



# Getting Started with Cogility Studio

Copyright © 2001-2011 Cogility Software Corporation.

All Rights Reserved.

*Getting Started with Cogility Studio* is copyrighted and all rights are reserved. Information in this document is subject to change without notice and does not represent a commitment on the part of Cogility Software Corporation. The document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Cogility Software Corporation.

Document version number 7.0

Cogility is a trademark of Cogility Software Corporation. Other brands and their products are trademarks of their respective holders and should be noted as such.

Cogility Software Corporation  
111 N. Market St. #888  
San Jose, CA 95113

[support@cogility.com](mailto:support@cogility.com)

Printed in the United States of America.

The software described in this book is furnished under a license agreement and may be used only in accordance with the terms of the agreement.



### Preface

About this book .....	5
Cogility Studio documentation .....	5

### Chapter 2 Set up a Cogility project

Installation .....	7
Project configuration .....	7
Setting up a basic project .....	8
Authoring repository .....	9
Configuring an authoring repository .....	9
Loading the authoring repository .....	10

### Chapter 3 Model development

Model container .....	11
Creating a model .....	11
Information model .....	12
Class diagrams .....	12
Class attributes .....	13
Class associations .....	13
Behavior model .....	13
Events .....	14
Messages .....	15
Web services .....	16
Transformations .....	16
Business logic .....	16
Change management .....	17

### Chapter 4 Deployment & execution

Deployment model .....	19
Push .....	20
Cogility Insight .....	21
Cogility Message Traffic Viewer .....	21
Cogility Web Service Exerciser .....	22
Cogility WatchDog .....	23





# Preface

---

Cogility Studio provides the tools to create a model for an enterprise information system, and deploy it as a J2EE application. The Cogility Studio documentation provides support for this endeavor.

## About this book

*Getting Started with Cogility Studio* provides a quick primer for those new to Cogility Studio and enterprise application integration modeling in general. It surveys current application integration technologies and describes the business problem that Cogility Studio solves. It also provides an overview of enterprise application integration modeling in Cogility Studio with a brief description of each phase in the modeling process.

## Cogility Studio documentation

Cogility Studio comes with several volumes of documentation to help you.

- *Installing and Configuring Cogility Studio* describes the installation and configuration of your application server, database and Cogility Studio.
- *Getting Started with Cogility Studio* is a brief overview of Cogility Studio.
- *Modeling with Cogility Studio* tells you how to build a model-driven enterprise application using Cogility Modeler and associated tools.
- *Using Actions in Cogility Studio* provides a reference to modeling action semantics for use with Cogility Studio.
- *Change Management in Cogility Studio* describes the change management system for models and model artifacts.
- *Model Deployment & Execution in Cogility Studio* is a guide to application monitoring, maintenance and migration, and describes the utilities that you can use to test and monitor your model deployed as a enterprise application.

Several white papers on various topics are also available to further your understanding of enterprise application integration, business process management, model driven architecture and other related topics. See the Cogility website:

<http://www.cogility.com>.





# Set up a Cogility project

---

This section briefly describes how to set up Cogility Studio before you start using the various tools. You need to install the software and create a project and an authoring repository. You may create several projects, perhaps one for each model you are developing, and for that project, you keep all configuration files, source code and libraries in a defined location. When you nominate a project as the active project, the tools of Cogility Studio automatically refer to the active project's files for information such as the location of the authoring repository and other configuration information. See [“Project Configuration Editor” on page 51](#) of the guide, *Model Deployment & Execution in Cogility Studio*.

## Installation

You will need a J2EE application server and database to run Cogility Studio. You first install Cogility Studio, then install (or create instances of) the database and application server. Typically, you install Cogility Studio from a CD. The Cogility Studio installation, application server and database configuration instructions are described the guide, *Installing and Configuring Cogility Studio*. Start with [“Cogility Studio” on page 7](#).

The license to run Cogility Studio is provided separately from the installation CD. With the license file, you will receive instructions for installing the license.

## Project configuration

Following installation, your first task is to establish a project directory where you will keep the files specific to your modeling project. Generally, one model or a set of related models may pertain to a project. Associated with those models are specific libraries, source files and configuration files held in the project directory. You use Cogility Project Configuration Editor to create each project and manage the project files. See [“Project Configuration Editor” on page 51](#) of the guide, *Model Deployment & Execution in Cogility Studio*.

This section describes setting up a basic project, creating a project-specific configuration file for the model's authoring repository and making the project the active project.

