

Corticon Server: Deploying Web Services with Java

Notices

Copyright agreement

© 2013 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Apama, Business Empowerment, Business Making Progress, Corticon, Corticon (and design), DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Empowerment Center, Fathom, Making Software Work Together, OpenEdge, Powered by Progress, PowerTier, Progress, Progress Control Tower, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress OpenEdge, Progress Profiles, Progress Results, Progress RPM, Progress Software Business Making Progress, Progress Software Developers Network, ProVision, PS Select, RulesCloud, RulesWorld, SequeLink, SpeedScript, Stylus Studio, Technical Empowerment, WebSpeed, Xcalia (and design), and Your Software, Our Technology—Experience the Connection are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, Apama Risk Firewall, AppsAlive, AppServer, BusinessEdge, Cache-Forward, DataDirect Spy, DataDirect SupportLink, Future Proof, High Performance Integration, OpenAccess, ProDataSet, Progress Arcade, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, Progress Responsive Process Management, Progress Software, PSE Pro, SectorAlliance, SeeThinkAct, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

See Table of Contents for location of Third party acknowledgements within this documentation.

Table of Contents

Preface.....	9
Progress Corticon documentation.....	9
Overview of Progress Corticon.....	11
 Chapter 1: Conceptual overview.....	 13
What is a web service?.....	13
What is a Decision Service?.....	14
What is the Corticon Server with Java?.....	14
What is a web services consumer?.....	14
 Chapter 2: Getting started.....	 15
 Chapter 3: Using the Corticon Server for Java installer.....	 17
Downloading the Corticon Server for Java packages.....	17
System requirements.....	18
Corticon Server for Java setup wizard.....	18
Limits of the default evaluation license.....	22
Registering your Corticon license for Java Server.....	23
 Chapter 4: Starting Corticon Server for Java.....	 25
 Chapter 5: Corticon Java Server files and API tools.....	 29
Basic server classes.....	30
Setting up Corticon Server use cases.....	30
Installing Corticon Server as a J2EE SOAP servlet.....	31
Installing Corticon Server as a J2EE enterprise Java bean (EJB).....	31
Installing Corticon Server as Java classes in-process or in a custom Java container.....	32
The Corticon home and work directories.....	33
Configurations that use global environment settings.....	33
The Corticon Server Sandbox.....	34
Testing the installed Corticon Server.....	34
Testing the installed Corticon Server as a J2EE SOAP servlet.....	35
As in-process Java classes.....	37
 Chapter 6: Deploying a Ruleflow to the Corticon Server.....	 41

Creating a Ruleflow.....	42
Creating and installing a Deployment Descriptor file.....	42
Using the Deployment Console tool's Decision Services on the Java Server.....	42
Installing the Deployment Descriptor file.....	45
Hot re-deploying Deployment Descriptor files and Ruleflows.....	46
Chapter 7: Consuming a decision service.....	47
Integrating and testing a Decision Service.....	48
Path 1- Using Corticon Studio as a SOAP client to consume a Decision Service.....	48
Creating a new Java server test in Corticon Studio.....	49
Executing the remote test.....	50
Path 2 - Using bundled sample code to consume a Decision Service.....	51
Sending a request message to Corticon Server.....	54
Path 3 - Using SOAP client to consume a Decision Service.....	57
Web services messaging styles.....	57
Creating a service contract using the Deployment Console.....	58
Creating a request message for a Decision Service.....	59
Sending a request message to Corticon Server.....	59
Troubleshooting.....	60
Chapter 8: Modeling and managing rules using Server Console.....	61
Lifecycle management Through the Server Console.....	62
Creating a new decision service version.....	62
Opening the new version.....	63
Modifying the new version.....	63
Promoting the new version to live.....	64
Removing the version from live.....	64
Telling the server where to find Deployment Descriptor files	64
Chapter 9: Using the Corticon Server Console.....	65
Launching and logging in to Corticon Server Console.....	66
Using LDAP authentication in Server Console.....	68
Console main menu page.....	71
Decision Services page.....	72
Decision Service actions and information.....	73
Execution and Distribution Visualizations.....	80
Deploy Decision Service page.....	82
Server state.....	82
Configure rule server.....	84
Logging tab.....	84
Deployment Directory tab.....	85
Decision Service Options tab.....	85
License tab.....	86

Monitor rule server page.....	86
Rules Server Stats tab.....	87
Rules Server Settings tab.....	87
Environment Settings tab.....	88
Memory Utilization tab.....	89
Rules Server Log.....	89
WSDLs.....	90
 Chapter 10: Summary of Java samples.....	91
 Appendix A: Third party acknowledgments	93

Preface

For details, see the following topics:

- [Progress Corticon documentation](#)
- [Overview of Progress Corticon](#)

Progress Corticon documentation

The following documentation, as well as a *What's New in Corticon* document, is included with this Progress Corticon release:

Corticon Tutorials	
<i>Corticon Studio Tutorial: Basic Rule Modeling</i>	Introduces modeling, analyzing, and testing rules and decisions in Corticon Studio. Recommended for evaluators and users getting started. <i>See also the PowerPoint-as-PDF version of this document that is accessed from the Studio for Analysts' Help menu.</i>
<i>Corticon Studio Tutorial: Advanced Rule Modeling</i>	Provides a deeper look into Corticon Studio's capabilities by defining and testing vocabularies, scope, collections, messages, filters, conditions, transient data, and calculations in multiple rulesheets that are assembled into a Ruleflow. <i>See also the PowerPoint-as-PDF version of this document that is accessed from the Studio for Analysts' Help menu.</i>
<i>Corticon Tutorial: Using Enterprise Data Connector (EDC)</i>	Introduces Corticon's direct database access with a detailed walkthrough from development in Studio to deployment on Server. Uses Microsoft SQL Server to demonstrate database read-only and read-update functions.

Corticon Studio Documentation: Defining and Modeling Business Rules	
<i>Corticon Studio: Installation Guide</i>	Step-by-step procedures for installing Corticon Studio and Corticon Studio for Analysts on computers running Microsoft Windows. Also shows how use other supported Eclipse installations for integrated development. Shows how to enable internationalization on Windows.
<i>Corticon Studio: Rule Modeling Guide</i>	Presents the concepts and purposes the Corticon Vocabulary, then shows how to work with it in Rulesheets by using scope, filters, conditions, collections, and calculations. Discusses chaining, looping, dependencies, filters and preconditions in rules. Presents the Enterprise Data Connector from a rules viewpoint, and then shows how database queries work. Provides information on versioning, natural language, reporting, and localizing. Provides troubleshooting and many <i>Test Yourself</i> exercises.
<i>Corticon Studio: Quick Reference Guide</i>	Reference guide to the Corticon Studio user interface and its mechanics, including descriptions of all menu options, buttons, and actions.
<i>Corticon Studio: Rule Language Guide</i>	Reference information for all operators available in the Corticon Studio Vocabulary. A Rulesheet example is provided for many of the operators. Includes special syntax issues, handling arithmetic and character precedence issues.
<i>Corticon Studio: Extensions Guide</i>	Detailed technical information about the Corticon extension framework for extended operators and service call-outs. Describes several types of operator extensions, and how to create a custom extension plug-in.
Corticon Server Documentation: Deploying Rules as Decision Services	
<i>Corticon Server: Deploying Web Services with Java</i>	Details installing the Corticon Server as a Web Services Server, and then deploying and exposing Decision Services as Web Services on Tomcat and other Java-based servers. Presents the features and functions of the browser-based Server Console.
<i>Corticon Server: Deploying Web Services with .NET</i>	Details installing the Corticon Server as a Web Services Server, and then deploying and exposing decisions as Web Services with .NET. Provides installation and configuration information for the .NET Framework and Internet Information Services (IIS) on various supported Windows platforms.
<i>Corticon Server: Integration & Deployment Guide</i>	An in-depth, technical description of Corticon Server deployment methods, including preparation and deployment of Decision Services and Service Contracts through the Deployment Console tool. Discusses relational database concepts and implementation of the Enterprise Data Connector. Goes deep into the server to discuss state, persistence, and invocations by version or effective date. Includes samples, server monitoring techniques, and recommendations for performance tuning.

Overview of Progress Corticon

Progress® Corticon® is the Business Rules Management System with the patented "no-coding" rules engine that automates sophisticated decision processes.

Progress Corticon products

Progress Corticon distinguishes its development toolsets from its server deployment environments.

- **Corticon Studios** are the Windows-based development environment for creating and testing business rules:
 - **Corticon Studio for Analysts.** is a standalone application, a lightweight installation that focuses exclusively on Corticon.
 - **Corticon Studio** is the *Corticon Designer* perspective in the **Progress Developer Studio** (PDS), an industry-standard Eclipse and Java development environment. The PDS enables development of applications integrated with other products, such as Progress OpenEdge.

The functionality of the two Studios is virtually identical, and the documentation is appropriate to either product. Documentation of features that are only in the *Corticon Designer* (such as on integrated application development and Java compilation) will note that requirement. Refer to the *Corticon Studio: Installation Guide* to access, prepare, and install each of the Corticon Studio packages.

Studio Licensing - Corticon embeds a time-delimited evaluation license that enables development of both rule modeling and Enterprise Data Connector (EDC) projects, as well as testing of the projects in an embedded Axis test server. You must obtain studio development licenses from your Progress representative.

- **Corticon Servers** implement web services for business rules defined in Corticon Studios:
 - **Corticon Server for deploying web services with Java** is supported on various application servers, and client web browsers. After installation on a supported Windows platform, that server installation's deployment artifacts can be redeployed on various UNIX and Linux web service platforms as Corticon Decision Services. The guide *Corticon Server: Deploying web services with Java* provides details on the full set of platforms and web service software that it supports, as well as installation instructions in a tutorial format for typical usage.
 - **Corticon Server for deploying web services with .NET** facilitates deployment of Corticon Decision Services on Windows .NET Framework and Microsoft Internet Information Services (IIS). The guide *Corticon Server: Deploying web services with .NET* provides details on the platforms and web service software that it supports, as well as installation instructions in a tutorial format for typical usage.

