678">The purpose of this setting is to avoid a situation where logging can itself prevent recovery from a failing condition, for example by contributing to timing issues.</p> <p data-bbox="405 696 1282 756">Logging occurs on a dedicated low-priority thread to further reduce its impact on the critical portions of the system.</p> <p data-bbox="405 774 515 800">Optional.</p> <p data-bbox="405 817 1282 852">Possible values are positive integers or zero (0). Zero implies no limit.</p> <p data-bbox="405 869 539 895">Default is 0</p>

Example Cluster Configuration Properties

The examples below show cluster configuration properties that are set to the same values in engine property files for all nodes in the cluster, and those that can be different in each node, and the Decision Manager related properties that are required only if Decision Manager is used, and that are set in one or more nodes as needed.

The examples show a set of cluster-level configuration properties enabling multicast discovery of cluster members, a distributed cache, and no backing store.

Other Lists of Engine Properties

The lists below focus on cluster-level properties. Other properties are listed in the following sections:

- [Summary List of Backing Store Related Properties on page 344](#)
- [Example Inference Agent Configuration Properties on page 318](#)
- [Query Agent Properties on page 327](#)



In the examples below, some example values are provided. Most properties have default values and can be omitted if the default value is suitable.

Cluster Properties That Must be the Same in All Nodes

```
# Cluster name:
java.property.tangosol.coherence.cluster Acme

# Multicast cluster member discovery:
java.property.tangosol.coherence.clusteraddress=224.3.3.1
java.property.tangosol.coherence.clusterport=33389
java.property.tangosol.coherence.ttl=4

# OR
# Well-known-address cluster member discovery:
# java.property.tangosol.coherence.wka1=hostname1
# java.property.tangosol.coherence.wka1.port=8088
# java.property.tangosol.coherence.wka2=hostname2
# java.property.tangosol.coherence.wka2.port=8088

# Cache cluster properties:
be.engine.cluster.hasBackingStore=false
be.engine.cluster.isCacheLimited=false
java.property.be.engine.limited.cache.back.size.limit
be.engine.cluster.multiEngineOn=true
java.property.tangosol.coherence.distributed.backupcount=2
```

```

be.engine.cluster.minCacheServers=2
java.property.tangosol.coherence.distributed.threads=10

# If Operational Descriptor Override File is used:
java.property.tangosol.coherence.override=file:/c:/tmp/my_tangosol
-coherence-override.xml

```

Use of an operational descriptor override is not generally required. For details see [Overriding and Extending the Operational Deployment Descriptor on page 411](#) and [Specifying Operational Override File Locations on page 413](#).

Cluster Properties That Can Differ Across Nodes

(The cache server property is not exactly a cluster property, but for sake of completeness it is included here.)

```

# Cache logging properties
java.property.tangosol.coherence.log.level=3
java.property.tangosol.coherence.log=stderr
java.property.tangosol.coherence.log.limit=4096

# Local Storage
java.property.tangosol.coherence.distributed.localstorage=true

# Deploy this node as a cache server?
be.engine.cacheServer=true

# If multiple hosts exist on one machine:
java.property.tangosol.coherence.localhost=
java.property.tangosol.coherence.localport=

```

Cluster Properties Set in Some Nodes, As Needed

```

# If Decision Manager is Used:
be.engine.cluster.externalClasses.classLoader=true
be.engine.cluster.externalClasses.path=c:/myfolder/

```

Chapter 21 **Configuring Inference Agents (Cache OM)**

This chapter explains how to configure inference agents within the cache cluster.

Note that In Memory and in Persistence object management options do not provide features at the inference agent level, so they don't require inference agent configuration.

Topics

- [*Inference Agent Configuration Overview, page 306*](#)
- [*Inference Agent Runtime Architecture, page 308*](#)
- [*Configuring Inference Agents—TIBCO Designer, page 309*](#)
- [*Configuring Inference Agents—Engine Properties, page 313*](#)
- [*Inference Agent Engine Property Reference, page 315*](#)
- [*Example Inference Agent Configuration Properties, page 318*](#)

Inference Agent Configuration Overview

For a definition and overview of inference agents and their role within a cache cluster, see [Inference Agents on page 270](#). This section provides more detailed information and instructions for inference agent configuration.

Local L1 Cache

For good performance, each inference agent also has a local cache, called the L1 cache, in addition to the distributed entity caches. An agent checks for an object first in the local cache. If an object is found there, the agent checks that it is current (comparing it with the object in the cache cluster). If the local object is current it is used, saving serialization time.

You can configure the size of the cache. See `Agent.AgentGroupName.L1CacheSize` in [Configuring Inference Agents—Engine Properties on page 313](#).

Access to Cached Objects

All inference agents (across all agent groups) and all query agents in a cache cluster share access to the same set of cached objects, which is maintained for the cache cluster by the object management layer.

Concepts, events, and scorecards are not shared in the same way. See [Designing With Multiple Active Inference Agents on page 274](#) for more details.

Note that when reading from cache, `BusinessEvents` first attempts to get data from the L1 cache. If the data is not available in this local cache, or it is not current, `BusinessEvents` retrieves the data from the cache cluster. If the data is not in the cache cluster, and a backing store is in place, the cache cluster transparently loads it from backing store and gives it to the requesting engine. (The agents do not interact with the backing store directly, except during recovery.)

Multiple Agent Write and Replication Step

Changes made in one instance of an agent must be replicated across all other instances.

Each agent in the cluster has an `AgentTxn-agentId` cache. (The `agentId` is internally generated.) This cache stores the change list for the agent. All changes to objects that occur in one RTC are grouped into one change list, or transaction.

All agents in the same group subscribe to the other agent's changes using this cache. They apply those changes to their working memories, so that the working memory of all agents stays synchronized.

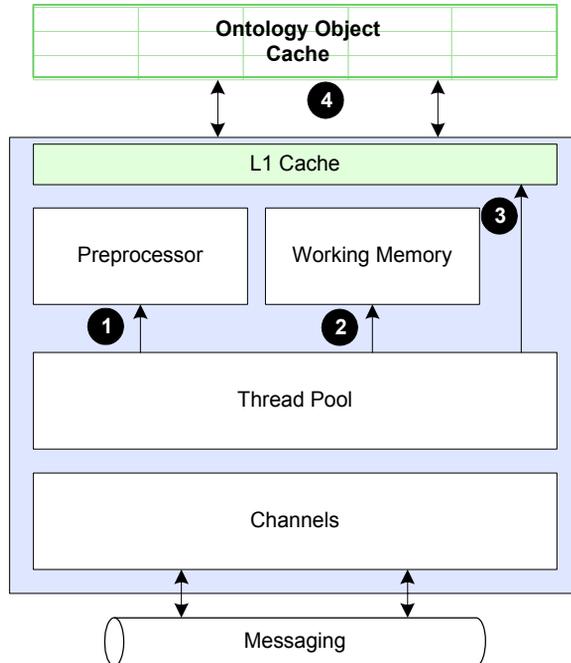
See [Designing With Multiple Active Inference Agents on page 274](#) for more information on the effect of multiple agents on project design and on runtime behavior.

Inactive Agents and Fault Tolerance Behavior

When an agent group is configured to have one or more inactive members, those agents act as secondary agents ready to take over in the event of failure of other agents. See [Fault Tolerance Between Inference Agents in a Group on page 272](#) for an overview.

Inference Agent Runtime Architecture

The diagram below shows how messages come into an inference agent, and are processed through the system, as explained below. The section [Multiple Agent Write and Replication Step on page 306](#) explains how changes in one agent's working memory are communicated to all instances of the agent.



Single Agent Flow

Each event entering a channel is handled by a worker thread in the thread pool. You can define a number of worker threads (see [Input Destinations on page 364](#)). The flow is as follows:

1. Preprocessors (if any) are executed first.
2. Then if any objects are to be asserted, they are asserted into working memory, using the same thread.
3. Changes to objects are written to the L1 cache in a batch at the end of the RTC.
4. Then changes are written to the cache cluster.
5. If there is a backing store, those changes are then written to the backing store.

Configuring Inference Agents—TIBCO Designer

Configure settings for each BAR that you will deploy as an inference agent to provide the desired behavior.

This section focuses on object management aspects of configuration. See [Configuring a BAR File for Deployment on page 360](#) for other aspects of configuring an inference agent.

To configure Cache object management for an inference agent, you must also configure engine properties. See [Configuring Inference Agents—Engine Properties on page 313](#).

To Configure Cache OM for an Inference Agent in TIBCO Designer

See [BusinessEvents Archive Resource Object Management Tab—Cache Settings Reference on page 311](#) for a reference table of cache-related settings.

1. Open your project in TIBCO Designer.
2. Add or open a BAR resource within an EAR resource in your project and select the BAR resource for configuration.
3. In the Type field on the Configuration tab select Inference. Configure settings on this and other tabs as explained in [Configuring a BAR File for Deployment on page 360](#).
4. Select the **Object Management** tab and do the following
 - a. From the Type drop-down list, select **Cache**.
 - b. Agent Group Name: Provide a name that is shared by all deployed instances of this agent. As a recommendation, use the BAR name.
 - c. Cache Configuration File and Resource fields: Not used in this release. Ignore these fields.

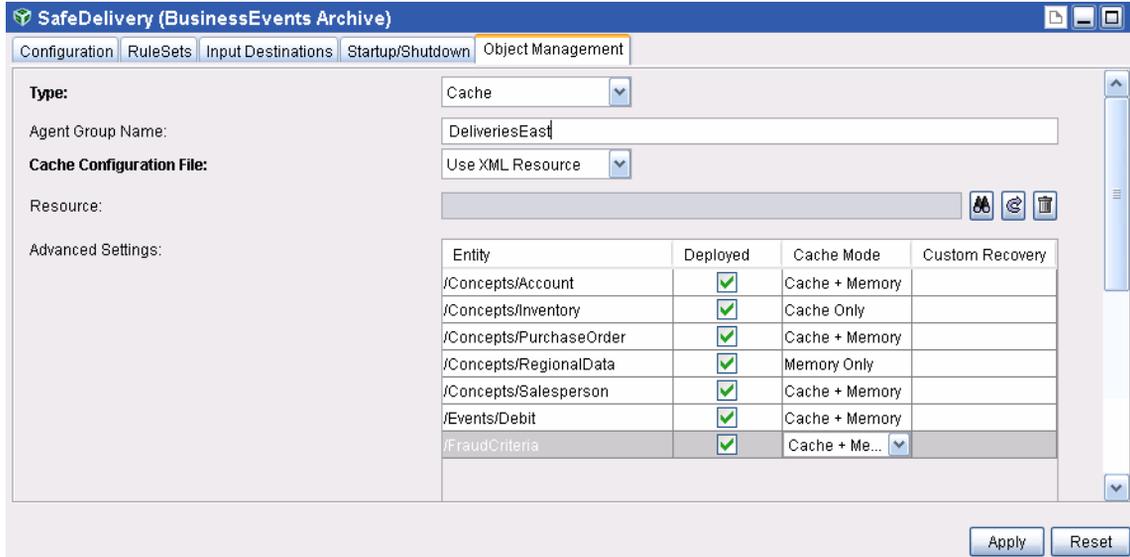
Advanced Settings— Cache Modes

The Entity column shows the project path of all concepts, events, and scorecards in the project (that is path to the item in the design-time project).

The Deployed setting and Custom Recovery setting are not used in this release.

See [Working With Cache Modes on page 282](#) and [Design Constraints With Cache Only Cache Mode on page 285](#) for more information about cache modes.

5. In the Cache Mode column, choose one of the options from the drop-down list for each entity:
 - Cache + Memory (default).
 - Cache Only
 - Memory Only



SafeDelivery (BusinessEvents Archive)

Configuration | RuleSets | Input Destinations | Startup/Shutdown | **Object Management**

Type: Cache

Agent Group Name: DeliveriesEast

Cache Configuration File: Use XML Resource

Resource:

Advanced Settings:

Entity	Deployed	Cache Mode	Custom Recovery
/Concepts/Account	✓	Cache + Memory	
/Concepts/Inventory	✓	Cache Only	
/Concepts/PurchaseOrder	✓	Cache + Memory	
/Concepts/RegionalData	✓	Memory Only	
/Concepts/Salesperson	✓	Cache + Memory	
/Events/Debit	✓	Cache + Memory	
/FraudCriteria	✓	Cache + Me...	

Apply Reset



Cache Modes and Related Concepts

Concepts can be related to each other concept through inheritance, containment or reference. All related concepts must use the same cache mode.

For example, if the parent is Cache plus Memory and the child is Memory Only, it inconsistent object graphs will result on recovery.

See [Working With Concepts and Concept Relationships on page 90](#) for more details about the relationships between concepts.

6. Click **Apply** and **Save**.

BusinessEvents Archive Resource Object Management Tab—Cache Settings Reference

Other tabs of the BusinessEvents Archive Resource are documented in [Chapter 26, Deploytime Configuration, on page 353](#).

Field	Global Var?	Description
Type	No	<p>One of In Memory, Persistence, or Cache. Default is In Memory. When you select Cache, cache configuration settings appear, as shown in this table.</p> <p>For In Memory, no additional configuration is required in this tab. See Configuring In Memory Object Management on page 243 for other aspects of In Memory OM. For Persistence, see Chapter 16, Configuring Persistence Object Management, on page 251.</p>
Agent Group Name	No	<p>In multi-engine mode, all agents deployed using this BAR belong to the same agent group and share this name. The name is used for various engine configuration properties. (BusinessEvents also maintains an internal ID in addition to this key to uniquely identify agents.)</p> <p>See Configuring Inference Agents—Engine Properties on page 313.</p> <p>A default value is provided if you don't provide a value. For the first BAR resource where no value is explicitly set, the value is set to agent1. For the next BAR resource where no value is explicitly set, the value is set to agent2, and so on.</p>
Cache Configuration File	No	<p>Not used in this release.</p> <p>See Configuring Caching Scheme, Multi-Engine, and Cluster Properties on page 296.</p>
Advanced Settings		
Entity	No	Shows the project path of all concepts, events, and scorecards in the project
Deployed	No	Not used in this release.

Field	Global Var?	Description
Cache Mode	No	Select one of the following: <ul style="list-style-type: none">• Cache + Memory (default).• Cache Only• Memory Only See Working With Cache Modes on page 282 for details and a warning about using these settings.
Custom Recovery	No	Not used in this release.

Configuring Inference Agents—Engine Properties

The table in this section explains what engine properties you need to set to configure various aspects of an inference agent.

This section builds on cluster-level configuration settings and assumes they are in place. See the following chapters for those settings:

- [Chapter 19, Configuring Cache Cluster Discovery, page 287](#)
- [Chapter 20, Configuring Cache Cluster Settings, page 295](#)

An inference agent is equivalent to one BAR in an EAR.

See also [Configuring Inference Agents—TIBCO Designer on page 309](#).

Defining an Agent Group

The agent group name is defined in the Agent Group Name field of the Object Management tab of the BusinessEvents Archive (BAR) resource (when Cache is selected as the object management type).

Thus the agent group name applies to the BAR (rule session) and not to the engine as a whole. One deployed engine can have multiple agent groups within it, each BAR having a different agent group name.

Note that the term *AgentGroupName* in tables references this name.

Defining a Unique Key for Each Agent

Scorecard information is instance-specific and is not shared across agents. The `Agent.AgentGroupName.key` property ensures that on recovery, correct scorecard values are returned to the correct agents. (This is different from the internal ID generated for an agent.)

Local Storage Generally Not Enabled on Inference Agents

For performance reasons, it is recommended that you disable local storage of cache data on inference agents and query agents. Instead use dedicated cache nodes for storage of cache data. By default the following property is set to true:

```
java.property.tangosol.coherence.distributed.localstorage=true
```

You must explicitly set it to false to disable local storage.

Configuring Fault Tolerance and Load Balancing for an Inference Agent Group

With multi-engine mode, you can configure a group of inference agents to have some active and some inactive members. However note that, as explained in the section [Load Balancing and Fault Tolerance Between Inference Agents on page 272](#), even if all agents are configured to be active, they perform fault tolerance implicitly: when one active agent fails, the other active agents rebalance the workload as needed.

To define the number of *active* agents in a group, set the property `Agent.AgentGroupName.maxActive`.

To configure some agents to remain inactive until needed (that is, until an active node fails) do the following:

- Set the `maxActive` value to a number lower than the number of deployed instances.
- Set each agent's `Agent.AgentGroupName.priority` property as needed to define which of the deployed agents become active, and which act as fault tolerant secondary agents.

With single-engine mode, you only need to use the `priority` property. Only one agent is active at a time, so the `maxActive` property is not required. Its default value is 1.

Tuning Agent Performance

As well as load balancing features, you can use the following properties to tune performance:

`Agent.AgentGroupName.l1CacheSize`

`Agent.AgentGroupName.threadcount`

`Agent.AgentGroupName.recoveryPageSize`

See [Table 29, Inference Agent Engine Properties, on page 315](#) for details.

Inference Agent Engine Property Reference

Table 29 Inference Agent Engine Properties (Sheet 1 of 3)

Property	Notes
<code>Agent.AgentGroupName.maxActive</code>	<p>Specifies the maximum number of active instances of the agent. This value is used to limit the number of active instances in an agent group.</p> <p>A value of 0 indicates an unlimited number of active instances.</p> <p>Must be the same for all nodes where instances of this agent (that is, agent group members) are deployed.</p> <p>Default is 1.</p>
<code>Agent.AgentGroupName.priority</code>	<p>Specifies the priority of the agent.</p> <p>The priority indicates the order in which inactive agents become active, and conversely, the order in which active agents become inactive, when new agents join the cluster.</p> <p>The <i>lower</i> the number, the higher the agent is in the activation priority list. For example, an agent with priority 2 has a higher priority than an agent with a priority of 6.</p> <p>Set differently in each node's TRA file to determine the order of each agent instance for startup, as well as failover and fallback in fault tolerance situations.</p> <p>Default is 1.</p>
<code>Agent.AgentGroupName.key</code>	<p>Specifies a value that uniquely identifies an instance of an agent within its agent group.</p> <p>Set differently in each node's TRA file to identify each agent uniquely. (BusinessEvents also maintains an internal ID in addition to this key to uniquely identify agents.)</p> <p>Required for recovery of scorecards. Recommended in all cases, for situations that require an agent instance to be uniquely identified.</p>

Table 29 Inference Agent Engine Properties (Sheet 2 of 3)

Property	Notes
<code>Agent.AgentGroupName.l1CacheSize</code>	<p>Specifies the maximum number of objects in each agent's L1Cache. The L1 cache is a local cache used by the inference agent for local access to recently used objects. It is used to optimize access to objects.</p> <p>The value can be 10,000 or above. Values less than 10,000 are interpreted as 10,000.</p> <p>Note that this setting does not relate to local storage. It is required for all inference agent nodes.</p> <p>Default is 10000 (unit is objects).</p>
<code>java.property.tangosol.coherence.distributed.localstorage</code>	<p>When you use a distributed cache scheme, you decide whether to use some JVM memory for local cache storage based on your needs.</p> <p>Set to true to enable the object management layer to use the BusinessEvents server's JVM memory for storing cached objects.</p> <p>Set to false to keep JVM memory free for other uses.</p> <p>TIBCO recommends that you disable local storage on inference agents and query agents, and enable it only on cache server nodes.</p> <p>Default is true.</p>
<code>Agent.AgentGroupName.threadcount</code>	<p>Specifies the number of threads used by the inference agent L1 cache to write to the cache cluster.</p> <p>The default value is the same number as there are processors available to the JVM.</p> <p>See also Configuring Caching Scheme, Multi-Engine, and Cluster Properties on page 296 for a related thread property.</p>

Table 29 Inference Agent Engine Properties (Sheet 3 of 3)

Property	Notes
<code>Agent.AgentGroupName.recoveryPageSize</code>	<p data-bbox="599 239 1278 303">Specifies the number of entries per page to be used while recovering objects from the cache.</p> <p data-bbox="599 321 1292 477">For example, if you set the value to 10,000, then the engine loads handles in blocks of 10,000, instead of trying to load them in a single batch. Smaller batch sizes result in slower recovery. Experiment with batch size to establish the best batch size to use for your environment.</p> <p data-bbox="599 494 1256 558">A value of 0 means that the objects are recovered in one iteration.</p> <p data-bbox="599 576 742 607">Default is 0.</p>

Example Inference Agent Configuration Properties

The example below shows inference agent configuration properties that are set in engine property files for all agents in an agent group with three active agents.

The file would also be configured with cluster-level settings as shown in [Example Cluster Configuration Properties on page 303](#).

Values are for example use only. Your settings may differ.

Agent 1

```
Agent.AcmeGroup.maxActive=3
Agent.AcmeGroup.priority=1
Agent.AcmeGroup.key=agent1
Agent.AcmeGroup.l1CacheSize=20000
java.property.tangosol.coherence.distributed.localstorage=false
Agent.AcmeGroup.threadcount=64
Agent.AcmeGroup.recoveryPageSize=10000
```

Agent 2

```
Agent.AcmeGroup.maxActive=3
Agent.AcmeGroup.priority=2
Agent.AcmeGroup.key=agent2
Agent.AcmeGroup.l1CacheSize=20000
java.property.tangosol.coherence.distributed.localstorage=false
Agent.AcmeGroup.threadcount=64
Agent.AcmeGroup.recoveryPageSize=10000
```

Agent 3

```
Agent.AcmeGroup.maxActive=3
Agent.AcmeGroup.priority=3
Agent.AcmeGroup.key=agent3
Agent.AcmeGroup.l1CacheSize=20000
java.property.tangosol.coherence.distributed.localstorage=false
Agent.AcmeGroup.threadcount=64
Agent.AcmeGroup.recoveryPageSize=10000
```

Chapter 22 **Configuring Cache Servers (Cache OM)**

This chapter explains how to configure cache servers in a cache cluster. Cache servers are used as storage nodes for cache data.

Topics

- [Cache Server Configuration Overview, page 320](#)
- [Configuring a Cache Server, page 321](#)
- [Engine Property Reference for Cache Server Settings, page 322](#)
- [Adding a Cache Server to a Running Production System, page 323](#)

Cache Server Configuration Overview

Unlike inference and query agents, dedicated cache server nodes are defined and deployed at the node level. No other types of agents are permitted on a dedicated cache server node. No configuration is required in TIBCO Designer.

For an overview of cache servers and their role within a cache cluster, see [Cache Server Nodes \(Storage Nodes\) on page 271](#). This chapter deals with cache server configuration.



Agent Nodes Functioning as Cache Servers It is possible, but not generally recommended, to enable inference and query agent nodes to act as cache servers in addition to their other functions. Using dedicated cache server nodes for data storage is more efficient and more scalable for production scenarios.

Deploying a Node as a Dedicated Cache Server

To deploy a node as a cache server only, ignoring any other agent-level functionality configured in the EAR, deploy any project EAR file using a correctly configured engine property file (see [Configuring a Dedicated Cache Server Node](#).)

If the `be.engine.cacheServer` property is set to true, any agent-related configuration in the EAR file is ignored and the node is internally configured to have a cache server agent. All entities (concepts and events) are available.

Memory and Heap Size Guideline for Cache Servers

The amount of memory you need for cache servers depends on factors such as how many objects you have, their object management configuration, and whether you are using limited or unlimited cache.

You must find an appropriate balance for your projects between too little memory, which leads to too much time spent in garbage collection, and too much memory, which leads to longer garbage collection cycles. The optimal heap size depends on the needs of your projects, such as how much data is kept in each cache server, how many cache servers are used, and whether the cache is limited or unlimited.

For example, if you use a JVM heap size of 1024 MB (1GB), in order to minimize the impact of garbage collection, about 75% of the heap can be used to store cache items, which means about 768 MB per heap. The other 25% is then available for garbage collection activities.

Different operating systems have different requirements so make adjustments as required.

Configuring a Cache Server

This section builds on cluster-level configuration settings and assumes they are in place. See the following chapters for those settings:

- [Chapter 19, Configuring Cache Cluster Discovery, page 287](#)
- [Chapter 20, Configuring Cache Cluster Settings, page 295](#)

Configuring a Dedicated Cache Server Node

Do the following to ensure that a node functions *only* as a cache server node.

1. Begin with an engine property (TRA) file that has been configured for the cache cluster, or add the properties to a supplementary property file you will use in addition to the configured `be-engine.tra` file.
2. Ensure that the following engine property is present and set to true:
`be.engine.cacheServer=true`
3. Ensure that the following engine property, if present, is *not* set to false:
`java.property.tangosol.coherence.distributed.localstorage`
(By default this property is set to true.)

Configuring an Agent Node to Also be a Cache Server

By default, an EAR file containing BAR files for inference or query agents (or both), also deploys as a cache server.

This enables simple deployment in test environments, with minimal configuration and using fewer machines. It is not generally recommended for production environments.

To configure an agent node to also be a cache server, do one of the following:

- Omit the following properties to use the default behavior.
- Specify the following properties with their default values, as shown:

```
be.engine.cacheServer=false
java.property.tangosol.coherence.distributed.localstorage=true
```

Engine Property Reference for Cache Server Settings

The following table provides a reference to all the engine property file settings that relate to setting up a cache server.



Refer to the file `BE_HOME/bin/be-cache-config.tra` for example usage of cache-related properties. This file is provided for your convenience. You can copy properties as needed into your engine TRA file and complete the configuration.

Table 30 *BusinessEvents Engine Properties for Configuring a Cache Server*

Property	Notes
<code>be.engine.cacheServer</code>	<p>You configure a server to work as a cache server <i>only</i> by setting <code>be.engine.cacheServer true</code>.</p> <p>When an EAR file is deployed with this setting, the deployed node functions as a cache server only. It does not perform any normal BusinessEvents work, that is, it does not process rules or have a Rete network. This is true even if the EAR file contains BAR resources configured for inference or query agents.</p> <p>Local Storage property is true by default Note that the property: <code>java.property.tangosol.coherence.distributed.localstorage</code> is set to true by default, so there is no need to set it explicitly in your TRA file. If a node is deployed with See Table 29, Inference Agent Engine Properties, on page 315 for details on this property.</p> <p>Default value is false.</p>

Adding a Cache Server to a Running Production System

As system load increases, you can add more cache servers to handle the work. This section explains how to add a cache server to a running production system.

Multicast Configuration

For multicast configuration, configure and deploy the new cache server in the usual way. You do not have to restart the system. The system behaves appropriately at the next restart. See [Discovering Cache Cluster Members Using Multicast on page 290](#).

Well-Known Address Configuration

For well-known address configuration, add the well-known address information for the existing set of new cache servers in the TRA file of the new server and deploy it in the usual way. However, before the next restart, you must update all TRA files to include well-known address information for the new cache server. See [Discovering Cache Cluster Members Using Well-Known-Addresses on page 292](#).

Chapter 23 **Configuring Query Agents (Cache OM)**

See [Query Agents on page 271](#) for an introduction to query agents.

See *TIBCO BusinessEvents Language Reference* for details on using the query language. This chapter explains only how to configure a query agent.

Topics

- [Configuring Query Agents—TIBCO Designer, page 326](#)
- [Configuring Query Agents—Engine Properties for Performance, page 327](#)

Configuring Query Agents—TIBCO Designer

Configuring a query agent is similar to configuring an inference agent, but simpler.

To Configure a Query Agent in TIBCO Designer

1. Open your project in TIBCO Designer.
2. Add or open a BAR resource within an EAR resource in your project and select the BAR resource for configuration.
3. In the Type field on the Configuration tab select **Query**.
(The Rule Sets tab is disabled because rule sets are not used by query agents.)
4. In the Input Destinations tab, select the destination or destinations that the agent listens to, for events that trigger queries. Also select destinations to which query results are sent.
5. Select the Startup/Shutdown tab and select rule functions that contain query definitions.

It is a best practice to create the query definitions at system startup, using startup rule functions. See *TIBCO BusinessEvents Language Reference* for details about query definitions.

- a. Click the plus sign to browse to and select a function. Only rule functions that take no argument and whose Validity field (in the rule function's Configuration tab) is set to Action appear in the list.
- b. If you have selected more than one function in one list, click the up and down arrows to arrange the functions as needed. BusinessEvents calls the functions in the order listed.

Object
Management

6. Select the **Object Management** tab. Enter an Agent Group Name. As a recommendation, use the BAR name.

Note that the Type field is preset to Cache, which is the only valid option for a query agent. The only field you can set for a query agent is the Agent Group Name.

7. Click **Apply** and **Save**.

See [BusinessEvents Archive Resource Object Management Tab—Cache Settings Reference on page 311](#) and [Configuring a BAR File for Deployment on page 360](#).

Configuring Query Agents—Engine Properties for Performance

Query agents have only a few agent-level properties. (Query agents are stateless and are not run in load-balanced or fault-tolerant groups and do not require properties relating to those features.)

This section builds on cluster-level configuration settings and assumes they are in place. See the following chapters for those settings:

- [Chapter 19, Configuring Cache Cluster Discovery, page 287](#)
- [Chapter 20, Configuring Cache Cluster Settings, page 295](#)

Query Agent Properties

Query agent properties are used to configure the query agent local cache:

```
be.agent.query.localcache.maxelements
be.agent.query.localcache.evictseconds
be.agent.query.localcache.prefetchaggressive
```

The query agent's local cache stores cache data locally for efficient reuse. The local cache listens to and synchronizes the locally stored entity instances with those in the main cache, so that the local cache stays up-to-date.

The prefetch feature improves performance, but CPU and memory usage increases as a result of the aggressive prefetching. You may have to try different values till you find the optimal settings for your environment.

If the default values provide good performance, you do not have to add these properties when configuring a query agent.

Table 31 *BusinessEvents Engine Properties for Query Agent Local Cache*

Property	Notes
<code>be.agent.query.localcache.maxelements</code>	Specifies the maximum number of entities that can be stored in the query-agent's local cache. When the threshold is reached, oldest entities are removed first. Default value is 50000.
<code>be.agent.query.localcache.evictseconds</code>	Specifies an age limit on the cached entities in seconds. After this period, they are removed from the local cache. Note Age resets each time an entity is accessed by the query engine. Default value is 900.

Table 31 *BusinessEvents Engine Properties for Query Agent Local Cache*

Property	Notes
<code>be.agent.query.localcache .prefetchaggressive</code>	If set to true, then objects required for a query are prefetched while the query is executing. Ensure that the cache size is large enough to accommodate objects that are prefetched. Default is false.

Chapter 24 **Setting up a Backing Store Database**

A backing store enables persistent backup of data in the cache. Use of a backing store enables recovery in the event of a system-wide failure. After you configure the caching solution for your project, you can then add a backing store.

This chapter covers setting up the backing store database.

You must also configure backing store settings. See [Chapter 25, Project Configuration for Backing Store](#), on page 341.

Topics

- [Backing Store Database Setup Overview](#), page 330
- [Resources Required for Setting Up the Database](#), page 332
- [Backing Store Database Configuration Tasks](#), page 334
- [Updating an Existing Backing Store Database Schema](#), page 338

Backing Store Database Setup Overview

You can implement a backing store for use with any Cache object management option. At system startup, data is loaded into the cache from the backing store. During regular operation, the cache persists the data that is written to it in the backing store. This happens at the end of each RTC.

If you use a limited-size cache, you must use a backing store so that data evicted from the cache is not lost.

This overview provides summary information about setting up the database, with references to the sections that provide detailed information.

You must also configure various properties to enable and configure backing store functionality. See [Chapter 25, Project Configuration for Backing Store, on page 341](#) for details.

Backing Store Requirements

Backing store functionality has been tested with Oracle 10g Enterprise Edition and Oracle Database 10g Express Edition (see the readme file for specific version information). You can download the Express Edition for development use from the Oracle web site.

Instructions in this chapter assume you are working with a local database for testing. For production deployments, you might have to ask a database administrator to create a database user for you. You should then be able to run the other SQL scripts yourself, logged on as the user created by the administrator.

Before You Begin Database Setup

- Develop your caching solution and test it. See [Chapter 17, Understanding Cache OM and Multi-Engine Features, on page 263](#) and chapters following.
- Ensure that you have access to a supported database management product and can create a user and tables.

Backing Store Database Setup Tasks

Backing store configuration tasks for each backing store, and background information are summarized below.

- [Resources Required for Setting Up the Database on page 332](#) outlines all the resources you need to set up a backing store.

- [Backing Store Database Configuration Tasks on page 334](#) explains several tasks you must complete in order to configure the database schema, including generating SQL scripts for your project and running them to create the database user and tables.

Extra Procedure to Handle Long Identifier Names

A known limitation in Oracle means that each identifier name cannot exceed 30 characters in length. If you have longer names, remember to complete [Task C, Shorten Long Names Using the Aliases File, on page 335](#), to give an alias to each long identifier name.

Every entity, property, or state machine whose name exceeds 30 characters in length has an entry in the generated *yourname.aliases* file (For example, *acme.aliases*). This file has no entries if all names have 30 characters or less.

Note that you must run the `be-oradeploy` utility again, after updating the *yourname.aliases* file (as mentioned in the procedure.)

After You Finish Database Setup

You must also configure various properties to enable and configure backing store functionality. See [Chapter 25, Project Configuration for Backing Store, on page 341](#) for details.

Maintaining a Backing Store—If Ontology Object Definitions Change

If you add, change, or delete ontology object definitions, the backing store schema will no longer match your ontology. You must update the backing store schema so it still matches the ontology. See [Updating an Existing Backing Store Database Schema on page 338](#) for details

Resources Required for Setting Up the Database

The table below lists resources required and sections following explain the procedures for setting up backing store tables.

Table 32 Resources Required for Backing Store Implementation

Resource	Default Location and Purpose
Oracle 10G	This feature has been tested with Oracle 10g Enterprise Edition and Oracle Database 10g Express Edition (see the readme file for specific version information).
Oracle JDBC Driver Oracle Thin driver recommended	Oracle JDBC drivers are not provided. Download the client from the Oracle web site or find the drivers in your Oracle Client installation. Copy the JAR files (for example, ojdbc14.jar) to <i>BE_HOME/lib/ext</i> or elsewhere in your class path. Backing store functionality has been tested with Oracle Thin driver.
<i>be-oracle.jar</i>	<i>BE_HOME/lib</i> A JAR file required for backing store functionality.
<i>be-oradeploy.exe</i> <i>be-oradeploy.tra</i>	<i>BE_HOME/bin</i> Executable and property files used to generate SQL scripts (see below).
Provided SQL Scripts	<i>BE_HOME/bin</i>
<i>initialize_database.sql</i>	The <i>initialize_database.sql</i> script drops the user (and therefore all the tables) and adds the user again. By default the user is called <i>be_user</i> with the password <i>be_user</i> and has DBA rights. Edit the script if you want the user to have a different name or different rights.
<i>base_types.sql</i>	The <i>base_types.sql</i> script defines the base types, corresponding to the BusinessEvents object data structure.
<i>create_tables.sql</i>	The <i>create_tables.sql</i> script creates the tables that are used to maintain the metadata.

Table 32 Resources Required for Backing Store Implementation (Cont'd)

Resource	Default Location and Purpose
Generated SQL Scripts	These scripts are generated when you run the <code>be-oradeploy</code> executable, as explained below, and they are located in the same directory where you run <code>be-oradeploy</code> .
<code>yourname.sql</code>	
<code>yourname.aliases</code>	You provide the value of <code>yourname</code> when you generate the scripts.
<code>yourname_remove.sql</code>	The <code>yourname.sql</code> script is executed after the provided scripts are executed, as explained in the procedures below. It creates schema tables and types.
<code>yourname_cleanup.sql</code>	
<code>yourname_alter.sql</code>	
	The <code>yourname.aliases</code> script has entries if your project has names longer than 30 characters. You must perform a procedure to provide aliases for long names, and then regenerate the SQL scripts again using the <code>be-oradeploy</code> utility. The procedure is explained in Task C, Shorten Long Names Using the Aliases File .
	The <code>yourname_remove.sql</code> script can be used as needed. It removes the database schema. You can use it to reset the project
	The <code>yourname_cleanup.sql</code> script can be used as needed. It truncates the tables.
	The <code>yourname_alter.sql</code> script is for use in schema migration. Generated only if properties are added to <code>be-oradeploy.tra</code> , to identify the existing database and its user and password. See Updating an Existing Backing Store Database Schema on page 338 .