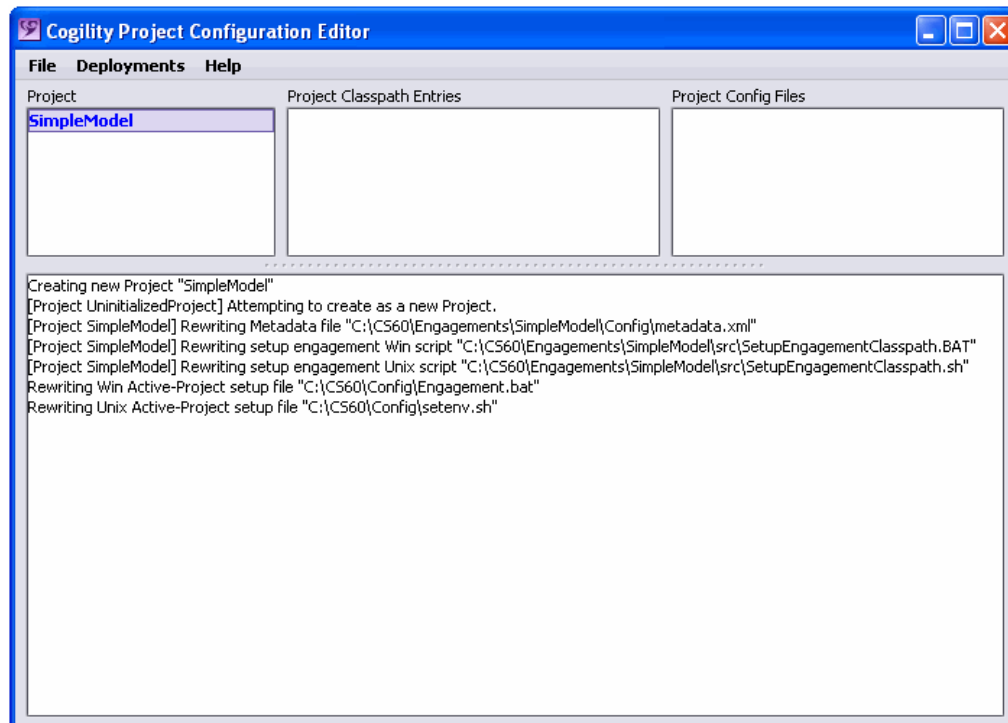


Figure 2-1: An active project in Cogility Project Configuration Editor

## Setting up a basic project

### To set up a basic project:

1. From the **Start** menu, select **All Programs > Cogility Studio > Project Configuration Editor**.

See [“Running Cogility Project Configuration Editor” on page 51](#) of the guide, *Model Deployment & Execution in Cogility Studio*.

2. Create a new project.

- a. In the **Project** field, right-click and select **Create New Project...**

- b. In the input dialog's name field, enter a project name and click **OK**.

Do not include spaces in the project name. See [“Creating a new project” on page 54](#) of the guide, *Model Deployment & Execution in Cogility Studio*.

3. In the **Project** field, select the project, right-click and select **Make <project name> the Active Project**.

See [“Rewriting configuration artifacts” on page 55](#) of the guide, *Model Deployment & Execution in Cogility Studio*. Your Cogility Project Configuration Editor console should look like the one shown in [Figure 2-1 on page 8](#). Next, you set up an authoring repository to work with the model for this project. See [“Authoring repository” on page 9](#).



## Authoring repository

The authoring repository is defined by a set of files on your system that hold the versions of your model artifacts during development. In this section, you establish the authoring repository with the Load Persistent Repository utility. See [“Authoring repository” on page 19](#) of the guide, *Modeling with Cogility Studio*.



Figure 2-2: The Load Persistent Repository utility

## Configuring an authoring repository

### To configure an authoring repository for your project:

1. Using a text editor such as Notepad, create a text file with the following:

```
com.ohana.object.persistence.db.count=1
com.ohana.object.persistence.db.connection=1

com.ohana.object.persistence.db.class.1=files.File
com.ohana.object.persistence.db.interface.1=FileRepository
com.ohana.object.persistence.db.servername.1=localhost
com.ohana.object.persistence.db.databasesname.1=%DCHOME%/Engagements/
<project>/Repository
com.ohana.object.persistence.db.username.1=
com.ohana.object.persistence.db.password.1=
```

2. In the text above, substitute `<project>` with the name of the project you created in [“Setting up a basic project” on page 8](#).
3. Save the file with the filename **Repository.txt**.  
You can save the file anywhere on your system, but for it to take affect, it must be copied into the Config folder of your project. This can be done using the editor.
4. In Cogility Project Configuration Editor, import the configuration file into your project.
  - a. In the **Project** field, select the project.
  - b. In the **Project Config Files** field, right-click and select **Import New Config File(s)**.
  - c. Navigate to the location of the **Repository.txt** file, select it and click **Open**.  
The file is imported into the `%DCHOME%\Engagements\<project>\Config\Files` directory. For more information, see [“Configuration files” on page 64](#) of the guide, *Model Deployment & Execution in Cogility Studio*
5. Create a directory (or folder) named **Repository** in the directory, `%DCHOME%\Engagements\<project>` where `%DCHOME%` is the location where Cogility Studio is installed, and `<project>` is the name of the project you created in [“Setting up a basic project” on page 8](#).

The directory you are creating maps to the value specified by the following configuration parameter:

```
com.ohana.object.persistence.db.databasesname.1=
```

For example, C:\CS60\Engagements\SimpleModel\Repository.

You now have a repository configured for the project you created. Next, you load that repository with the files required for use with the model.

## Loading the authoring repository

### To load the authoring repository:

1. From the **Start** menu, select **All Programs > Cogility Studio > Cogility Modeler > Load Persistent Repository**.
2. From the drop-down list next to **Loading Into**, select the repository you wish to load into the current context.