Server Licensing - Corticon embeds a time-delimited evaluation license that enables evaluation and testing of rule modeling projects on supported platform configurations. You must obtain server deployment licenses and server licenses that enable the Enterprise Data Connector (EDC) from your Progress representative.

Conceptual overview

This guide is a walkthrough of fundamental concepts and functions of Corticon Server for Java. The examples focus on the default Apache Tomcat web server that the product installs to provide Java Web Services.

For details, see the following topics:

- [What is a web service?](#)
- [What is a Decision Service?](#)
- [What is the Corticon Server with Java?](#)
- [What is a web services consumer?](#)

What is a web service?

From the business perspective: A Web Service is a software asset that automates a task and can be shared, combined, used, and reused by different people or systems within or among organizations.

From the information systems perspective: A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. [From <http://www.w3c.org>.]

What is a Decision Service?

A Decision Service automates a discrete decision-making task. It is implemented as a set of business rules and exposed as a Web Service (or Java Service). By definition, the rules within a Decision Service are complete and unambiguous; for a given set of inputs, the Decision Service addresses every logical possibility uniquely, ensuring decision integrity.

A Ruleflow is built in Corticon Studio. Once deployed to the Corticon Server, it becomes a Decision Service.

What is the Corticon Server with Java?

The Corticon Server is a high-performance, scalable and reliable system resource that manages pools of Decision Services and executes their rules against incoming requests. The Corticon Server can be easily configured as a Web Services server, which exposes the Decision Services as true Web Services.

What is a web services consumer?

A Web Services Consumer is a software application that makes a request to, and receives a response from, a Web Service. Most modern application development environments provide native capabilities to consume Web Services, as do most modern Business Process Management Systems.

Getting started

This Tutorial steps through the procedures for running the Corticon Java Server as a Web Services server, deploying Ruleflows to the Server, exposing the Ruleflows as Decision Services, and then testing them with document-style SOAP requests. There are other installation, deployment and integration options available beyond the SOAP/Web Services method described here, including Java-centric options using Java objects and APIs. More detailed information on all available methods is contained in the *Integration & Deployment Guide*.

This Tutorial consists of four main sections:

Using the Corticon Server for Java installer

Describes downloading and installing Corticon Server for Java on your designated server machine.

Installing the Corticon Server for Java as a web services server

Describes deploying the Corticon Java Server to the Tomcat web server, which is installed by default during the installation process described in the first section.

Deploying a Ruleflow to the Corticon Server

Describes deploying Ruleflows to the Server and exposing them as web services, which we call *Decision Services*. Once a Ruleflow becomes a Decision Service, it may be consumed by any external application or process capable of interacting with standard document-style web services.

Consuming a Decision Service

Describes integrating and testing your deployed Decision Service by creating and sending request messages to the Server, and viewing the response messages. Two methods of integration and testing are discussed: one method assumes you have access only to the tools contained in the default Server installation, while the other method assumes you have a commercially available SOAP client, application or tool to perform these tasks.

Using the Corticon Server for Java installer

For details, see the following topics:

- [Downloading the Corticon Server for Java packages](#)
- [System requirements](#)
- [Corticon Server for Java setup wizard](#)
- [Limits of the default evaluation license](#)
- [Registering your Corticon license for Java Server](#)

Downloading the Corticon Server for Java packages

Corticon Server for Java and its Service Packs are packaged in executable installer applications:

- The 5.3 installer, `PROGRESS_CORTICON_5.3_SERVER.exe`, to perform a new installation.
- The 5.3.4 Service Pack installer, `PROGRESS_CORTICON_5.3.4_SERVER.exe`, to update a 5.3.0 installation.

To download the required installers:

1. Get credentials to access and download packages on the Progress Software Electronic Software Download (ESD) site.
2. Connect to the ESD, and then navigate to the Corticon 5.3 pages.
3. Locate, download, and save the required installers to a temporary location accessible by the target machine.

System requirements

Corticon Server for Java requirements include:

- All Java-supported processors
- 512 MB RAM recommended
- 10 MB disk space (core libraries)
- 150 MB disk space (full install)
- Java Runtime Engine, version 6 or version 7.

Progress Corticon Server for Java is supported on several Microsoft Windows platforms. Servers can be deployed onto a variety of UNIX/Linux platforms and Application Servers. See the Progress Software web page [Progress Corticon 5.3 - Supported Platforms Matrix](#) for more information.

See the Corticon KnowledgeBase entry [Corticon Server 5.X sample EAR/WAR installation for different Application Servers](#) for detailed instructions on configuring Apache Tomcat, JBoss, WebSphere, WebLogic on all supported platforms.