Backing Store Database Configuration Tasks

As with any procedure that modifies your data, ensure that you have made backups before you begin.



Existing Backing Stores: After Ontology Object Definition Changes

See [Updating an Existing Backing Store Database Schema on page 338](#) for the impact of different kinds of changes and how you can update the backing store schema.

Task A Prepare Files

1. Open your project in TIBCO Designer, and build the EAR file. Model information in the EAR will be used to build tables in the database.
2. Ensure that `be-oracle.jar` is located in `BE_HOME/lib` (or other location in your class path).
3. Copy your JDBC drivers file to `BE_HOME/lib/ext` (or other location in your class path). These files are part of the Oracle Client software.

Task B Generate the SQL Scripts

Open a command window and navigate to `BE_HOME\bin`. Run `be-oradeploy.exe` using a command with the following format:

```
be-oradeploy [-p property file] [-o Oracle schema output file] [EAR Path] [-h]
```

For example:

```
be-oradeploy -o acme c:/BEProjects/MyEar.ear
```



If you are not running from the default folder (`BE_HOME/bin`) or if you are not using the default files, you must provide both the `--propFile` and the `-p` parameters, and pass them the fully qualified name of the `tra` file. For example:

```
be-oradeploy --propfile c:\mypath\myfile.tra -p
c:\mypath\myfile.tra
```

The options are explained in the following table:

Option	Description
<code>-p, /p,</code> <code>-property, or</code> <code>/property</code>	Specifies the property file. If not specified, the default property file is used, that is, <code>be-oradeploy.tra</code> in the current directory. See note above.
<code>-o</code>	Specifies the Oracle schema output filename for deployment.
<code>-h, /h, or /help</code>	Displays this help.

In the command window, you see various messages as schema definition commands are created in the generated scripts and the scripts are created.

The generated scripts appear in the directory where you ran the executable. For example, if you provided the schema output filename `acme`, you would see files called `acme.sql`, `acme.aliases`, `acme_cleanup.sql`, and `acme_remove.sql`.

The user-defined part of the database schema is created in an Oracle schema file as schema definition commands. In [Task E](#) you run this script (together with provided scripts) to build the schema in the database.



Keep the `yourname.sql` file for future reference. If you modify your ontology, you must also modify your database schema. You can compare the original with the newly generated `yourname.sql` and run only those commands that are different in the new file.

Task C Shorten Long Names Using the Aliases File

If the aliases file has entries, do the following to provide short aliases for all names longer than 30 characters (see [Extra Procedure to Handle Long Identifier Names on page 331](#) for more details):

1. Open the `yourname.aliases` file for editing.
2. For each entity, property, and state machine name that exceeds 30 characters in length, provide an alias using a name that is shorter than 30 characters. Ensure the name is unique.

For example, you would modify the following entry:

```
TABLE.D_NewConceptNewConceptNewConceptNewConcept.alias=
D_NewConceptNewConceptNewConceptNewConcept
```

With a short name such as:

```
TABLE.D_NewConceptNewConceptNewConceptNewConcept.alias=
D_NewConceptNewConceptNewCon
```

3. Perform [Task B, Generate the SQL Scripts](#), again. This time, the aliases you created are used.

Task D Run the Initialize Database Script to Create the Oracle User



Running the `initialize_database.sql` script drops all existing backing store tables. If you have data you need to retain, contact TIBCO support for assistance.

In the script `initialize_database.sql` script, the Oracle user is set to `BE_USER`, with password `BE_USER`. You can edit the script as needed to change these default settings. The documentation uses the default username and password.

1. Login to Oracle Server as the system user.
2. Navigate to the location of the scripts (or copy them all to the `BE_HOME/bin` directory) and open an SQLPlus prompt. (Open a command window, type `SQLPLUS` then provide the system user credentials.)
3. Type `@initialize_database.sql` to run the provided script, `initialize_database.sql`. You see messages like the following:

```
DROP USER be_user CASCADE
*
ERROR at line 1:
ORA-01918: user 'BE_USER' does not exist

User created.
Grant succeeded.
SQL>
```

Note that the script assumes that it has been run before. It deletes the user before creating it again. This means you can run the script again without having to take extra steps. This is useful for testing purposes.

Task E Login as the Oracle User and Run SQL Scripts

In this step, you run scripts to create the database schema under the user you created. The schema combines the definitions in `base-types.sql`, `create-tables.sql`, and the generated *Oracle schema output file* file (`acme.sql` as an example).

Note that these scripts also perform cleanup before creating the schema. The first time you run the scripts, you see harmless error or warning messages because there is nothing to delete.

1. Login to the Oracle server as `BE_USER`, password `BE_USER` (or whatever username and password you set in the script in [Task D](#)).
2. Navigate to the location of the scripts and open an SQLPlus prompt. Identify yourself as `be_user` with password `be_user`.
3. At the SQL prompt, type the following to run each script in turn:
 - a. `@base_types.sql`
 - b. `@create_tables.sql`
 - c. `@yourname.sql` (for example, `@acme.sql`)

Your database tables are now configured for use. Additional configuration activities are explained in [Chapter 25, Project Configuration for Backing Store](#), on page 341.

Updating an Existing Backing Store Database Schema

If you change the project ontology, that is, if you create, alter or delete a concept or an event, you must update the backing store schema so it matches the updated ontology. You must do this before you deploy the updated project.

Not all changes can be automatically migrated. Manual migration is required for such changes.

What the Schema Migration Utility Can and Can't Handle Automatically

The migration utility handles the following:

- Addition of entity types and attributes. New entity types and attributes are added to the database schema.
- Deletion of entity types and attributes. Deleted entity types and attributes are dropped from the database schema.

The utility can handle only certain changes to existing entities and attributes, depending on the datatype and on Oracle functionality. The utility does not handle changes to the data type of an existing attribute, for example, changing a String attribute to a number attribute.



If the utility encounters any change in the schema that cannot be migrated automatically, then the migration script is not generated. In this case, migrate the data and the schema manually. Contact TIBCO support for assistance about handling of specific changes.

To Update an Existing Backing Store Database Schema

Before you begin:

- Gracefully shut down the deployed application (all agents and cache servers).
 - Back up your existing database.
1. Generate the updated EAR file for the modified project.
 2. Save existing copies of *yourname.sql* and *yourname.aliases* so you can compare them with the files you will generate for the changed project.
 3. Open the *be-oradeploy.tra* file for editing and add the following properties:

```
be.oracle.schemamigration.url=SourceDbURL
be.oracle.schemamigration.user=username
be.oracle.schemamigration.pswd=password
```

Where the username and password are those you set up in [Task D, Run the Initialize Database Script to Create the Oracle User](#), on page 336. Examples of the database URL are provided in the section [Adding a JDBC Connection Resource](#) on page 345.

The properties enable the program to compare the schema of the existing database with the schema of the project EAR file, to generate the schema alteration script.

4. Run the `be-oradeploy.exe` utility as explained in [Task B, Generate the SQL Scripts](#), using the updated EAR file.
5. If any of the new or changed definitions result in entries in the `yourname.aliases` file, follow instructions in [Task C, Shorten Long Names Using the Aliases File](#), on page 335. You must use the same aliases again for entity types (definitions) that used aliases before. Remember to generate the scripts again, as instructed in [Task C](#).
6. Run the newly-generated script `yourname_alter.sql` script.

Your database tables are now configured for use.

Chapter 25 **Project Configuration for Backing Store**

A backing store enables persistent backup of data in the cache. Use of a backing store enables recovery in the event of a system-wide failure. After you configure the caching solution for your project, you can then add a backing store.

This chapter explains how to configure your project to use a backing store.

You must first set up the backing store database. See [Chapter 24, Setting up a Backing Store Database, on page 329](#).

Topics

- *[Project Configuration for Backing Store—Overview, page 342](#)*
- *[Adding a JDBC Connection Resource, page 345](#)*
- *[Setting JDBC Connection and Pool Properties, page 347](#)*
- *[Enabling Backing Store and Setting Cache Characteristics, page 348](#)*
- *[Configuring How Backing Store Data is Loaded at Startup, page 349](#)*
- *[Engine TRA File Reference for Backing Store Settings, page 351](#)*

Project Configuration for Backing Store—Overview

You can implement a backing store for use with any cache option. Various aspects of backing store behavior are configurable, as explained in this chapter.

Backing Store Configuration Builds on Cache Configuration

It's a good idea to configure and test caching features before adding backing store functionality. This section explains only the properties relating to backing store functionality.

For cache-related configuration see [Chapter 17, Understanding Cache OM and Multi-Engine Features](#), on page 263 and chapters following.

Backing Store Configuration Summary

To set up a backing store you must complete the following activities:

- First configure the backing store. See [Chapter 24, Setting up a Backing Store Database](#), on page 329.
- Add the database connection details to your TIBCO Designer project. See [Adding a JDBC Connection Resource](#) on page 345.
- Enable the backing store and set properties relating to the cache cluster. See the following sections:
 - [Enabling Backing Store and Setting Cache Characteristics](#) on page 348
 - [Configuring How Backing Store Data is Loaded at Startup](#) on page 349
 - [Engine TRA File Reference for Backing Store Settings](#) on page 351

Backing Store Runtime Behavior

During regular operation, cache data is written to the backing store by the cache cluster. Only cache servers write to the backing store. Inference agents and query agents are not involved in writes to the database.

At system startup, the data is loaded into the cache from the backing store, using any one of the nodes (including nodes running inference agents and query agents).

Options to configure these behaviors are explained next.

Reading From the Backing Store at Startup

At system startup, after the minimum number of cache servers have started, the cache is loaded with backing store data. This occurs before the agents begin to process events or execute startup functions.



Any node in the cluster can load data from the backing store to the cache at startup. This is why you must add all the backing store properties to all the engine property files.

You can configure the system so that only certain objects are loaded into the cache from the backing store at startup. See [Preloading Options for Cache Only Objects on page 350](#).

See [Configuring How Backing Store Data is Loaded at Startup on page 349](#).

Reading From the Backing Store At Runtime

At runtime, if an object is not in working memory it is requested from the cache. If it is not in the cache, the cache server retrieves that object from the backing store.

At runtime, only the cache servers interact with the backing store.

Writing to the Backing Store

After an RTC, data is written to the cache, and then the cache writes the data to the backing store.

Only nodes that store cache data (also known as storage nodes), write to the backing store. Any node can be configured as a storage node (cache server) using the following property:

```
java.property.tangosol.coherence.distributed.localstorage=true
```

However, it is a best practice to enable storage only on specialized nodes, called cache server nodes.

See [Cache Server Configuration Overview on page 320](#) for details.

Additional Configuration Options

Other configuration options relating to performance and debugging are listed next. They are described in the reference table, [Engine TRA File Reference for Backing Store Settings on page 351](#).

- Number of rows to pre-fetch during database reads:
`be.engine.tangosol.oracle.prefetch`

- Batch execution size for database insertions and updates: `be.engine.tangosol.oracle.batchSize`
- Debugging and analyzing performance: `be.oracle.debug`

Summary List of Backing Store Related Properties

The following is a complete listing backing store properties discussed in this chapter:

```
# Backing Store Related Properties
be.engine.cluster.hasBackingStore
be.engine.cluster.isCacheLimited
java.property.be.engine.limited.cache.back.size.limit

be.oracle.dburi.0
be.oracle.dburi.pool.initial.0
be.oracle.dburi.pool.min.0
be.oracle.dburi.pool.max.0
be.oracle.dburi.pool.enforce.0

be.oracle.debug

be.engine.tangosol.oracle.prefetch
be.engine.tangosol.oracle.batchSize

# Only one preload choice is allowed. Pick one: none, include, or
# exclude. All options are shown below as examples only:
be.engine.cluster.preload=none

# OR
be.engine.cluster.preload=include
be.engine.cluster.EntityClassName.preload=true

# OR
be.engine.cluster.preload=exclude
be.engine.cluster.EntityClassName.preload=false
```

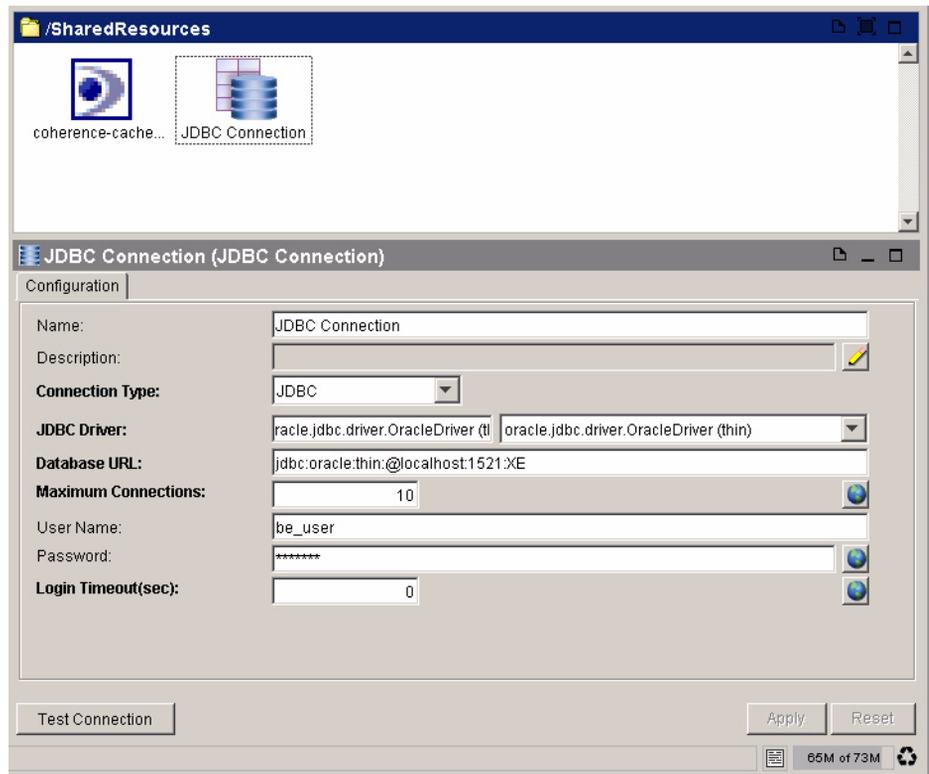
Adding a JDBC Connection Resource

Add a JDBC Connection resource to your project and configure it to connect to the backing store. Details below explain how to connect to a local instance of Oracle 10g Express Edition database. Adapt the instructions as needed for your database.



You must also specify the location of this resource in the engine property `be.oracle.dburi.0`. See [Specifying the JDBC Connection on page 347](#).

1. In TIBCO Designer, open your project, and open the folder where you keep shared resources. Add a JDBC Connection resource (from the JDBC palette).



2. In the drop-down list to the right of the JDBC Driver field, select **oracle.jdbc.driver.OracleDriver (thin)**. The driver appears in the JDBC Driver field and the Database URL format appears as:

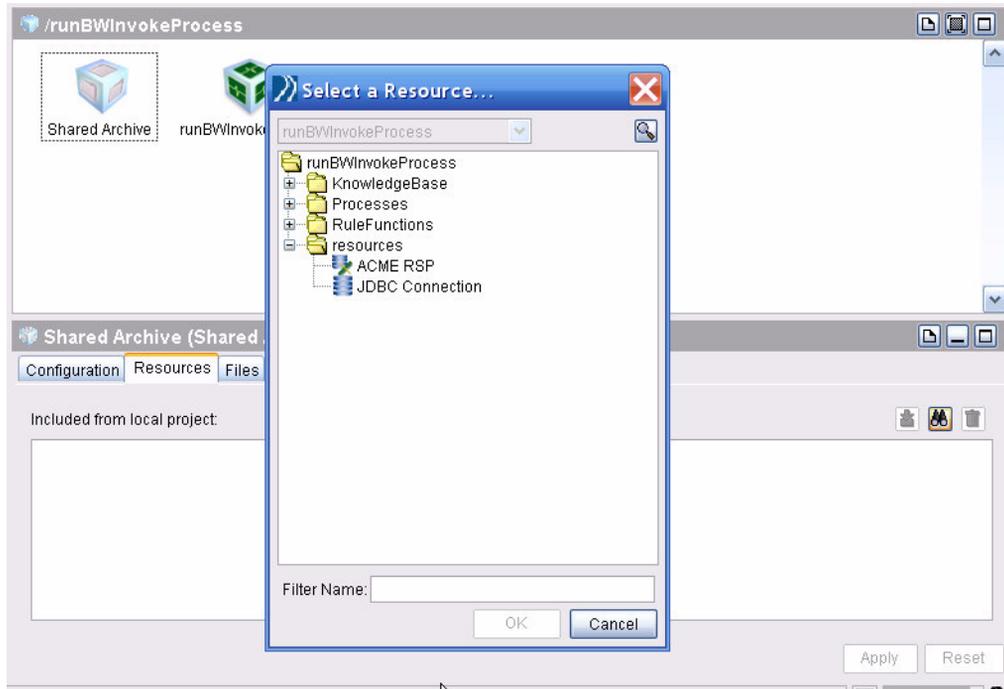
```
jdbc:oracle:thin:@<host>:<port#>:<db_instancename>
```

3. In the Database URL field, configure the provided format. For example:

```
jdbc:oracle:thin:@localhost:1521:XE
```

where 1521 is the default port, and XE is the default instance name for Oracle Database 10g Express Edition. (The default instance name for Oracle Database 10g is ORCL).

4. In the User Name and Password fields, enter the username and password of the database user you created (see [Task D, Run the Initialize Database Script to Create the Oracle User, on page 336](#)). The example username and password used in the documentation are both `be_user`.
5. Click **Apply**, then click **Test Connection**. If the details are correct, you see a success message. Save the project.
6. Add the resource to the shared archive as follows. Select the Shared Archive resource (SAR) within the EAR resource. Select the Resources tab, browse to the location of the JDBC Connection resource you just configured, and select it.



7. Click **Apply** then click **Save** to save the project. When you build the EAR file, the JDBC Connection resource is packaged into the shared archive (SAR) file.
8. Rebuild the EAR file.

Setting JDBC Connection and Pool Properties

Add the following properties to the engine property file.

Specifying the JDBC Connection

To set the JDBC connection, add the property `be.oracle.dburi.0` to your engine property file and provide the project path (that is path to the item in the design-time project) to the JDBC connection resource you configured (see [Adding a JDBC Connection Resource on page 345](#)). For example:

```
be.oracle.dburi.0 /SharedResources/JDBC Connection.sharedjdbc
```



Here is one way to get the correct URI value. Open the project and open the EAR resource. Open the Shared Archive resource and look at the Resources tab. All shared resources are listed there, using the correct path.

Setting Connection Pool Size

The connection pool settings are as follows (showing sample values):

```
be.oracle.dburi.pool.initial.0 2  
be.oracle.dburi.pool.min.0 2  
be.oracle.dburi.pool.max.0 5  
be.oracle.dburi.pool.enforce.0 true
```

BusinessEvents performs backing store operations in bursts after each RTC. Monitor the database performance and adjust the pool size as needed.

Having more connections can also speed up recovery in the event of a failure.

Enabling Backing Store and Setting Cache Characteristics

This section briefly focuses on cache properties related to use of a backing store, for the benefit those who are adding backing store support to an existing project configured for cache OM.

See [Chapter 20, Configuring Cache Cluster Settings, on page 295](#) for more details about these and other cache cluster properties.

Enabling Backing Store Functionality

To enable backing store functionality you must set the following property in all cluster nodes so that the entire cluster is backing store aware:

```
be.engine.cluster.hasBackingStore=true
```

Specifying Limited Cache Size

```
be.engine.cluster.isCacheLimited=true  
java.property.be.engine.limited.cache.back.size.limit
```

If you want to limit the size of the cache, set the `isCacheLimited` property.

You only need to set the size limit if you want to use a size other than the default size.

Configuring How Backing Store Data is Loaded at Startup

At system startup, one node loads data from the backing store to the cache cluster. A cache cluster is made up of all the dedicated cache servers and all the storage-enabled nodes, which act as cache servers as well as inference or query agents.

Any node in the cluster can perform the cache loading.



Because any node can perform cache loading, set the relevant properties in all nodes in the cluster.

A Quorum of Cache Servers

Before cache loading begins, you must ensure that enough cache servers to hold the data to be loaded from the backing store. (Note that storage-enabled nodes are considered to be cache servers, in addition to dedicated cache servers.)

To do this, you specify a quorum, that is, a minimum number of cache servers that must be started before cache loading begins.

After the quorum is reached, one of the nodes in the cluster performs the cache loading. The cluster does not start processing incoming data until the required data has been loaded into the cache.

To specify this quorum of cache servers set the following property on all nodes, using the same value:

```
be.engine.cluster.minCacheServers
```

After the quorum is reached, whichever node acquires the lock first performs the cache loading. All agents wait until backing store data has finished loading before they start.

See [Configuring Caching Scheme, Multi-Engine, and Cluster Properties on page 296](#).



The setting does not affect runtime operation of the deployed application. Deployed applications continue to run even if one or more cache servers fails and the quorum is no longer met. A warning message is written to the log file. See [Reliability of Cache Object Management on page 264](#) for related information.

Preloading Options for Cache Only Objects

Objects set to the cache mode Cache plus Memory are always loaded at startup. However you can configure preloading options for objects set to Cache Only cache mode (see [Working With Cache Modes on page 282](#) for more on cache modes).

You can configure the system so that at startup, you load all, some, or none of the Cache Only objects from the backing store into the cache. These options allow you to control the size of the cache.

Objects not loaded at startup time are loaded on an as-needed basis, when needed during an RTC, when they are not found in the cache.

To Preload All Cache-Only Objects or No Objects

To load all Cache Only objects at startup, you do nothing: this is the default behavior.

To load no objects at startup, you set the following property:

```
be.engine.cluster.preload=none
```

To Control What Is and What Is Not Preloaded

You can specify the subset using *either* inclusion or exclusion. Choose the method that best suits your needs.

To include only a specific subset of objects when preloading, set the following property:

```
be.engine.cluster.preload=include
```

And specify the list of entities to load in a list as follows:

```
be.engine.cluster.EntityClassName.preload=true
be.engine.cluster.EntityClassName2.preload=true
```

To exclude a specific subset of objects from preloading, set the following property:

```
be.engine.cluster.preload=exclude
```

And specify the list of entities to exclude from the load in a list as follows:

```
be.engine.cluster.EntityClassName.preload=false
be.engine.cluster.EntityClassName2.preload=false
```

In the list of entities, replace *EntityClassName* with the appropriate generated class names.



You can use the function `Coherence.C_ClassName()` to obtain the generated class name for an object, given the entity's project path (that is, the path to the item in the design-time project).

Engine TRA File Reference for Backing Store Settings

This table explains all the engine properties used to configure backing store functionality. In addition see [Table 27, Cache Cluster Properties, on page 297](#) for properties that define whether a backing store is used, and if a limited cache is used.

Table 33 *BusinessEvents Engine Properties for Configuring a Backing Store (Sheet 1 of 2)*

Property	Notes
<code>be.oracle.dburi.0</code>	Specifies the path from the project root to the JDBC Connection resource that defines the connection to the backing store. For example: <code>be.oracle.dburi.0 /SharedResources/JDBCConnection.sharedjdbc</code>
<code>be.oracle.dburi.pool.initial.0</code>	Initial number of JDBC connections in the JDBC connection pool used for the backing store. For example: <code>be.oracle.dburi.pool.initial.0 10</code> See Setting Connection Pool Size on page 347 for more details.
<code>be.oracle.dburi.pool.min.0</code> <code>be.oracle.dburi.pool.max.0</code>	Minimum and maximum number of JDBC connections in the JDBC connection pool used for the backing store. See Setting Connection Pool Size on page 347 for more details.
<code>be.oracle.dburi.pool.enforce.0</code>	Set this property to true if you want the other <code>be.oracle.dburi.pool</code> properties to be enforced. By default the <code>be.oracle.dburi.pool</code> properties are ignored. Default is false.
<code>be.engine.tangosol.oracle.prefetch</code>	Number of rows to pre-fetch during database reads. Default is 1000
<code>be.engine.tangosol.oracle.batchSize</code>	Batch execution size for database insertions and updates. Default is 10
<code>be.oracle.debug</code>	Optional. Dumps Oracle database reads and writes to the console and log file. Set to true or false (Optional setting). Default is false.

Table 33 *BusinessEvents Engine Properties for Configuring a Backing Store (Sheet 2 of 2)*

Property	Notes
<code>be.engine.cluster.preload</code>	<p>Specifies what Cache Only objects are loaded into the cache on system startup. See Preloading Options for Cache Only Objects on page 350 for more details.</p> <p><code>all</code>: All objects are loaded.</p> <p><code>none</code>: No objects are loaded.</p> <p><code>include</code>: The only objects loaded are those specified by their generated class names, in properties of the following format :</p> <p><code>be.engine.cluster.entityClassName.preload=true</code></p> <p><code>exclude</code>: No objects are loaded except those specified in properties of the following format:</p> <p><code>be.engine.cluster.entityClassName.preload=false</code></p> <p>The function <code>Coherence.C_ClassName()</code> returns the generated class, given the project path to the entity (that is, the path to the entity in the design-time project).</p> <p>Default is <code>all</code>.</p>
<code>be.engine.cluster.EntityClassName.preload</code>	<p>See above.</p>

Chapter 26 **Deploytime Configuration**

This chapter explains configuration that is done before deploying the nodes. It also presents two methods of configuration and management of deployed applications.

Topics

- *Deploytime Configuration Overview, page 354*
- *Understanding Methods of Configuring Engine Properties, page 356*
- *Defining and Using Global Variables, page 358*
- *Configuring a BAR File for Deployment, page 360*
- *BusinessEvents Archive Resource Reference, page 362*
- *Shared Queue and Threads Properties, page 367*
- *Configuring to Run Outside of a TIBCO Administrator Domain, page 368*
- *Configuring to Run in a TIBCO Administrator Domain, page 370*
- *Deploying Many Agents on One Machine With TIBCO Administrator, page 373*
- *Customizing the List of Properties on the Advanced Tab, page 374*

Deploytime Configuration Overview

Deploytime configuration settings affect the behavior of the deployed nodes and agents. Deploytime settings can apply at the agent level, node level, or at a higher level such as that of a cache cluster.

Some deploytime configuration is done in TIBCO Designer, when you configure the EAR and BAR resources, for example, choosing which rule sets to deploy.

An EAR file deploys as one BusinessEvents node. A node can contain one or more inference agents, or one or more query agents, or both, or it can be deployed as a cache server. You can deploy one ear multiple times with different (or the same) deploytime settings.

Some deploytime configuration is done using properties. You can configure properties using property files, using TIBCO Administrator, or both. See [Understanding Methods of Configuring Engine Properties on page 356](#) to understand how to use each method.

The methods you use for configuring your projects and managing the deployed applications depend on whether the application is deployed to a TIBCO Administrator domain or not.

See Also

[Chapter 27, Configuring Logging Settings, on page 375](#)

[Chapter 28, Deploying a TIBCO BusinessEvents Project, on page 379](#)

[Chapter 29, Configuring and Using Hot Deployment, on page 391](#)



For details about deploying Decision Manager classes (implemented virtual rule functions) see *TIBCO BusinessEvents Decision Manager*.

Two Approaches to Configuring and Managing BusinessEvents Applications

There are two general approaches to managing the BusinessEvents servers:



TIBCO recommends that you use only one of these approaches to avoid potential configuration conflicts.

For test environments, it is common to deploy outside of a domain. For production it is common to deploy to a domain.

Outside a TIBCO Administrator Domain

TIBCO BusinessEvents includes a command-line utility for starting and stopping BusinessEvents servers and setting deploytime properties. The properties are set in the engine property file (`be-engine.tra`) and (optionally) supplementary property files. These files are used by the command-line utility.

In a TIBCO Administrator Domain

You can use TIBCO Administrator for deploying and undeploying, starting and stopping BusinessEvents servers, and for setting deploytime properties.

All properties that are set in the engine property file can be added to the Advanced tab for the merged BAR file, which appears as a single service within the application when viewed in the TIBCO Administrator user interface. See [Customizing the List of Properties on the Advanced Tab on page 374](#) for the procedure.

You can also use global variables in your project, and set their values at deploy time. All global variables set in a project automatically appear in the Advanced tab of the application in the TIBCO Administrator user interface. See [Defining and Using Global Variables on page 358](#).

See [Understanding Methods of Configuring Engine Properties on page 356](#) for more details on this topic.

Understanding Methods of Configuring Engine Properties

How you set engine properties depends in part on whether you are deploying to a TIBCO Administrator domain or not. See [Two Approaches to Configuring and Managing BusinessEvents Applications on page 354](#) for an introduction to this topic, then explore each option in more detail below.

Setting Engine Properties for Deployment Outside of a Domain

For deployment outside of a domain you set properties in the engine property files. The main configuration file is the `be-engine.tra` file. You can also define sets of properties in supplementary files.

One approach is to configure common values in the main engine property file, and override or extend them using supplementary files, as needed for a specific node's requirements. See [Configuring to Run Outside of a TIBCO Administrator Domain on page 368](#) for more details.



You can also set values for global variables in the TRA file. See [Defining and Using Global Variables on page 358](#).

Setting Engine Properties for Deployment to a Domain

When you deploy to a TIBCO Administrator domain, you can set properties using engine property files or using the TIBCO Administrator user interface, or both. Whether you use the TRA file or TIBCO Administrator to set properties depends on what is convenient in your situation.

In TIBCO Administrator the individual BARs are, in effect, merged into one, and one set of BAR dialogs is available for configuration. You cannot deploy a BAR file individually. In effect, you are setting properties at the node level. However some property names contain an agent identifier so they can apply at the agent or agent group level.



In TIBCO Administrator, you can also define global variable values. They are set in the Advanced tab for the application. You can provide agent-level identifiers as global variables and provide values at deploy-time.

To set the deploytime engine properties in TIBCO Administrator, you go to the Advanced tab for the merged BAR. Expand Application Management > *application_name* > Configuration > *application_name.bar*, and then click the Advanced tab.

The default value of *application_name* is provided by the name of the enterprise archive resource, defined in TIBCO Designer. You can change the value when you upload the EAR file in TIBCO Administrator.

You can customize the list of engine properties that appears on the Advanced tab of the merged BAR to suit your needs. For instructions see [Customizing the List of Properties on the Advanced Tab on page 374](#).



Properties set in the BAR file's Advanced tab override those set in the *BE_HOME/bin/be-engine.tra* file. It's a good idea to set individual properties in one location (TRA file or BAR Advanced tab) to avoid errors.

Also note that TIBCO Administrator generates a TRA file on deployment and then reads properties from that generated file. See [Location of TIBCO Administrator-Generated Property File on page 389](#).

Using AppManage for Scripted Deployment to a Domain

Instead of using the TIBCO Administrator user interface, you can perform scripted deployment to a TIBCO Administrator domain using the AppManage utility. Use of AppManage is not explained in BusinessEvents documentation. See TIBCO Runtime Agent Scripting Deployment User's Guide for details.

Defining and Using Global Variables

You can define and use global variables in your TIBCO Designer project and then provide the variable values at deploy time.

You can also provide values in the `BE_HOME/bin/be-engine.tra` engine property file.

Values provided in TIBCO Administrator override values provided in the engine property file (`BE_HOME/bin/be-engine.tra`) and TIBCO Designer.

This section provides the basic procedures for defining and using global variables as a convenience. These topics are covered in more detail in documentation for TIBCO Administrator.

To Add a Global Variable to a Project in TIBCO Designer

To add a global variable, select the Global Variables tab, click the pencil icon to display the editing window, and add the variable name and, as desired, a default value. See TIBCO Designer documentation for more details.

For example, you define a global variable to define the location of a custom caching scheme. You call the variable `CachingSchemeLoc` and provide a default location of `C:/beCachingScheme/myschemes.xml`.

To Use a Global Variable in TIBCO Designer

To use a global variable as the value for a project setting, drag it from the list of variables into the text box for the setting, or enter it manually using the syntax `%%Variable_Name%%`. For example, to use the caching scheme global variable in the File Path field, enter `%%CachingSchemeLoc%%`.

Note that this syntax is not used with the system functions, `System.getGlobalVariableAsType`.

To Specify a Value in TIBCO Administrator

Global variables you define in a project appear automatically in TIBCO Administrator, on the application's Advanced tab. To navigate to this tab, expand Application Management > *Application_Name* > Configuration, click the application name in the Configuration Builder panel, and then click Advanced. You see the list of global variables. Provide a value in the Value field. Save and deploy.

To Specify a Value in the Engine Property File

Open the engine property file (*BE_HOME/bin/be-engine.tra*) and add the variable name and the value to be used at deploytime. Use the syntax:

```
tibco.clientVar.variablename defaultvalue
```

Alternatively, you could also put the name and value in an override property file and specify the appropriate file using `-p` option at deploy time.

Configuring a BAR File for Deployment

The procedure in this section explains how to set up a BAR file as an inference agent or query agent. Most of the actions require background understanding. Object management configuration is explained in detail in the referenced sections.

To Configure a BAR File for Deployment

1. Open your project in TIBCO Designer.
 2. If you have not yet added a BAR to the EAR resource, open the EAR, right-click in the design panel, and select **Add Resource > BusinessEvents Workbench > BusinessEvents Archive**.
 3. Open the BusinessEvents Archive you want to configure.
- Configuration
4. In the Configuration tab, provide a name and optionally a description. You can also add a person's name in the Author field.
 5. In the Type field, select the type of agent you are configuring:
 - Select Inference to configure an inference agent.
 - Select Query to configure a query agent. Query agents are available for Cache object management only, and only in TIBCO BusinessEvents Enterprise Suite.

See [Configuration on page 362](#) for a reference to the settings.
- Rule Sets
6. If you are configuring an inference agent, select the Rulesets tab and do one of the following:
 - Select All RuleSets if you want to deploy all rule sets in the project in this BAR.
 - Select A Selection of RuleSets and then browse to and select the rule sets you want to include in this BAR. See [Deploying Many Agents on One Machine With TIBCO Administrator on page 373](#) for more information.

See [Rule Sets on page 363](#) for a reference to the settings.
- Input Destinations
7. Select the Input Destinations Tab and do one of the following:
 - Select Custom and then check the Enable checkbox for all enable the destination listeners you want to use.
 - Select Defaults to enable only the default destinations.

BusinessEvents checks the Default column if the default event for that destination is used in a rule set selected for deployment on the RuleSets tab.

Preprocessors for
Input Destinations

8. If you want to assign a preprocessor to a destination, do the following:
 - a. Click in the Preprocessor field. You see the Search button. Browse to and select the rule function you have set up to be the preprocessor for the destination.

Only functions with one argument of type simple event appear in the list. See [Working With Event Preprocessors on page 145](#).

To remove a preprocessor from the Preprocessor field, click the Search button, then click OK without making a selection.
 - b. In the Workers field, select a number of worker threads to be created, or choose one of the other values.
 - c. If you selected a number of worker threads in the Workers field, in the Queue Size field, select a queue size.

See [Input Destinations on page 364](#) for a reference to the settings.

Startup and
Shutdown

9. Select the Startup/Shutdown tab if you want to select rule functions that perform startup and shutdown actions.
 - a. Click the plus sign to browse to and select a function. Only rule functions that take no argument and whose Validity field (in the rule function's Configuration tab) is set to Action appear in the list.
 - b. If you have selected more than one function in one list, click the up and down arrows to arrange the functions. BusinessEvents calls the functions in the order listed.

See [Startup and Shutdown on page 366](#) for a reference to the settings.

Object
Management

10. Select the **Object Management** tab. From the Type drop-down list, select one of the options:
 - Persistence: See [Chapter 16, Configuring Persistence Object Management, on page 251](#).
 - Cache: See [Chapter 17, Understanding Cache OM and Multi-Engine Features, on page 263](#) through [Chapter 23, Configuring Query Agents \(Cache OM\), on page 325](#) for details on caching features and configuration.
 - In Memory: This option requires no additional configuration.