There should be only one selection, that for the authoring repository you configured in [“Configuring an authoring repository” on page 9](#). Cogility Studio looks for the repository location based on the configuration of the active project. If there were more than one repository configured, you would select a repository from the list. Note that the repository is empty at this point, no model has been created for it yet.

3. Click **Load**.

The utility creates the files that comprise the authoring repository. Your Load Persistent Repository dialog should look similar to the one in [Figure 2-2 on page 9](#). You now have an authoring repository established for use with the model in your project. Next, you begin developing the model. See [“Model development” on page 11](#).



# Model development

---

A model in Cogility Studio represents an enterprise system in terms of its component parts. Programatic objects like events, messages, class instances, and so on have their representation as artifacts in a model. Cogility Modeler provides the integrated development environment for building a model using these artifacts, then deploying the model as a run-time application composed of program objects based on the model artifacts.

In the last sectionw, “[Set up a Cogility project](#)” on page 7, you created a project and configured an authoring repository for the model in the project. This section describes some of the basics of model development using Cogility Modeler. For more information see “[Introduction to Cogility Modeler](#)” on page 15 of the guide, *Modeling with Cogility Studio*.

## Model container


To get started, you first run Cogility Modeler, select the repository, and begin adding model artifacts to the model. The model container is the first of these artifacts. A model is a schematic representation of an application that performs various functions between and within the modeled systems. The functional areas of a model, the information model, the message model, the web services model, the behavior model and the transformation model are described following this sections.

## Creating a model

**To create a model in Cogility Modeler:**

1. From the Windows **Start** menu, select **All Programs > Cogility Studio > Cogility Modeler > Cogility Modeler Login**.

If you have continued from the previous section, “[Set up a Cogility project](#)” on page 7, you are logging into an empty context, and you don’t need to specify the repository or the model version. Cogility Modeler opens with the default deployment model shown in the tree view. For more information, see “[Configuration management](#)” on page 16 of the guide, *Modeling with Cogility Studio*.

2. In Cogility Modeler, click the **Add a Model** button , or from the **Selection menu**, select **Domain Modeling > Add a Model**.

The model container is the top level model artifact. See “[Model container](#)” on page 21 of the guide, *Modeling with Cogility Studio*.

---

**Note:** If you have the DeploymentModel object selected, the **Add a Model** button will not be enabled. Hold the Control key and click on the DeploymentModel object to de-select it.

---

3. In the editor view, under the **Model** tab, in the **Name** field, enter the model container name.

The name must begin with an alphabetic character and can contain any combination of alphanumeric characters or underscores. The name cannot contain spaces. For this tutorial, the model container is named SimpleModel.

You now have a model container in which you can create model artifacts.

## Information model

The information model describes the data that interact across the enterprise. The common information model, for all data objects interacting with the distinct, integrated systems, is the master information model (MIM). Each integrated system has its own information model, described in a distinct information model (DIM). While a DIM is optional, all models must have a MIM. See [“Information model” on page 29](#) of the guide, *Modeling with Cogility Studio*.

Also, a model must have a MIM class. The class describes a business object such as a Customer or Product. Classes may have attributes and operations. Define a class for each type of business object in your model. For more information, see [“Classes” on page 33](#) of the guide, *Modeling with Cogility Studio*.