Corticon Server for Java setup wizard

Installation type

There are three ways to create Corticon Server for Java installations:

- **New 5.3.4 installation** - Perform a [new installation](#), then update it with the [5.3.4 Service Pack](#).
- **Update a 5.3.3, 5.3.2, 5.3.1, or 5.3.0 installation** - Apply the [5.3.4 Service Pack](#). Note that a higher Service Pack in a version rolls up the features of all preceding Service Packs.
- **Upgrade a 5.2 or earlier installation to the latest 5.3 Service Pack** - An installed version of Corticon Server for Java 5.2 or earlier installed on the target machine must be [uninstalled](#), then [Corticon 5.3.0 Server installed](#), followed by applying the [5.3.4 Service Pack](#).

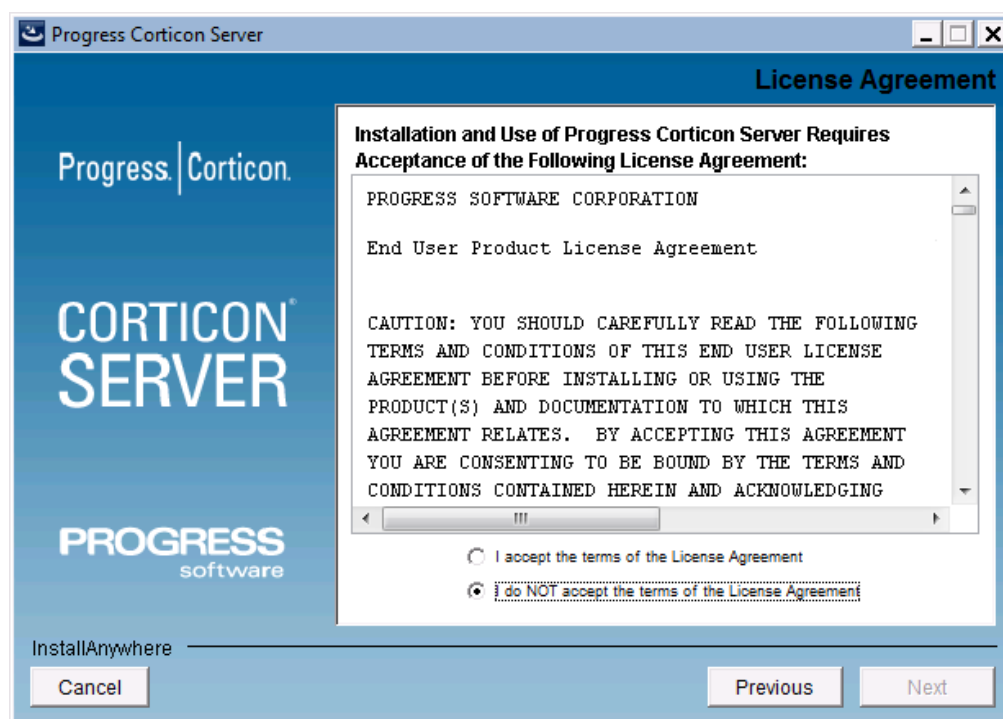
Consult with Progress Corticon Support or your Progress representative to consider your migration strategies for existing assets before you take action.

New installation

To perform a new installation of Corticon Server for Java:

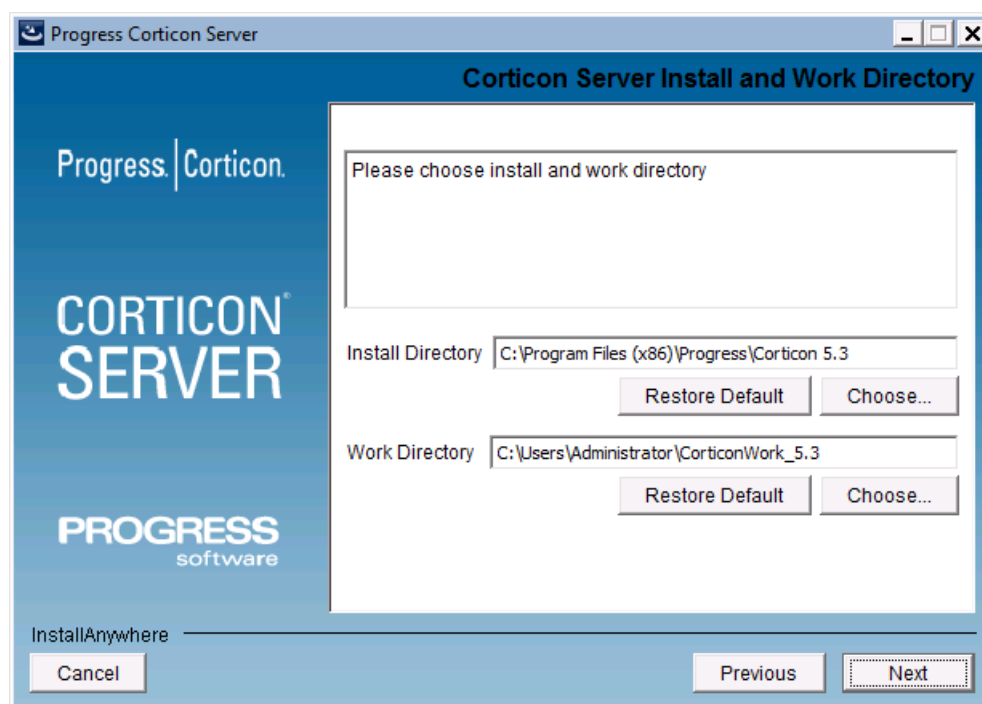
1. Double click on `PROGRESS_CORTICON_5.3_SERVER.exe` to open the Progress Corticon Server Setup Wizard.
2. Click **Next** to continue.