For Persistence and Cache options, setting up object management is a larger task than simply configuring this tab. See the referenced sections for details.
11. Click **Apply** and save the project.

BusinessEvents Archive Resource Reference



Each BusinessEvents Archive resource in a project configures deploy-time settings for one inference agent or one query agent. A BusinessEvents Archive resource exists within an Enterprise Archive resource (EAR). The EAR is a standard TIBCO Designer component that is required to create an EAR file for project deployment (see *TIBCO Designer User's Guide* for details).

A BusinessEvents Archive resource allows you to configure the following for an inference agent:

- Rule sets to deploy
- Input destinations (enabling listeners, specifying preprocessors and workers)
- Object management settings
- Startup and shutdown actions

For a query agent, you specify input destinations and startup and shutdown actions. On the Object Management tab, you specify the query agent group name.



Projects saved in TIBCO BusinessEvents Enterprise Suite are not intended for use with BusinessEvents Inference Edition.

Configuration

The Configuration tab has the following fields.

Field	Global Var?	Description
Name	No	The name to appear as the label for the resource. (This name does not have to follow rules for identifiers, which are explained in the section Identifiers (Names) in <i>TIBCO BusinessEvents Language Reference</i> .)
Description	No	Short description of the resource.
Author	No	Name of the person who created the resource.

Field	Global Var?	Description
Type	No	Indicates type of agent: Inference or Query. See Chapter 21, Configuring Inference Agents (Cache OM) , on page 305 and Chapter 22, Configuring Cache Servers (Cache OM) , on page 319 for information.

Rule Sets

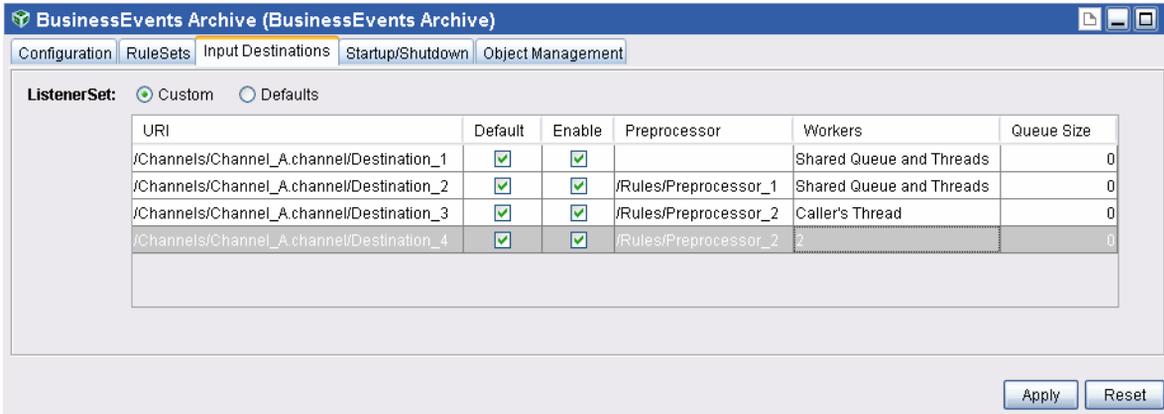
On the Rule Sets tab, you specify which rule sets you want to include in the deployed project, all rule sets or a selection of rule sets:



The main reason to choose a selection of rule sets is to configure multiple inference agents, dividing the rule sets between the BAR resources as appropriate. You can include multiple inference agents in one EAR, or deploy multiple EAR files, each with differently configured inference agents.

Input Destinations

All destinations configured for the project appear in the Input Destinations tab.



You can enable and disable destination listeners. You can also specify a function for use as a preprocessor by the destination. (Destinations are configured within the Channel resource. For information about configuring destinations, see [Chapter 2, Working With Channels and Destinations, on page 23](#).)

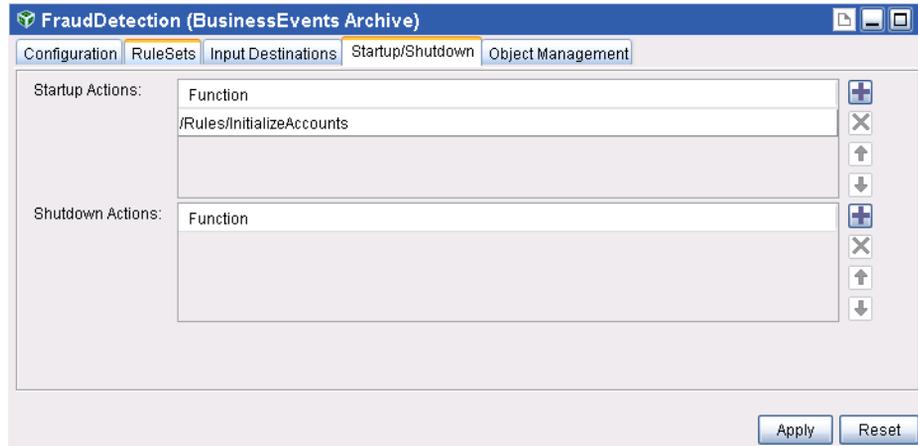
The Input Destinations tab has the following fields

Field	Global Var?	Description
URL	No	Project path to the destination (that is path to the destination in the design-time project).
Default	No	This field is for information only. You can't change the value. BusinessEvents checks the Default checkbox if the following is true: <ul style="list-style-type: none"> This destination is used as the default destination for an event AND That event is specified in the declaration of a rule AND That rule is in a rule set that is included in the BAR.
Enable	No	Check the Enable checkbox to enable this destination listener for deployment. Uncheck to disable this destination listener for deployment. It will be unavailable in the deployed project.

Field	Global Var?	Description
Preprocessor	No	<p>Specifies the rule function used as the preprocessor for messages entering this destination.</p> <p>Specifying a preprocessor is optional. If you specify a preprocessor, also specify worker thread settings.</p> <p>See Working With Event Preprocessors on page 145.</p>
Workers	No	<p>Specifies the number of worker threads to use for the preprocessor.</p> <p>See Worker Threading and Queue Options on page 146 for more details and a discussion of the advantages and disadvantages of each option.</p> <p>Shared Queue and Threads Uses the BusinessEvents system-wide shared queue and threads. See Shared Queue and Threads Properties on page 367 for related property settings.</p> <p>Caller's Thread Uses the thread (and queue size) provided by the channel resource client.</p> <p>Dedicated Workers (Numbers 1-8) Specifies a number of worker threads for the input destination. Also specify the Queue Size.</p> <p>A value is required and used only if a preprocessor is specified.</p> <p>Default is Shared Queue and Threads</p>
Queue size	No	<p>Specifies the queue size of each worker thread, if you choose a number of workers from 1-8.</p> <p>0 means unlimited.</p> <p>A value is required in this field only if the Workers field specifies a numeric value.</p> <p>Default is 0</p>

Startup and Shutdown

You can add rule functions to the Startup/Shutdown tab to perform startup and shutdown actions. You can use only rule functions that take no argument and whose Validity field (in the rule function's Configuration tab) is set to Action. BusinessEvents calls the functions in the order listed. See [Working With Startup and Shutdown Rule Functions on page 142](#) for more details.



Object Management

The option you select in the object management tab determines how BusinessEvents data is stored, and how the data is protected from failures. Use of the Object Management tab is documented in chapters dealing with the different object management options.

The object management options are:

- **In Memory** (default) The standard JVM memory management features are used. If a machine fails, the data is lost. For In Memory, no additional configuration is required. In Memory object management is documented in [Chapter 15, Configuring In Memory Object Management, on page 243](#).
- **Persistence** Backs up data at checkpoint intervals to an internal database. See [Chapter 16, Configuring Persistence Object Management, on page 251](#).
- **Cache** Different caching options provide for various levels of failover and failback, using different deployment architectures. see [Chapter 14, Understanding Object Management and Fault Tolerance, on page 233](#) and chapters following. See [Table 27, Cache Cluster Properties, on page 297](#) for the reference table explaining the cache options.

Shared Queue and Threads Properties

The table below describes the shared queue and thread properties.

In this release, shared queue and threads are used for the preprocessor only, and are used only if the following are all true:

- The node contains agents that use channels (inference agents or query agents).
- The agent BAR resource uses the Shared Queue and Threads setting in the Workers field in the Input Destinations tab of the BAR resource in the EAR. (See [Input Destinations on page 364](#).)
- Default settings are not adequate (they generally are adequate).

Table 34 Shared Queue and Threads Properties

<code>com.tibco.cep.runtime.scheduler.default.numThreads</code>	<p>Specifies the number of system-wide shared threads.</p> <p>The default value is the same number as there are processors available to the JVM.</p> <p>See also notes for <code>com.tibco.cep.runtime.scheduler.queueSize</code></p>
<code>com.tibco.cep.runtime.scheduler.queueSize</code>	<p>Specifies the queue size for the system-wide shared queue.</p> <p>For more details, see notes for <code>com.tibco.cep.runtime.scheduler.default.numThreads</code></p> <p>If set to 0 (zero), the queue size is unlimited.</p> <p>By default the queue size is the number of threads multiplied by 128.</p>

Configuring to Run Outside of a TIBCO Administrator Domain

Before configuring and starting a TIBCO BusinessEvents application, you configure an enterprise archive resource (EAR) file for your BusinessEvents project within TIBCO Designer.

This section gives a general procedure for configuring deploytime properties for command-line startup. See [Configuring to Run in a TIBCO Administrator Domain on page 370](#) for the equivalent procedure if you want to configure for deployment using TIBCO Administrator.

Use of Supplementary Property Files

If you will start the BusinessEvents application at the command-line, you can configure various settings in the `be-engine.tra` property file, in supplementary configuration files, or both. How you distribute the properties between the main property file and supplementary ones depends on your needs.

For example, you might configure properties for a cache server in one property file, and properties for a regular BusinessEvents engine in another. When you start the server (and run the application), you would specify the appropriate supplementary configuration file, as explained in [Deploying a Project Outside a TIBCO Administrator Domain on page 380](#).



The values of properties specified in the supplementary file override values for same-named properties set in the `be-engine.tra` file.

To Configure Properties for Command-Line Startup

- Do one of the following:
 - Open the `be-engine.tra` file for editing.
 - Open a supplementary property file for editing. You can use supplementary files to configure sets of options for different needs as explained in the section introduction.
- Set properties and values as explained in the relevant section of this chapter and elsewhere in the guide. See [Deploytime Configuration Overview on page 354](#) for a summary and references to related information. Also see [Table 36, Command-line Engine Startup Options, on page 381](#) for a list of available properties.



If a property name includes spaces, escape them using a back slash. For example, `be.engine.om.berkeleydb.cacheweight.My\ Rulesession`.

3. Optionally, specify startup options in the property file, instead of at the command line. Use the `tibco.env.APP_ARGS` global variable using the following format:

```
tibco.env.APP_ARGS [--propFile startup_property_file] [-p propertiy_file] [-n  
engine_name] [earfilename]
```

See [Deploying a Project Outside a TIBCO Administrator Domain on page 380](#) for the deployment procedure.

Configuring to Run in a TIBCO Administrator Domain

Before configuring and deploying a TIBCO BusinessEvents project, you configure an enterprise archive resource (EAR) file for your BusinessEvents project within TIBCO Designer.

This section gives a general procedure for configuring deploytime properties for deployment using TIBCO Administrator. See [Configuring to Run Outside of a TIBCO Administrator Domain on page 368](#) for the equivalent procedure if you want to configure for command-line startup.

On the Advanced tab for the application you can configure generic options and global variable values.

On the Advanced tab for the merged BAR file (Application Management > *application_name* > Configuration > *application_name.bar*) you can set various engine properties. See [Customizing the List of Properties on the Advanced Tab on page 374](#) for details on adding options to the Advanced tab.



Properties configured in the default property file (*BE_HOME/bin/be-engine.tra*) are used when you deploy using TIBCO Administrator. However, properties set in TIBCO Administrator override those set in the property file.

Rule Session Configuration Options

When you deploy a BusinessEvents EAR file, all the separate BAR files within it are merged.

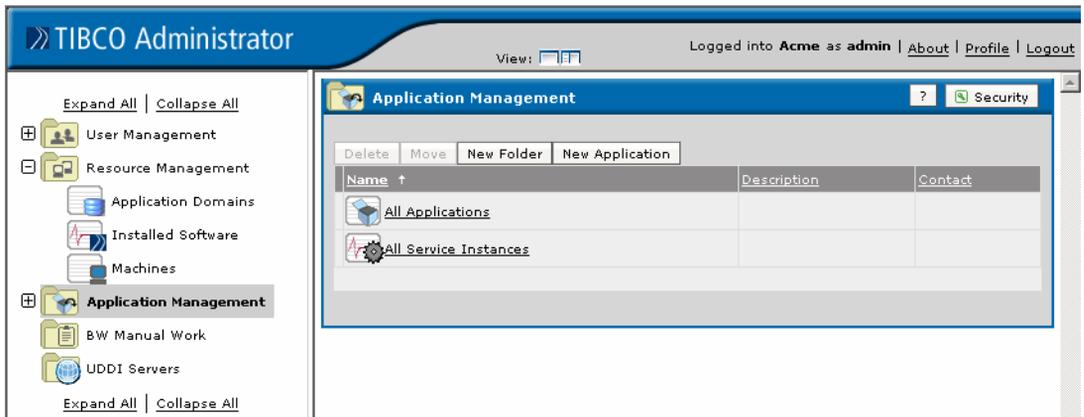
TIBCO Administration Domains

When you install TIBCO Administrator, you are prompted to create an administration domain and to create a username and password. You can use this domain or create another one (using Domain Utility) for your BusinessEvents applications and the hardware they run on. The first time you log in to the TIBCO Administrator user interface, you must use the username and password entered during installation of the TIBCO Administrator software. You can then create additional users and passwords as needed.

If TIBCO Administrator was already installed before you installed BusinessEvents, you may have to contact the person responsible for administering the software to get login credentials for an existing administration domain.

To Configure (But Not Deploy) a Project EAR Using TIBCO Administrator

1. Start the TIBCO Administrator GUI:
 - Windows: Start > Programs > TIBCO > TIBCO Administrator Enterprise Edition *version* > TIBCO Administrator
 - Web browser: `http://host-name:port/` (where *host-name* is the machine name and *port* is the HTTP port specified during installation, 8080 by default)
2. Select the administration domain for the application and provide the username and password assigned during installation, or other administrator user credentials. (See [TIBCO Administration Domains on page 370](#) for more information about domains.)
3. If you are configuring a project for the first time, do the following:
 - a. Click **Application Management** (in the left panel).
 - b. Click the **New Application** button.



- c. At the Upload EAR File dialog, click **Browse**, select the EAR file you want to deploy, and click **OK**.
 - d. At the New Application Configuration dialog, set the Application Parameters and Services settings as desired (click Help for details) and click **Save**. By default, Name is set to the EAR file name and Deployment Name is set to the EAR file name prepended with the domain name.
 4. Expand to **Application Management** > *application_name* > **Configuration**.
- Setting Engine Properties
5. As desired, in the Configuration Builder panel, click the merged BAR name (*application_name.bar*) and select the **Advanced** tab. Set properties as desired. Click **Save**.

6. As desired, in the Configuration Builder panel, click the application name and set values in the Advanced tab, including global variable values. Click **Save**.
See [Deploying a Project in a TIBCO Administrator Domain on page 383](#) for the deployment procedure.

Deploying Many Agents on One Machine With TIBCO Administrator

When you use TIBCO Administrator to deploy a multi-BAR EAR file to a machine, or to deploy the same EAR multiple times to one machine, issues relating to BAR-level property settings can arise. Each BAR represents one agent.

When you look at an uploaded EAR file in TIBCO Administrator, you see that all BAR files in the EAR are represented by one merged bar file, which appears at the level of a service under the application. This is because it is not possible to deploy selected BARs to different servers.

Because the BAR files are merged, you can't set properties differently for each BAR file using TIBCO Administrator. Certain object management configuration options, however, are set differently for each BAR file (rule session).

In general, here are some approaches you can take to setting values at the rule session level when you need to set values for multiple rule sessions differently:

- Create a different global variable for each BAR file's configuration field, and put the appropriate variable in each BAR's configuration field in TIBCO Designer. At deploy time, set values as needed using the global variables that appear on the application's Advanced tab. See [Defining and Using Global Variables on page 358](#) for more details.
- Some property file properties have names to which you can append rule-session or agent-specific names. Add and set those properties in the `BE_HOME/bin/be-engine.tra` files. See [Engine Property File Settings for Persistence Object Management on page 260](#) for some examples.

A similar configuration issue can arise if you want to deploy one EAR file to the same machine more than once, for example, setting the database environment directory for a BAR's persistence data store, when Persistence is used for object management. Guidelines to handle this scenario are provided in the section [Database Environment Directory on page 257](#).

Customizing the List of Properties on the Advanced Tab

You can customize the list of engine properties that is available in TIBCO Administrator for deploytime configuration. These properties appear in the Advanced tab for the merged BAR file.

To view the properties, expand Application Management > *application_name* > Configuration > *application_name.bar*, and then click the Advanced tab.

To make engine properties available for deploytime configuration in TIBCO Administrator, you add a new `<property>` element in the `be-engine.xml` file. You can also modify and delete elements.



Default values set in the `be-engine.xml` file (and values manually set in TIBCO Administrator) override values set in the `be-engine.tra` file.

The properties you add to `be-engine.xml` are available in TIBCO Administrator for all new BusinessEvents projects. To make the `be-engine.xml` properties available for existing projects, you must first regenerate their EAR files in TIBCO Designer and reload the project EARs in TIBCO Administrator.

To Customize the List of Properties on the Application's Advanced Tab

1. Open the following file for editing:

```
BE_HOME\lib\com\tibco\deployment\be-engine.xml
```

2. Use the `<property>` element in the `be-engine.xml` file to add properties. The `<property>` element has the following structure:

```
<property>
  <name>name of property</name>
  <default>default value</default>
  <description>short description of property</description>
</property>
```

For example, to include the `be.trace.enable` property in a deployment configuration, add the following element to the `be-engine.xml` file:

```
<property>
  <name>be.trace.enable</name>
  <default>true</default>
  <description>Global switch for tracing</description>
</property>
```

3. In TIBCO Designer, rebuild the project EAR file.
4. In TIBCO Administrator, reload the project EAR file.

Chapter 27 **Configuring Logging Settings**

This chapter explains how to configure settings relating to logging.

Topics

- [Configuring BusinessEvents Logging Settings, page 376](#)

Configuring BusinessEvents Logging Settings

To Make Logs Easier to Use

It's a good idea to use a separate log directory for each server running on a machine. You can do this by running the BusinessEvents servers with the `-name` option, or by setting a log directory name in the TRA files (or both). For example, in the TRA file for a specific inferencing node, you could set the directory as follows:

```
# Trace Properties
Engine.Log.Dir Inferencel
```

The rest of this section provides a reference to the properties you can use to define logging behavior.

Table 35 BusinessEvents Engine Properties to Determine Logging Behavior (Sheet 1 of 3)

Property	Notes
<code>be.trace.enable</code>	<p>Enables tracing features. All other trace properties are ignored if <code>be.trace.enable</code> is set to false.</p> <p>Available in TIBCO Administrator by default.</p> <p>Default is true.</p>
<code>be.trace.log.enable</code>	<p>Enables tracing to log files.</p> <p>Available in TIBCO Administrator by default.</p> <p>Default is true.</p>
<code>be.trace.log.fileName</code>	<p>Defines a log file name.</p>
<code>be.trace.log.maxnum</code>	<p>Number of log files to keep. When <code>be.trace.log.maxsize</code> is reached, a new log file is created for the next log entries. Files are created up to the <code>be.trace.log.maxnum</code>. The oldest file is deleted when a new file is added after this value is reached.</p> <p>Available in TIBCO Administrator by default.</p> <p>Note: BusinessEvents uses this property (and not <code>engine.Log.MaxNum</code>).</p> <p>Default is 10.</p>

Table 35 BusinessEvents Engine Properties to Determine Logging Behavior (Sheet 2 of 3)

Property	Notes
<code>be.trace.log.maxsize</code>	<p>Maximum size of one log file.</p> <p>Available in TIBCO Administrator by default.</p> <p>Note: BusinessEvents uses this property (and not <code>engine.Log.Maxsize</code>).</p> <p>Default is 10000000.</p>
<code>be.trace.publish.daemon</code>	<p>Value of the daemon parameter when tracing to Rendezvous.</p> <p>Available in TIBCO Administrator by default.</p> <p>Default is <code>tcp:7500</code>.</p>
<code>be.trace.publish.enable</code>	<p>Enable tracing to TIBCO Rendezvous. When enabled, log information is published as Rendezvous messages.</p> <p>Available in TIBCO Administrator by default.</p> <p>Default is <code>false</code>.</p>
<code>be.trace.publish.network</code>	<p>Value of the network parameter when tracing to Rendezvous.</p> <p>Available in TIBCO Administrator by default.</p> <p>No default value.</p>
<code>be.trace.publish.service</code>	<p>Value of the service parameter when tracing to Rendezvous.</p> <p>Available in TIBCO Administrator by default.</p> <p>Default is <code>7500</code>.</p>
<code>be.trace.publish.subject</code>	<p>Defines the publish subject when tracing to Rendezvous.</p> <p>Available in TIBCO Administrator by default.</p> <p>Default is <code>be.engine.trace</code>.</p>
<code>be.trace.roles</code>	<p>Comma-separated list of trace roles to enable.</p> <p>Available in TIBCO Administrator by default.</p> <p>Allowed values: <code>infoRole</code>, <code>errorRole</code>, <code>warnRole</code>, <code>userRole</code>, <code>debugRole</code></p> <p>Default is <code>infoRole, errorRole, warnRole, userRole</code>.</p>

Table 35 BusinessEvents Engine Properties to Determine Logging Behavior (Sheet 3 of 3)

Property	Notes
<code>be.trace.term.enable</code>	Enables tracing to terminal. Available in TIBCO Administrator by default. Default is true.
<code>Engine.Log.Dir</code>	Specifies the location of the engine log file. (The property <code>be.trace.log.dir</code> is not used). See Default Location of Log Files and Other Files and Directories on page 389 . Default is <code>logs</code> (that is, a directory called <code>logs</code> under the working directory).

Deploying a TIBCO BusinessEvents Project

This chapter explains how to deploy a TIBCO BusinessEvents project.

Also see [Chapter 26, Deploytime Configuration](#), on page 353.



Procedures for loading external classes from Decision Manager RMS server are documented in the *TIBCO BusinessEvents Decision Manager* guide.

Topics

- [Deploying a Project Outside a TIBCO Administrator Domain](#), page 380
- [Deploying a Project in a TIBCO Administrator Domain](#), page 383
- [TIBCO Hawk Application Management Interface](#), page 387
- [Determining Various Names and Locations](#), page 388

Deploying a Project Outside a TIBCO Administrator Domain

To run an engine outside of a TIBCO Administrator domain, you start the engine at the command line. Before starting the server and running a TIBCO BusinessEvents engine at the command line, you configure an enterprise archive resource (EAR) file for your BusinessEvents project within TIBCO Designer, and, as needed, you configure deploytime properties.

Read [Chapter 26, Deploytime Configuration, on page 353](#) to understand the deploytime configuration tasks. See [Configuring to Run Outside of a TIBCO Administrator Domain on page 368](#) for the procedure. Also see [Chapter 15, Configuring In Memory Object Management, on page 243](#) for information about fault tolerance as used for In Memory object management.

For information about deploying a BusinessEvents project using TIBCO Administrator, see [Deploying a Project in a TIBCO Administrator Domain on page 383](#).

A simple command-line interface allows you to start the BusinessEvents engine and run the application specified in a project EAR file, with some startup options.

Optionally, you can provide startup options in the `be-engine.tra` file itself (or a supplementary property file). Putting the options in the property file allows you to start the engine using the simple command with no arguments.

To Deploy a Project Outside of an Administration Domain

1. Open a command window and navigate to the location of the `be-engine` executable file. It is generally in `BE_HOME\bin`, for example:
`c:/tibco/be/3.0/bin.`
2. Enter a command using the following format:

```
be-engine [--propFile startup_property_file] [-p property_file] [-n engine_name]
[-h] [earfilename]
```



The ear file name must be the last argument



Alternatively, you can specify the startup properties in the `be-engine.tra` file itself using a line of the same format:

```
tibco.env.APP_ARGS [--propFile startup_property_file] [-p property_file] [-n
engine_name] [earfilename]
```

See [Command-line Engine Startup Options on page 381](#) for details on using the options.

You see engine startup messages in the console, similar to the following:

```
Using property file: C:\tibco\be\version\bin\be-engine.tra
*****
Using arguments :-n acme C:\BEProjects\FraudDetectionExample.ear
```

You also see project initialization messages. When the engine starts you see a message like: Engine *engine_name* started.

Command-line Engine Startup Options

[Table 36, Command-line Engine Startup Options](#), provides detailed information about the options.

Table 36 Command-line Engine Startup Options

Option	Description
<code>--propFile</code> <i>path_to_startup_properties_file</i>	When you execute <code>be-engine</code> , by default it looks in the working directory for a property file of the same name (<code>be-engine.tra</code>). This property file provides startup values and other parameters to the executable. You can specify a different startup property file using the <code>--propFile</code> parameter. See Default Location of Log Files and Other Files and Directories on page 389 for more on how working directory is determined.

Table 36 Command-line Engine Startup Options (Cont'd)

Option	Description
<i>-p path_to_properties_file</i>	<p>Allows you to pass one or more supplementary property files to <code>be-engine</code>. Supplementary property files can be used in addition to <code>be-engine.tra</code> (or alternate file you specified using <code>--propFile</code>).</p> <p>Supplementary property files typically have a <code>.cfg</code> or <code>.tra</code> extension. Properties are defined as a list of name-value pairs (<i>property=value</i>).</p> <p>Values in supplementary property files override the values in the startup property file. Values provided at the command line override values in the supplementary property files.</p> <p>If you specify multiple property files that include different values for the same parameters, BusinessEvents uses the value in the left-most file in the command line. For example, consider this command line:</p> <pre>be-engine -p first.tra -p second.tra -p third.tra</pre> <p>If <code>second.tra</code> and <code>third.tra</code> set different values for (for example) <code>tibco.clientVar.BUILDEAR</code>, and <code>first.tra</code> does not include this parameter, BusinessEvents uses the value in <code>second.tra</code>. However, if <code>first.tra</code> also includes a value for <code>tibco.clientVar.BUILDEAR</code>, BusinessEvents uses the value in <code>first.tra</code>.</p>
<i>-n engine_name</i>	<p>Allows you to provide an alternate name for the BusinessEvents engine. The name provided here is used in the console and in log files.</p>
<i>-h</i>	<p>Displays command-line help.</p>
<i>earfilename</i>	<p>To start an application, provide the archive (EAR file) path and name, for example:</p> <pre>be-engine c:\beprojects\demo.ear</pre> <p>The ear file name and path must be the last argument.</p>

Deploying a Project in a TIBCO Administrator Domain

Before deploying a TIBCO BusinessEvents project, you configure an enterprise archive resource (EAR) file for your BusinessEvents project within TIBCO Designer, and, as needed, you configure deploytime properties.

Read [Chapter 26, Deploytime Configuration, on page 353](#) to understand the deploytime configuration tasks, and [Configuring to Run in a TIBCO Administrator Domain on page 370](#) for the property configuration procedure.

This section discusses deployment using TIBCO Administrator. For information about running a BusinessEvents application at the command line, see [Deploying a Project Outside a TIBCO Administrator Domain on page 380](#).

You can configure your BusinessEvents engine to allow you to replace the EAR file without shutting down the engine. This is known as a *hot deployment*. See [Configuring and Using Hot Deployment on page 391](#).

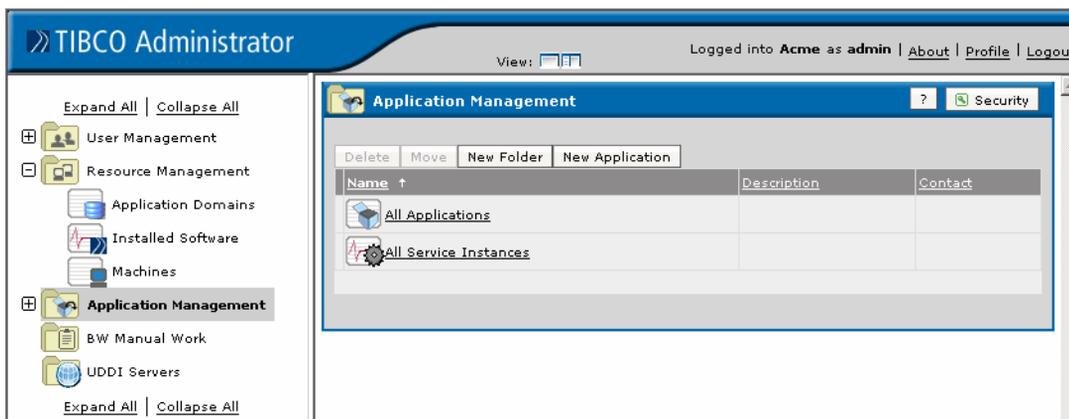


Rule Session (Agent) Configuration Options When you deploy a BusinessEvents EAR file, all the separate BAR files within it are merged.

To Deploy a Project EAR in a TIBCO Administrator Domain

1. As needed, ensure that all the following are started on the machine whose engine properties you want to change:
 - TIBCO Administrator service for the administration domain
 - TIBCO Hawk service for the administration domain
2. Start the TIBCO Administrator GUI:
 - Windows: Start > Programs > TIBCO > TIBCO Administrator Enterprise Edition *version* > TIBCO Administrator
 - Web browser: `http://host-name:port/` (where *host-name* is the machine name and *port* is the HTTP port specified during installation, 8080 by default)
3. Select the administration domain for the application and provide the username and password assigned during installation, or other administrator user credentials.

4. If you are deploying a project for the first time, do the following:
 - a. Click **Application Management** (in the left panel).
 - b. Click the **New Application** button.



- c. At the Upload EAR File dialog, click **Browse**, select the EAR file you want to deploy, and click **OK**.
 - d. At the New Application Configuration dialog, set the Application Parameters and Services settings as desired (click Help for details) and click **Save**. By default, Name is set to the EAR file name and Deployment Name is set to the EAR file name prepended with the domain name.
5. Expand to **Application Management** > *application_name* > **Configuration**.

Setting Engine Properties

6. As desired, in the Configuration Builder panel, click the merged BAR name (*application_name.bar*) and select the Advanced tab. Set properties as desired. See [Configuring to Run in a TIBCO Administrator Domain on page 370](#) for a more detailed procedure and references to related sections, especially [Customizing the List of Properties on the Advanced Tab on page 374](#). Click **Save**.



When you deploy the BusinessEvents engine using TIBCO Administrator, the property values set on the Advanced tab override values set for the same properties in the engine property (TRA) file.

7. As desired, in the Configuration Builder panel, click the application name and set values in the Advanced tab, for example, for global variables. Click **Save**.
8. As desired, select the machines in the administration domain to which you will deploy the application: In the Configuration Builder panel, click the

application name. In the General tab, Target Machines panel, the current machine is available by default.

Fault Tolerance Settings —for In Memory OM Only

9. As desired, select **Add to Additional Machines** to deploy the application to multiple machines. Select the machines to which you will deploy. You can also select the same machine more than one time, if you want to deploy the application more than once on a machine.
10. Fault tolerance features appear in the FT Group Setting pane when you select multiple machines. If you are configuring BusinessEvents engines as a fault tolerant group, available only for In Memory object management, do the following:

- a. For each machine, define a fault tolerance weight (priority).

The higher the number, the higher priority of the server. The primary engine has the highest priority (that is, the number with the largest value, when configuring using TIBCO Administrator). Secondary engines have lower priorities (and values closer to 1).

See [Chapter 15, Configuring In Memory Object Management, on page 243](#) for more configuration details.



If you give two or more engines the same priority value, then the actual priority is determined by join (server startup) time. The engine that joined the cluster first has the highest priority.

- b. In the FT Group Settings pane, check the **Run Fault Tolerant** checkbox.



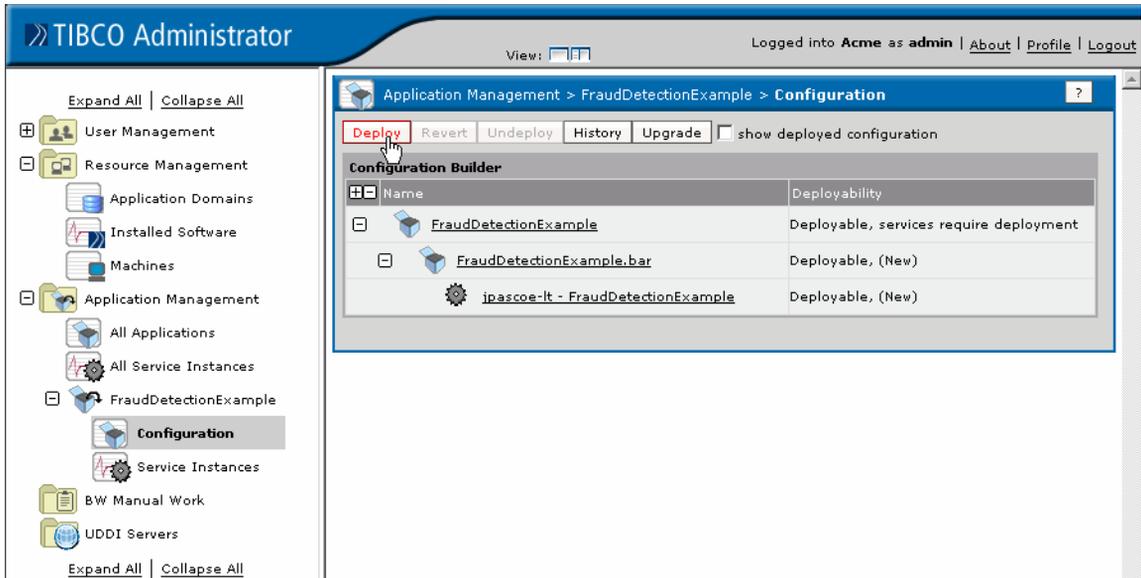
BusinessEvents does not use these TIBCO Administrator fault tolerance settings. They appear in the engine property file generated by TIBCO Administrator (see [Location of TIBCO Administrator-Generated Property File on page 389](#)), with the property names shown in parentheses:

- Heartbeat Interval (Engine.FT.HeartbeatInterval)
- Activation Interval (Engine.FT.ActivationInterval)
- Activation Delay. (Engine.FT.ActivationDelay)

The above settings are relevant only when TIBCO Rendezvous is used for fault tolerance. BusinessEvents fault tolerance for In Memory object management is managed by the object management layer, not by Rendezvous. See [Chapter 15, Configuring In Memory Object Management, on page 243](#) for more details.

11. Click **Save**.

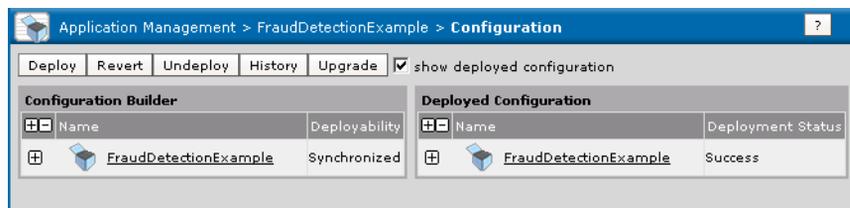
- Navigate up to the application's main Configuration dialog and click the **Deploy** button.



- At the Deploy Configuration dialog, configure settings if desired then click **OK**.

The application deploys, and the Configuration dialog displays again.

- Check the **Show deployed configuration** checkbox to display the Deployed Configuration panel and verify success:



TIBCO Hawk Application Management Interface

TIBCO BusinessEvents includes a set of TIBCO Hawk microagent methods that allow you to manage your TIBCO BusinessEvents deployment using TIBCO Hawk. These functions are listed and described in [Appendix I, TIBCO Hawk Microagent Methods, on page 481](#).