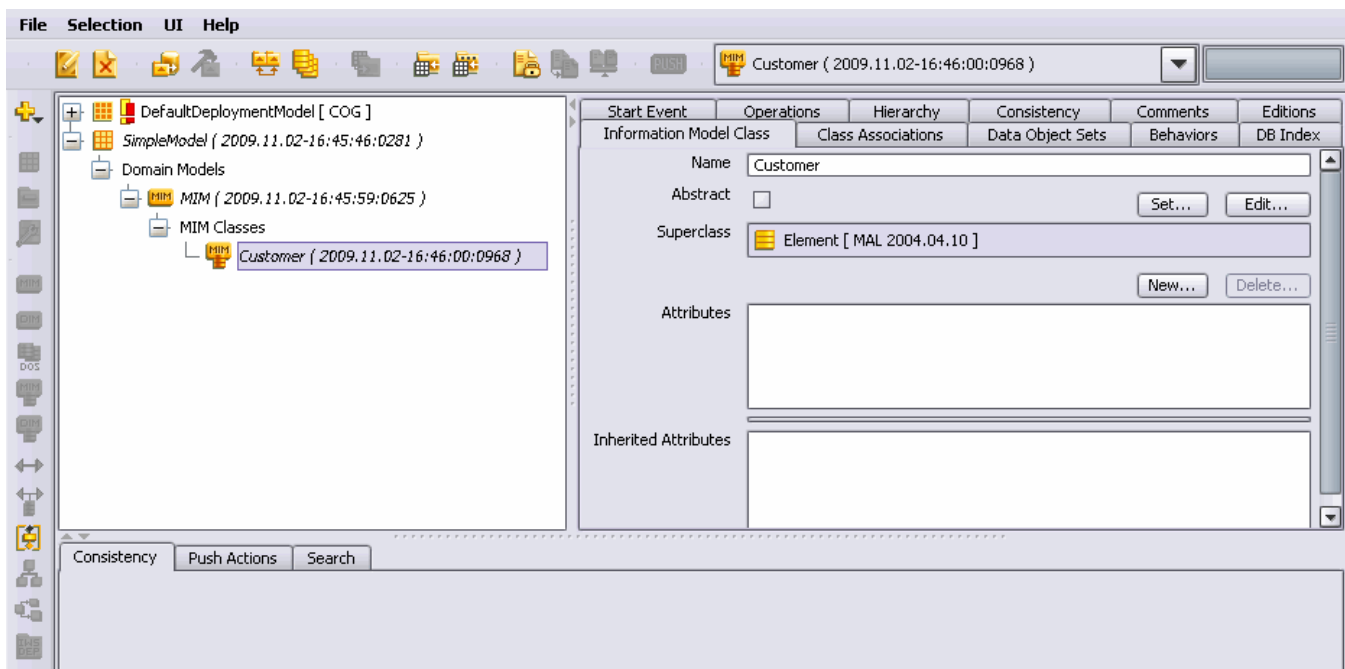


Figure 3-3: The Customer class in the MIM

## Class diagrams

Class diagrams provide a graphical representation of an information model. See [“Class diagrams” on page 49](#) of the guide, *Modeling with Cogility Studio*. The class diagram can include classes, their attributes and operations, and their relationships with other classes in your model. The relationship may take the form of a superclass to subclass, or an association between classes. Associations describe the role of each class in the relationship and the number of instances of each role. For example, a Customer may have many Orders, as shown in [“A class diagram showing a one-to-many relationship between two classes” on page 13](#).

## Class attributes

Classes usually have attributes. At run time, it is the values of these attributes that define the instances of the class type. For more information, see [“Attributes” on page 35](#) of the guide, *Modeling with Cogility Studio*. The class diagram shown in [Figure 3-4 on page 13](#) shows the Customer class with the customerID, firstName and lastName attributes.

## Class associations

Two class artifacts are related to each other by means of an association artifact. Associations describe the roles of each class and the number of instances which can participate in the relationship between the classes. For more information, see [“Associations” on page 51](#) of the guide, *Modeling with Cogility Studio*.

For this example, the Customer class is the source of the association, and the Order class is the target. One Customer may have many (any number of) Orders. The class diagram illustrates this as follows.

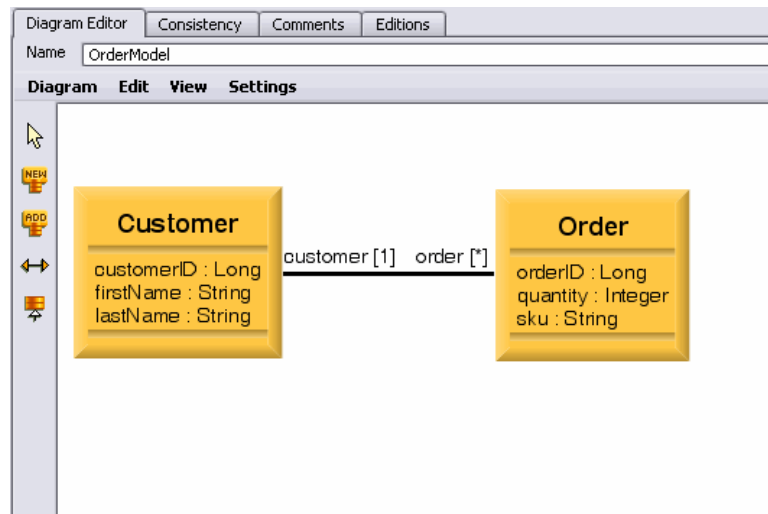


Figure 3-4: A class diagram showing a one-to-many relationship between two classes

## Behavior model

The behavior model describes business processes that involve business objects across the enterprise. Central to the behavior model is a state machine artifact that describes a business object's interactions, and conditions for those interactions with other model artifacts. A state machine is

based on the UML state diagram, indicating an object's state in a business process. See *“Behavior model” on page 107* of the guide, *Modeling with Cogility Studio*.

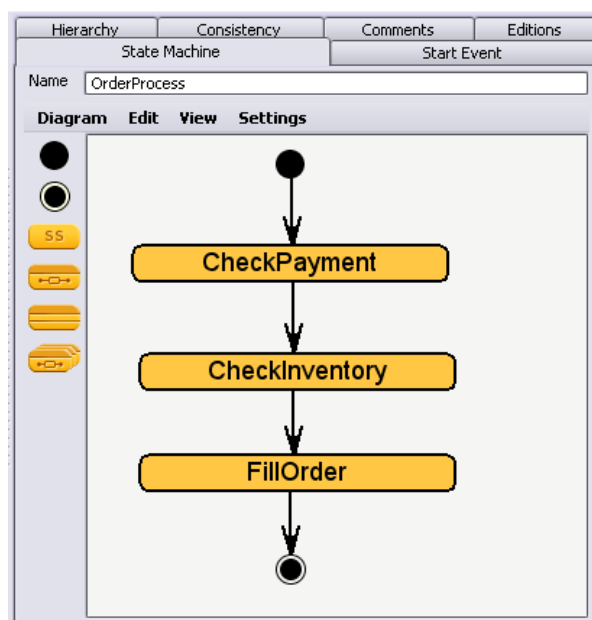


Figure 3-5: State machine

## Events

The state machine described in *“Behavior model” on page 13* requires an event to initiate the business process. Such an event is called a start event. A business process may also generate events. Both the data used as an input to the business process and the data generated by a business process populate events. For example, an event passes new customer information contained in a message to the state machine, which generates another event containing additional information. For more