The **License Agreement** panel opens.



- After you have read, understood, and agreed to the terms of End User License Agreement, choose **I accept the terms of the license agreement**, and then click **Next**.

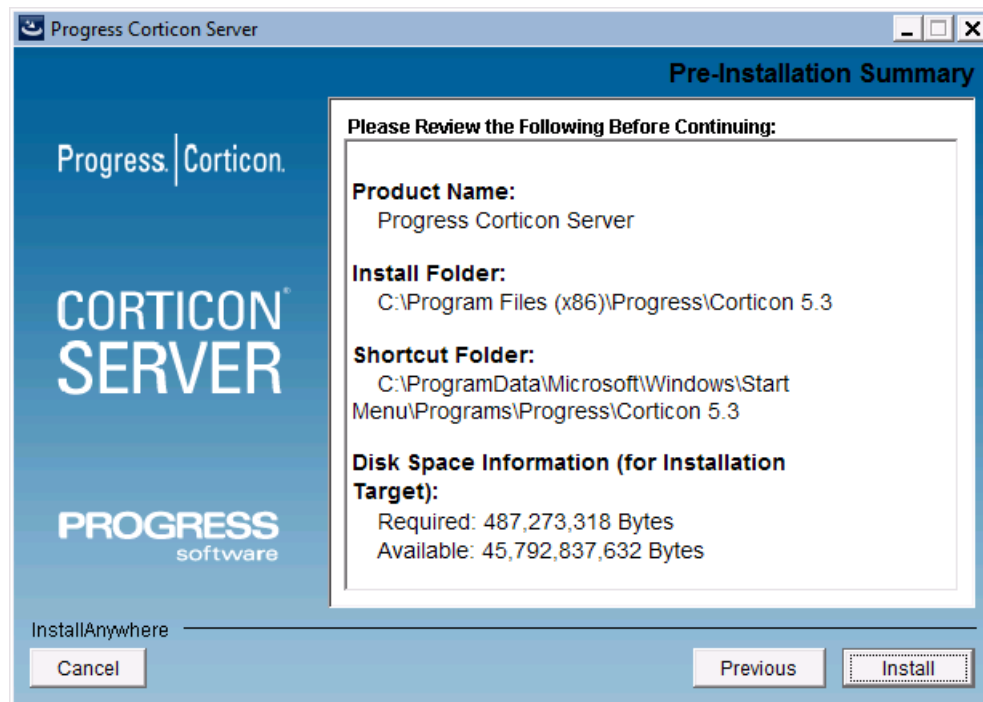
The **Choose Install Folder** panel opens.



- Select the installation directory. To accept the default locations, click **Next**.

To specify preferred locations, click **Choose**, navigate to each preferred directory, click **OK** to close the chooser dialog, and then click **Next**.

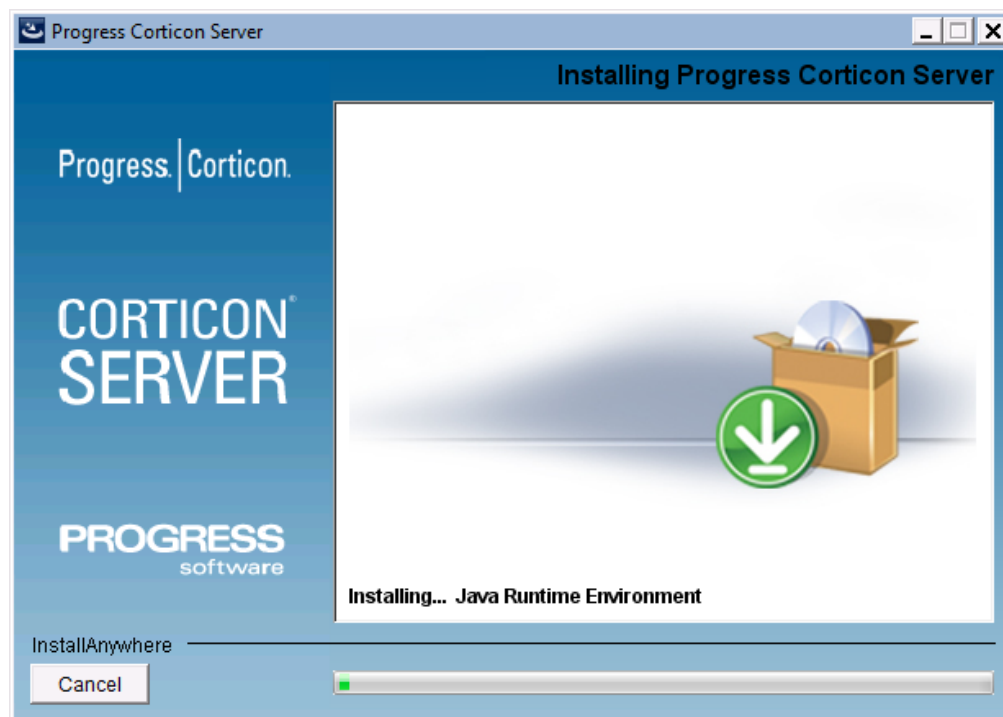
The **Pre-Installation Summary** page opens.