Determining Various Names and Locations

This section explains where various runtime items are located, and how the engine name is determined at runtime.

Determining the Engine (Node) Name

When establishing the engine name, BusinessEvents software searches for a value from one of the settings in the order shown, accepting the first value it finds:

1. API setting. If BusinessEvents is started using the public API, and a non-null instance name is provided when getting the `RuleServiceProvider` with `RuleServiceProviderManager.newProvider(String instanceName, Properties env)`—this takes precedence over all other name settings.
2. The engine name set at the command line using the `-name` option. An engine name set at the command line overrides the engine name property set in the `be-engine.tra` or supplementary property file (see next).
3. the engine name set by the `be.engine.name` property. This property can be set in the usual ways:
 - For command-line startup it can be set in the `be-engine.tra` file or in a supplementary property file.
 - For deployment to a TIBCO Administrator domain, it can be added to the Advanced tab (see [Customizing the List of Properties on the Advanced Tab on page 374](#)).
4. The name of the TIBCO Hawk microagent instance. This name exists if TIBCO Hawk is enabled at runtime. The microagent name can also be set in the `be-engine.tra` file using the property `Hawk.AMI.DisplayName`.
5. The host name.
6. This string: `engine`.

Determining the TIBCO Repo URL for BusinessEvents

When applications are started outside of a TIBCO Administrator domain, the TIBCO repo URL is the file location of the project EAR file. You add a property to the engine property file, `be-engine.tra` to specify the location:

```
tibco.repourl location of EAR file
```

When TIBCO Administrator is used to deploy a BusinessEvents application, the `tibco.repourl` property is added to the generated TRA file (see [Location of](#)

[TIBCO Administrator-Generated Property File on page 389](#)).

The value of this property is the URL of the project repository. The URL format depends on the deployment transport used. Supported formats for the URL are `tibcr`, `http`, `https`, and `file` (local).

You can use a `tibcr://` URI to identify the server-based repository location. For full details about its allowable parameters, refer to TIBCO Adapter SDK™ documentation.



One way to determine the repo URL is to temporarily deploy the BusinessEvents EAR. The repo URL is in the deployed BusinessEvents instance TRA file. However, you must remove the escape characters (back-slashes). You must then undeploy the BusinessEvents instance after obtaining the `tibco.repourl` value.

Location of TIBCO Administrator-Generated Property File

When it deploys a BusinessEvents project, TIBCO Administrator reads the properties from the engine property file located in `BE_HOME/bin/be-engine.tra` and then incorporates the deployment configuration values set in the TIBCO Administrator user interface. (Property values set in TIBCO Administrator override those set in the `BE_HOME/bin/be-engine.tra` file).

TIBCO Administrator then generates a property file for runtime use in this location:

```
TIBCO_HOME/tra/domain/domain_name/application/application_name/application_name.tra
```

After deployment, the generated file is used, and not the file in `BE_HOME`.

Default Location of Log Files and Other Files and Directories

The working directory and various file locations depend on the method of deployment: startup at the command line, or deployment using TIBCO Administrator.

Path names that do not start with the root directory are assumed by the operating system to start from the working directory.

Also see [BE Working Directory \(and Log File Location\) on page 457](#) for a note about how working directory is determined when you use BusinessEvents Tools

File or Directory	TIBCO Administrator Deployment	Start with <code>be-engine.exe</code>
Engine startup property file	<code>domain_name.domain_name-deployment_name.BAR_Name_prefix2Ebar</code>	<code>BE_HOME/bin/be-engine.tra</code>
Working Directory	<code>BE_HOME</code>	Directory in which you start the <code>be-engine.exe</code> executable
Engine log files	<p><code>TIBCO_DOMAIN_HOME/DOMAIN_NAME/application/logs</code></p> <p>Where <code>TIBCO_DOMAIN_HOME</code> is the location configured when creating the administration domain. For example, <code>c:/tibco/tra/domain/Acme/application/logs</code></p> <p>Use <code>Engine.Log.Dir</code> to change the log file directory.</p>	<p><code>working_directory/logs/engine-name.log</code></p> <p>Use <code>Engine.Log.Dir</code> to change the log file directory</p>
Database Environment Directory (for Persistence object management)	<code>BE_HOME/db/session_name</code>	<code>working_directory/db/session_name</code>
TIBCO Designer log files	N/A	<code>designer_home/bin/logs</code>

Chapter 29 **Configuring and Using Hot Deployment**

This chapter explains how to use the hot deployment feature to deploy changes to a running application.

Topics

- *Hot Deployment Overview, page 392*
- *Hot Deployment Supported Modifications, page 393*
- *Enabling and Disabling Hot Deployment, page 395*
- *Performing Hot Deployment in a TIBCO Administrator Domain, page 397*
- *Performing Hot Deployment Outside a TIBCO Administrator Domain, page 399*

Hot Deployment Overview

You can make certain changes to a BusinessEvents project and apply them to engine that is running the application, without stopping the application. This is known as hot deployment.

Modifications Allowed in Hot Deployment

You can make only certain changes during a hot deployment. They are listed in [Table 37, Hot Deployment Supported Modifications](#). Modifications allowed are also different for each type of object management.

If you attempt to deploy an EAR file that includes unsupported modifications without first stopping the BusinessEvents engine, BusinessEvents rejects the EAR file.

In some cases, if you want to make a change that is not supported by the hot deployment feature, for example, if you want to change an event expiry action for an event definition, you may be able to work around the limitation. For example, instead of changing the event expiry action in the event definition, you may be able to change the rule function used in the currently configured event expiry action.

Hot Deployment in a Fault Tolerance Group (In Memory OM)

In a fault tolerance group, you must first perform the hot deployment on the secondary servers. When the process is complete on the secondary servers, you can then safely perform the hot deployment on the primary server.

When hot deployment is enabled, the deployed application listens for changes in the EAR file. Therefore do not use the same copy of the EAR file (that is, the EAR file in the same directory location) for deploying primary and secondary servers. If you do, the hot deployment may occur out of order, and anomalous behavior may occur in the caches.

How Hot Deployment Occurs

Performing hot deployment requires changing the execution code at runtime. This is made possible using the `-javaagent` option. The `-javaagent` option is provided in the `be-engine.tra` file as shipped.

In an active agent, the hot deployment process waits for the current RTC cycle to complete and then injects the changes before the next RTC cycle starts.

Hot Deployment Supported Modifications

Check marks in [Table 37](#) identify the project modifications that are supported by the BusinessEvents hot deployment feature for In Memory and Persistence object management. For Cache object management, see the note following the table.



When using the hot deployment feature to deploy a project, BusinessEvents ignores changes to global variables.

Table 37 Hot Deployment Supported Modifications

Resource	New	Modify	Delete
Rule***	✓	✓	✓
Rule Function***	✓	✓	✓
Simple Event	✓		
Time Event	✓		
Concept	✓		
Channel			
Destination			
Rule Set***	✓	✓	✓
Concept View	✓	✓	✓
Score Card	✓		
BusinessEvents Archive		✓*	
Global Variable	✓		
State Machine	✓**		
State Machine Transition	✓**		
State Machine Timeout Expression	✓**		
State Machine Timeout Action	✓**		

Table 37 Hot Deployment Supported Modifications (Cont'd)

Resource	New	Modify	Delete
State Machine Entry Action	✓**		
State Machine Exit Action	✓**		
State Machine	✓**		

***** For Cache object management** Hot deployment is available only for rules, rule functions, and rule sets.

✓* The hot deployment feature only supports additions and/or changes to rule sets information in BusinessEvents Archives.

✓** The hot deployment feature only supports new state machines and state machine components within new concepts.

Enabling and Disabling Hot Deployment

As a safety measure you can enable and disable hot deployment. To enable and disable hot deployment, you use the engine property `be.engine.hotDeploy.enabled`. The property is available in TIBCO Administrator by default. Its default value is `false`.

You can set the property value to `true` or `false` in the `be-engine.tra` file, or using TIBCO Administrator. If you deploy using TIBCO Administrator, set the value using TIBCO Administrator.



You set the property value, then deploy and start the application, so that *subsequent* deployments can be done without stopping the engine. (In other words, you can't set the hot deployment property to `true` and immediately perform a hot deployment.)

To Enable Hot Deployment – Engine Property File

1. Open the `BE_HOME/bin/be-engine.tra` file for editing.
2. Locate the hot deployment property and set the value to **true**:

```
# Hot deployment
be.engine.hotDeploy.enabled true
```

3. Save and close the file.
4. Start or restart the BusinessEvents engine.



If you deploy using TIBCO Administrator, it is likely that the value you set in the `be-engine.tra` file for `be.engine.hotDeploy.enabled` will be overridden by the same-named property, which has a default value, on the Advanced tab of the BAR file in TIBCO Administrator. See [Customizing the List of Properties on the Advanced Tab on page 374](#) for more details.

To Enable Hot Deployment – TIBCO Administrator



This procedure assumes that you have generated the project EAR and uploaded it to TIBCO Administrator. You can configure before you deploy.

1. In the left pane of TIBCO Administrator, click **Application Management**>*project_name*>**Configuration**.
2. In the Configuration Builder panel, click *project_name.bar*.

3. Click **Advanced**.
4. Locate the hot deployment property, `be.engine.hotDeploy.enabled`.
5. Change the property value to **true**.
6. Click **Save**.
7. Deploy and start the BusinessEvents application.

You can now perform hot deployment to the running application. See [Performing Hot Deployment in a TIBCO Administrator Domain on page 397](#) and [Performing Hot Deployment Outside a TIBCO Administrator Domain on page 399](#) for details.

Performing Hot Deployment in a TIBCO Administrator Domain

This section explains how to perform hot deployment when the BusinessEvents project has been deployed to a TIBCO Administrator domain.

Modify the Project as Needed and Build the EAR File

In TIBCO Designer, modify the BusinessEvents project according to your needs. Not all modifications are supported with hot deployment. See [Table 37, Hot Deployment Supported Modifications](#) for a list of supported modifications.

Next, regenerate the project EAR file.



The new EAR file must have the same name as the existing one.

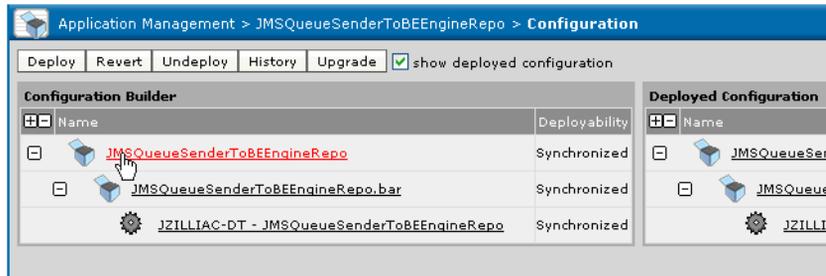
Enable Hot Deployment (As Needed)

Ensure that you have enabled hot deployment, as explained in [Enabling and Disabling Hot Deployment on page 395](#).

Perform Hot Deployment

1. As needed, ensure that all the following are started on the machine whose engine properties you want to change:
 - TIBCO Administrator service for the administration domain
 - TIBCO Hawk service for the administration domain
2. Start the TIBCO Administrator GUI:
 - Windows: Start > Programs > TIBCO > TIBCO Administrator Enterprise Edition *version* > TIBCO Administrator
 - Web browser: `http://host-name:port/` (where *host-name* is the machine name and *port* is the HTTP port specified during installation, 8080 by default)
3. Select the administration domain for the application and provide the username and password assigned during installation, or other administrator user credentials.
4. Expand to **Application Management** > *application_name* > **Configuration**.

- In the Configuration Builder panel, select the application (at the base of the tree).



- In the Edit Application Configuration dialog, click **Upload New EAR File**.
- At the Upload EAR File dialog, click **Browse**, select the EAR file you want to deploy, and click **OK**.
- Confirm the upload by clicking **OK** again, then click **Save**. Verify that the Deployability column displays *Deployable*.
- Click **Deploy**. You see the Deploy Configuration dialog.
- Uncheck* these checkboxes (if they are checked):
 - **Stop running services before deployment**
 - **Start successfully deployed services**
 - **Force redeployment of all services**
 (When the Stop running services before deployment checkbox is checked, you see an additional setting, Kill services that haven't stopped after (seconds). It is removed when you uncheck the checkbox.)
- Click **OK**. TIBCO Administrator performs the hot deployment of your modified BusinessEvents project.

Performing Hot Deployment Outside a TIBCO Administrator Domain

This section explains how to perform hot deployment when the BusinessEvents project has been deployed to a TIBCO Administrator domain.

Modify the Project as Needed and Build the EAR File

In TIBCO Designer, modify the BusinessEvents project according to your needs. Not all modifications are supported with hot deployment. See [Table 37, Hot Deployment Supported Modifications](#) for a list of supported modifications.

Next, regenerate the project EAR file.



The new EAR file must have the same name as the existing one.

Enable Hot Deployment (As Needed)

Ensure that you have enabled hot deployment, as explained in [Enabling and Disabling Hot Deployment on page 395](#).

Add a Property to the Engine Property File

Add a line to the `be-engine.tra` file to specify the location of the EAR file:

```
tibco.repourl location of EAR file
```

Perform Hot Deployment

Place the modified EAR file in the location specified in the `be-engine.tra` file:

```
tibco.repourl location of EAR file
```

The engine notices the changed file and performs the hot deployment at the next RTC cycle.

Appendix A Engine Startup and Shutdown Sequence

This section helps you understand the main actions that occur during engine startup and shutdown (in normal circumstances). In any particular project only some of the actions may be required. For example, a project may have no startup rule functions. See [Working With Startup and Shutdown Rule Functions on page 142](#) for related information.

This section assumes cache OM. It provides the main milestones only and focuses on nodes running inference agents.

Startup Sequence

During engine startup the following actions occur:

1. System information displays (in consoles) and is recorded in the log file:
 - The property file and EAR file that were used to start the engine.
 - The version of the JAR files it is using, and the version of the JAR files that the EAR file was built with.
 - If persistence OM is used, the location of the Berkeley DB software it is using, and information about what was recovered from the database.
2. All inference agents build their Rete networks and working memory starts.

If a backing store is in use (Cache OM only), cache data is recovered from the backing store and populates cache servers (and JVMs of any other storage enabled agents).

(Cache OM only) Inactive Nodes If all agents in an engine (node) are inactive, then this ends the startup sequence for that engine.
3. Channels start up for outbound traffic (inbound listeners do not start yet).
4. Scorecards are created.
5. Startup functions execute (for example, they initialize values of scorecards).
6. The first RTC cycle occurs and all rule actions that are eligible to execute now execute. (Scorecards and startup rule functions can cause rules to be eligible to execute. Depending on the state of entities recovered from the backing store, the RTC will take more or less time.) See [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#) for more details about RTC cycles.
7. The engine startup advisory event is asserted, and its RTC occurs (as needed).

8. Time events (if any) are asserted:
 - The clock starts for repeating time events and they are created and asserted at the specified intervals.
 - Rule-based events (recovered or scheduled in a startup action) are asserted after the specified delay. The delay begins when the rule or rule function action executes, so at startup, it is possible for time events to have passed their start time, and they are asserted immediately.
9. Finally, inbound channel listeners activate and accept incoming events and the system is now fully started up.

Shutdown Sequence

During engine shutdown the following main actions occur:

1. Inbound channels and listeners shut down
2. Shutdown rule functions execute
3. An RTC occurs (as needed).
4. Outbound channels shut down.

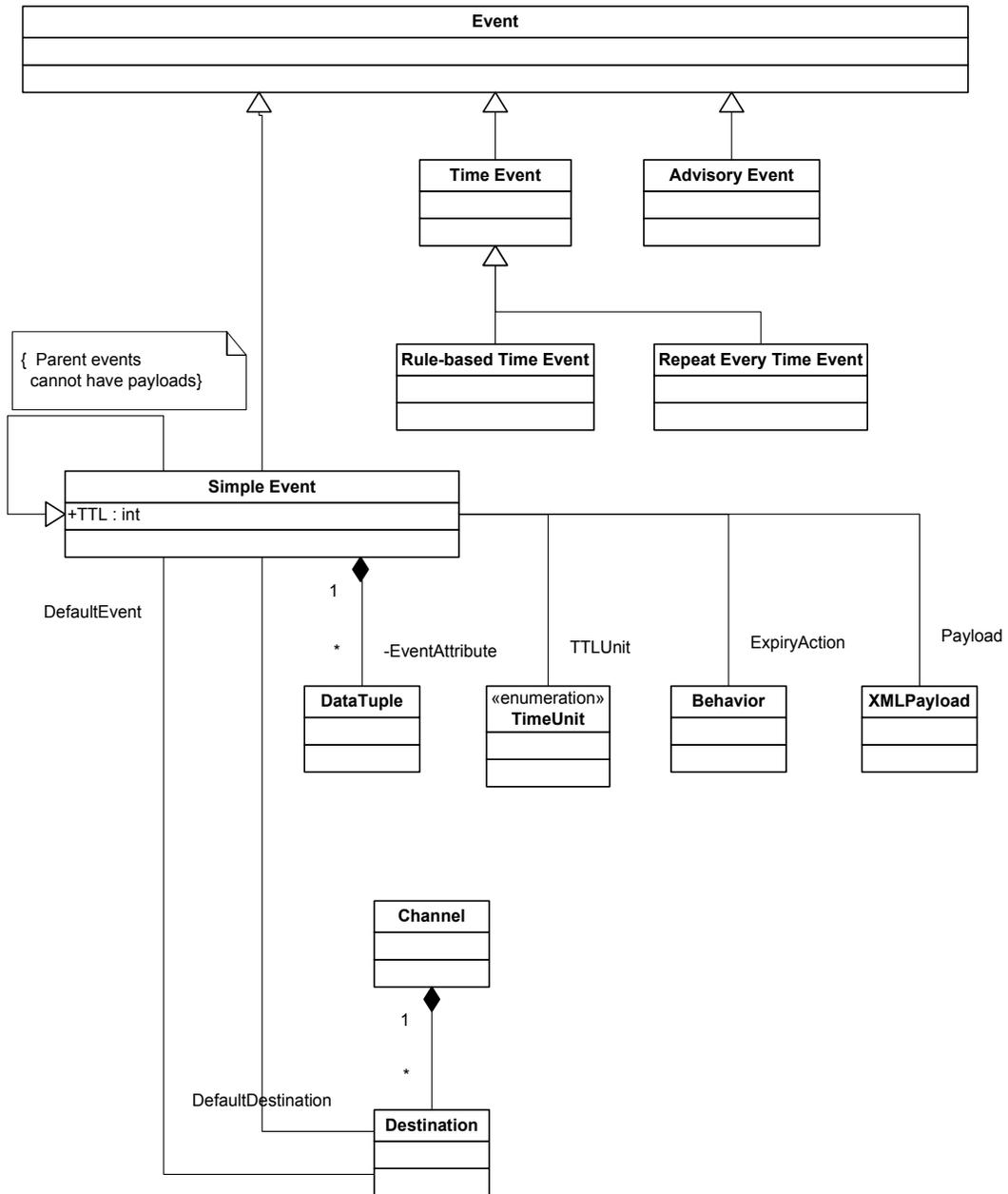
Appendix B **Object Models**

This appendix contains class diagrams showing the object models for rule, event, concept and scorecard entity types, and for state models (available only in BusinessEvents Enterprise Suite).

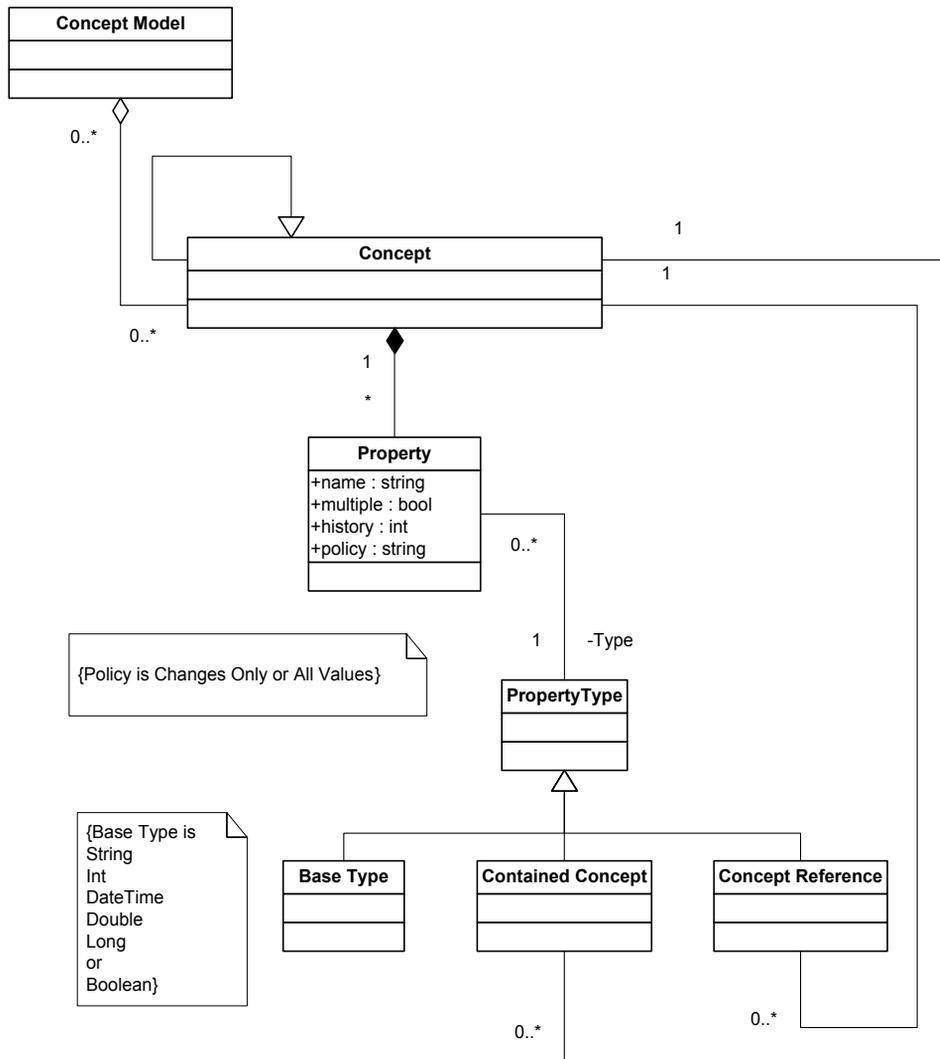
Topics

- [Event Model, page 404](#)
- [Concept Model, page 405](#)
- [Rule and Ruleset Model, page 406](#)
- [State Model—Overview, page 407](#)
- [State Model—State Detail, page 408](#)

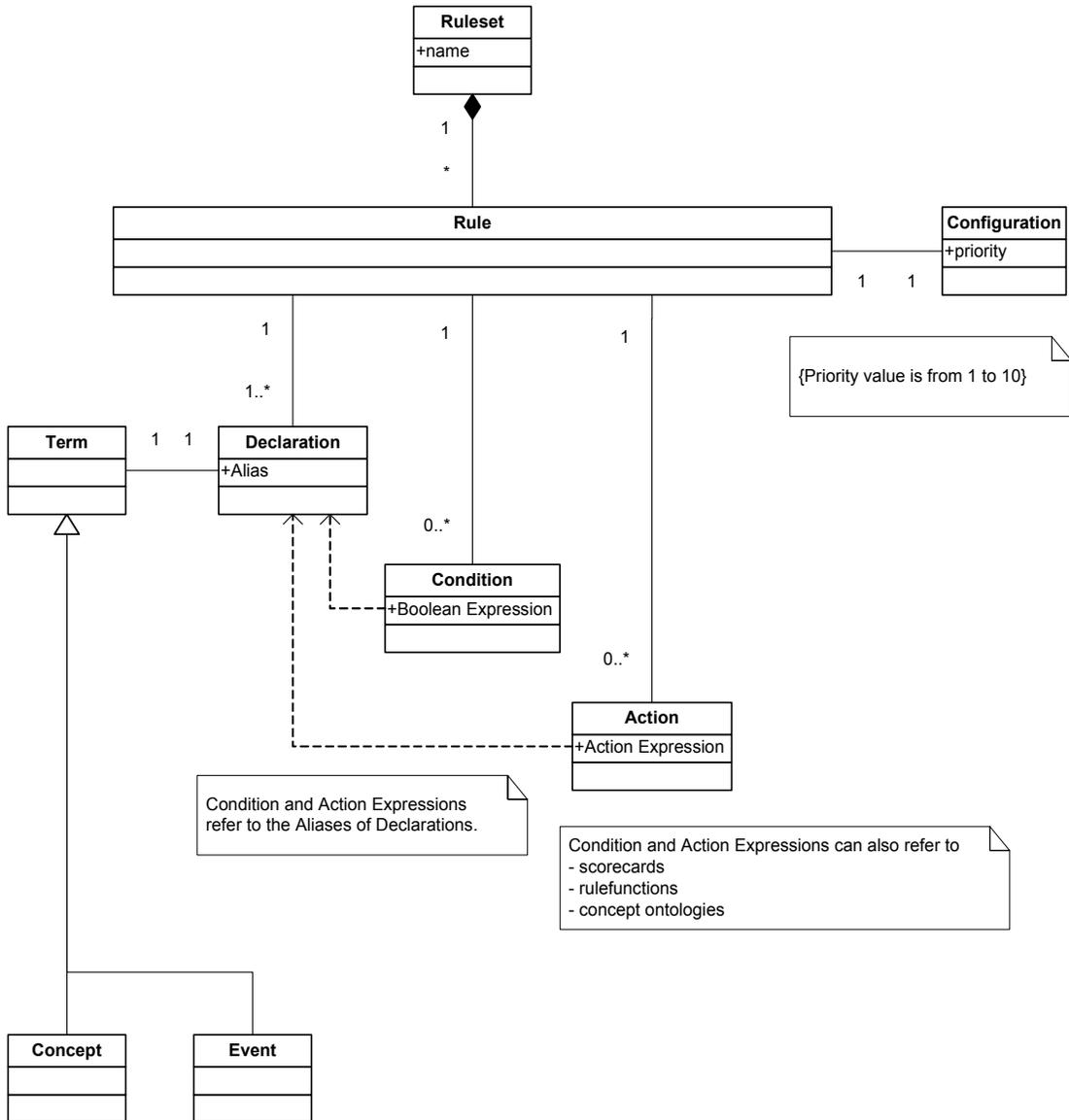
Event Model



Concept Model

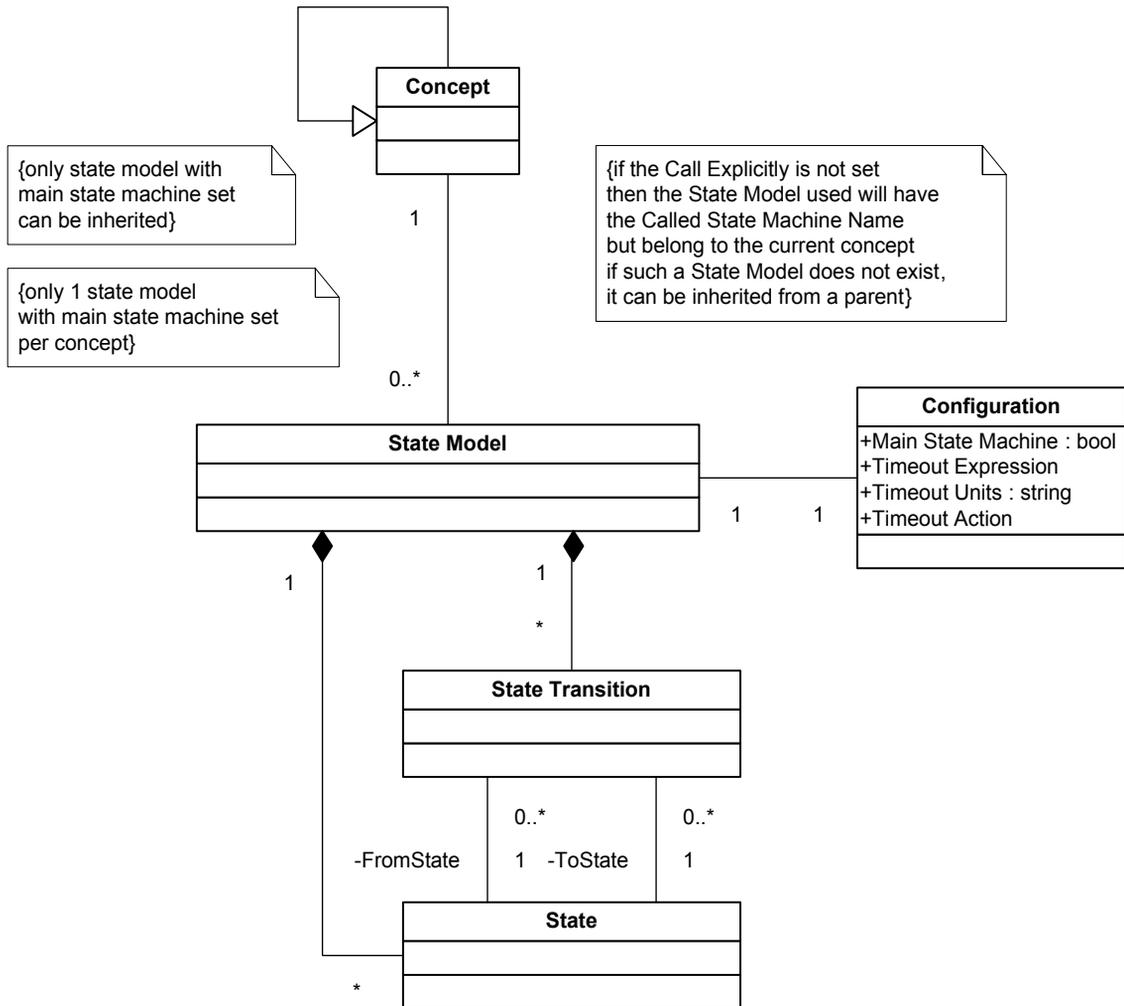


Rule and Ruleset Model



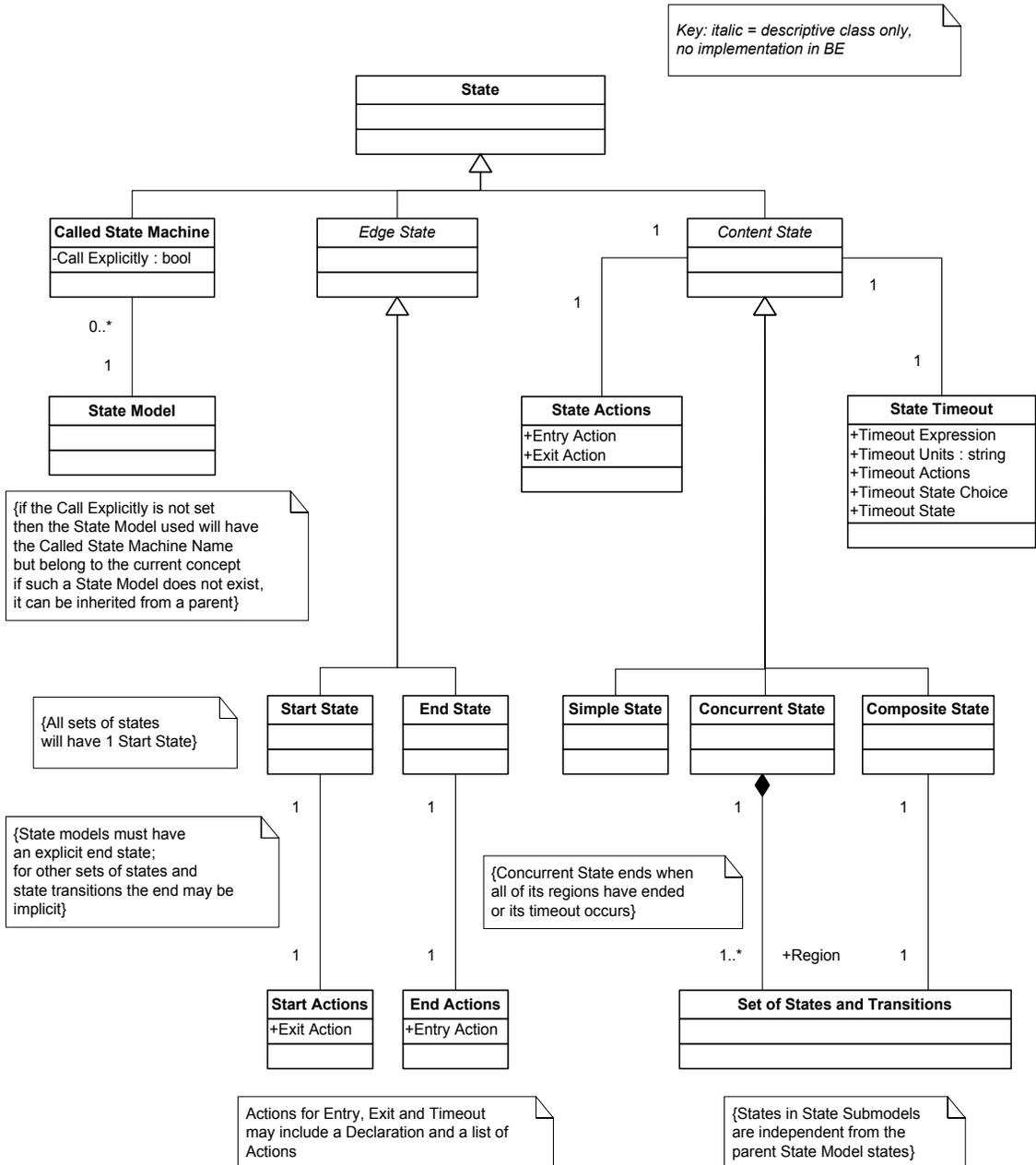
State Model—Overview

See also, [State Model—State Detail on page 408](#).



State Model—State Detail

See also, [State Model—Overview on page 407](#).



Appendix C **Advanced Caching Topics**

This appendix provides details about caching topics that are useful only in some circumstances or provide background information for interested readers.

Topics

- *Provided Caching Schemes, page 410*
- *Overriding and Extending the Operational Deployment Descriptor, page 411*
- *Specifying Operational Override File Locations, page 413*
- *Understanding Entity Caches, page 416*

Provided Caching Schemes

Caching schemes determine the specific characteristics and behavior of the cache. The provided schemes have been configured for BusinessEvents use. Common configuration options are available using properties (see [Configuring Caching Scheme, Multi-Engine, and Cluster Properties on page 296](#)). In most cases there is no need to be aware of the caching scheme that underlies your configuration options.

All caching schemes are for a distributed cache, which has been determined to be the best option for BusinessEvents. The table below provides basic information in case you need to work with Support to configure the caching scheme options beyond the provided configuration options.

Backing Store Schemes Use Write-Behind	BusinessEvents performs backing store operations in bursts after each RTC. Write-behind is used for both backing store schemes, with a write-delay of one second. With write-behind, which is asynchronous, set the JDBC connection pool size to match (or exceed) the number of entity <i>types</i> you expect to modify. Each entity type has its own write-behind queue.
--	---

Table 38 Provided Caching Scheme Names

Cache Name (in TRA)	Scheme Name	Notes
dist-unlimited-bs	default-distributed	Distributed cache with backing store and no expiry and eviction. Requires implementation of a backing store.
dist-limited-bs	default-distributed-limited	Distributed cache with backing store and limited entries (for use with backing store only. If you don't use backing store, use a scheme with unlimited entries) Requires implementation of a backing store.
dist-unlimited-nobs	distributed-nobs	Distributed cache with no backing store and no expiry and eviction This is the default value.

Overriding and Extending the Operational Deployment Descriptor

The operational deployment descriptor is called `tangosol-coherence.xml`. This file is provided in `BE_HOME/lib/ext/coherence.jar`. In order to configure certain cache-related settings, you may need to override the values of certain elements in the operational deployment descriptor. This is explained in [Overriding Element Values in Engine Property Files on page 411](#).

You may also need to define additional elements, and system properties for existing elements that lack them.