information, see [“Events” on page 80](#) of the guide, *Modeling with Cogility Studio*. The following figure illustrates an event that starts a business process.

The screenshot shows the 'Event' configuration window in Cogility Studio. The 'Name' field is 'NewOrder\_Event'. The 'Is Internal Event' checkbox is checked. The 'Abstract' checkbox is unchecked. The 'Superclass' is 'Element [ MAL 2004.04.10 ]'. The 'Attributes' section lists two attributes: 'quantity' (type Integer) and 'sku' (type String). The 'Attribute' tab is selected, showing details for the 'quantity' attribute: Name 'quantity', Type 'Integer [ MAL 2004.04.10 ]', Size '0', and 'Is Required' unchecked. The 'Inherited Attributes' section is empty.

Figure 3-6: Start event

## Messages

Message artifacts in Cogility Studio represent either JMS messages, which are converted to or from events or SOAP messages which define the interface to web services. See [“Messages” on page 101](#) of the guide, *Modeling with Cogility Studio*. A JMS message containing data from an external system may be modeled as a message artifact and converted to an event. Likewise, events may populated messages via the conversion process. See [“Conversions” on page 89](#) of the guide, *Modeling with Cogility Studio*.

The screenshot shows the 'Message' configuration window in Cogility Studio. The 'Name' field is 'arg'. The 'Superclass' is 'Element [ MAL 2004.04.10 ]'. The 'Attributes' section lists two attributes: 'quantity' (type Integer) and 'sku' (type String). The 'Attribute' tab is selected, showing details for the 'sku' attribute: Name 'sku', Type 'String [ MAL 2004.04.10 ]', Size '0', and 'Is Required' unchecked. The 'Inherited Attributes' section is empty.

Figure 3-7: Message artifact

## Web services

In Cogility Studio, an outbound web service is one that the model application calls to invoke some external functionality, such as looking up a zip code or verifying an e-mail address. An inbound web service is one that users of the model application call to perform model functionality, such as submitting an order or creating a new customer. Both are represented with web service artifacts. See [“Web Services Model” on page 199](#) of the guide, *Modeling with Cogility Studio*.

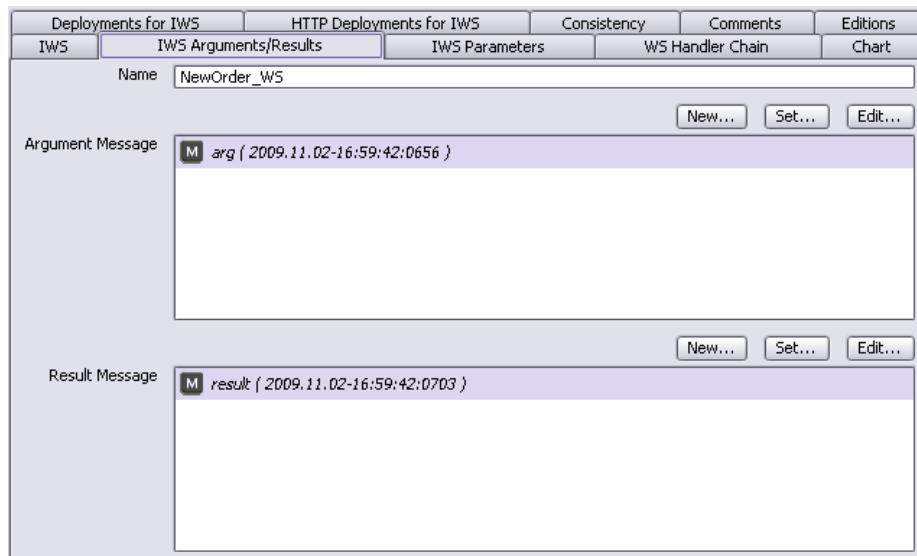


Figure 3-8: Web service artifact

## Transformations

Disparate systems may represent data differently: a date may be formatted 12/27/2006 on one system, 2006.12.27 on another. Data transformation artifacts provide for the homogenization of such data in the model and its transmission between disparate systems. These artifacts may be reused and combined in different transformation chains. See [“Transformation model” on page 143](#) of the guide, *Modeling with Cogility Studio*.

## Business logic

Within the states and transitions of a state machine, within the actions of a web services, and within transformations you define the business logic using action semantics, a scripting language similar to



Javascript, but with elements of an object-oriented language like Java. See [“Action semantics” on page 9](#) of the guide, *Using Actions in Cogility Studio*.