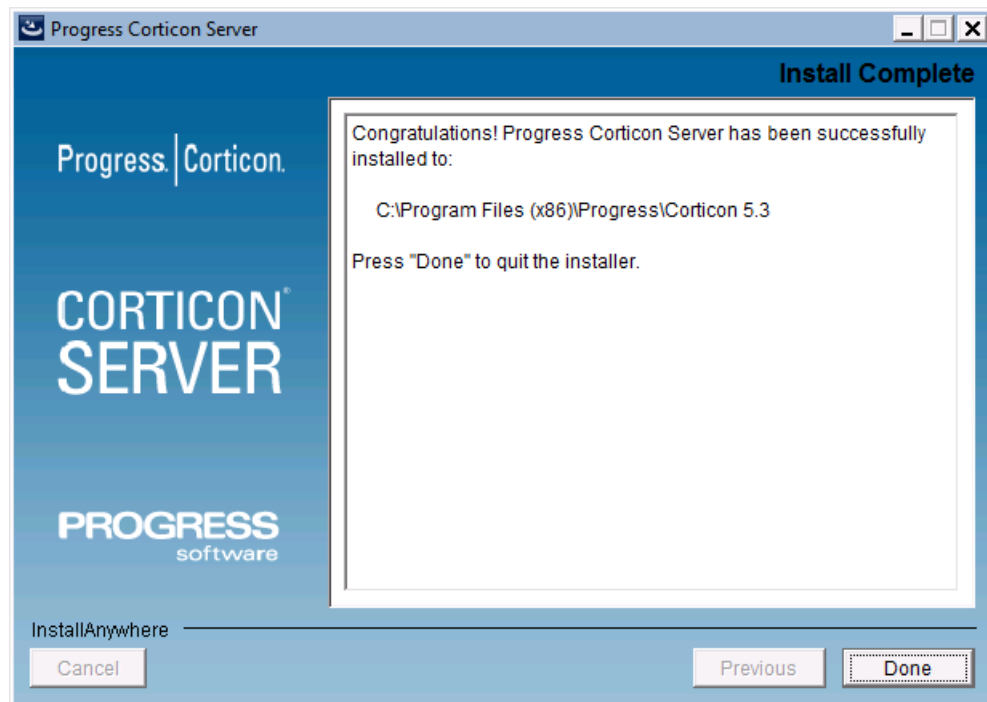
Verify your selections in the **Pre-Installation Summary** panel.

5. Click **Install** to continue.

The installation status window opens.



This panel displays a status bar during the installation process. When done, the **Install Complete** panel opens.