Overriding Element Values in Engine Property Files

BusinessEvents overrides the values of elements in the operational descriptor using their `system-property` attributes. The `system-property` attribute values are used in Java command line options, and those options are included in some standard BusinessEvents engine properties.

For example, the `system-property` attribute assigned to the element `<cluster-name>` is `tangosol.coherence.cluster`. BusinessEvents uses this attribute to specify a value for `<cluster-name>` in the engine property (TRA) files as follows:

```
java.property.tangosol.coherence.cluster myclustername
```

System properties are predefined for many operational elements. You can override the value of any element that has a `system-property` attribute by adding a Java command line option in the engine property files, as shown in the example above.

For detailed documentation of this feature, and a list of elements for which `system-property` attributes are predefined, see the section "Line Setting Override Feature" in the online reference, *TIBCO BusinessEvents Cache Configuration Guide*.

Defining Additional Elements and System Properties in Override Files

If there is no `system-property` attribute for a setting you want to override, or if you need to add more elements than are available in the operational deployment descriptor, you must create an override file (or a series of override files) to contain the `system-property` attributes and any new elements you require. For example, you might add more elements to provide well-known addresses for the servers in a cache cluster.

You can store default values in an override file, and you can also create engine properties (using the `system-property` attributes) that override those file values. The settings you enter in engine property files override settings in the override file, and settings in the override file override those in the operational descriptor.

Override file elements that also exist in the operational descriptor must use the same structure as the elements in the operational descriptor.

Example: Adding Well Known Address Attributes

You would need to add elements if you use the well-known address method of defining a cache cluster (see [Configure the Cache Cluster Discovery Settings on page 279](#)) and need more than six well-known addresses. (Six well-known addresses are declared in the default file, `tangosol-coherence.xml`.) Here is an example definition:

```
<well-known-addresses>
  <socket-address id="1">
    <address system-property="tangosol.coherence.wka"></address>
    <port system-property="tangosol.coherence.wka.port">8088</port>
  </socket-address>
</well-known-addresses>
```

In the override file, for example, `tangosol-coherence-override-prod.xml`, you add similar entries to the one provided in `tangosol-coherence.xml`, each specifying a unique `system-property` setting. For example:

```
<well-known-addresses>
  <socket-address id="7">
    <address system-property="tangosol.coherence.wka7"></address>
    <port system-property="tangosol.coherence.wka.port">8089</port>
  </socket-address>
  <socket-address id="8">
    <address system-property="tangosol.coherence.wka8"></address>
    <port system-property="tangosol.coherence.wka1.port">8089</port>
  </socket-address>
</well-known-addresses>
```

You can then use the `system-property` values in engine properties, and (in the case of the example shown) define the well known address values.

Specifying Operational Override File Locations

As explained in [Overriding and Extending the Operational Deployment Descriptor on page 411](#), in order to define additional elements or add system properties for existing elements, you use an override file.

You can actually use two or more tiers of override files depending on your needs:

- The first tier settings override and extend the operational deployment descriptor
- The second tier settings override and extend the first tier overrides.
- More tiers can be added as needed, though this is unlikely.

The reason for using multiple tiers is to enable all (or many) nodes to use the first tier overrides, while providing additional second tier overrides to selected nodes. For example, you may want to enable verbose logging on only one or two nodes for diagnostic purposes. To do so you would provide those settings in a second tier override file that you then reference in the selected nodes' engine property files.

How the First Tier Override File Default Location is Specified

The operational descriptor file, `tangosol-coherence.xml` (located in `BE_HOME/lib/ext/coherence.jar`) contains an `xml-override` property configured to point to a default override file:

```
<coherence xml-override="{tangosol.coherence.override
/tangosol-coherence-override-{mode}.xml}">
```

The structure of the property is:

```
xml-override={property default-property-value}
```

where *property* is the name of the override property, and *default-property-value* is the filepath and name of the file.

How BusinessEvents Uses the Override Property

The object management layer looks for the specified override file in the classpath.

The default location of the file specified by `tangosol.coherence.override` is a file at the root level of the class path. BusinessEvents uses the first instance of this file that it finds at the root level of the classpath. For example, if you put an instance of this file at the root level of a different JAR file that is located closer to the beginning of the classpath, then that file is used instead of the one in `coherence.jar`.

If a user overrides the default override location, BusinessEvents looks for the file specified in the same way.

Optional User-Defined Property for Flexibility in the Filename

You can use a user-defined property such as `{mode}` to define name patterns that provide flexibility. You can place multiple files that use the name pattern in the specified override location. For example, each of the following has a different value for `{mode}`: `tangosol-coherence-override-dev.xml`, `tangosol-coherence-override-test.xml`, and `tangosol-coherence-override-prod.xml` files. Then you can easily switch between files that have values appropriate for those environments at engine start-up.

You specify the value to use for the user-defined property in your TRA file, or at the command line.

For example, using `{mode}` as the user defined property, to specify the value in the TRA file, use:

```
java.property.mode Mode_Value
```

To override the value at the command line, use

```
-Dmode Mode_Value
```

Where (still using the example defined above) the *Mode_Value* is one of `test`, `prod`, or `dev`.

The default value for `{mode}` is `prod` (defined in the `tangosol-coherence.xml` descriptor). Therefore the default value for the first tier override file is `/tangosol-coherence-override-prod.xml`.

How to Specify a Different Location for the First Tier Override File

To override the default location of the first tier override file, you provide a value for the `tangosol.coherence.override` property either at the command line or using an engine property. You can do this one time only, on engine startup. The value and file contents must be the same on all nodes in a cluster.

The value of the property can be a file path or a JAR URL. It must be a location in the classpath.

To specify the override using an engine property, add the property name (specified in the `xml-override` property) and its value to the TRA files. All nodes in a cache cluster must specify the same file. For example:

```
java.property.tangosol.coherence.override=file:/c:/tmp/my_tangosol-coherence-override.xml
```

For URL locations inside any JAR, specify the path as in the following example:

```
java.property.tangosol.coherence.override=file:/home/jsmith/tmp/client/lib/coherence.jar!/my_tangosol-coherence.override.xml
```

You can alternatively specify the override as a system property, that is, a command line parameter at engine start-up. For example,:

```
-Dtangosol.coherence.override=file:/C:/tmp/my_tc-override.xml
```

How a Second Tier Override File Default Location is Specified and Overridden

The first tier override file can itself specify the default location and name of a second tier override file, again using the `xml-override` property:

```
xml-override={property default-property-value}
```

You would configure the property using different values, but the mechanism is the same. For example, you might use the property name `be.coherence.override`. You can also use another user property for part of the filename, if you want to provide that flexibility, for example:

```
<coherence xml-override="{be.coherence.override  
/be-coherence-override-{be.coherence.environment}.xml}"
```

As explained above, if you want to specify a file of this pattern, for example, `be-coherence-override-dev`, you could specify the value in the TRA files or at the command line on engine start-up. Here is an example showing how the system property value is defined at the command line:

```
-Dbe.coherence.environment=dev
```

Similarly, if you want to specify a different location and filename for the second-tier override file, you would use the `be.coherence.override` property, for example:

```
-Dbe.coherence.override=file:C:/tmp/tango-coherence-override.xml
```

Understanding Entity Caches

This appendix supplements information provided in [Chapter 17, Understanding Cache OM and Multi-Engine Features, on page 263](#). It is provided for those who want to understand the internal structures of the caches used in Cache object management. This information is not required for configuration tasks.

For each entity in working memory, a corresponding cache exists in the cache cluster. Internal entities also have caches for various purposes, explained in this section.

Entity Cache Names Format

Each entity cache has a name, which uses the following format:

cache-type.cluster-name.AgentGroupName.entity-name

The elements of the above name are explained below

Cache Type (Caching Scheme)

Cache type is the type of caching scheme (as defined by its cache name in the `coherence-cache-config.xml` descriptor), for example, `dist-unlimited-bs`. Several cache types are provided. They are described in [Table 38, Provided Caching Scheme Names, on page 410](#).

Cluster Name

Cluster name is the value of the following property:

```
java.property.tangosol.coherence.cluster
```

Agent Name

This field of the cache name is blank because BusinessEvents does not support agent-specific entity caches.

All entities are globally scoped and available to all agents.

Entity Name

Two types of entities have caches:

- Internal entities
- Ontology entities

Internal entity names and caches are listed and described in [Table 39, Internal Entity Caches, on page 417](#).

The ontology entity field of the entity cache name uses the entity's generated class name, which is similar to its design-time folder path and name, prefixed by `be.gen`. For example:

```
be.gen.Concepts.LargeConcepts.ThisLargeConcept
```

Caches for Ontology Objects

These caches are used to store the objects of types defined in the ontology of the project.

The types of caches created for ontology objects depend on the caching scheme used. If the `dist-unlimited-bs` caching scheme is used, then the cache names look like this:

```
dist-unlimited-bs$foo$be.gen.Order
```

Where `foo` is the cluster name.

Caches for Internal Entities

The following internal caches use a pre-defined scheme in the cache configuration file. Do not change this scheme. This information is provided for reference only.

Table 39 Internal Entity Caches

Entity (Cache) name	Purpose of the Cache
Master	Maintains the cluster state and is shared by all nodes.
Catalog	Maintains a cached copy of all ontology definitions shared by all nodes.
TypeIDs	Stores the mapping between type IDs and class names. All ontology objects are tagged with a unique integer ID. Use of IDs avoids the need to serialize and send class name strings between nodes.
ObjectTableIDs	Stores the key mapping for all objects in the cluster. The objects themselves are stored in their respective caches.
ObjectTableExtIDs	Stores the external key mapping for all objects that have an external ID (<code>extId</code>).
AgentTable	Stores all the agents and their respective states across all cluster nodes and identifies the currently active and inactive nodes.

Table 39 Internal Entity Caches

Entity (Cache) name	Purpose of the Cache
<i>AgentTxn-agentId</i>	Each agent in the cluster has an <i>AgentTxn-agentId</i> cache. The <i>agentId</i> is internally generated. It stores the change list for the agent. The change list is used to replicate changes between active-active and active-passive sets of agents in the cluster so that they stay synchronized.
<i>TimeQueue</i>	Maintains all entries that are time bound, for example, state machines that can have timeouts at a state machine level or at a state level. This cache maintains an index to all objects that must be re-evaluated after a certain period of time.

Appendix D **BusinessEvents Tools Overview**

This chapter gives you an overview of the BusinessEvents Tools applications, Rule Analyzer and Rule Debugger.

See the following chapters for related topics:

- [Appendix E, Working With Rule Analyzer, page 433](#)
- [Appendix F, Working With Rule Debugger, page 449](#)
- [Appendix G, BusinessEvents Tools Reference, page 461](#)

Topics

- [BusinessEvents Tools Overview, page 420](#)
- [BusinessEvents Tools—Rule Analyzer User Interface Overview, page 423](#)
- [BusinessEvents Tools Graph Elements, page 427](#)
- [BusinessEvents Tools—Rule Debugger User Interface Overview, page 428](#)

BusinessEvents Tools Overview

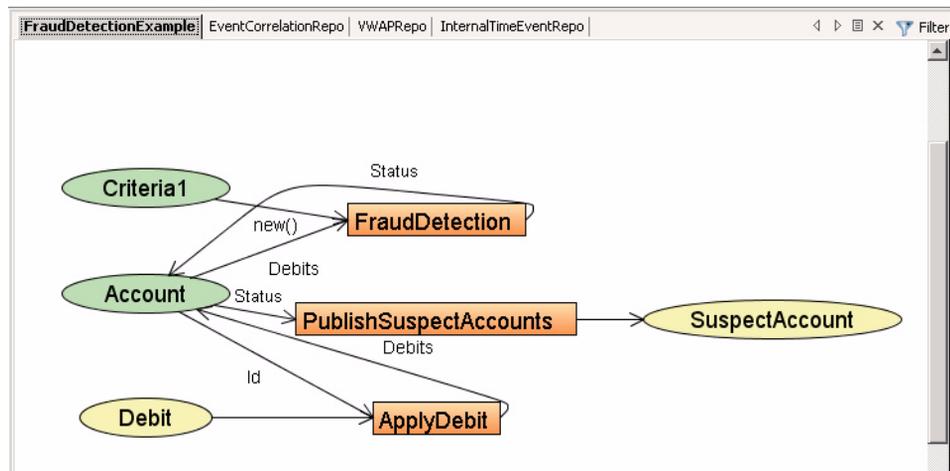
Two tools help you analyze and troubleshoot your BusinessEvents projects: Rule Analyzer and Rule Debugger. These tools are provided in an application called BusinessEvents Tools, which runs independently. You can start the application from within TIBCO Designer or at the command line. You can switch between Rule Debugger and Rule Analyzer when you are using the BusinessEvents tools application.

Rule Analyzer lets you check the structure of a project before you build its EAR and deploy it. You can examine several projects in Rule Analyzer simultaneously, using a tabbed interface. Unlike Rule Debugger, Rule Analyzer does not work with facts, just with the ontology.

Rule Debugger has all the features of Rule Analyzer, except some filtering and viewing features. Rule Debugger has additional features that enable you to examine a BusinessEvents application in action. By providing test data to the running application, you can examine its facts and behavior, as well as its ontology.

Rule Debugger works against EAR files that contain one or more BusinessEvents archives. You can work with several projects. Each Rule Debugger session corresponds to one engine. Similarly, for projects that use multiple rule sessions, you can view each rule session using tabs.

Ontology Graphs



Central to the Rule Analyzer and Rule Debugger tools are the graphs generated for each BusinessEvents project that you open. Each graph summarizes possible rule evaluation paths based on assertion of facts into working memory.

The graphs show the relationships between ontology entities (concepts, events, scorecards, and rules) using color-coded shapes and various kinds of linking lines. You can click each object in a graph to view its properties and other information in different panels of the user interface.

In Rule Analyzer (but not Rule Debugger), you can also display (or hide) class diagram relationships between entities in the graph, and you can display state machine graphs. You can filter the graph to focus on a subset of entities.

Many layout styles are available for graphs. It is recommended that you try different layouts until you find the ones that render the structure of your BusinessEvents projects most clearly.

Rule Analyzer Benefits

Users who are familiar with BusinessEvents rules can use the graphs (and associated information) to follow the logic of rule execution and find areas for improvement. Users who are less familiar with the product can learn more by close examination of the graphs.

Using Rule Analyzer, you can make changes in the project and view the results before deploying the project. Working with Rule Analyzer can reveal useful information such as the following:

- How changes to the project might affect object interdependencies. For example, how a rule firing would cause rule conditions to be re-evaluated.
- How changes in inheritance, property types, conditions, and dependencies may affect existing rule execution.
- Which rules are independent from one another. This information is helpful for partitioning your projects into independent rule sets for better scaling, and for better load balancing, in the case of multiple rule sessions (agents) in one EAR.
- Where there are inefficiencies in conditions and overall design such as unused identifiers or properties, most-frequently-used objects, and bottleneck rules.

Rule Debugger Benefits

Rule Debugger shows you how a deployed BusinessEvents application will behave in response to data. It shows potential problems in execution of rules. You can run without pausing, or in step mode.

When you run Rule Debugger in step mode, you can examine the current state of the rule engine at each rule execution. You can see information about the facts related to the rule. You can see the context of the rule execution—which rules have already fired, and which are going to fire (given the current evaluation). Step mode can be used as a learning tool, as well as a diagnostic tool.

Rule Debugger can use any project archive (EAR) that contains a BusinessEvents archive. The EAR file can be local or on a remote machine accessible through the file system.

BusinessEvents Tools—Rule Analyzer User Interface Overview

Rule Analyzer provides a clickable ontology graph and three additional panels with related information. Rule Debugger includes two additional panels, documented in the section [BusinessEvents Tools—Rule Debugger User Interface Overview on page 428](#).

The screenshot displays the TIBCO BusinessEvents Rule Analyzer application window. The interface is divided into several panels:

- Properties Panel:** Located on the left, it shows a table of properties for the selected rule set.

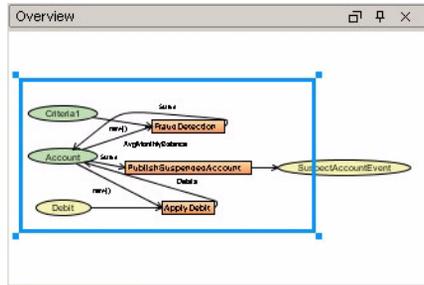
Name	Value
Name	PublishSuspendedAccount
Folder	/Rules/RuleSet
Desc	
Author	
Priority	5
- Source Panel:** Located at the top right, it displays the source code for the rule set.


```
rule Rules.RuleSet.PublishSuspendedAccount {
  declare {
    Concepts.Account account;
  }
  when {
    account.Status=="Suspended";
  }
  then {
    System.debugOut("-----");
  }
}
```
- Overview Panel:** Located at the bottom left, it shows a small, simplified version of the ontology graph.
- Analyzer/Dependency Panel:** Located at the bottom right, it shows a detailed ontology graph with nodes and relationships.
 - Nodes: Criteria1, Account, Debit, FraudDetection, PublishSuspendedAccount, ApplyDebit, Sus.
 - Relationships:
 - Criteria1 → new() → FraudDetection
 - Account → AvgMonthlyBalance → FraudDetection
 - Account → Status → PublishSuspendedAccount
 - Account → new() → PublishSuspendedAccount
 - Debit → new() → ApplyDebit
 - PublishSuspendedAccount → Debits → ApplyDebit
 - PublishSuspendedAccount → Sus

Labels with arrows point to each of these panels: Properties Panel, Source Panel, Overview Panel, and Analyzer/Dependency Panel.

A summary of the purpose of each panel in Rule Analyzer follows. For a detailed reference to the user interface, see [Appendix G, BusinessEvents Tools Reference, on page 461](#).

Overview Panel

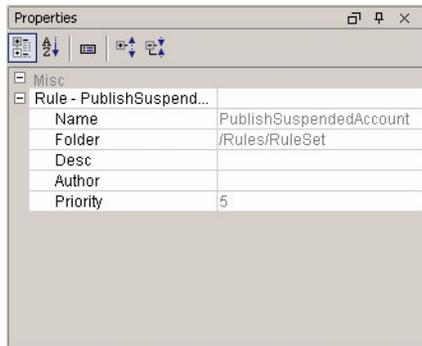


The overview panel (in the lower left of the user interface) shows the entire graph for the project. The blue box shows what is currently displayed in the Analyzer/Dependency panel.

When you drag the blue box over different parts of the overview graph, the parts of the graph that are contained in the blue box display in the Analyzer/Dependency panel (see [Analyzer/Dependency Panel on page 425](#)).

When you resize the blue box, the zoom level in the Analyzer/Dependency panel changes accordingly, to display the part of the graph contained in the blue box. The blue box also resizes when you use the interactive zoom and pan tools to resize the graph in the Analyzer/Dependency panel.

Properties Panel



The properties panel (in the upper left of the user interface) displays the properties of the entity that is currently selected in the graph (displayed in the Analyzer/Dependency panel).

The Folder property shows the position of the entity in the TIBCO Designer project folder hierarchy. Other properties are set in TIBCO Designer, using the Configuration and Properties tabs or are derived from related information, for example, the name of the rule author.

Source Panel

```

Source - /Rules/RuleSet/PublishSuspendedAccount
rule Rules.RuleSet.PublishSuspendedAccount {
  declare {
    Concepts.Account account;
  }
  when {
    account.Status=="Suspended";
  }
  then {
    System.debugOut("-----");
  }
}

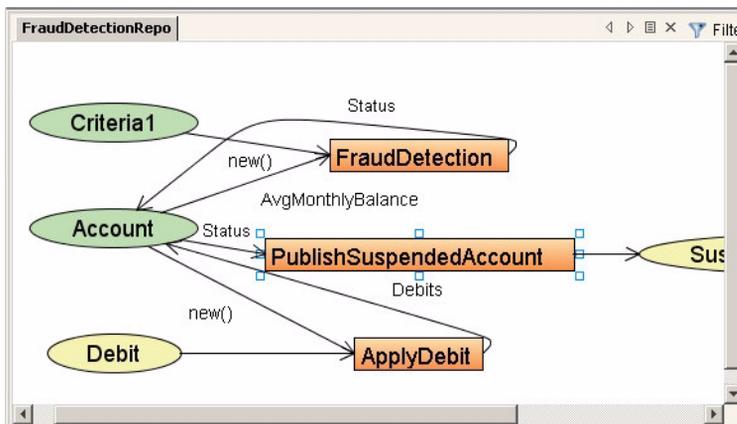
```

The Source panel (in the top right of the user interface) displays the details of the rule selected in the Analyzer/Dependency panel.

To change the rule, you open it in the rule editor in the TIBCO Designer project. The rule is presented differently in TIBCO Designer and in the Source panel:

- The Source panel **declare** block corresponds to the rule editor **Declaration** area.
- The Source panel **when** corresponds to the rule editor **Conditions** area.
- The Source panel **then** block corresponds to the rule editor **Actions** area.

Analyzer/Dependency Panel



In Rule Analyzer this panel has no name. It contains the main view of the dependency graph. Tabs just above the graph allow you to switch to a different project or application if multiple are open.

See [BusinessEvents Tools Graph Elements on page 427](#) for details about the shapes and links that can appear in a graph.

You can click an entity to display its information in the other panels (such as the Source and Properties panels). You can click a link to highlight its path. You can also enable the Link Navigator mode (click the  button) and then click on any part of a link to follow it from start to end. Property names on links indicate the property used by a rule or affected by a rule, depending on the direction of the arrow.

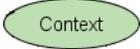
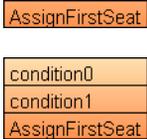
You can use various styles of layout. See [BusinessEvents Tools Layout Options Reference on page 476](#) to understand the uses of the different options.

Orphaned Entities

In some projects or services, you may notice some entities not connected by links. This is because the tools may not generate links for certain relationships. For example, the tools do not introspect custom Java functions.

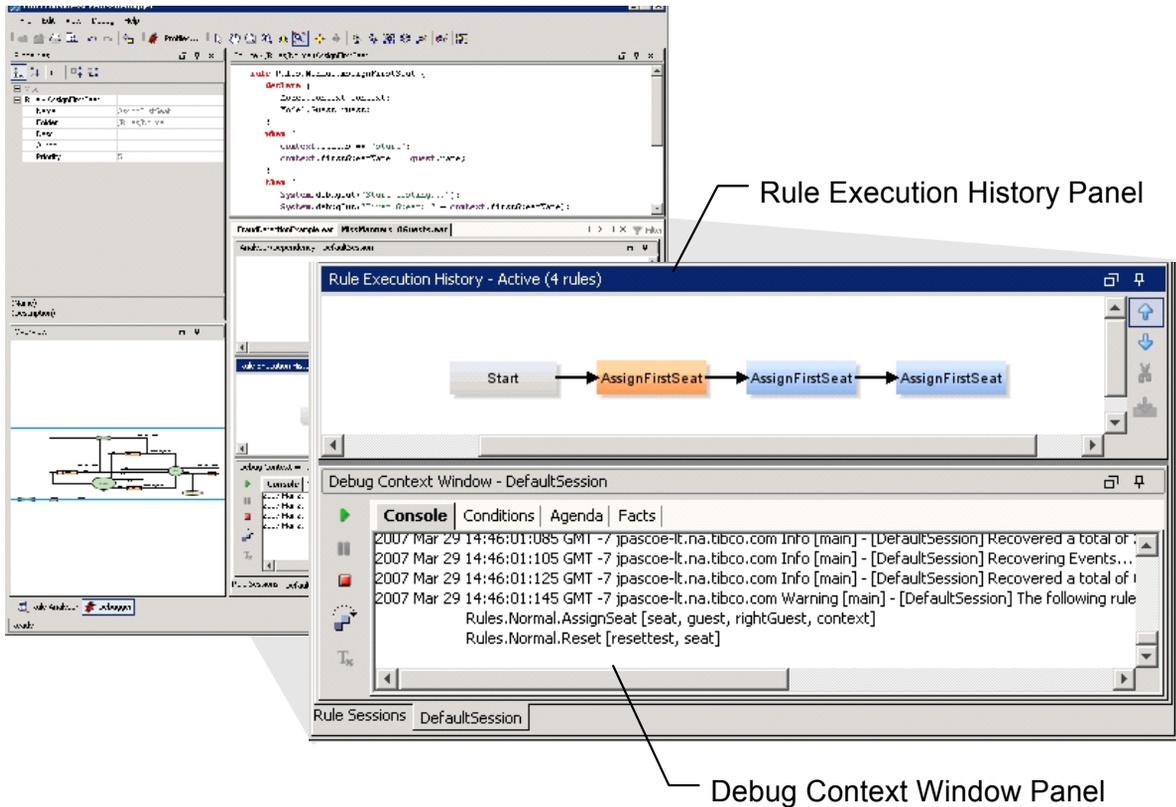
BusinessEvents Tools Graph Elements

Table 40 Explanation of Graph Elements in the Analyzer/Dependency Panel

Element	Explanation
	A simple event is shown as a yellow oval.
	A time event is shown as a green oval.
	A concept is shown as a green oval.
	A scorecard is shown as a green oval.
	A rule is shown in an orange rectangle. You can click on a rule to expand it, showing each condition of the rule. When the rule is expanded, you can see the links that relate to each condition.
	<p>Dependency or modification link. A <i>dependency</i> link is an arrow from an entity to a rule. It indicates that a property of the entity (or the entity itself) is used in the rule. The property name is shown on the line.</p> <p>A <i>modification</i> link is an arrow from a rule to an entity. It indicates that a rule action (when triggered) modifies a property of the entity (or the entity itself). The property name appears on the line. A modification link can also create an instance of an entity. In that case, <code>new()</code> appears on the line.</p>
	Concept reference link. The hollow diamond points to an entity that references (has a concept reference property for) the entity at the other end of the line. The concept reference property name appears on the line. This is a class diagram relationship. See Reference Relationships on page 86 .
	Contained concept link. The solid diamond points to an entity that contains the entity at the other end of the line. The name of the contained concept property appears on the line. This is a class diagram relationship. See Containment Relationships on page 85 .
	Inheritance link. The solid arrowhead points to an entity from which the entity at the other end of the line inherits properties. The words <code>Inherits from</code> appear on the line. This is a class diagram relationship. See Inheritance Relationships on page 84 .

BusinessEvents Tools—Rule Debugger User Interface Overview

Rule debugger has all the panels found in Rule Analyzer (shown in [BusinessEvents Tools—Rule Analyzer User Interface Overview on page 423](#)), plus two more: the Rule Execution History and Debug Context Window panel.



Step Mode

In step mode, the engine pauses just before and just after each rule fires. You click the Step  button to step to the next pause. Pausing before the rule fires lets you see how the rule conditions are matched by entities in working memory. Pausing after the rule fires lets you see the effect of the rule action or actions on the entities, and potentially on the rules in the agenda. The Rule Debugger panels show most information when you use step mode.

Rule Execution History Panel

When you use step mode, the Rule Execution History window shows you each rule in the rule agenda, in order of execution, for one conflict resolution cycle (see [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#)).

You can click a rule box icon to display information about the rule in the Properties and Source panels. A yellow border appears around the selected rule. In addition, a pop-up window of facts relating to the rule displays.

You can also review past rule execution cycles (using up and down arrows).

Rule Box Color The current rule is shown in an orange box. Rules that have executed appear in grey boxes to the left of the orange box. Rules in the rule agenda that may fire are shown in blue boxes to the right of the orange box (and also in the Agenda tab). See [Table 44, Rule Debugger Rule Execution History Panel Toolbar Options, on page 465](#) for details on the toolbar options.

Debug Context Window Panel

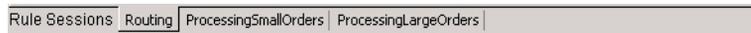
When you use step mode, Debug Context Window provides information about all rules that have fired and, in the case of the Agenda tab, are about to fire. This panel contains four tabs providing information about the current rule execution context, and a toolbar to control rule execution.



Step Mode and Run Mode When you use run mode, only the Console tab displays information, and only if the engine is local. It is only in step mode that the detailed information becomes available.

Services with Multiple Rule Sessions

The rule execution details for each rule session are shown separately. To switch from one rule session to another, click the tabs at the bottom of the Debug Context panel. Each tab shows the name of one rule session:



See [Table 45, Debug Context Panel Toolbar Buttons, on page 466](#) for details on the toolbar options.

Debug Context Window Panel Tabs

This section describes each of the tabs available in the Debug Context Window panel.

Console Tab

Shows information about the engine startup, rules being added, and so on. Displays messages printed to the console, as specified in particular rules, as they execute.



The information is similar to what you see when you start the BusinessEvents engine at the command line. This information is also saved in the engine log file (see [BE Working Directory \(and Log File Location\) on page 457](#) for information on location of the log files).

The first line shows the location of the property file used for this session. The property file for a debugger session is generated from the one specified in the debugger profile and is stored in the user's temp area. For example, on the Windows platform, you might see:

```
Using property file:
C:\DOCUME~1\jsmith\LOCALS~1\Temp\be-engine33146.tra
```

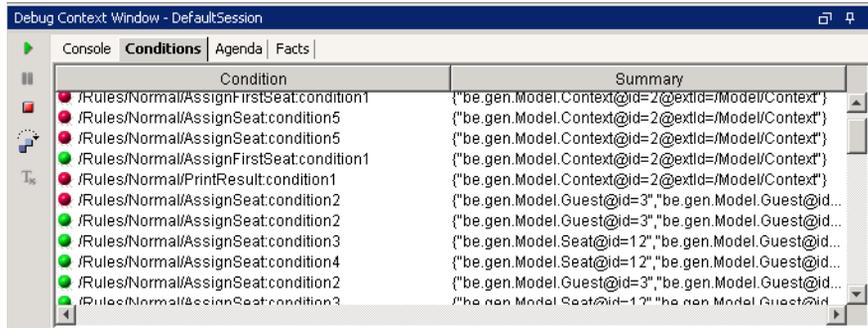


When you use run mode, only the Console tab is used. The other tabs in the Debug Context Window panel are used only in step mode.

The Console tab is not available when you are debugging against a remote engine.

Conditions Tab

Shows how the conditions for the rule that just fired were resolved. Click on a condition to view facts used to resolve the condition.



The most recent conditions appear at the top. Conditions with a red dot evaluated to false. Conditions with a green dot evaluated to true.

You can click a condition to expand it and see the facts that were associated with that condition at the time it was evaluated.

Agenda Tab

The Agenda tab shows the name of each of the rules in the agenda. You can expand the rules in the Rule column to see more information. Click a rule to see a single-line summary about each object affected by the rule actions. You can then expand the object nodes to see more details.



The rule at the top of the list corresponds to the rule directly to the right of the current rule (shown in the orange box). It is the next rule that will fire (unless a re-evaluation changes the agenda).

The Summary, Timestamp, and RTC Cycle columns are not currently used.

Facts Tab

The Facts tab shows facts used by the last rule that executed.



The same information appears in a pop-up window when you click on that rule's box in the Rule Execution History window. (The pop-up window shows facts relating to rules that fired earlier when you click on their boxes.)

Appendix E Working With Rule Analyzer

This chapter explains how to run Rule Analyzer and perform various procedures such as printing graphs.

See the following chapters for related topics:

- [Appendix D, BusinessEvents Tools Overview, page 419](#)
- [Appendix F, Working With Rule Debugger, page 449](#)
- [Appendix G, BusinessEvents Tools Reference, page 461](#)

Topics

- [Running Rule Analyzer, page 434](#)
- [Using Rule Analyzer, page 435](#)
- [Rule Analyzer Filtering and Viewing Options, page 437](#)
- [Finding Entities in a BusinessEvents Tools Graph, page 441](#)
- [Exporting a Graph as an Image, page 445](#)
- [Printing a Graph, page 447](#)

Running Rule Analyzer

You can run Rule Analyzer from the command line or from TIBCO Designer, when you are displaying a BusinessEvents project. You run Rule analyzer against the project repository, not the EAR file for the project.



Rule Analyzer and Rule Debugger are two utilities provided within the same application—BusinessEvents Tools. Once Rule Analyzer or Rule Debugger is running, you can switch between the two utilities and you can open multiple projects (Rule Analyzer) or services (Rule Debugger) as desired.

To Run Rule Analyzer From the Command Line

1. At a command prompt, execute the following:

```
BE_HOME/bin/be-tools
```

You see the Rule Analyzer user interface.

2. Click **File > Open** and select the project directory of the project whose rules you want to examine.

You see the graph of the project's ontology.

To Run Analyzer From TIBCO Designer

With your project open in TIBCO Designer, you can run Rule Analyzer on the project in any of the following ways:

- Click **Tools > BusinessEvents Tools > Rule Analyzer**
- Right-click on an EAR resource and select **Tools or Multiuser > BusinessEvents Tools > Rule Analyzer**
- Click **F11**.

See Also

[Using Rule Analyzer on page 435](#)

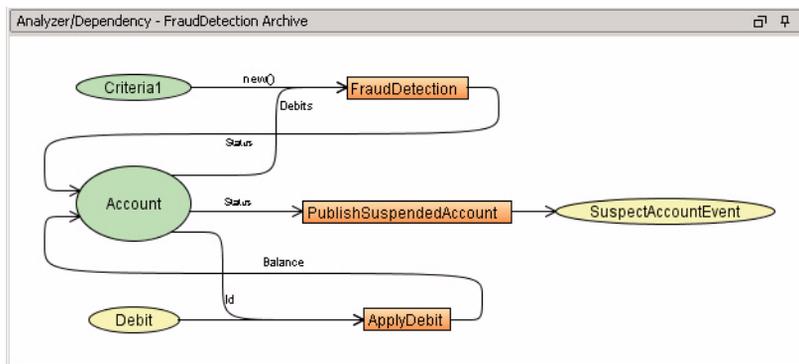
Using Rule Analyzer

Rule Analyzer is not a procedural tool. You use it to explore the structure of a project's ontology and to understand how rules and events interact to affect concepts and scorecards. This section contains a few ideas to get you started, and provides pointers to other sections for more information.



State timeouts do not appear in Rule Analyzer (BusinessEvents Enterprise Suite only).

Using the Analyzer/Dependency Graph



The graph shown in the Analyzer/Dependency panel helps you understand the relationships between entities in the ontology, and the rules that affect and are affected by them. See [Table 40, Explanation of Graph Elements in the Analyzer/Dependency Panel, on page 427](#) for more details on how items are represented in a graph.

If you are new to BusinessEvents, read the following analysis to learn how the graph exposes various relationships. For example the graph can help you understand:

How Events Affect Rules Any rule whose declaration contains an event that is asserted into the working memory is evaluated. For example, the arrow from the `Debit` event (yellow oval) to the `ApplyDebit` rule (orange rectangle) indicates that the `ApplyDebit` rule is evaluated when a `Debit` event is asserted into working memory.

How Rules Affect Events Events can also be created internally by rule actions. This action can then trigger other rules. For example, a `Suspend` event is created by an action in the `PublishSuspendedAccount` rule. In a real-world application, the `Suspend` event would perhaps change or create a concept, causing another rule to be evaluated, and so on.

How Concepts and Scorecards Affect Rules If a rule's declaration references a concept, then when an instance of that concept is created or changed, the rule will be evaluated or re-evaluated. Concept creation is indicated by the text `New()` on the link between the rule and the concept. It is not necessary to put scorecards in the declaration because there is only one instance of each scorecard in an application. Any change causes all rules that use the scorecard in their conditions to be evaluated.

How Rules Affect Concepts and Scorecards Rules can create, modify and delete concept instances. Rules can modify scorecards. The `FraudDetection` rule (shown in the illustration) has a link to the `Account` concept, and the word `Status` appears on the line. This means that one action of the `FraudDetection` rule action is to change the `Status` property of an `Account` instance, when rule conditions are met.

Using the Properties and Source Panels

You can click each shape in the Analyzer/Dependency graph to see related information about it.

- Click a concept, event, rule, or scorecard to see its properties in the Properties panel. You can see user-defined and system properties.
- Click a rule to see its code in the Source panel. the code continues to display until you click a different rule.

Examining the graph, together with this textual information helps you to understand how the application is structured and how it reacts to events and changes in the entities.