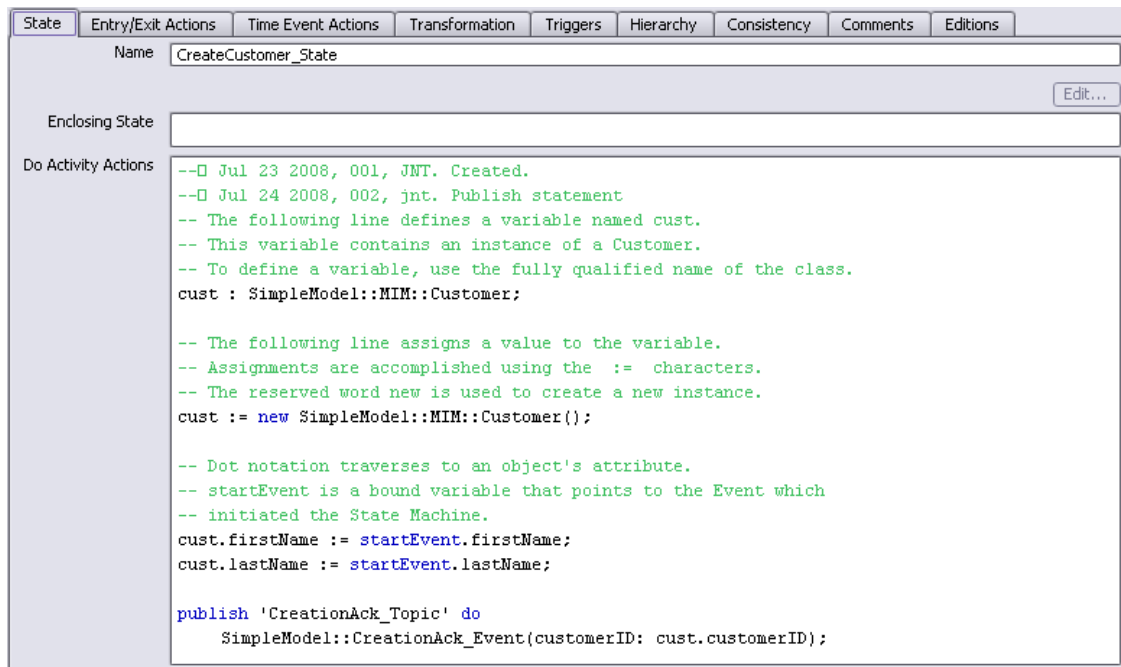


Figure 3-9: Business logic defined in the Do Activity of a state machine

## Change management

To maintain model artifact versions in the authoring repository, Cogility Modeler includes a change management facility. It provides for comparison of model objects, including three-way comparisons, and difference merging. See [“Change management” on page 5](#) of the guide, *Change Management in Cogility Studio*.





## Deployment & execution

---

Once you have finished modeling your enterprise application in Cogility Modeler, and it is free of inconsistencies, you deploy the model as a run-time application on a J2EE application server. The deployment process creates the required database structures, populated with application metadata, as well as deployment descriptors, in order to execute the modeled application on a J2EE application server. The application server and the data base for the run time repository are modeled as deployment model artifacts, thereby allowing you to choose from several application server and database combinations. Once deployed, the model application may be monitored with Cogility Insight. Also, other Cogility tools such as Cogility Action Pad, Cogility Message Traffic Viewer, and Cogility Web Service Exerciser let you run business logic against the application and work with the run-time repository, publish JMS messages, and invoke model web services. See the guide, *Model Deployment & Execution in Cogility Studio*.

### Deployment model

Within a deployment model there may be several application server and database artifacts, each representing any of the supported servers and databases. For example, your deployment model may have artifacts for each of the IBM WebSphere, BEA WebLogic and JBoss application servers, as well as one for the Oracle database. When you deploy your model application, you choose the deployment with a particular application server and database combination. See “[Configuration Parameters](#)” on [page 25](#) of the guide *Modeling with Cogility Studio*.

**Note:** A Deployment object may contain one or more ApplicationServer objects and a single Database object.

The screenshot shows the 'Deployment' configuration window in Cogility Studio. It is divided into several sections:

- Deployment Tab:** Contains fields for 'Name' (WebLogic10\_localhostOracle), buttons for 'New...', 'Add...', 'Edit...', 'Remove', 'Move Up', 'Move Down', and 'Enable/Disable', and a list of 'Application Server Instances' (0) WebLogic10 [ COG ].
- Database Section:** Includes a checkbox 'If an application server instance push fails, do not continue with the remaining instances', buttons for 'New...', 'Set...', and 'Edit...', and a 'Database' field (localhostOracle [ COG ]).
- Database Tab:** Contains fields for 'Name' (localhostOracle), 'User ID' (pear), 'Password' (\*\*\*\*), and 'Connect String' (jdbc:oracle:thin:@localhost:1521:ord).
- Application Server - General Tab:** Contains fields for 'Name' (WebLogic10), 'Type' (WebLogic v9/v10), 'Enabled for Push' (checked), 'Target Host' (localhost), 'Connection URI', 'JNDI Initial Context Factory' (weblogic.jndi.WLInitialContextFactory), 'JNDI Provider URL' (t3://<targethost>:7001), 'JNDI URL Pkg Prefixes', 'HTTP Port' (7001), 'Webservice Endpoint Prefix' (http://<targethost>:7001), 'Username' (weblogic), and 'Password'.
- Remote Machine Actions:** A section titled 'If push to a remote machine, also do these actions:' with two checked checkboxes: 'Update active database & application server information to configuration files on the remote host' and 'Update active application server information on the local machine'.