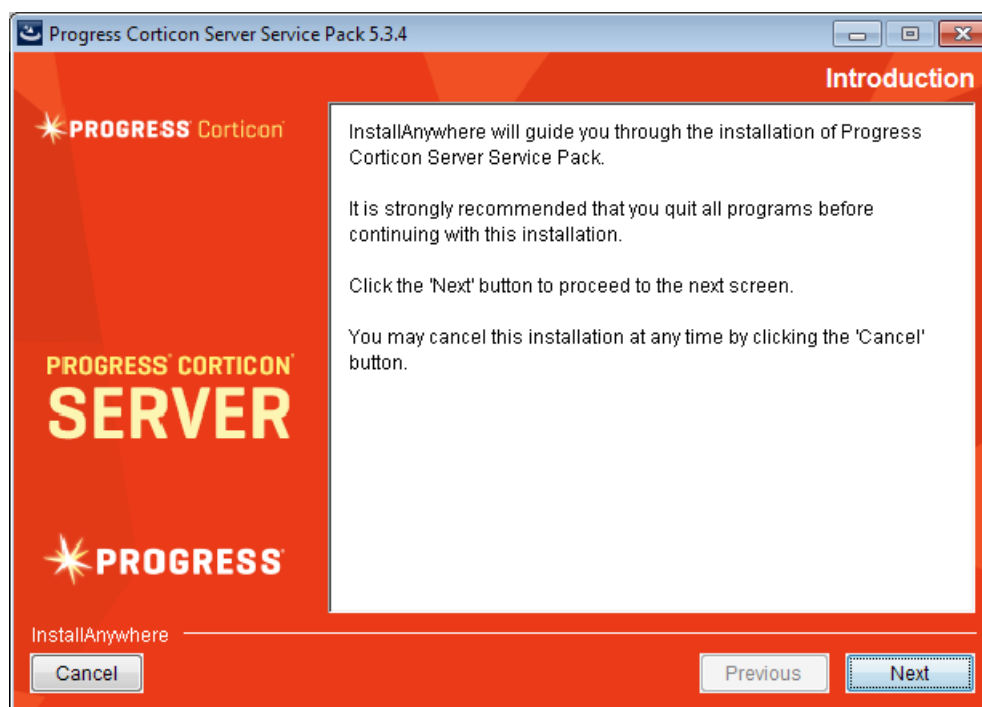


6. Choose **Done** to complete the Corticon Server for Java installation and close the installer.

Service Pack updater

To update a 5.3.3, 5.3.2, 5.3.1, or 5.3.0 installation of Corticon Server for Java to a Service Pack:

1. If the installed Corticon Server you want to update is running, stop it now.
2. Double click on `PROGRESS_CORTICON_5.3.4_SERVER.exe` to open the Progress Corticon Server Service Pack Setup Wizard. Note the revised Progress branding scheme in the 5.3.4 update wizard:



3. Click **Next** through the panels, and then click **Install**.

When the 5.3.3, 5.3.2, 5.3.1, or 5.3.0 installation is detected, and it is determined that this Service Pack has not already been applied, the updater proceeds to complete the update of the installation.

Uninstalling Corticon Server

To remove a version of the server to prepare for a new installation (or if you want to fully remove it), use this procedure.

Note: Uninstall removes a complete major.minor version. You cannot uninstall just a Service Pack.

To uninstall Corticon Server for Java:

1. Stop the server.
2. Backup any files you want to retain.
3. Navigate to `[CORTICON_HOME]\Uninstall Progress Corticon Server`.
4. Run `Uninstall Progress Corticon Server.exe`.

The installed files are removed. Note that files you created are NOT removed or replaced during this process.

If the Uninstaller program is unable to fully remove components (usually because they are open), it will display messages, and might require a reboot to complete the process.

Limits of the default evaluation license

The license included in the default Corticon Server installation has pre-set limits on certain Corticon Server and Decision Service parameters. These limits are:

- **Number of Decision Services** – Up to 20 Decision Services may be deployed at any given time. This means the sum total of all Decision Services loaded via `.odd` files, Web Console, or APIs cannot exceed 20.
- **Pool Size** – No Decision Service may have a maximum pool size setting of greater than 1. Pool size is measured on a Decision Service-by-Decision Service basis, so you may have 20 Decision Services deployed (compliant with the Decision Service limitation above), each with 1 Reactor in its pool, without violating the default license restrictions.
- **Number of Rules** – All rules in all deployed Ruleflows (that is, all deployed Decision Services) must not exceed 500. A rule generally consists of a single Condition/Action Column or a single Action row in Column 0. Filter expressions do not count because they only modify other rules.

The Corticon Server log captures errors and exceptions caused by expired or "under-strength" licenses. Such log messages are detailed in the *Troubleshooting* section of the *Rule Modeling Guide*.

If you are Progress Corticon customer, you should have access to an unlimited license that will lift these restrictions. If you are an evaluator, and discover that these limitations are preventing or inhibiting your evaluation, contact your Progress Software representative to get a license with expanded capabilities.

Registering your Corticon license for Java Server

Progress Corticon embeds an evaluation license in its products to help you get started.

- Corticon Studio evaluation licenses let you use database access (Enterprise Data Connector or "EDC"), and are timed to expire on a preset date.
- Corticon Server evaluation licenses do not enable use of Enterprise Data Connector, and limit the number of decision services, rules, and pools in use. They too are timed to expire on a preset date.

When you obtain a license file, it applies to Studios as well as Servers. You must perform configuration tasks to record it for each Corticon Studio, each Corticon Server, and each Deployment Console. If you intend to use EDC on your Corticon Servers, your Corticon license must allow it. Contact Progress Corticon technical support if you need to acquire a license.

The Corticon Server license is placed at two locations in the installation to enable use and -- when specified in the license -- enable EDC functions for:

- Corticon Server
- Corticon Server Console
- Corticon Deployment Console

To configure Corticon Java Server to access its license file:

1. Copy the license JAR by its default name, `CcLicense.jar`.
2. Navigate to the installation's `Server\lib` directory to paste the file and overwrite the existing file in that location.
3. Navigate to the installation's `Server\Tomcat\webapps\axis\WEB-INF\lib` directory to paste the file and overwrite the existing file in that location.

When you launch the Corticon Deployment Console, your license with its registration information is registered for the Corticon Deployment Console. When your license enables EDC, the **Database Access** fields and functions are enabled.

Note:

You can choose to locate the license by another JAR name at a preferred location, and then expressly identify it to the server.