Using Find, Filter, Export, and Printing Features

The following procedures explain how to do certain Rule Analyzer tasks:

- [Rule Analyzer Filtering and Viewing Options on page 437](#). You can use these options to view or hide class diagrams, for example.
- [Finding Entities in a BusinessEvents Tools Graph on page 441](#)
- [Exporting a Graph as an Image on page 445](#)
- [Printing a Graph on page 447](#)

Rule Analyzer Filtering and Viewing Options



The filtering and viewing options described in this section are available in Rule Analyzer only (not in Rule Debugger).

You can use the filtering and viewing options for various purposes such as:

- To focus on one aspect of the ontology such as a set of related entities, using entity filtering.
- To add state machine transitions into the graph (does not support entity filtering).
- To view the class diagram relationships only.
- To view the graph with all relationships except class diagram relationships.

Filtering Entities in a Graph

This procedure relates to use of the Include Entities field in the Specify Filter Parameters dialog. In addition, you can display:

- The filtered or unfiltered graph with or without class diagram relationships.
- *Only* the class diagram relationships for the filtered or unfiltered graph.

The Analyze State Machine option does not support filtering.

To Filter Entities in a Graph

1. From the Rule Analyzer menu, select **View > Filter**.

You see the Specify Filter Parameters dialog (see [Specify Filter Parameters Reference on page 439](#) for details).

2. Do one of the following:
 - If you know the project path and name of the entities you want to filter, type them in the Include Entities field. (The project path is the path to the item in the design-time project).
 - If you don't know the project path and name of the entities, click the Filter  button to the right of the Include Entities field. You see the Entity Chooser pop-up. Select entities from the project tree as desired. You can also use text options to narrow the choice. See [Using the Entity Chooser on page 442](#) for more details.

3. Click **Filter**. The graph displays again, showing the entities you specified. If no entities match, a blank graph window displays.



To clear filter and viewing options, display the Specify Filter Parameters dialog, clear all entries, and click **Filter**.

Using Class Diagram and State Machine Viewing Options

To Analyze State Machine Information

Instead of viewing rule execution information, you can view state machine information (Enterprise Suite only). This option can't be combined with entity filtering.

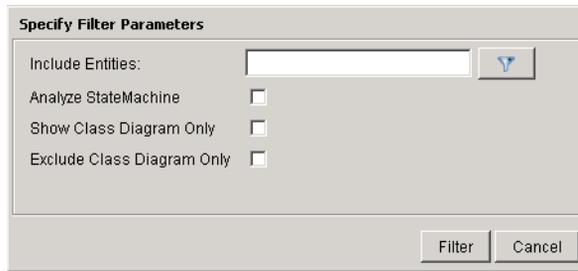
1. From the Rule Analyzer menu, select **View > Filter**.
You see the Specify Filter Parameters dialog.
2. Check the Analyze State Machine checkbox.
3. Click **Filter**.

To View or Hide Class Diagram Information

By default, class diagram information is shown in addition to the rule execution information. You can filter entities as well as set class diagram options.

1. From the Rule Analyzer menu, select **View > Filter**.
You see the Specify Filter Parameters dialog.
2. Do one of the following:
 - To show only the class diagram and hide the rule execution information, check the Show Class Diagram Only checkbox.
 - To show rule execution information and hide class diagram information, check the Exclude Class Diagram Only check box.
3. Click **Filter**.

Specify Filter Parameters Reference



Include Entities

You can determine what entities (concepts, scorecards, events, and rules) to graph by specifying them here. All entities except the selected entities are hidden when you click the **Filter** button. Viewing a small set of entities can help you understand the structure of complex projects.

You can directly type a comma-separated list of entities into the field, using the complete project path, beginning with a forward slash. The following example specifies two entities, a concept and an event:

```
/Concepts/Account , /Events/SuspectAccountEvent
```

For large projects, it is easier to specify entities using the Entity Filter dialog. To access this dialog, click the filter button . See [Entity Chooser Dialog on page 440](#). Entities selected using the filter appear in the Include Entities field.

Analyze State Machine

Choose this option to display the project's state machine. (Enterprise Suite only). See [Chapter 10, Working With The State Modeler, on page 163](#) for more information on state machines.



This option works only on unfiltered graphs. If you specify an entity filter, then state machine information does not display.

Show Class Diagram Only

By default, graphs show not only class diagram information, but also the rules and their relations with the members of the class diagram (which are concepts, scorecards, and events). Select this option to show the class diagram information and hide all other information, such as rules.

Exclude Class Diagram Only

Select this option to hide class diagram information (see the section [Show Class Diagram Only](#) for more details).

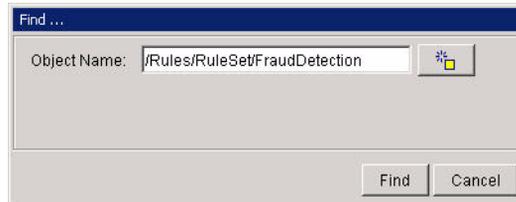
Entity Chooser Dialog

In the Include Entities field, click the filter button  to use the Entity Chooser dialog to select a list of entities to display.

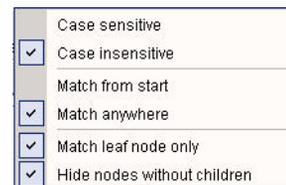
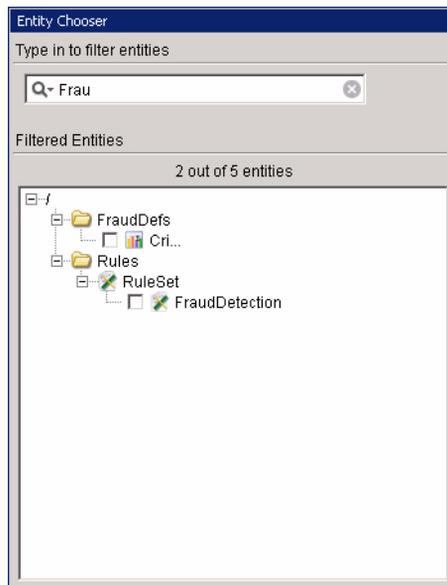
This dialog is the same one used in the Find feature. See [Using the Entity Chooser on page 442](#).

Finding Entities in a BusinessEvents Tools Graph

BusinessEvents projects often generate large, complex graphs. The Find feature enables you to locate a single entity (object) when you are using Rule Analyzer or Rule Debugger. You can use simple find if you know the object name and project path:



If you don't know the project path or complete entity name, you can use the Entity Chooser and provide whatever information you can. The chooser finds entities that contain the string you provide. To refine the search conditions, click the magnifying glass icon and choose options (shown on the right in the figure below).



To Find an Entity in a Graph



Find locates only entities currently displayed in the graph. If you have filtered the entities in the graph (Rule Analyzer only), then Find locates only entities that are shown within the filtered display. To find an entity in the unfiltered graph, first clear the filter, then find the entity.

1. Do one of the following:
 - From the Rule Analyzer menu, select **Edit > Find**.
 - From the Rule analyzer toolbar, click the Find  button.
 - Click **Ctrl-F**.

You see the Find dialog.
2. Do one of the following:
 - If you know the project path and name of the entity you want to find, type it in the Object Name field and click **Find**.
 - If you don't know the project path and name of the entity you want to find, click the Find  button to the right of the Object Name field. You see the Entity Chooser pop-up. Select an entity from the project tree.

You can use the text options to narrow the choice. See [Using the Entity Chooser on page 442](#) for more details.
3. Click **Find**. The graph displays again, showing the entity you specified. The entity is selected. If no matching entity is found, you see the message: No Objects Found.

Using the Entity Chooser

The Entity Chooser is available to help you locate entities while you are using the Find and Filter features.

To Choose Entities Using Text (Type-Ahead)

For simple projects, it is easy to choose entities using the project tree: just select a checkbox (or multiple checkboxes if you are using Entity Chooser while filtering).

However, for complex projects you might find the type-ahead feature and the drop-down text options helpful. They narrow the search. Only matching entities display in the project tree so they are easier to find and select.

1. To learn how to access the Entity Chooser dialog, see these procedures:

[To Filter Entities in a Graph on page 437](#)

[To Find an Entity in a Graph on page 442](#)

2. As desired, click the spyglass icon to reveal a drop-down list of matching options. Select one or more matching options to define how text strings are used. See [Entity Matching Options on page 443](#) for details.
3. In the field below the text, "Type in to filter entities," enter a text string according to the text matching options you chose. For example, if you choose the "Match from start" option, you must enter a project path beginning with the forward slash.

The project tree displays only matching entities.

To remove all text from the field, click the X that displays towards the right of the field.

4. Select one entity from the tree if you are using the Find feature. Select one or more entities from the tree if you are using the Filter feature.

All selected entities' project paths appear in the Find dialog's Object Name field, or in the Filter dialog's Include Entities field.

Entity Matching Options

Specifying a matching option affects how the text you enter in the Type in to Include Entities field is used to match entities and folders in the project tree. You can use multiple options, for example, Case Insensitive and Match Anywhere.

Option	Description
Case Sensitive	Select Case Sensitive for a case sensitive match (for example, Dog matches Dog but does not match dog).
Case Insensitive	Select Case Insensitive for a case insensitive match (for example, Dog matches Dog and dog). Default: Case Insensitive.
	Note: This setting also affects text you type directly into the Object Name (Find dialog) or Include Entities (Filter dialog) fields.

Option	Description
Match from Start	Select Match from Start if you want to narrow your search by project path.
Match Anywhere	<p>Specify the project path (beginning with forward slash) to show all matching entities in the specified path.</p> <p>Select Match Anywhere to narrow your search using a text string that is part of a folder or entity name. Enter the text to show all folders and entities that contain the text you type.</p> <p>Default: Match Anywhere.</p>
Match Leaf Node Only	<p>Leaf nodes are nodes that have no child nodes. This option filters out any matches not located in a leaf node.</p> <p>Selected by default.</p>
Hide Nodes without Children	<p>Nodes that don't contain child objects are not useful for most searches.</p> <p>Selected by default.</p>

Exporting a Graph as an Image

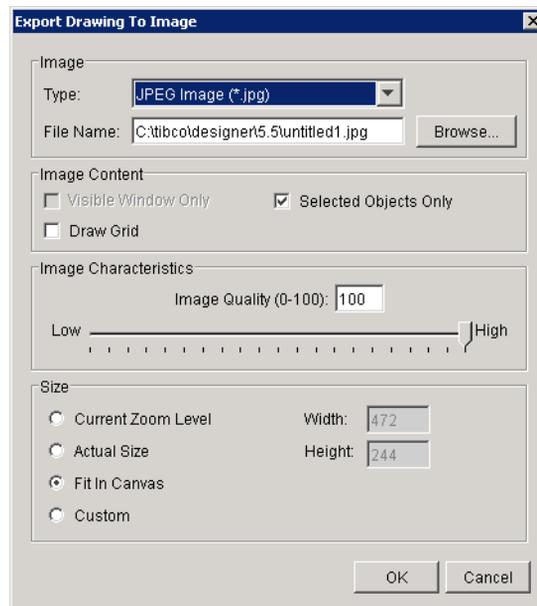
This feature is available in Rule Analyzer and Rule Debugger. You can export the graph displayed in the Analyzer panel (or part of it) to an image file.

(The Analyzer panel is the name of the panel in Rule Analyzer where the graph displays. It has no name Rule Debugger.)

You can then use the image file, for example, to communicate with colleagues about project issues, or in project documentation.

To Export a Graph as an Image

1. With the graph displayed as desired, select **File > Export As**. You see the Export Drawing To Image dialog:



2. In the Type field, select a file format:
 - JPEG image (*.jpg)
 - Portable Document Format (*.pdf)
 - Portable Network Graphics (*.png)
3. In the File Name field, browse to the destination directory and provide a name for the image.

4. In the Image Content area, make selections as appropriate. Different options are available depending on the state of the drawing in Analyzer panel. Unavailable options are grayed-out.
 - Visible Window Only—If only part of the model diagram is currently displayed in the Analyzer panel, export only the visible part.

This option is available only when you also choose Current Zoom Level in the Size area.
 - Selected Objects Only—If you have selected some objects in the Analyzer panel, check the checkbox to export only the selected objects.
 - Draw Grid—If you have displayed a grid in the Analyzer panel, check the checkbox to include the grid in the exported image.
5. In the Image Characteristics area, select the desired image quality. If 100 (best quality) results in very large files, experiment with lower quality to get adequate quality at a smaller file size. For example, at 50, you may find the color in the JPEG files is not smooth, but at a higher value it becomes smooth.
6. In the Size area select the option desired:
 - Current Zoom Level—The zoom level currently used in the Analyzer panel.

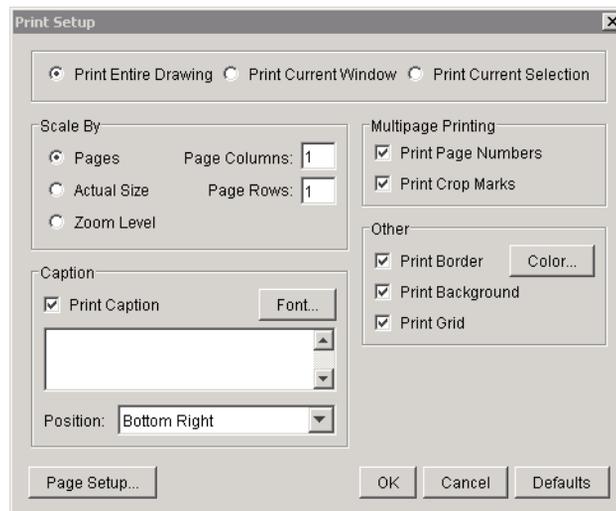
When you select this option, the Visible Window Only option in the Image Content area becomes available.
 - Actual Size—The zoom level set internally by the software.
 - Fit in Canvas—The zoom level necessary to fit the whole graph in the current panel size.
 - Custom—Select Custom then set the width or height in pixels. The proportions are maintained, so when you set one measurement, the other adjusts accordingly. Exports the graph (or partial graph selected in the Image Content area) to an image of these dimensions.
7. Click **OK**.

Printing a Graph

You can print all or some of a graph, and print large graphs on multiple pages.

To Print a Graph

1. As needed, prepare the graph to suit the print options you want to use. If you want to print part of a graph, or print at the current zoom level, display the graph as you want to print it. If you want to print selected objects only, select them.
2. From the menu select **File > Page Setup**. You see the Print Setup dialog:



3. Select a print option:
 - Print Entire Drawing.
 - Print Current Window—Prints the part of the graph that is currently displayed in the Analyzer panel.
 - Print Current Selection—Prints only selected objects.
4. Select a scaling option. Scale by:
 - Pages—Prints to a single page or to multiple pages that you can tile to show the whole printout. In the Page Columns field, enter a number of pages to

define the width of the tiled printout. In the Page Rows field, enter a number of pages to define the height of the tiled printout.

- Actual Size—The zoom level set internally by the software.
 - Zoom Level—Prints at the graph's current zoom level.
5. Set other options as desired to include page numbers, crop marks, captions, a border (of a selected color), and a grid.
 6. Click **Page Setup**. At the Page Setup dialog, select paper size, page orientation and other options as desired. You can also click Printer and define printer settings.
 7. Click **OK** to close each setup dialog in turn.
 8. From the menu select **File > Print**. At the Print dialog, select a printer and other options as desired, and click **OK**. The graph prints to the selected printer.

Appendix F Working With Rule Debugger

This chapter explains how to setup and run Rule Debugger and perform supporting procedures such as configuring profiles.

See the following chapters for related topics:

- [Appendix D, BusinessEvents Tools Overview, page 419](#)
- [Appendix E, Working With Rule Analyzer, page 433](#)
- [Appendix G, BusinessEvents Tools Reference, page 461](#)

The following procedures in [Appendix E, Working With Rule Analyzer](#) also apply to Rule Debugger:

- [Finding Entities in a BusinessEvents Tools Graph, page 441](#)
- [Exporting a Graph as an Image, page 445](#)
- [Printing a Graph, page 447](#)

Topics

- [Rule Debugger Setup Overview, page 450](#)
- [Running Rule Debugger, page 451](#)
- [Configuring Rule Debugger Profiles, page 454](#)
- [Rule Debugger Profiles Settings, page 456](#)
- [Configuring a BusinessEvents Engine for Remote Debugging, page 460](#)

Rule Debugger Setup Overview

You can run Rule Debugger without additional setup when you invoke it from TIBCO Designer while a BusinessEvents project is displayed. However for other uses, some additional setup is required. This section summarizes the setup and points to other sections for more details.

Creating and Managing Profiles

When you run Rule Debugger at the command line, you must select a profile after the user interface displays. Each debugger profile specifies an EAR file that contains a BusinessEvents application configuration, as well as some other information about property files and startup options. See [Configuring Rule Debugger Profiles on page 454](#) for more details.

Running Rule Debugger Against a Remote Engine

You can run rule debugger against a BusinessEvents application running on a remote BusinessEvents engine. Note the following differences when you run against a remote engine:

Remote Engine Configuration Before running debugger remotely you must ensure that the remote engine's TRA file is configured to enable remote debugging. See [Configuring a BusinessEvents Engine for Remote Debugging on page 460](#).

EAR File and TRA File Location If the remote application's EAR file and TRA file are not accessible to the debugger, copy them to the local machine (or to a shared network drive) and specify that location in the debugger profile. For accurate debugging, do not change the copied files or filenames.

Console Tab is Unavailable When you use the debugger on a remote BusinessEvents engine, the debugger console tab (which shows the engine's console output) is not available.

Providing Data to the Application

You may have to provide sample messages to trigger rules to fire. In *TIBCO BusinessEvents Getting Started*, the first tutorial explains how to using ActiveMatrix BusinessWorks activities and the ActiveMatrix BusinessWorks tester feature to provide input to a BusinessEvents application.

Running Rule Debugger

You can run Rule Debugger from the command line. You can also run it from TIBCO Designer, when a BusinessEvents project is open.



Rule Debugger does not run on the AIX platform.

In both cases, once Rule Debugger is displayed, you can use the Launch Rule Debugger button or select Debug > Launch to start a new Rule Debugger session with the same or a different configuration profile. You can then switch between the application tabs to work with each application.



This section explains the basic procedure for running Rule Debugger. Before you can use Rule Debugger, however, you must configure a profile specifying details of the application you want to debug. The procedures in this section refer to the procedure for setting up profiles, where you can find more detailed information.

Before you can run Rule Debugger against a remote BusinessEvents engine, you must configure that engine's TRA file to enable remote access. See [Configuring a BusinessEvents Engine for Remote Debugging on page 460](#).

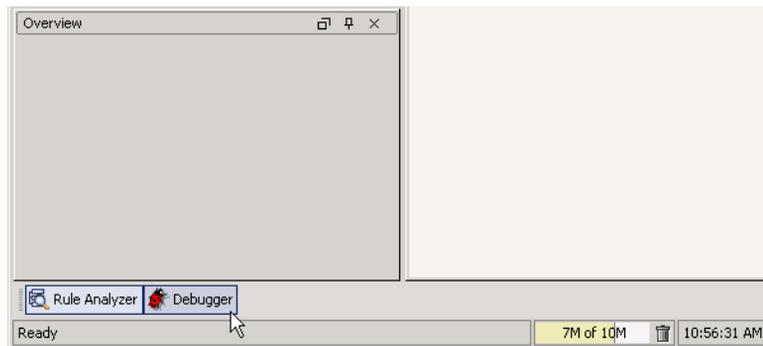
To Run Rule Debugger From the Command Line

1. At a command prompt, execute the following:

```
BE_HOME/bin/be-tools
```

You see the Rule Analyzer user interface, which displays by default.

2. Click the **Debugger** button (at the bottom left of the user interface).



The Rule Debugger and Profiles options are now available from the menu.

The next step is to figure one or more profiles. Profiles provide information about the application you want to debug. To begin debugging, you will launch a Rule Debugger session with a profile. See [Configuring Rule Debugger Profiles on page 454](#).

If your profile is ready to use, see [To Start a Rule Debugger Session—After Configuring Profiles on page 453](#) for the next step.

To Run Rule Debugger From TIBCO Designer



In the context of TIBCO Designer, you can always run Rule Debugger when you select the project's EAR resource. If the focus is elsewhere in the project, you can run Rule Debugger only when there are no changes to the BusinessEvents project since the last time the EAR file was built. If the tool is unavailable from the menu, rebuild the EAR file to make it available or select the EAR file resource.

With the desired project open in TIBCO Designer, do the following.

1. Navigate to and select the project EAR file (its icon is the blue box .
2. Start a Rule Debugger session in any of the following ways:
 - Click **Tools > BusinessEvents Tools** and select **> Rule Debugger**
 - Right-click on an EAR resource and select **Tools or Multiuser > BusinessEvents Tools** and select **> Rule Debugger**
 - Press **F12**

You see the Rule Debugger user interface, displaying the application. The console window shows activity in the engine.

If you want to run additional sessions from within Rule Debugger, you must configure one or more profiles. See [Configuring Rule Debugger Profiles on page 454](#).

If your profile is ready to use, see [To Start a Rule Debugger Session—After Configuring Profiles on page 453](#) for the next step.

To Start a Rule Debugger Session—After Configuring Profiles

You can launch Rule Debugger from a BusinessEvents application open in TIBCO Designer without specifying a profile. See [To Run Rule Debugger From TIBCO Designer on page 452](#) for details.

But for all other cases, you must start a Rule Debugger session using a profile that specifies details about the application you want to debug. For details on adding and modifying profiles, see [Configuring Rule Debugger Profiles on page 454](#).

If the profile you want to use has been configured, then follow this procedure after you have started Rule Debugger. (See [To Run Rule Debugger From the Command Line on page 451](#) and [To Run Rule Debugger From TIBCO Designer on page 452](#) for startup details.)

1. Do one of the following to start a Rule Debugger session.
 - From the Rule Debugger menu, select **Debug > Launch**
 - Click the Launch Rule Debugger  button in the top toolbar:



You see the Launch Debugger with Configurations dialog.

2. Select a profile from the list on the left and do one of the following:
 - Click **Run** to run the debugger without pausing between execution of each rule.
 - Click **Step** to run the debugger in step mode, pausing between execution of each rule.

You see the Rule Debugger user interface, displaying the application. The console window shows activity in the engine.

Note that you can launch and run multiple debugger sessions in one BusinessEvents Tools instance. Each session is displayed on a separate tab.

Configuring Rule Debugger Profiles

When you start a Rule Debugger session, you must specify a profile that provides information about the application you want to work with and the BusinessEvents engine that runs the application. You can create multiple profiles for the same application. For example, you may want to test the effect of different startup arguments.



When you start a Rule Debugger session from the context of a project open in TIBCO Designer you don't have to create a profile (see [To Run Rule Debugger From TIBCO Designer on page 452](#)).

To Configure Rule Debugger Profiles

This procedure assumes you are viewing the Rule Debugger user interface. See [Running Rule Debugger on page 451](#) for instructions on accessing the Rule Debugger user interface.

1. Access the Edit Debug Configurations dialog in one of the following ways:
 - Click the **Profiles** button in the top toolbar:



Or select **Debug > Profiles** from the menu.

- Click the Launch Rule Debugger  button in the top toolbar:



Or select **Debug > Launch** from the menu.

When you use the Profiles option, you can work with profiles but cannot start a debugger session. When you use the Launch option, you can work with profiles, and you can also start a debugger session using a selected profile.

2. To create or edit a profile, do one of the following:

- **To create a new profile based on defaults**, click the **New Profile** button .



Click **Edit Defaults** to change defaults used when creating new profiles. See [Edit Default Configuration Properties Dialog on page 459](#) for details.

- **To create a new profile based on the currently selected profile**, click the **Copy Profile** button .

- **To edit an existing profile**, select it from the list on the left and modify its values in the fields on the right.

See [Rule Debugger Profiles Settings on page 456](#) for details on completing the settings to enable local or remote debugging.

3. To organize the list of profiles, you can do the following:

- **To delete a profile**, select the profile name in the left panel and click the **Delete** button . Delete profiles you don't need any more to keep the list manageable.

- **To move a profile up or down in the list**, select the profile name in the left panel and click the up and down arrows. You can put frequently-used profiles near the top, or organize profiles by project, and so on.

4. As appropriate, choose an option to save, apply, or cancel all changes made in all profiles since you opened the Edit Default Configuration Properties dialog:

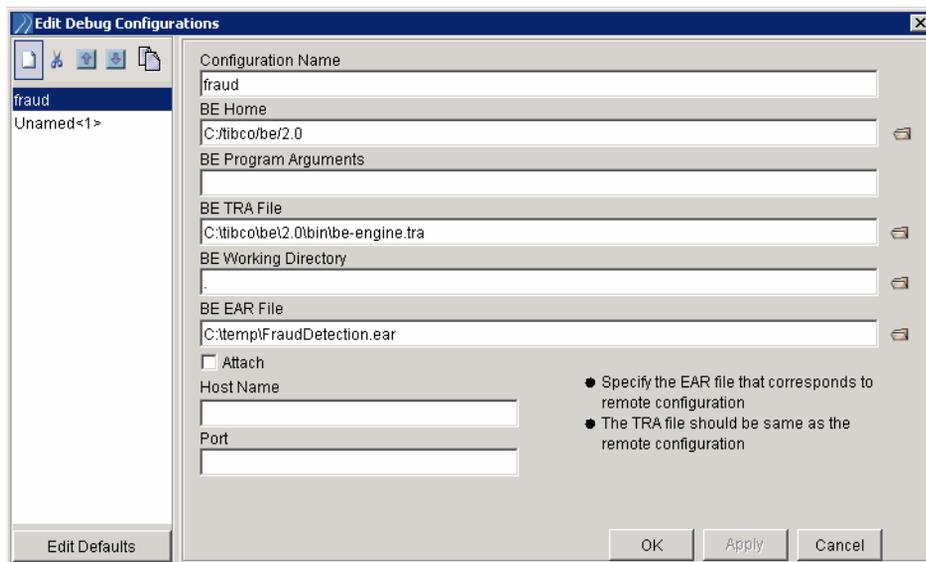
- **To apply changes to a profile in memory**, click **Apply**. You can then select a different profile to work with.
- **To save changes, additions, and deletions to disk**, click **Apply** (to save the last changes to memory), then click **OK** (to save the work to disk).
- **To cancel changes, additions, and deletions**, click **Cancel**. None of the work done since you opened the Edit Default Configuration Properties dialog (or last clicked OK) is saved to disk.



Note that clicking the Apply button does not save changes, additions, and deletions to disk. Click Apply then click OK to save these actions to disk.

Rule Debugger Profiles Settings

This section explains how to configure profile settings for local or remote debugging. See [Running Rule Debugger on page 451](#) and [Configuring Rule Debugger Profiles on page 454](#) for related procedures.



A profile specifies the EAR file whose rules you want to debug, and other supporting information. You can configure and save multiple profiles to point to different EAR files, or to the same EAR file but with a different configuration.



Remote Debugging To enable debugging against a remote BusinessEvents engine, you must specify its host name and Java Debug Interface (JDI) port number. All sections below apply to both local and remote debugging, unless specific information for remote debugging is provided in the text below. Configuration is also required. See [Configuring a BusinessEvents Engine for Remote Debugging on page 460](#).

Configuration Name

Provide a descriptive name for the profile.

BE Home

Specify the directory in which BusinessEvents is installed, for example, `C:\tibco\be\3.0`.

Remote Debugging This setting is not used for remote debugging.

BE Program Arguments

Provide options and parameters as you would if you were starting the BusinessEvents engine at the command-line.

Remote Debugging This setting is not used for remote debugging.

BE TRA File

Identify the path and name of the property file (`be-engine.tra`) used when starting the engine, for example:

```
c:/tibco/be/3.0/bin/be-engine.tra
```

Remote Debugging Specify the location of the property file used to start the remote engine if it is available to the debugger session using the file system. If it is not available on the file system, specify the location of a copy of the original file that you save to an accessible location. Do not change the copy in any way.



When you manage the engine for your BusinessEvents application using TIBCO Administrator, then by default the property file used to start the engine is located here:

```
TIBCO_HOME/tra/domain/domain-name/application/app-name/app-name.tra
```

Note that a different domain home can be specified when a domain is created.

BE Working Directory (and Log File Location)

The location of the working directory for the BusinessEvents engine. The default default value is a dot (`.`), which means that the working directory is the directory where the BusinessEvents engine is started. The `be-tools` and `be-engine` executables are normally started in the `TIBCO_HOME\be\3.0\bin` directory.



Working Directory and Log File Location If you use the dot value, be aware of the following difference in the location of the working directory, and therefore `engine-name.log` file:

- When you start BusinessEvents tools from the command line (and in its default location in the BusinessEvents `bin` directory), the working directory will be `BE_HOME\bin\logs`.
- When you start BusinessEvents Tools from TIBCO Designer, the working directory will be `TIBCO Designer_HOME\bin\logs`.

Also see [Default Location of Log Files and Other Files and Directories on page 389](#).

Path names that do not start with the root directory are assumed by the operating system to start from the working directory.

Remote Debugging This setting is not used for remote debugging.

BE EAR File

Location of the BusinessEvents EAR file of the project you want to debug.

Remote Debugging Specify the location of the EAR file of the project you want to debug if it is available to the debugger session using the file system. If it is not available on the file system, specify the location of a copy of the original file that you save to an accessible location. Do not change the copy in any way.

Attach

This setting is used only for remote debugging.

Remote Debugging Check this checkbox if you are configuring a profile for remote debugging.

Host Name

This setting is used only for remote debugging.

Remote Debugging Provide the name of the machine on which the remote BusinessEvents engine runs.

Port

This setting is used only for remote debugging.

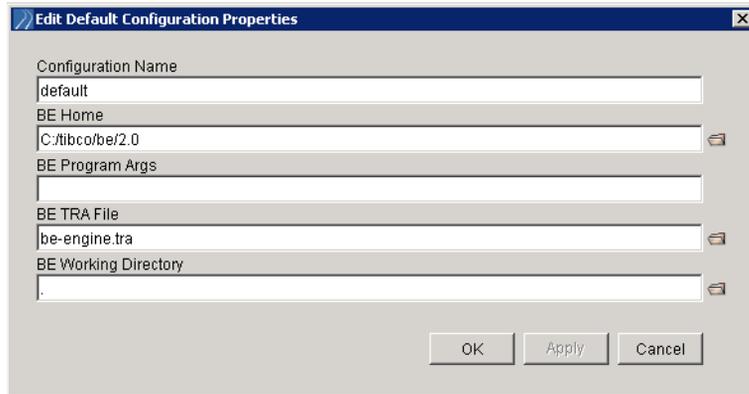
Remote Debugging Provide the JDI port number, as specified in the remote engine's property file. See [Configuring a BusinessEvents Engine for Remote Debugging on page 460](#).

Edit Defaults

Click to display the Default Configuration Properties dialog. See [Edit Default Configuration Properties Dialog](#), next.

Edit Default Configuration Properties Dialog

Complete the values you want to use as default properties when you click the New Profile button  in the Edit Debug Configurations dialog. See above for details on the meaning of each setting.



Profiles XML File

Profile settings are saved in XML format in the `be-tools.debugger.config` file, stored in the user's `.TIBCO` directory. For example, for a Windows user called `jsmith`, the location would be `C:\Documents and Settings\jsmith\.TIBCO`.

Configuring a BusinessEvents Engine for Remote Debugging

You must configure the BusinessEvents engine property file so the engine uses Java Debug Interface (JDI) for remote debugging.

To Configure Java Debug Interface (JDI)

For each BusinessEvents engine you want to enable for remote debugging, do the following.

1. Open the `BE_HOME/bin/be-engine.tra` file for editing.
2. Specify the port on which you want the engine to listen, using the environment variable `tibco.env.JDI_PORT`, for example:

```
tibco.env.JDI_PORT 5192
```

Where 5192 is the default value. If multiple engines run on the same machine, ensure that each has a unique port.

3. Uncomment the following line:
`-Xrunjdpw:transport=dt_socket,address=%JDI_PORT%,suspend=na,server=y`
4. Start or restart the engine.

You specify the same JDI port number in the profile you will use to connect to the remote engine (see [Port on page 458](#)).

Appendix G **BusinessEvents Tools Reference**

This chapter provides a reference to the BusinessEvents Tools application user interface toolbars, menus, preferences, and layout options.

See the following chapters for related topics:

- [Appendix D, BusinessEvents Tools Overview, page 419](#)
- [Appendix E, Working With Rule Analyzer, page 433](#)
- [Appendix F, Working With Rule Debugger, page 449](#)

Topics

- [BusinessEvents Tools Toolbar Reference, page 462](#)
- [BusinessEvents Tools Menu Reference, page 467](#)
- [BusinessEvents Tools Preferences Reference, page 472](#)
- [BusinessEvents Tools Layout Options Reference, page 476](#)

BusinessEvents Tools Toolbar Reference

Tables in this section describe options in the main toolbar at the top of the user interface, options in the bottom toolbar, and options in the toolbars located in the Properties, Rule Execution History, and Debug Context Window panels.

Main Toolbar Options

Table 41 BusinessEvents Tools—Main Toolbar Options

Button	Description
General Tools	
	Open. Rule Analyzer only. You can open one or more projects in Rule Analyzer. Enables you to browse to a project folder to select a project. All projects open are available in the Analyzer panel, using tabs. The project name is used as the tab label.
	Reload. Rule Analyzer only. If you change a project in TIBCO Designer and save the changes, clicking Reload reflects these changes in Rule Analyzer.
	Print. Prints the graph. See Printing a Graph on page 447 .
	Print Preview. Lets you see how the printed page would look.
	Undo. Undoes changes one per mouse click, most recent first. You can set an undo limit. See BusinessEvents Tools Preferences Reference on page 472 .
	Redo. Redoes changes, one per mouse click, in reverse order (that is, last change first).
	Find. Tool for finding entities in a large graph. See Finding Entities in a BusinessEvents Tools Graph on page 441 .
	Launch. Rule Debugger only. Starts a Rule Debugger session using a profile. See Running Rule Debugger on page 451 .
	Edit profiles. Rule Debugger Only. Opens the Edit Configurations dialog. See Configuring Rule Debugger Profiles on page 454 .

Table 41 BusinessEvents Tools—Main Toolbar Options (Cont'd)

Button	Description
Cursor Tools	
	Select (Pointer) tool.
	Pan tool.
Zoom Tools	
	Marquee Zoom Drag the cursor diagonally to define the area to which you want to zoom.
	Interactive Zoom Drag the cursor up to zoom out. Drag the cursor down to zoom in.
	Link Navigator tool Shows the path of a link using a simple animation.
	Fit In Window Zooms the graph to fit the size of the current Analyzer/Dependency panel.
	Expand All Rule Nodes Expands rule nodes to show rule conditions and their related links in the graph.
	Collapse All Rule Nodes Hides all rule conditions Rule nodes (boxes) show only the rule name. All links enter and leave the rule without indicating specific rule conditions.
Layout Tools. See BusinessEvents Tools Layout Options Reference on page 476 for details.	
	Hierarchical Layout with Orthogonal Routing See Hierarchical Layout on page 476 .
	Hierarchical Layout with Normal Routing See Hierarchical Layout on page 476 .
	Orthogonal Layout See Orthogonal Layout on page 477
	Circular Layout See Circular Layout on page 477 .
	Symmetric Layout See Symmetric Layout on page 477 .
	Incremental Layout See Incremental Layout on page 476 .