Figure 4-10: Deployment model

## Push

Deployment consists of pushing the model application to the target database and application server. This is the automated process whereby the application artifacts are created for the designated database and J2EE environment. See [“Deployment” on page 26](#) of the guide, *Modeling with Cogility Studio*. To push a model, you select the target deployment model. See [“Deployment model” on page 19](#) of the guide, *Modeling with Cogility Studio*. The push process is straight forward; see the guide,

*Modeling with Cogility Studio.* However, there are variations on the process that save you time and effort when you need to push only specific model artifacts or push to only the application server or database. See “[Push modes](#)” on page 27 of the guide, *Model Deployment & Execution in Cogility Studio*.

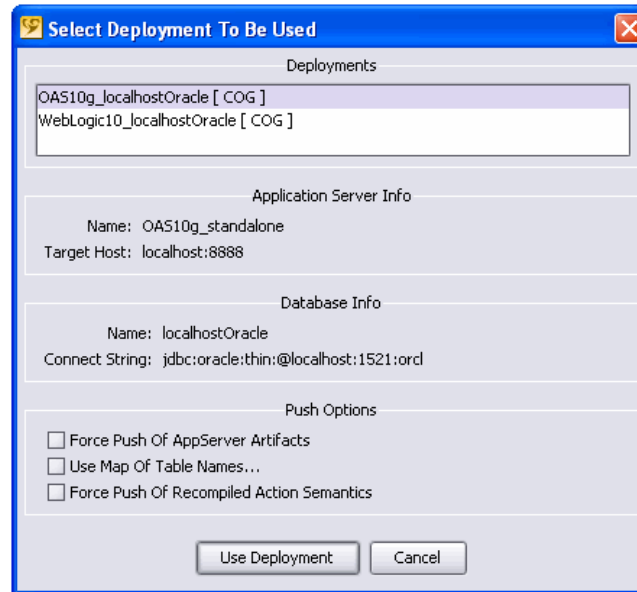


Figure 4-11: Pushing the model into execution

## Cogility Insight

Deployed with the model on the application server, Cogility Insight provides a web browser-based application monitoring utility. Your model application’s business processes, run-time data and configuration settings may be viewed or accessed through this interface. See “[Cogility Insight](#)” on page 99 of the guide, *Model Deployment & Execution in Cogility Studio*.

## Cogility Message Traffic Viewer

For testing and simulation scenarios, when you want to see how your model application responds to JMS messages, you can use Cogility Message Traffic Viewer. For a production scenario, the publication of JMS messages occurs between external processes and the model application or between multiple objects within the model application. Before you deploy a model into actual production, you can test it without connections to the external, integrated systems by simulating JMS

messages with this utility. See [“JMS Message Utilities”](#) on page 129 of the guide, *Model Deployment & Execution in Cogility Studio*.

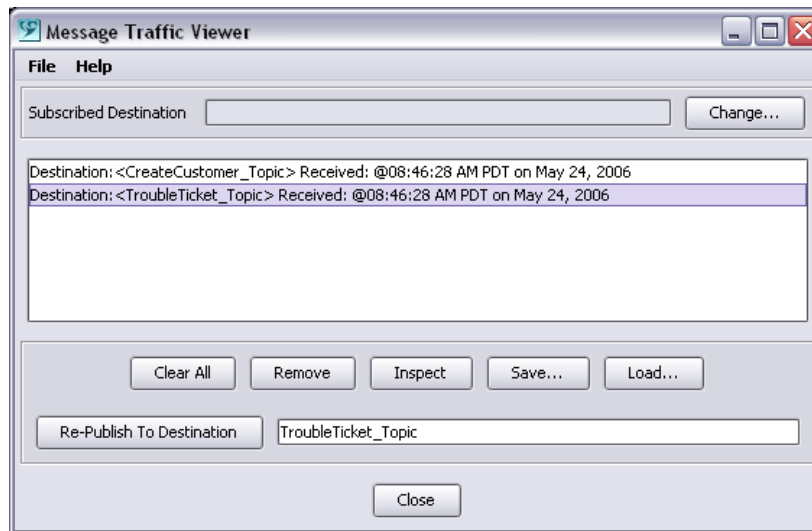


Figure 4-12: Cogility Message Traffic Viewer

## Cogility Web Service Exerciser

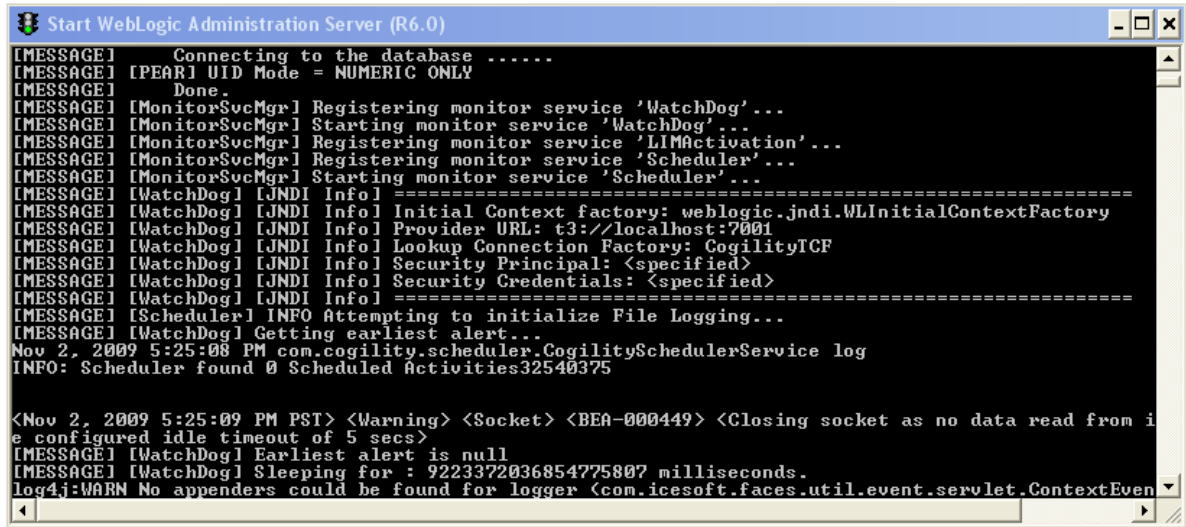
Cogility Web Service Exerciser lets you invoke web services independently of a model. You can also import modeled web services and run them with this utility. See [“Cogility Web Service Exerciser”](#) on page 137 of the guide, *Model Deployment & Execution in Cogility Studio*.



Figure 4-13: Cogility Web Service Exerciser

## Cogility WatchDog

Business processes may time out by virtue of a time event associated with a transition or an action. Cogility WatchDog is a service that monitors and manages these events at run time. See “[Cogility WatchDog](#)” on page 143 of the guide, *Model Deployment & Execution in Cogility Studio*.



```

Start WebLogic Administration Server (R6.0)
[MESSAGE] Connecting to the database .....
[MESSAGE] [PEAR] UID Mode = NUMERIC ONLY
[MESSAGE] Done
[MESSAGE] [MonitorSvcMgr] Registering monitor service 'WatchDog'...
[MESSAGE] [MonitorSvcMgr] Starting monitor service 'WatchDog'...
[MESSAGE] [MonitorSvcMgr] Registering monitor service 'LIMActivation'...
[MESSAGE] [MonitorSvcMgr] Registering monitor service 'Scheduler'...
[MESSAGE] [MonitorSvcMgr] Starting monitor service 'Scheduler'...
[MESSAGE] [WatchDog] [JNDI] Info! =====
[MESSAGE] [WatchDog] [JNDI] Info! Initial Context factory: weblogic.jndi.WLInitialContextFactory
[MESSAGE] [WatchDog] [JNDI] Info! Provider URL: t3://localhost:7001
[MESSAGE] [WatchDog] [JNDI] Info! Lookup Connection Factory: CogilityTCF
[MESSAGE] [WatchDog] [JNDI] Info! Security Principal: <specified>
[MESSAGE] [WatchDog] [JNDI] Info! Security Credentials: <specified>
[MESSAGE] [WatchDog] [JNDI] Info! =====
[MESSAGE] [Scheduler] INFO Attempting to initialize File Logging...
[MESSAGE] [WatchDog] Getting earliest alert...
Nov 2, 2009 5:25:08 PM com.cogility.scheduler.CogilitySchedulerService log
INFO: Scheduler found 0 Scheduled Activities32540375

<Nov 2, 2009 5:25:09 PM PST> <Warning> <Socket> <BEA-000449> <Closing socket as no data read from i
e configured idle timeout of 5 secs>
[MESSAGE] [WatchDog] Earliest alert is null
[MESSAGE] [WatchDog] Sleeping for : 9223372036854775807 milliseconds.
log4j:WARN No appenders could be found for logger <com.icesoft.faces.util.event.servlet.ContextEven

```

Figure 4-14: Cogility WatchDog







### A

active project 8  
Authoring repository 9

### B

basic project 8  
Behavior model 13  
business processes 13

### C

Class associations 13  
Class attributes 13  
Class diagrams 12  
Cogility Message Traffic Viewer 21  
Cogility WatchDog 23  
Cogility Web Service Exerciser 22

### D

Deployment 19

### E

Events 14

### I

Information model 12  
initiate the business process 14  
install Cogility Studio 7

### M

Message artifacts 15  
MIM class 12  
Model container 11

### O

outbound web service 16

### P

Project configuration 7  
project-specific configuration file 8

### S

set up Cogility Studio 7  
state machine 16

### T

Transformations 16