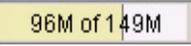
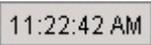
Table 41 BusinessEvents Tools—Main Toolbar Options (Cont'd)

Button	Description
	<p>Link Routing Redraws the graph, attempting to redraw only the links, leaving entity nodes in the same size and position where possible.</p> <p>The behavior of link routing is affected by the preference options Fix Node Size and Fix Node Position. See BusinessEvents Tools Preferences Reference on page 472 for details.</p>

Lower Toolbar Options

The lower part of the user interface contains the following buttons and informational icons

Table 42 BusinessEvents Tools—Bottom Toolbar Options

Bottom Toolbar	
 Rule Analyzer	<p>Rule Analyzer When viewing the Rule Debugger interface, click to view the Rule Analyzer interface.</p>
 Debugger	<p>Rule Debugger When viewing the Rule Analyzer interface, click to view the Rule Debugger interface.</p>
 96M of 149M	<p>Memory Usage Shows the total memory allocation (heap size) and the amount of memory currently consumed by all projects and services open in this BusinessEvents tools instance. The yellow portion of the bar indicates how much of the total available memory is consumed.</p>
	<p>Force GC Click to run the Java garbage collector for the BusinessEvents Tools application. Doing so frees up memory.</p>
 11:22:42 AM	<p>Time The clock in the lower right corner displays the current time, using a 12-hour format.</p>

Properties Panel Toolbar Options

The properties panel provides the following toolbar options:

Table 43 *BusinessEvents Tools Properties Panel Toolbar Options*

Button	Description
 and 	Categorized and Alphabetical sort options. This feature is not used.
	Show or Hide Description The description area is the gray area below the properties list. When you click a property, the property name and description (if any) display in the description area.
 and 	Expand all and Collapse all Toggle between these two options as desired to open and close the property trees.

Rule Execution History Panel Toolbar Options

During a Rule Debugger session, you can view past conflict resolution cycles as well as the current one. (See [Understanding Conflict Resolution and Run to Completion Cycles on page 130](#) for more information on conflict resolution.)

Table 44 *Rule Debugger Rule Execution History Panel Toolbar Options*

Button	Description
	Displays the previous conflict resolution cycle (CRC). If you click Previous while the first cycle in the session is displayed, then the last cycle displays and you can continue to click back through the cycles again.
	Displays the next conflict resolution cycle. If you click Next while the last cycle in the session is displayed, then the first cycle displays and you can continue to click forward through the cycles again.
	Deletes the currently displayed conflict resolution cycle. Removing completed cycles frees up memory in Rule Debugger (you can see memory information in the user interface footer).
	Jump to the current conflict resolution cycle.

Debug Context Window Panel Toolbar Options

These options enable you to control rule execution during a Rule Debugger session.

Table 45 *Debug Context Panel Toolbar Buttons*

Button	Description
	Used in step mode to switch to run mode, and run the cycle to completion without pausing.
	Used in step mode to pause rule execution. Execution can be resumed using either the run or step buttons.
	Used in both step and run modes to stop rule execution. Rule execution cannot be restarted after it is stopped. You must start a new debugger session to resume debugging.
	Used in step mode to resume rule execution, pausing after each rule executes. Used in run mode to switch to step mode.
	Clear the list of rules executed. Not used in this release.

BusinessEvents Tools Menu Reference

Tables in this section describe the menu options available in BusinessEvents tools. Options on each menu are shown in the following sections:

- [File Menu Options, page 468](#)
- [Edit Menu Options, page 469](#)
- [View Menu Options, page 470](#)
- [Debug Menu Options, page 471](#)

File Menu Options

Table 46 BusinessEvents Tools File Menu Options

Option	Description
Open	<p>Rule Analyzer only. You can open one or more projects in Rule Analyzer. Enables you to browse to a project folder to select a project. All projects open are available in the Analyzer panel, using tabs. The project name is used as the tab label.</p> <p>This functionality is also provided by the Open  button.</p>
Reload	<p>Rule Analyzer only. If you change a project in TIBCO Designer and save the changes, clicking Reload reflects these changes in Rule Analyzer.</p> <p>This functionality is also provided by the Reload  button.</p>
Close Tab	Closes the currently displayed project tab.
Close All	Closes all project tabs.
Export as	You can export the model tab to an image file, for example, to share the view with others or for use in project documentation. See Exporting a Graph as an Image on page 445 .
Print	<p>Prints the graph. See Printing a Graph on page 447.</p> <p>This functionality is also provided by the Print  button.</p>
Print Preview	<p>Lets you see how the printed page would look.</p> <p>This functionality is also provided by the Print Preview  button.</p>
Page Setup	Enables you to configure the printer for the print job. See Printing a Graph on page 447 .
Exit	Exits the BusinessEvents Tools application.

Edit Menu Options

Table 47 *BusinessEvents Tools Edit Menu Options*

Option	Description
Undo	<p>Undoes changes, one change per mouse click, most recent first. You can set an undo limit. See BusinessEvents Tools Preferences Reference on page 472.</p> <p>This functionality is also provided by the Undo  button.</p>
Redo	<p>Redoes changes, one per mouse click, in reverse order (that is, last change first).</p> <p>This functionality is also provided by the Undo  button.</p>
Find	<p>Tool for finding entities in a large graph. See Finding Entities in a BusinessEvents Tools Graph on page 441.</p> <p>This functionality is also provided by the Undo  button.</p>
Preferences	<p>Allows you to set various preferences. See BusinessEvents Tools Preferences Reference on page 472.</p>

View Menu Options

Table 48 BusinessEvents Tools View Menu Options

Option	Description
Filter	<p>Tool used in Rule Analyzer only. You can also access the tool from the Analyzer/Dependency panel's toolbar. Allows you to filter the entities that appear in the graph; to add state machine transitions; to view only class diagram relationships; to hide class diagram relationships. See Rule Analyzer Filtering and Viewing Options on page 437.</p> <p>This functionality is also provided by the Filter  button (in the Analyzer/Dependency panel).</p>
Refresh	Restores the graph to a more clear layout arrangement.
Interactive Tools	<p>See BusinessEvents Tools Toolbar Reference on page 462 for details on these tools, which are also available on the toolbar:</p> <p>Select and Pan</p> <p>Marquee Zoom, Interactive Zoom, Fit in Window</p> <p>Link Navigator</p>
Layout	See Layout Options Reference on page 130 for details on the layout options.
Expand All Rule Nodes	<p>Displays each rule condition and its related links in the graph.</p> <p>This functionality is also provided by the Expand Rule Nodes  button.</p>
Collapse All Rule Nodes	<p>Hides all rule conditions. All links enter and leave the rule without indicating specific rule conditions.</p> <p>This functionality is also provided by the Collapse Rule Nodes  button.</p>
Show Properties	Shows or hides the named panel. See BusinessEvents Tools—Rule Analyzer User Interface Overview on page 423 for details about each panel.
Show Overview	
Show Source	
Show Console	Shows or clears the Console panel. The console panel displays BusinessEvents engine activity.
Clear Console	

Debug Menu Options

Table 49 *BusinessEvents Tools Debug Menu Options*

Option	Description
Launch	<p>Rule Debugger Only. Starts a Rule Debugger session using a profile. See Running Rule Debugger on page 451.</p> <p>This functionality is also provided by the Launch Debugger  button.</p>
Profiles	<p>Rule Debugger Only. Opens the Edit Configurations dialog. See Configuring Rule Debugger Profiles on page 454.</p> <p>This functionality is also provided by the Profiles  button.</p>

BusinessEvents Tools Preferences Reference

To set all preferences to their default values, click the **Defaults** button.

Interactive Preferences

Table 50 BusinessEvents Tools Interactive Preferences

Option	Description
Auto Hide Scrollbars	<p>If checked, then when the complete contents can display in the space available, scrollbars are hidden.</p> <p>If not checked, scrollbars remain.</p> <p>Default: Auto hide scrollbars is checked.</p>
Undo limit	<p>Number of levels of undo enabled.</p> <p>Default: 30.</p>
Grid	<p>Options are None, Lines, or Points. A grid can help you arrange nodes in an orderly way or see how nodes are positioned relative to each other.</p> <p>Default: None.</p>
Collapse Rule Nodes	<p>If checked, rule nodes (boxes) show only the rule name when the graph is first displayed.</p> <p>If unchecked, rule nodes are expanded when the graph is first displayed, showing conditions and their related links in the graph.</p> <p>Default: Collapse Rule Nodes is checked.</p>
Links	<p>Style to use for the links (that is, the lines in the graph connecting the rules and entities). Options are:</p> <p>Polyline—Line segments are joined using angles.</p> <p>Curved—line segments are joined using curves.</p> <p>Default: Curved.</p>
Link Colors	<p>Sets the color of the links. You can choose a color for links used in class diagrams (also known as concept views) and a different color for all other links. Text color is not affected by this setting.</p>

Layout Preferences

Table 51 *BusinessEvents Tools Layout Preferences*

Option	Description
Hierarchical Layout—Quality	<p>Options are Draft (lowest quality), Medium, and Proof (highest quality). It takes longer to generate a higher quality graph than a lower quality graph. A different algorithm is used in each case.</p> <p>Default: Medium</p>
Hierarchical Layout—Orientation	<p>Defines the general direction in which the links display, to reflect hierarchical relationships between the entities. BusinessEvents graphs are not particularly hierarchical, but setting this option defines the general direction of the layout.</p> <p>Options are: Left to Right, Top to Bottom, Right to Left, and Bottom to Top.</p> <p>Default: Left to Right</p>
Hierarchical Layout—Routing	<p>Determines how links are routed when the hierarchical layout is used:</p> <p>Orthogonal—Routes the links using horizontal and vertical straight line segments (that is, using right angles).</p> <p>Polyline—Routes the links using straight line segments with arbitrary angles.</p> <p>Default: Orthogonal.</p> <p>Overlapping lines are more likely with orthogonal routing than with polyline routing. With polyline routing the routing algorithm adds path nodes as needed to avoid overlapping lines.</p> <p>Note that the line segments can be joined using straight lines or curves in both cases. See the Links setting in Layout Preferences on page 473.</p>

Table 51 BusinessEvents Tools Layout Preferences (Cont'd)

Option	Description
Hierarchical Layout—Fix Node Sizes	<p>This setting affects link routing behavior for the hierarchical layout option only. See below for the setting used for all other layout options.</p> <p>If checked, node sizes do not change when you use the link routing feature.</p> <p>If unchecked, link routing changes node sizes as needed for clarity.</p> <p>Default: Unchecked</p>
Link Routing—Fix Node Sizes	<p>This setting affects link routing behavior for all layout options except hierarchical (see above for the setting that applies to hierarchical layouts).</p> <p>If checked, node sizes do not change when you use the link routing feature.</p> <p>If unchecked, link routing changes node sizes as needed for clarity.</p> <p>Default: Unchecked</p>
Link Routing—Fix Node Positions	<p>This setting affects link routing behavior for all layout options.</p> <p>If checked, node positions do not change when you use the link routing feature.</p> <p>If unchecked, link routing changes node positions as needed for clarity.</p> <p>Default: Checked</p>
Orthogonal Layout, Circular Layout, Symmetric Layout - Quality	<p>Options are Draft (lowest quality), Medium, and Proof (highest quality). It takes longer to generate a higher quality graph than a lower quality graph. A different algorithm is used in each case.</p> <p>Default: Medium</p>

Look and Feel Preferences

Choose a look-and-feel option for the user interface, to suit your needs:

- Vsnet
- Windows XP (Default)
- Office 2003 Blue
- Office 2003 Metallic
- Eclipse
- Xerto
- Metal

BusinessEvents Tools Layout Options Reference

Layout options refer to the style of the graph that is rendered in the Rule Analyzer and Rule Debugger Analyzer/Dependency panel.



Additional Layout Options

Various options affect the appearance of a graph, no matter what layout is chosen. (The hierarchical layout offers additional options.) See the following sections for details:

- [Interactive Preferences on page 472](#)
- [Layout Preferences on page 473](#)

The main layout options are:

- [Hierarchical Layout, page 476](#)
- [Orthogonal Layout, page 477](#)
- [Circular Layout, page 477](#)
- [Symmetric Layout, page 477](#)

Incremental Layout

Choose Incremental Layout if you want the main arrangement of the graph to remain stable when you make changes to your project and only re-route the entities and links that have changed since the last rendering.

Keeping most things in the same place makes it easier to see how changes you have made in the project affect the graph.

If you want to refresh the entire graph, click the desired layout option. The program has greater freedom to optimize the layout. The result is likely to be a more pleasing arrangement.

To perform an incremental layout update, do one of the following:

- Select View > Layout > Incremental Layout
- Click the Incremental Layout button  on the tool bar.

Hierarchical Layout

The hierarchical layout style indicates dependencies by positioning the nodes at different levels. The hierarchical layout style is useful when it is useful to show

precedence relationships in the ontology. The hierarchical layout style indicates dependencies by positioning the nodes at different levels. You can use Layout Preferences to determine in the direction of the hierarchy. Preferences and options specific to the hierarchical layout are:

- Orientation Options (Edit > Preferences)—Left to Right, Top to Bottom, Right to Left, Bottom to Top
- Routing Options (View > Layout)—Orthogonal or normal (polyline) routing
- Routing Options (Edit > Preferences)—Orthogonal or polyline routing.

See [BusinessEvents Tools Preferences Reference on page 472](#) for more details.

Orthogonal Layout

The orthogonal layout style uses only horizontal and vertical links. Graphs are drawn quickly.

The algorithm places highly connected nodes closer together, resulting in a more compact graph. Even when lines overlap, the algorithm ensures that they are still easy to follow. Because this style has no hierarchical or other visual constraints, the resulting graphs are often very clear.

Circular Layout

This layout is useful for ontologies where nodes tend to have a clustered (ring or star) structure (where each main node has a starburst of related nodes).

Symmetric Layout

The symmetric layout style looks for and emphasizes the symmetries in a project topology. It can produce a pleasing visual result, if there are no reasons to arrange the nodes for other reasons, for example there is no hierarchy or ring-clustering inherent in the structure of the nodes.

Appendix H **Deprecated and Unused Properties**

The following BusinessEvents engine properties are either deprecated or are not used by BusinessEvents in this release.

Table 52 *Deprecated and Unused Engine Properties*

Property	Description
<code>be.engine.cluster.cacheType</code>	<p>Deprecated property. Ignored if present. In 3.0.0, this property specifies which of the following provided caching schemes to use: <code>dist-unlimited-bs</code>, <code>dist-limited-bs</code>, or <code>dist-unlimited-nobs</code> (default)</p> <p>Instead various properties are used to internally select the correct caching scheme, for example, <code>be.engine.cluster.isCacheLimited</code> and <code>be.engine.cluster.hasBackingStore</code></p> <p>See Chapter 20, Configuring Cache Cluster Settings, on page 295 for more details.</p>
<code>be.engine.om.recovery.threads</code>	Not used in this release. Ignored if present.
<code>be.ft.cluster.name</code>	Deprecated property. Instead use <code>Engine.FT.GroupName</code> for In Memory OM fault tolerance.
<code>be.ft.enabled</code>	Deprecated property. Instead use <code>Engine.FT.UseFT</code> for In Memory OM fault tolerance.
<code>be.ft.failback.waitmilliseconds</code> <code>be.ft.failover.waitmilliseconds</code>	Deprecated Properties. If present work with In Memory OM fault tolerance, but has not purpose. These properties were introduced in a 2.x release to define a wait period, ensuring that cache was fully initialized before failing over to a secondary or failing back to the recovered primary.
<code>be.ft.priority</code>	Deprecated property. Instead use <code>Engine.FT.Weight</code> for In Memory OM fault tolerance. (Unlike the currently used property, in this deprecated property, the lower the number the higher the priority.)

Table 52 *Deprecated and Unused Engine Properties (Cont'd)*

Property	Description
<code>be.locale.country</code>	<p>Sets the country code to use for localization. Use upper case. Uses the ISO 3166 standard.</p> <p>Note: BusinessEvents is not fully localized in this release.</p>
<code>be.locale.language</code>	<p>Sets the language code to use for localization. Uses the ISO 639 standard.</p> <p>Note: BusinessEvents is not fully localized in this release.</p>
<code>be.locale.variant</code>	<p>Optional extension to the locale language, for example, if you set <code>be.locale.language</code> to <code>en</code>, you might set <code>be.locale.variant</code> to <code>US</code>, which is interpreted as <code>en_US</code></p> <p>Note: BusinessEvents is not fully localized in this release.</p>
<code>be.trace.log.append</code>	<p>This property is ignored in BusinessEvents. The behavior is: log files are appended. to the end of the current log file when the BusinessEvents server starts. This behavior cannot be changed.</p>
<code>be.trace.log.dir</code>	<p>This property is ignored in BusinessEvents. Instead the property <code>Engine.Log.Dir</code> is used.</p>
<code>Engine.Log.MaxNum</code>	<p>This property is ignored in BusinessEvents. Instead the property <code>be.trace.log.maxnum</code> is used.</p> <p>The <code>Engine.Log.MaxNum</code> property appears in the TRA file that is generated by TIBCO Administrator. In the TIBCO Administrator user interface, the value for this setting is added to the following location: the Server Settings tab of the merged BAR file instance. (The structure within the application configuration builder area is: application at the top level, then merged bar file, then merged bar file instance).</p>
<code>Engine.Log.MaxSize</code>	<p>This property is ignored in BusinessEvents. Instead the property <code>be.trace.log.maxsize</code> is used.</p> <p>The comments for <code>Engine.Log.MaxNum</code> also apply to the <code>Engine.Log.MaxSize</code> property.</p>
<code>java.property.tangosol.coherence.cacheconfig</code>	<p>Not needed or used in this release except as advised by TIBCO Support.</p>

TIBCO Hawk Microagent Methods

TIBCO Administrator is the preferred monitoring and management application for TIBCO BusinessEvents. However, the BusinessEvents engine is instrumented with a TIBCO Hawk microagent that can be used to perform many administrative functions. This appendix describes the microagent methods available for the BusinessEvents engine.

Topics

- [TIBCO Hawk Methods Overview, page 483](#)
- [activateRuleSet\(\), page 484](#)
- [activateTraceRole\(\), page 485](#)
- [deactivateRuleSet\(\), page 486](#)
- [deactivateTraceRole\(\), page 487](#)
- [execute\(\), page 488](#)
- [forceOMCheckpoint\(\), page 489](#)
- [getChannels\(\), page 490](#)
- [getDestinations\(\), page 491](#)
- [getEvent\(\), page 492](#)
- [GetExecInfo\(\), page 493](#)
- [getHostInformation\(\), page 494](#)
- [getInstance\(\), page 495](#)
- [getMemoryUsage\(\), page 496](#)
- [getNumberOfEvents\(\), page 497](#)
- [getNumberOfInstances\(\), page 498](#)
- [getOMInfo\(\), page 499](#)
- [getRuleSet\(\), page 500](#)

- [getRuleSets\(\), page 501](#)
- [getScorecard\(\), page 502](#)
- [getScorecards\(\), page 503](#)
- [getSessionInputDestinations\(\), page 504](#)
- [getSessions\(\), page 505](#)
- [getStatus\(\), page 506](#)
- [getTotalNumberRulesFired\(\), page 507](#)
- [getTraceSinks\(\), page 508](#)
- [reconnectChannels\(\), page 509](#)
- [resetTotalNumberRulesFired\(\), page 510](#)
- [resumeChannels\(\), page 511](#)
- [resumeDestinations\(\), page 512](#)
- [stopApplicationInstance\(\), page 513](#)
- [suspendChannels\(\), page 514](#)
- [suspendDestinations\(\), page 515](#)

TIBCO Hawk Methods Overview

BusinessEvents embeds a TIBCO Hawk microagent whose methods enable you to monitor and manage deployed BusinessEvents applications. You can use TIBCO Hawk or the Hawk Console in TIBCO Administrator.

Types of Methods

The methods documented in this appendix are provided for the following purposes:

- To enable TIBCO Administrator to perform certain actions, for example, `GetExecInfo()`, `stopApplicationInstance()`, `getHostInformation()`
- To provide information about what is happening in the BusinessEvents engine, for example, `getRuleSets()`, `getDestinations()`, `getTotalNumberRulesFired()`
- To make certain changes in the BusinessEvents engine without stopping it, for example, `activateRuleSet()`, `forceOMCheckpoint()`, `reconnectChannels()`

Enabling TIBCO Hawk Microagent

Before using the Hawk methods, you must enable the TIBCO Hawk microagent in the BusinessEvents engine property file, `be-engine.tra`. To do this, set the `Hawk.Enabled` property to true and ensure that it is uncommented.

If you are using non-default transport parameters for TIBCO Hawk, you must also set the `repo.hawkDaemon`, `repo.hawkNetwork`, and `repo.hawkService` properties to the values for the transport you are using.

For More Information

TIBCO Administrator Server Configuration Guide has more details on working with microagents and methods using TIBCO Administrator. *TIBCO Hawk Methods Reference* provides detailed documentation about TIBCO Hawk microagents and methods.

activateRuleSet()

Purpose Activate a RuleSet in the Session

Type ACTION

Parameters

Name	Description
Session	Name of the Session (optional).
URI	URI of the RuleSet.

Returns

Type	Description
Session	Name of the Session (optional).
URI	URI of the RuleSet.
Activated	Is the RuleSet activated?

activateTraceRole()

Purpose Enable a Trace Role

Type ACTION

Parameters

Name	Description
Role Name	Name of a Role

Returns Returns nothing.

deactivateRuleSet()

Purpose Deactivate a RuleSet in the Session

Type ACTION

Parameters

Name	Description
Session	Name of the Session
URI	URI of the RuleSet

Returns

Type	Description
Session	Name of the Session.
URI	URI of the RuleSet.
Deactivated	Is the RuleSet deactivated?

deactivateTraceRole()

Purpose Disable a Trace Role

Type ACTION

Parameters

Name	Description
Role Name	Name of a Role

Returns Returns nothing.

execute()

Purpose Runs a special command.

Type ACTION_INFO

Parameters

Name	Description
Command	The special command to execute
Parameters	Parameters (optional)

Returns

Type	Description
Line	Line Number.
Name	Name.
Value	Value.

forceOMCheckpoint()

Purpose Forces a Object Store checkpoint of a Session.

Type ACTION

Parameters

Name	Description
Session	Name of the Session

Returns Returns nothing.

getChannels()

Purpose Retrieves Channel Info.

Type INFO

Parameters

Name	Description
URI	URI of the Channel (optional)

Returns

Type	Description
Line	Line Number
URI	URI of the Channel.
State	Current state of the Channel

getDestinations()

Purpose Retrieves Destination Info.

Type INFO

Parameters

Name	Description
Channel URI	URI of the Channel (optional).
Destination Name	Name of the Destination (optional).

Returns

Type	Description
Line	Line Number.
Channel URI	URI of the Channel.
Destination URI	URI of the Destination.
Nb in	Number of Events in.
Rate in	Rate of Events in.
Nb out	Number of Events out.
Rate out	Rate of Events out.

getEvent()

Purpose Retrieves an Event from a Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session
Id	Id of the Event
External	True if using the event's external Id, false if using the internal Id

Returns

Type	Description
Line	Line number.
Session	Name of the Session.
Type	Attribute or Property.
Name	Name of the Attribute or Property.
Value	Value of the Attribute or Property.

GetExecInfo()

Purpose Gets engine execution information

Type INFO

Parameters No parameters.

Returns

Type	Description
Status	Engine status (ACTIVE, SUSPENDED, STANDBY or STOPPING)
Uptime	Elapsed time since RuleSessionProvider was started (milliseconds)
Threads	Number of RuleSessions in engine.
Version	Project version

getHostInformation()

Purpose Gets host information properties.

Type INFO

Parameters

Name	Description
Name	Name of host information property to get (optional).

Returns

Type	Description
Name	Property Name
Value	Property Value

getInstance()

Purpose Retrieves an Instance from the Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session
Id	Id of the Instance.
External	True if using the instance's external Id, false if using the internal Id.

Returns

Type	Description
Line	Line number.
Session	Name of the Session.
Type	Attribute or Property.
Name	Name of the Attribute or Property.
Value	Value of the Attribute or Property.

getMemoryUsage()

Purpose Gets engine memory usage information.

Type INFO

Parameters No parameters.

Returns

Type	Description
Max	Maximum memory size of the JVM, in bytes.
Free	Estimate of the free memory available to the JVM, in bytes.
Used	Estimate of the memory used in the JVM, in bytes.
PercentUsed	Estimate of the percentage of max memory used.

getNumberOfEvents()

Purpose Get the total number of events existing in a Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session

Returns

Type	Description
Line	Line number.
Session	Name of the Session.
Number	Total Number of Events

getNumberOfInstances()

Purpose Get the total number of instances existing in a Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session

Returns

Type	Description
Line	Line number.
Session	Name of the Session.
Number	Total Number of Instances

getOMInfo()

Purpose Retrieves Object Store information of a Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session

Returns

Type	Description
Line	Line number.
Session	Name of the Session
Property	Property name.
Value	Property value.

getRuleSet()

Purpose Retrieves the Rules of a given RuleSet.

Type INFO

Parameters

Name	Description
Session	Name of the Session
URI	URI of the RuleSet

Returns

Type	Description
Line	Line Number.
Session	Name of the Session.
URI	URI of the RuleSet
Rule	Name of the Rule
Priority	Priority of the rule.

getRuleSets()

Purpose Retrieves a RuleSets from the Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session

Returns

Type	Description
Line	Line Number.
Session	Name of the Session.
URI	URI of the RuleSet.
Activated	Is the RuleSet activated.

getScorecard()

Purpose Retrieves a Scorecard of a Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session
URI	URI of the Scorecard.

Returns

Type	Description
Line	Line number.
Session	Name of the Session.
Type	Attribute or Property.
Name	Name of the Attribute or Property.
Value	Value of the Attribute or Property.

getScorecards()

Purpose Retrieves all the Scorecards of a Session.

Type INFO

Parameters

Name	Description
Session	Name of the Session

Returns

Type	Description
Line	Line Number.
Session	Name of the Session.
Id	Id of the Scorecard.
External Id	External Id of the Scorecard.
Type	Class of the Scorecard.

getSessionInputDestinations()

Purpose Retrieves destinations enabled for input.

Type INFO

Parameters

Name	Description
Session	Name of the Session (optional).

Returns

Type	Description
Line	Line number.
Destination	Destination URI.
Preprocessor	Destination preprocessor URI.

getSessions()

Purpose Retrieves session names.

Type INFO

Parameters No parameters.

Returns

Type	Description
Line	Line number.
Session	Name of the Session.

getStatus()

Purpose Retrieves basic status information about the engine.

Type INFO

Parameters No parameters.

Returns

Type	Description
Instance ID	Instance ID of the application.
Application Name	Name of the application.
Uptime	Time elapsed since startup.
Process ID	Process ID of the application.
Host	Name of host machine on which this application is running.

getTotalNumberRulesFired()

Purpose Retrieves the total number of rules fired.

Type INFO

Parameters

Name	Description
Session	Name of the Session

Returns

Type	Description
Line	Line Number.
Session	Name of the Session.
Number of Rules Fired	Total number of rules fired since the last reset.

getTraceSinks()

Purpose Gets information about trace sinks.

Type INFO

Parameters

Name	Description
Role Name	Name of a Role (optional)
Sink Name	Name of a Sink (optional)

Returns

Type	Description
Line	Line Number
Instance ID	Instance ID of the application
Application Name	Name of the application
Sink Name	Sink Name
Sink Type	Sink Type (for example, fileSink, rvSink)
Description	Sink Description (for example, filename=file)
Role	Sink Role (for example, error, warn, debug)

reconnectChannels()

Purpose Restarts all channels or a single channel.

Type ACTION

Parameters

Name	Description
URI	URI of the channel to restart (all channels are restarted if this is empty).

Returns Returns nothing.

resetTotalNumberRulesFired()

Purpose Resets the total number of rules fired to zero.

Type ACTION

Parameters

Name	Description
Session	Name of the Session

Returns Returns nothing.

resumeChannels()

Purpose Resumes channels.

Type ACTION

Parameters

Name	Description
URI	URI of the Channel to resume (optional).

Returns Returns nothing.

resumeDestinations()

Purpose Resumes Destinations.

Type ACTION

Parameters

Name	Description
Channel URI	URI of the Channel that contains the Destination.
Destination Name	Name of the Destination (optional).

Returns Returns nothing.

stopApplicationInstance()

Purpose	Shuts down the engine. All checkpoint files will be preserved and the engine's operating system process will exit.
Type	ACTION
Parameters	No parameters.
Returns	Returns nothing.

suspendChannels()

Purpose Suspends channels.

Type ACTION

Parameters

Name	Description
URI	URI of the Channel to suspend (optional).

Returns Returns nothing.

suspendDestinations()

Purpose Suspends Destinations.

Type ACTION

Parameters

Name	Description
Channel URI	URI of the Channel that contains the Destination.
Destination Name	Name of the Destination (optional).

Returns Returns nothing.

Index of Engine Properties

A

Agent.AgentGroupName.key 313, 315
 Agent.AgentGroupName.l1CacheSize 314, 316
 Agent.AgentGroupName.maxActive 314, 315
 Agent.AgentGroupName.priority 314, 315
 Agent.AgentGroupName.recoveryPageSize 314, 317
 Agent.AgentGroupName.threadcount 278, 314, 316

B

be.agent.query.localcache.evictseconds 327, 327
 be.agent.query.localcache.maxelements 327, 327
 be.agent.query.localcache.prefetchaggressive 327, 328
 be.channel.tibjms.queue.ack.mode 44
 be.channel.tibjms.topic.ack.mode 44
 be.engine.cacheServer 321, 321, 322
 be.engine.cluster.cacheType (deprecated) 479
 be.engine.cluster.EntityClassName.preload 344, 350, 352
 be.engine.cluster.externalClasses.classLoader 299, 304
 be.engine.cluster.externalClasses.path 299, 304
 be.engine.cluster.hasBackingStore 297, 344, 348, 479
 be.engine.cluster.isCacheLimited 297, 344, 348, 479
 be.engine.cluster.minCacheServers 298, 349
 be.engine.cluster.multiEngineOn 297
 be.engine.cluster.preload 344, 350, 352
 be.engine.hotDeploy.enabled 395, 395
 be.engine.name 388
 be.engine.om.berkeleydb.cacheweight.RuleSession 260
 be.engine.om.berkeleydb.dbenv 261
 be.engine.om.berkeleydb.internalcachepersent 260
 be.engine.om.eventcache.defaultmaxsize 261
 be.engine.om.eventcache.maxsize.rule_session 261
 be.engine.om.recovery.threads 479
 be.engine.profile.*.duration 224
 be.engine.profile.*.enable 223
 be.engine.profile.*.file 224
 be.engine.profile.*.level 225
 be.engine.profile.BAR_Name.duration 224
 be.engine.profile.BAR_Name.enable 223
 be.engine.profile.BAR_Name.file 223
 be.engine.profile.BAR_Name.level 225
 be.engine.tangosol.oracle.batchSize 344, 344, 351
 be.engine.tangosol.oracle.prefetch 343, 344, 351
 be.ft.cluster.name (deprecated) 479
 be.ft.enabled (deprecated) 479
 be.ft.failback.waitmilliseconds (deprecated) 479
 be.ft.failover.waitmilliseconds (deprecated) 479
 be.ft.nodename 247, 248
 be.ft.priority (deprecated) 479
 be.hawk.microagent.name 203
 be.jms.reconnect.msgCodes 33
 be.jms.reconnect.timeout 32
 be.locale.country 480
 be.locale.language 480
 be.locale.variant 480
 be.oracle.dburi.n 351
 be.oracle.dburi.n> 344, 347
 be.oracle.dburi.pool.enforce.0 344, 347, 351
 be.oracle.dburi.pool.initial.n 344, 347
 be.oracle.dburi.pool.max.n 344, 347, 351
 be.oracle.dburi.pool.min.n 344, 347, 351
 be.oracle.debug 344, 344, 351
 be.oracle.pool.initial.n 351
 be.oracle.schemamigration.pswd 338
 be.oracle.schemamigration.url 338
 be.oracle.schemamigration.user 338
 be.trace.enable 376
 be.trace.log.append (not used) 480
 be.trace.log.dir (not used) 480
 be.trace.log.enable 376
 be.trace.log.fileName 376
 be.trace.log.maxnum 376
 be.trace.log.maxsize 377

be.trace.publish.daemon [377](#)
 be.trace.publish.enable [377](#)
 be.trace.publish.network [377](#)
 be.trace.publish.service [377](#)
 be.trace.publish.subjec [377](#)
 be.trace.roles [377](#)
 be.trace.term.enable [378](#)

C

com.tibco.cep.runtime.scheduler.default.numThreads [367](#)
 com.tibco.cep.runtime.scheduler.queueSize [367](#), [367](#)
 com.tibco.tibjms.connect.attempts [32](#)

E

Engine.FT.GroupName (used for in Memory OM only) [248](#)
 Engine.FT.UseFT(used for in Memory OM only) [248](#)
 Engine.FT.Weight (used for in Memory OM only) [248](#)
 Engine.Log.Dir [378](#)
 Engine.Log.MaxNum (not used) [480](#)
 Engine.Log.MaxSize (not used) [480](#)

H

Hawk.AMI.DisplayName [202](#)

J

java.property.be.engine.limited.cache.back.size.limit [2](#)
[97](#), [297](#), [344](#), [348](#)
 java.property.java.net.preferIPv4Stack [279](#), [288](#)
 java.property.mode [414](#)
 java.property.palettePath [199](#)
 java.property.tangosol.coherence.cacheconfig [480](#)

java.property.tangosol.coherence.cluster [290](#), [291](#), [293](#),
[303](#)
 java.property.tangosol.coherence.clusteraddress [290](#),
[290](#), [291](#), [303](#)
 java.property.tangosol.coherence.clusterport [290](#), [290](#),
[291](#), [303](#)
 java.property.tangosol.coherence.distributed.backupcount [298](#)
 java.property.tangosol.coherence.distributed.localstorage [313](#), [316](#), [321](#), [321](#), [343](#)
 java.property.tangosol.coherence.distributed.threads [299](#)
 java.property.tangosol.coherence.localhost [294](#)
 java.property.tangosol.coherence.localport [294](#)
 java.property.tangosol.coherence.log [301](#)
 java.property.tangosol.coherence.log.level [301](#)
 java.property.tangosol.coherence.log.limit [302](#)
 java.property.tangosol.coherence.override [414](#)
 java.property.tangosol.coherence.ttl [290](#), [291](#), [303](#)
 java.property.tangosol.coherence.wkan [292](#), [293](#)
 java.property.tangosol.coherence.wkan.port [292](#), [293](#)
 java.property.tibco.be.schema.exclude.null.props [105](#)
 java.property.tibco.be.schema.treat.null.values [105](#)

T

tibco.be.jdbc.connection.retry.count [117](#)
 tibco.be.jdbc.test.connections.interval [117](#)
 tibco.be.property.double.null.value [106](#)
 tibco.be.property.int.null.value [106](#)
 tibco.be.property.long.null.value [106](#)
 tibco.bwengine.name [201](#), [201](#)
 tibco.bwreporurl [202](#)
 tibco.clientVar.variableName [359](#)
 tibco.env.APP_ARGS [369](#)
 tibco.env.BW_HOME [198](#)
 tibco.env.BW_MIGRATION_APPEND_VERSION [198](#)
 tibco.env.BW_PLUGINS_HOME [198](#)
 tibco.env.BW_PLUGINS_HOME_OLD [198](#)
 tibco.env.CUSTOM_EXT_APPEND_CP [198](#)
 tibco.env.CUSTOM_EXT_PREPEND_CP [198](#), [198](#)
 tibco.repourl [388](#), [388](#), [399](#)

Glossary

A

advisory event

A notice from BusinessEvents about activity in the engine, for example, an exception. See also [event](#), [simple event](#), [simple event definition](#), [time event](#).

agenda

A prioritized list of rule actions that may fire. BusinessEvents recreates the agenda each time a change in working memory requires rules to be re-evaluated, for example, when a simple event is asserted. A rule action in an agenda may disappear without firing when the agenda is recreated.

agent

A deployed BAR. See [cache node](#), [inference agent](#), [query agent](#).

assert

Put facts into working memory. When object instances or events are asserted into working memory, rules may fire as a result. See also [fact](#), [retract](#).

B

BusinessEvents Archive (BAR)

A BusinessEvents Archive (BAR) file stores design-time resources used at runtime by agents. See also [agent](#), [rule session](#).

C

cache

In BusinessEvents, a form of object management. Refers to storage of facts (objects) used by the runtime engine. Various types of cache-based object management options are available. See also [object management](#), [cache mode](#).

cache cluster

A group of nodes configured for use by the object management system to store cache data and backups of cache data. See [object management](#), [cache](#).

cache mode

Options within cache-based object management. You can tune the performance of your application, and reduce its footprint in memory using different cache modes. Modes are cache plus memory, cache only, and in memory only. See also [object management](#), [cache only](#), [cache plus memory](#), [in memory only](#).

cache node

A node configured with a non-reasoning agent used as a storage node only. Used with Cache object management. See also [agent](#), [cache](#).

cache only

One of the cache modes, available for cache-based object management configuration. Instances of entity types that use cache only are serialized and kept in the cache until needed. They must be explicitly loaded into working memory when needed for rule processing. See also [object management](#), [cache](#), [cache mode](#)

cache plus memory

One of the cache modes, available for cache-based object management configuration. Instances of entity types that use cache plus memory are serialized and kept in cache until needed. There is one object in the cache (in a logical sense), and any number of references to that object in each JVM. See also [object management](#), [cache](#), [cache mode](#).

cache server

See [cache node](#).

caching scheme

In the cache configuration descriptor file, caching schemes provide configuration details for instances of the cache-based object management options. See also [object management](#), [cache](#).

complex event

An event that happens only when many other events have also happened. Or, an event that you infer from other events. See also [simple event](#).

complex event processing (CEP)

Correlation of multiple events from an event cloud, with the aim of identifying meaningful events and taking appropriate action.

channel

A named configuration that allows BusinessEvents to listen to a stream of events from a given type of source. Similar to a radio channel.

Example: An administrator might configure a channel that would allow BusinessEvents to listen to EMS messages. A channel contains one or more destinations. See also [destination](#).

checkpoint

Used with persistence object management. A checkpoint is the point in time at which working memory data is written to disk. The term checkpoint also encompasses all the activities involved in writing the data to disk.

complex event

An abstraction that results from patterns detected among simple events.

Example: A complex event might result from the following simple events that all occurred within one week's time: A stock broker buys shares of xyz stock. The same broker submits a very large order for xyz stock on behalf of a customer. The same broker sells shares of xyz stock at a profit.

concept

An abstract entity similar to the object-oriented concept of a class. A concept is a description of a set of properties that, when grouped together, create a meaningful unit. Concepts can be organized in a hierarchical structure.

Example: *Department*, *employee*, *purchase order*, and *inventory item* are all concepts. The term *concept type* refers to the definition and the term *concept instance* refers to the actual object.

concept reference

A property within one concept that references the ID of another concept.

concept view

A way to view and define groupings of concepts and their relationships in the BusinessEvents user interface.

conflict resolution cycle

A cycle of activities during which the engine executes rule actions on the currently asserted facts. The engine re-evaluates what rules need to fire after each action is executed, based on changes in working memory and re-creates the

agenda. The cycle is complete when all actions triggered by all conditions specified in rules have been met and no more actions remain. See also [agenda](#).

contained concept

A concept that exists entirely within another concept.

custom function

You can add Java-based custom functions as needed to supplement the library of standard functions provided with BusinessEvents. See also [ontology function](#), [rule function](#), [standard function](#).

D

decision artifact

Any entity which is part of a decision project and whose life cycle is managed by RMS. See also [RMS](#).

Decision Manager

an Eclipse-based Rich Client Platform (RCP) application that enables non-technical users to author rules. Each rule is represented by a row in a decision table. See also [decision table](#).

decision project

A BusinessEvents project ontology, plus decision tables, domain model, test data, access control, and other decision artifacts. Used within Decision Manager.

decision table

A tabular form presenting a set of conditions and their corresponding actions. A graphical tool for building rules. Used in Decision Manager.

deserializer

A class that performs conversion tasks. In BusinessEvents, a deserializer converts messages to events. See also [serializer](#)

destination

A channel property that defines a contact point on a given channel. For example, for a TIBCO Rendezvous channel, the destination properties would specify the subjects on which to listen.

distributed cache

In BusinessEvents, a form of cache-based object management. In a distributed cache, cached object data is partitioned between the nodes (JVMs) in the cache cluster for efficient use of memory. See also [object management](#), [cache](#).

E

entity

A concept, simple event, or scorecard. Entity instances are instances of a concept or scorecard, whereas entity types are the definition of the entity.

event

An object representing some occurrence or point in time. See also [advisory event](#), [simple event](#), [simple event definition](#), [time event](#).

evict

Term used in object management. To remove an object or entry from a cache. For example, in the near cache caching scheme, an eviction policy defines when an object is removed from the local cache. In the case of the cache only cache mode (advanced cache option), the term is also used to mean "remove the entity and its references from working memory." See also [object management](#).

expires (event)

At the end of the event's time to live period, the event is said to expire. It is removed from the working memory and (as needed) acknowledged. Other actions depend on the type of object management used. See also [time to live](#).

F**fact**

An instance of an event or concept or scorecard in working memory. See also [working memory](#).

I**in memory**

In BusinessEvents, a form of object management. Refers to storage of facts (objects) used by the runtime engine in JVM memory. See also [object management](#).

in memory only

One of the cache modes, available for cache-based object management configuration. Instances of entity types that use in memory only are available only in the engine's local JVM memory. See also [object management](#), [cache](#), [cache mode](#).

inference agent

In a deployed system, inference agents process incoming events using a Rete network as the inferencing engine, and a set of rules that are triggered by conditions in incoming events. Inference agents in Cache OM systems allow fault tolerance and load balancing. See also [Rete algorithm](#), [rule session](#).

inheritance

In BusinessEvents, a relationship between two concepts in which one concept automatically includes all the properties of another.

instance

In BusinessEvents, a usage of a concept definition. Similar to the Java term "object instance." Example: The Austen Elder object is an instance of the *employee* concept.

L**lambda transition**

A transition without a condition. This term is used in state machine configuration.

M**multiplicity**

A boolean parameter indicating whether or not BusinessEvents needs to allow for a collection or list of values in a concept property. For example, a purchase order concept would allow for only one PO number, but multiple line items.

O**object management**

The system in BusinessEvents that manages all the facts used in the runtime engine. Facts can be kept in memory only, kept in memory caches (with optional backing store), or persisted to a database.

ontology function

BusinessEvents generates ontology functions for each entity type in a project. There are three types of ontology functions: *constructors*, to create a simple event or concept instance; *time events*, to create and schedule time events, and *rule functions*, to invoke rule functions. See also [custom function](#), [rule function](#), [standard function](#).

P

payload

Similar to a JMS message, a simple event can contain properties and a payload. The payload holds the content of the message. You can define the XML schema for the payload when you configure the simple event definition. Payloads can also contain strings and Byte arrays.

persistence

In BusinessEvents, a form of object management. Refers to storage of facts (objects) used by the runtime engine, in a database. See also [object management](#).

Q

query agent

A query agent is a non-reasoning agent that and has read-only access to the underlying objects in the cache cluster. A query agent has no Rete network. Available only in Enterprise Suite. See also [agent](#).

R

Rete algorithm

Dr Charles L. Forgy developed the Rete algorithm for expert systems See also [Rete network](#).

Rete network

An in-memory network of objects based on the Rete algorithm which enables fast matching of facts with rule dependencies. "A Rete-based expert system builds a network of nodes, where each node (except the root) corresponds to a pattern occurring in the left-hand-side (the condition part) of a rule. The path from the root node to a leaf node defines a complete rule left-hand-side. Each node has a memory of facts which satisfy that pattern."

(http://en.wikipedia.org/wiki/Rete_algorithm)

retract

Remove facts from working memory. See also [assert](#), [fact](#), [working memory](#).

RMS

Rules Management Server. The server component of Decision Manager. RMS manages the rules management repository.

RTC (Run to Completion)

This term has the same meaning as the term "conflict resolution cycle." See [conflict resolution cycle](#).

rule

A declaration, with a set of conditions and actions. If all the conditions in the rule are satisfied by facts in working memory, BusinessEvents executes the action. See also [working memory](#).

rule based time event

See [time event](#).

rule function

Custom functions written using the BusinessEvents rule language. A rule function resource provides the editor for writing rule functions. See also [custom function](#), [ontology function](#), [standard function](#).

Also used to refer to a type of ontology function.

rule session

The run-time equivalent of a BusinessEvents Archive resource. Includes preprocessors, destinations, working memory, and the object manager. See also [BusinessEvents Archive \(BAR\)](#), [inference agent](#), [working memory](#).

rule set

A group of rules. At deployment time, you can select rule sets to use at runtime. See also [rule](#).

S**serializer**

A class that performs conversion tasks. In BusinessEvents, a serializer converts events to messages. See also [deserializer](#)

simple event

An object representing a business activity that happened at a single point in time. A Simple event includes information for evaluation by rules, meta-data that provides context, and a separate payload — a set of data relevant to the activity. See also [advisory event](#), [event](#), [simple event definition](#), [time event](#), [time to live](#). See also [complex event](#)

simple event definition

A description of the channel, destination, properties, and payload for a simple event. See also [simple event](#).

standard function

A library of standard functions is provided with BusinessEvents for use in rules and rule functions. See also [custom function](#), [ontology function](#), [rule function](#).

T**time event**

A type of event definition, used as a timer. Two types are available: rule based, and interval based (repeat every). Rule based time events are scheduled by rules. Interval based time event occur at the specified time interval. See also [advisory event](#), [event](#), [simple event](#), [simple event definition](#), [time to live](#).

Also used to refer to a type of ontology function.

time to live

A simple event property that defines the delay after which a simple event expires.

U**URI**

Unique resource identifier. Every event and instance has a URI for identification within the BusinessEvents server.

UML (Univied Modeling Language)

A language that assists in building a diagram of any complex entity. BusinessEvents state-model functionality is UML compliant. The

BusinessEvents term, *concept*, is similar to a UML class.

V

virtual rule function

A rule function whose signature is defined in TIBCO Designer and which is implemented in Decision Manager, using decision tables. The classes are then loaded into BusinessEvents engines. See also [rule function](#).

W

Workbench

The component of TIBCO BusinessEvents that integrates with TIBCO Designer and includes BusinessEvents palettes.

working memory

The runtime processing area for rules, objects, and actions. Rules apply only to data in the working memory. See also [BusinessEvents Archive \(BAR\)](#), [rule session](#).

Index

Symbols

@closure 68
 @extId 62
 @id 62, 68
 @interval 68
 @parent 95
 @payload 62
 @scheduledTime 68
 @ttl 62, 68

A

action-only functions 157
 actions
 entry and exit 177
 in rule editor 137, 152
 overview of rule actions 12
 activateRuleSet() 484
 activateTraceRole() 485
 activities palette 184
 advanced caching options 266
 and concept relationships 84, 282, 310
 cache only 284
 cache plus memory 282
 design requirements when using 285
 in memory only 283
 understanding 282
 advanced settings for cache-based object management 311
 advisory event (glossary) 519
 advisory events 9, 72, 141
 agenda (glossary) 519
 Agenda tab in Rule Debugger Context Window panel 431
 agent (glossary) 519
 agent group name (cache-based object management

 setting) 309, 326
 AIX, engine property for cache OM 279, 288
 aliases 152
 all values history policy 82, 93
 Analyzer/Dependency graph, using 435
 application management interface, TIBCO Hawk 387
 applications
 running at the command-line 380
 AppManage 357
 architecture
 components 6
 runtime 18
 assertion (glossary) 519

B

backing store 265
 BAR. See BusinessEvents archive resources
 BE_HOME xxix
 be-oradeploy 332
 BusinessEvents Archive (glossary) 519
 BusinessEvents archive resources
 functions of 362
 overview 15
 search utility for 141
 BusinessEvents servers, runtime purpose of 8

BusinessEvents Tools

- Analyzer/Dependency graph, using 435
 - bottom toolbar options 464
 - Debug Context Window panel toolbar 466
 - Debug menu options 471
 - Edit menu options 469
 - exporting graphs 445
 - File menu options 468
 - finding entities in graphs 441
 - graph elements 427
 - layout options reference 476
 - layout preferences 473
 - layout tools 463
 - Link Navigator 463
 - log file location 457
 - look and feel preferences 475
 - memory usage 464
 - menu reference 467
 - orphaned entities in 426
 - overview 420
 - preferences 472
 - printing graphs 447
 - Properties panel 436
 - Properties panel toolbar 465
 - Rule Execution History panel toolbar 465
 - Source panel 436
 - toolbar reference 462
 - View menu options 470
 - viewing or hiding panels 470
 - working directory depends on how started 457
 - zoom tools 463
- BusinessEvents Workbench palette
- purpose of 7
- BytesMessageSerializer 27

C

- cache (glossary) 519
- cache cluster (glossary) 519

- cache clusters
 - configuring 279
 - localport auto incrementing feature 294
 - multicast configuration 290
 - well-known addresses example 292
 - well-known-address configuration 292
- cache configuration file
 - setting for 309, 311
- cache mode 312
- cache mode (advanced caching option) 310
- cache mode (glossary) 519
- cache modes 282
- cache only (advanced caching option) 284
- cache only (glossary) 519
- cache plus memory (advanced caching option) 282
- cache plus memory (glossary) 520
- cache server (glossary) 519, 520
- cache servers
 - example properties for 320
- cache-based object management 255, 311, 366
 - advanced caching options 266
 - advanced caching options, understanding 282
 - advanced settings 311
 - agent group name (setting) 309, 326
 - and AIX, property required 279, 288
 - and message acknowledgement 26
 - backing store 265
 - cache configuration file (setting) 309, 311
 - cache mode (advanced caching option) 310
 - engine properties for 322
 - fault tolerance scenarios 241
 - overview 236
 - reliability 264
 - See also advanced caching options
 - understanding 264
- caching scheme (glossary) 520
- caching schemes
 - cache configuration file (setting) 309, 311
- call explicitly, check box 168
- Caller's Thread (preprocessor workers option) 365
- CallStateMachine 168
- changes only history policy 82, 93
- channel (glossary) 520

- channels 9
 - and multiple rule sets 9
 - overview 12
 - ChannelURI, variable 43
 - checkpoint (glossary) 520
 - checkpoint interval (persistence-based object management setting) 255
 - recommendations 257
 - circular layout 463, 477
 - class diagrams
 - viewing and hiding in Rule Analyzer 438, 439
 - class names, generated 343
 - ClientID (channel property) 36
 - cluster heartbeat 288
 - Collapse All Rule Nodes 463
 - command line
 - options for starting applications 381
 - running applications at 380
 - running Rule Analyzer from 434
 - running Rule Debugger from 451
 - command-line server management 355
 - command-line startup 356
 - configuring for 356
 - property configuration procedure for 368
 - complex element 59
 - complex event (glossary) 520
 - complex event processing
 - defined 2
 - requirements for 3
 - complex transitions 173
 - entering 174
 - components 6
 - composite state 170
 - composite state transitions 174, 174
 - concept (glossary) 520
 - concept model 4
 - concept reference 86, 86
 - concept reference (glossary) 520
 - concept relationships 85, 86
 - and advanced caching options 84, 282, 310
 - inheritance 84
 - concept view (glossary) 520
 - concept views
 - concept relationships 96
 - overview 12
 - concepts
 - and ontology functions 156
 - deleting explicitly 172
 - effect on rules 436
 - history parameter 80
 - how affected by rules 436
 - multiple checkbox for concept properties 80
 - overview 11
 - relationships 96
 - state modeler 164
 - concurrent states 171
 - conditions 137
 - in rule editor 152
 - overview 12
 - Conditions tab in Rule Debugger Context Window panel 431
 - Configuring 296, 296
 - conflict resolution cycle (glossary) 520
 - conflict resolution cycles 130
 - console information, in Rule Debugger 430
 - Console tab in Rule Debugger Debug Context Window panel 430
 - consumeEvent(event) 26
 - contained concept 85, 85
 - contained concept (glossary) 521
 - cross-subnet cache cluster configuration 292
 - custom recovery field 312
 - customer support xxxii
- ## D
- daemon (channel property) 35
 - database environment directory (persistence-based object management setting) 257
 - database location (persistence-based object management) 257
 - date functions 154
 - DateTime functions 154
 - deactivateRuleSet() 486
 - deactivateTraceRole() 487

- Debug Context Window panel [429](#)
 - Agenda tab [431](#)
 - Conditions tab [431](#)
 - Console tab [430](#)
 - Facts tab [432](#)
 - toolbar [466](#)
- debugging projects. See Rule Debugger [420](#)
- decision artifact (glossary) [521](#)
- Decision Manager [8](#)
- Decision Manager (glossary) [521](#)
- decision project (glossary) [521](#)
- decision table (glossary) [521](#)
- declaration [136](#)
 - rule editor [152](#)
- default events [42](#)
- default locations (various) [389](#)
- delete retracted objects from database (persistence-based object management setting) [256](#)
- deployment
 - hot [17](#), [17](#), [392](#)
 - in TIBCO Administrator [383](#)
 - overview [15](#), [16](#)
 - scripted [357](#)
- deploytime configuration
 - overview [16](#), [354](#)
- deserializer (glossary) [521](#)
- design requirements, when using advanced caching options [285](#)
- destination (glossary) [521](#)
- destinations [9](#)
 - and channels [24](#), [37](#)
 - default destination for event [57](#)
 - input [364](#)
- DestinationURI, variable [43](#)
- disabling hot deployment [395](#)
- distributed cache (glossary) [521](#)
- do not recover on restart (persistence-based object management setting) [257](#)
- domains [370](#)
- durable subscriber name, and special variables [43](#)
- DurableSubscriberName [43](#)

E

- EAR files [15](#)
 - deploying [383](#)
- Edit Default Configuration Properties dialog (for debugger profiles) [459](#)
- enabling hot deployment [395](#)
- end states [170](#)
- Engine category of advisory event [72](#)
- engine functions [154](#)
- engine log files, location [389](#)
- engine name, how determined at runtime [388](#)
- engine properties
 - adding to TIBCO Administrator [374](#)
 - See also Index of Engine Properties
 - setting in TIBCO Administrator [384](#)
- engine property files
 - enabling hot deployment in [395](#)
 - generated, location [389](#)
 - location of generated files [389](#)
 - properties for cache-based object management [322](#)
 - properties for persistence-based object management [260](#)
- Engine.FT.Weight [247](#)
- engine.primary.activated type of advisory event [72](#)
- Engine.Profiler.startCollectingToFile() [225](#)
- Engine.Profiler.stopCollecting() [226](#)
- EngineName, variable [43](#)
- engines
 - configuring for remote debugging [460](#)
 - startup options [381](#)
- enterprise archive resources [15](#)
- entities, setting cache mode options for [282](#)
- Entity (advanced caching setting) [309](#)
- entity (glossary) [521](#)
- Entity Chooser dialog [440](#)
 - using [442](#)
- entry actions [177](#)
- event (glossary) [521](#)
- event functions [154](#)
- event model [3](#)
- event preprocessors [145](#), [361](#)
- Event.routeTo [57](#)
- Event.sendEvent [57](#)

- events 9, 57
 - advisory 9, 72, 141
 - affected by rules 436
 - and ontology functions 156
 - default 42
 - default destination 57
 - effect on rules 435
 - expiry action 54
 - inherits from, setting 57
 - naming restrictions 58, 66
 - payload parameters 59
 - simple 9, 10
 - time 9
 - time events 10
 - types 9
- Exception category of advisory events 72
- execute() 488
- exit actions 177
- Expand All Rule Nodes 463
- expiry action 54
- exporting
 - graphs 445
 - state models to SVG 180

F

- Facts tab in Rule Debugger Context Window
 - panel 432
- facts, in Rule Debugger Conditions tab 431
- failover and failback
 - and distributed cache data 268
- fault tolerance
 - and cache-based object management 241
 - and in memory object management 239
 - and persistence-based object management 240, 253
 - configuring in TIBCO Administrator 385
 - hot deployment order in 392
 - not available for in memory object management 244
 - not available for in memory only advanced caching
 - option 273, 283
 - overview 246

- fault tolerance clusters
 - configuring 246
 - join time 246
- filtering entities in a graph 437
- filtering options in Rule Analyzer 439
- Find feature in BusinessEvents Tools 441
- Force GC 464
- forceOMCheckpoint() 489
- functions
 - action only 157
 - for use in rules 148, 153
 - mapper 139, 157
 - overview 13
 - See also rule functions
 - standard 154
 - state modeler functions 167
 - temporal 13, 160
 - tool tips for 157
 - types 13
 - types and usage 154

G

- garbage collection 464
- garbage collection guidelines 320
- generated class names 343
- generated class names, obtaining 350, 352
- getChannels() 490
- getDestinations() 491
- getEvent() 492
- GetExecInfo() 493
- getHostInformation() 494
- getInstance() 495
- getMemoryUsage() 496
- getNumberOfEvents() 497
- getNumberOfInstances() 498
- getOMInfo() 499
- getRuleSet() 500
- getRuleSets() 501
- getScorecard() 502
- getScorecards() 503
- getSessionInputDestinations() 504
- getSessions() 505

- getStatus() [506](#)
- getTotalNumberRulesFired() [507](#)
- getTraceSinks() [508](#)
- global variables
 - defining and using [358](#)
 - in the rule editor [138](#)
- graph elements in BusinessEvents Tools [427](#)
- graphs
 - exporting as images [445](#)
 - filtering entities in [437](#)
 - layout options [476](#)
 - layout preferences [473](#)
 - layout tools for [463](#)
 - printing [447](#)
- graphs of projects. See Rule Analyzer

H

- heap size [320](#)
- heartbeat, cluster [288](#)
- hierarchical layout [476](#)
- hierarchical layout with normal routing [463](#)
- hierarchical layout with orthogonal routing [463](#)
- history parameter
 - for concept properties [80](#)
 - history policy [82](#)
 - ring buffer [81](#)
- history policy, effect on rule evaluation [83](#)
- hot deployment [17](#), [392](#)
 - enabling and disabling [395](#)
 - limited modifications allowed with [392](#)
 - order of, in a fault tolerance group [392](#)
 - overview [17](#)
 - supported modifications [393](#)

I

- in memory (glossary) [522](#)

- in memory object management [255](#), [311](#), [366](#)
 - and message acknowledgement [26](#)
 - configuration procedure [245](#)
 - fault tolerance scenarios [239](#)
 - overview [235](#)
 - understanding [244](#)
- in memory only (advanced caching option) [283](#)
- in memory only (glossary) [522](#)
- incremental layout [463](#), [476](#)
- index (glossary) [524](#)
- indexes for property arrays [140](#)
- inference agent (glossary) [522](#)
- inheritance [84](#), [84](#)
- inheritance (glossary) [522](#)
- inherits from event setting [57](#)
- initialize_database.sql [336](#)
- input destinations
 - functions of [364](#)
- instance (glossary) [522](#)
- instances
 - aliases for [152](#)
 - concepts [11](#)
- INVOKE BW PROCESS type of advisory event [72](#)
- INVOKE BW PROCESS, type of BEBW advisory event [74](#), [213](#)
- IO functions [155](#)
- isTransacted (channel property) [36](#)

J

- Java Debug Interface (JDI) for remote debugging [460](#)
- JMS
 - channel naming restrictions [58](#), [66](#)
 - channels [12](#)
 - converting messages to non-default events [42](#)
 - durable subscriber name [43](#)
 - header properties [58](#), [66](#)
- JMS reconnect attempts [32](#)
- JMS reconnect codes [33](#)
- join time and fault tolerance priority [246](#)

K

key metrics. See scorecards

L

L1Cache 316

lambda transition (glossary) 522

layout options

 circular 477

 hierarchical 476

 incremental 476

 orthogonal 477

 symmetric 477

layout tools for BusinessEvents Tools 463

Link Navigator 463

Link Routing 464

links in BusinessEvents Tools, view path of 463

local channels 12

 maximum events in queue 41

local storage 343

localhost 294

localport 294

 auto incrementing feature 294

location of various files and directories 389

lockWM 211

log files

 adding backing store database entries to 351

 location 390

 location for BusinessEvents Tools 457

look and feel preferences in BusinessEvents Tools 475

M

main state machine 166

mapper functions 139, 157

math functions 155

maxActive property to define load balancing and fault tolerance 314

maximum (history parameter) 94

memory

 and cache only advanced caching option 284

 heap size guideline 320

 persistence property cache size 256

 tuning tips for persistence object management 253

 usage in BusinessEvents Tools 464

 using persistence-based object management as virtual memory 257

memory-based object management. See in memory object management

menu reference, BusinessEvents Tools menus 467

message acknowledgement with object management 26

message delivery messages 289

microagent, enabling 483

modifications supported for hot deployment 393

multicast

 and cache cluster configuration 290

multi-engine features 266

multiple rule sessions

 in Rule Debugger 429

 object management and fault tolerance scenarios 239

multiplicity (glossary) 522

N

namespace 42

network (channel property) 35

network traffic with multicast 290

nodes (JVMs) 239

number functions 155

O

object details, in Rule Debugger Agenda tab 431

- object management
 - advanced caching options [84, 266, 282](#)
 - and fault tolerance scenarios [239](#)
 - and message acknowledgement [26](#)
 - backing store [265](#)
 - functions of and types [255, 311, 366](#)
 - in memory object management configuration [245](#)
 - options introduced [5](#)
 - overview [234](#)
 - persistence-based object management
 - configuration [254](#)
 - See also in memory object management, persistence-based object management and cache-based object management
- object management (glossary) [522](#)
- object management and fault tolerance scenarios
 - cache-based object management [241](#)
 - in memory object management [239](#)
 - persistence-based object management [240](#)
- Object Management tab
 - cache-based object management settings (advanced) [309](#)
- ontology functions [13](#)
 - location of [156](#)
 - purpose and types of [156](#)
- ontology graphs [420](#)
- orphaned entities in BusinessEvents Tools [426](#)
- Orthogonal Layout [463](#)
- orthogonal layout [477](#)
- outstanding database operations (setting) [256](#)

P

- palettes
 - BusinessEvents Workbench [7](#)
 - TIBCO BusinessEvents activities [184](#)
 - TIBCO BusinessEvents State Modeler palette [178](#)
- payload (glossary) [523](#)
- payload parameters
 - cardinality [59](#)
 - complex element [59](#)
- payloads [59](#)
 - all, parameter [60](#)
 - attribute of type, parameter [60](#)
 - choice, parameter [60](#)
 - element of type, parameter [59](#)
 - including in TIBCO Rendezvous messages [30](#)
 - sequence, parameter [60](#)
 - validation, parameter [61](#)
 - XML element reference, parameter [59](#)
 - XML group reference, parameter [60](#)
- performance tuning
 - for query agents [327](#)
- persistence (glossary) [523](#)
- persistence-based object management [255, 311, 366](#)
 - and fault tolerance [253](#)
 - and message acknowledgement [26](#)
 - checkpoint interval (setting) [255](#)
 - configuration procedure [254](#)
 - database environment directory (setting) [257](#)
 - delete retracted objects from database (setting) [256](#)
 - do not recover on restart (setting) [257](#)
 - engine properties for [260](#)
 - fault tolerance scenarios [240](#)
 - overview [236](#)
 - property cache size (setting) [256](#)
 - understanding [252](#)
- port conflicts, avoiding [294](#)
- preprocessors [145, 361](#)
 - adding [361](#)
 - removing [361](#)
 - workers [365](#)
- printing graphs [447](#)
- profiles
 - configuring in Rule Debugger [454](#)
 - creating, editing, deleting in Rule Debugger [455](#)
 - default settings for [458](#)
 - in Rule Debugger [450](#)
 - XML file for [459](#)
- Properties panel toolbar, in BusinessEvents Tools [465](#)
- Properties panel, in BusinessEvents Tools [436](#)
- properties, adding to TIBCO Administrator [374](#)
- property array indexes [140](#)
- property arrays [80](#)
- property cache size (persistence-based object management setting) [256](#)

property files, supplementary [368](#)
 ProviderURL (channel property) [36](#)

Q

query agent (glossary) [523](#)
 query agents
 performance tuning [327](#)
 query language [14](#)
 queue
 local channel [41](#)

R

reconnect to JMS, attempts [32](#)
 reconnect to JMS, which messages [33](#)
 reconnectChannels() [509](#)
 reevaluating rules [83](#)
 relationships between concepts. See concept views
 reload (Rule Analyzer) [462](#)
 remote debugging
 configuring engine for [460](#)
 Java Debug Interface and [460](#)
 remote debugging, profile settings [456](#)
 remote engine configuration for Rule Debugger [450](#)
 Rendezvous. See TIBCO Rendezvous
 repository locator string [389](#)
 resetTotalNumberRulesFired() [510](#)
 resumeChannels() [511](#)
 resumeDestinations() [512](#)
 Rete network [19](#)
 Rete network (glossary) [523](#)
 retraction (glossary) [523](#)
 ring buffer [81](#)
 RMS (glossary) [523](#)
 RTC [130](#)
 rule (glossary) [523](#)
 rule agenda [19](#)
 rule agenda, in Rule Debugger [429](#)

Rule Analyzer

Analyzer/Dependency panel [425](#)
 benefits [421](#)
 class diagram and state machine options [438, 439](#)
 filter parameters [439](#)
 filtering and viewing options [437](#)
 filtering entities in [439](#)
 overview [420](#)
 Overview panel [424](#)
 Properties panel [424](#)
 reload option [462](#)
 running from the command line [434](#)
 running from TIBCO Designer [434](#)
 See also BusinessEvents Tools
 Source panel [425](#)
 state machines in [439](#)
 user interface overview [423](#)
 using [435](#)
 using find, filter, export, and printing features [436](#)
 rule based time events [10](#)

Rule Debugger

- Analyzer/Dependency panel 425
 - benefits 421
 - configuring engine for remote debugging 460
 - Debug Context Window 429
 - Debug Context Window, Agenda tab 431
 - Debug Context Window, Conditions tab 431
 - Debug Context Window, Console tab 430
 - Debug Context Window, Facts tab 432
 - multiple rule sessions 429
 - overview 420
 - profile default settings 458
 - profile settings 456
 - profiles 450
 - profiles XML file 459
 - profiles, configuring 454
 - profiles, creating, editing, deleting 455
 - providing sample data for 450
 - remote debugging configuration 456
 - Rule Execution History panel 429
 - run mode 429, 429
 - running against a remote engine 450
 - running from the command line 451
 - running from TIBCO Designer 452
 - See also BusinessEvents Tools
 - setup actions 450
 - starting a session 453
 - step mode 428
 - user interface overview 428
- rule editor
- actions 137
 - alias 152
 - conditions 137
 - declaration 136
 - global variables in 138
 - search utility 141
- rule engines
- conflict resolution cycles 130
- rule evaluation 19
- Rule Execution History panel 429
- Rule Execution History panel toolbar, in BusinessEvents Tools 465

- rule functions 4
 - for startup and shutdown 366
 - ontology functions 156
 - purpose and usage of 129
 - virtual 129
 - rule model 4
 - rule session (glossary) 524
 - rule sessions
 - See also BusinessEvents archive resources
 - rule sets
 - functions of 363
 - overview 12
 - rules
 - actions 12, 152
 - affected by events 435
 - and advisory events 72
 - and functions 148, 153
 - and ontology functions 156
 - conditions 12, 152
 - declaration 152
 - effect on concepts and scorecards 436
 - effect on events 436
 - how affected by concepts and scorecards 436
 - overview 12
 - reevaluating 83
 - rule resources 12
 - writing for advanced caching options 285
- Rules Management Server 8
- ruleset model 4
- run mode 429

S

- schedule a checkpoint if outstanding DB ops greater than (setting) 256
- scorecards
 - effect on rules 436
 - how affected by rules 436
 - overview 12
- scripted deployment 357
- search utility 141
- self transitions 173
- sending to another application 57

sense and respond [2](#)
 serializer (glossary) [524](#)
 serializers [25](#)
 service (channel property) [35](#)
 shared archive resources [15](#)
 Shared Queue and Threads (preprocessor workers option) [365](#)
 Shared Queue and Threads setting [367](#)
 simple event (glossary) [524](#)
 simple event definition (glossary) [524](#)
 simple events [9, 10](#)
 simple state [170](#)
 situation awareness [2](#)
 socket exception with AIX and cache [279, 288](#)
 Source panel, in BusinessEvents Tools [436](#)
 standard functions [13, 154](#)
 categories of [154](#)
 start states [170](#)
 configuring [171](#)
 StartFileBasedProfiler() [227](#)
 startup options, engine [381](#)
 startup. See command-line startup. See also deployment.
 State Machine resource [166](#)
 state machines [168](#)
 call explicitly, checkbox [168](#)
 configuring [169](#)
 explicitly deleting concepts from [172](#)
 functions [167](#)
 graphs of
 inheritance [167](#)
 main [166](#)
 See also state modeler
 viewing in Rule Analyzer [439](#)
 state model [4](#)
 state modeler
 example [165](#)
 functions [167](#)
 overview [13, 164](#)
 purpose of [7](#)
 See also states
 See also transitions
 State Machine resource [166](#)
 State Modeler palette [178](#)

state models
 exporting [180](#)
 states [170](#)
 transitions [173](#)
 stateful rule engine [4](#)
 states
 composite [170](#)
 concurrent [171, 175](#)
 entry and exit actions [177](#)
 in a state model [170](#)
 simple [170](#)
 start and end [170](#)
 timeouts [171](#)
 transitions [173](#)
 step mode [428, 429](#)
 stopApplicationInstance() [513](#)
 StopFileBasedProfiler() [227](#)
 string functions [155](#)
 subnets
 and configuration for well-known addresses [292](#)
 supplementary property files [368](#)
 support, contacting [xxxii](#)
 supported modifications for hot deployment [393](#)
 suspendChannels() [514](#)
 suspendDestinations() [515](#)
 SVG [180](#)
 exporting a state model to [180](#)
 symmetric layout [463, 477](#)
 system functions [155](#)

T

tangosol.coherence.localport.adjust [294](#)
 task summary [20](#)
 technical support [xxxii](#)
 temporal functions [155](#)
 arguments [160](#)
 overview [13](#)
 TextMessageSerializer [28](#)
 TIBCO administration domains [370](#)

TIBCO Administrator

- adding engine properties to [374](#)
- and server management [355](#)
- configuring deploy-time properties in [370](#)
- deployment procedure [383](#)
- deployment using [356](#)
- deploytime configuration in [356](#)
- deploytime configuration procedure [371](#)
- enabling hot deployment in [395](#)
- location of generated property files [389](#)
- purpose of [8](#)
- rule session configuration in [370](#)

TIBCO BusinessEvents terminology [9](#)

TIBCO BusinessWorks

- connecting to [7](#)
- process definitions, and TIBCO BusinessEvents activities [184](#)

TIBCO Designer

- purpose of [6](#)
- running Rule Analyzer from [434](#)
- running Rule Debugger from [452](#)

TIBCO Hawk

- application management interface [387](#)
- enabling microagent [483](#)
- list of methods [481](#)
- methods overview [483](#)

TIBCO Rendezvous

- channels [12](#)
- converting messages to non-default events [42](#)
- including a payload in a message [30](#)
- tracing properties [377](#)

TIBCO_HOME [xxix](#)tibcr [389](#)time events [9, 10](#)

- and ontology functions [156](#)
- rule based [10](#)
- time-interval based [10](#)

time to live

- example [53](#)

timeouts

- local channels [41](#)
- state machines and states [167](#)
- states [171](#)

tool tips, for functions [157](#)

tracing

- enabling tracing to terminal [378](#)
- enabling tracing to TIBCO Rendezvous [377](#)

track and trace [2](#)transitions [173, 174](#)

- complex [173](#)
- configuring [176](#)
- rules for concurrent states [175](#)
- self [173](#)

type (object management setting) [255, 311](#)**U**UML (glossary) [524](#)URI (glossary) [524](#)**V**variables for use with durable subscriber name [43](#)virtual rule function (glossary) [525](#)virtual rule functions [129](#)VRF (Virtual Rule Function) functions [155](#)**W**

well-known addresses

- and cache cluster configuration [292](#)
- example [292](#)

Workbench (glossary) [525](#)

Workbench palette. See BusinessEvents Workbench palette

workers (preprocessor) [365](#)working directory [389](#)working memory (glossary) [525](#)

X

XPath functions [155](#)

XSLT template [139](#), [157](#)