To custom configure Corticon Java Server's license location:

1. Navigate in the file system to the Corticon Java Server installation's, `Server` subdirectory.
 2. Double-click on `testServer.bat`, then do the following:
 - a. Type `416` and then press **Enter**.
 - b. Enter (or copy/paste) the complete path to the location of the license JAR file, as in this example, `C:\licenses\myCorticon531EDC_CcLicense.jar`. The command echoes back `Transaction completed`.
 - c. To confirm the setting, type `415` and then press **Enter**. The path is echoed back (you might need to scroll up to the command line.)
 3. Navigate in the file system to the Corticon Java Server installation's, `Server\Tomcat` subdirectory.
 4. Double-click on `testServerAxis.bat`.
 5. Perform the same `416` and `415` tasks as in Step 2 above
-

Starting Corticon Server for Java

When you installed Corticon Server, it deployed into its Tomcat during installation by pre-loading Corticon Server's `axis.war` file into Tomcat's `webapps` directory:

```
[CORTICON_HOME]\Server\Tomcat\webapps
```

Then, when you start the bundled Tomcat for the first time, it unpacks `axis.war`, and creates a new `axis` directory inside `\webapps`. This new folder contains the complete Corticon Server web application.

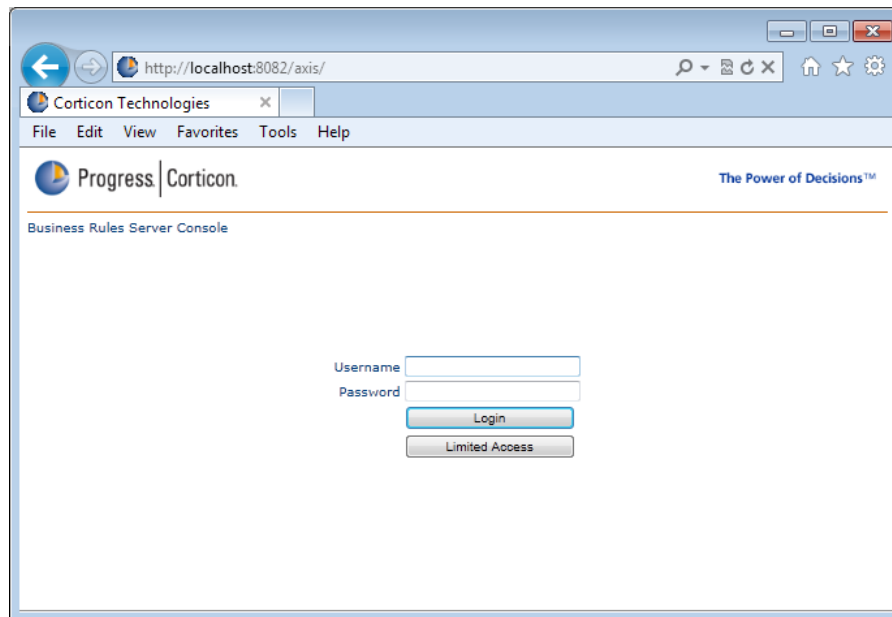
Note: If you intend to install Corticon Server on a web server other than the bundled Tomcat, refer to the *Integration & Deployment Guide* for details on using the standard `axis.war` package in your preferred web server.

To verify that Corticon Server is installed correctly on Tomcat:

1. Start Tomcat from the Windows **Start** menu by choosing **All Programs > Progress > Corticon > Corticon Server > Local Server (Tomcat) > Start Server**
2. In an Internet browser, enter the URL <http://localhost:8082/axis>.

The Business Rules Server Console opens to its login page, as shown:

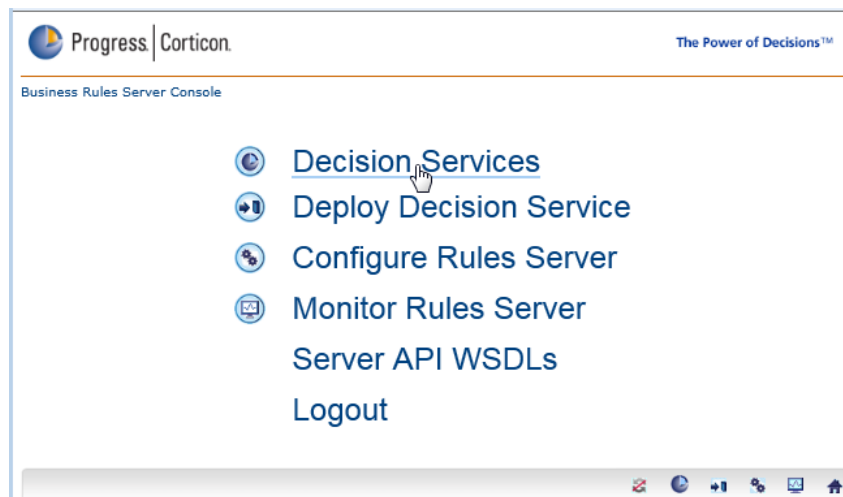
Figure 1: Server web console login page



3. Enter the Username `admin` and the Password `admin`. Click **Login**.

The Business Rules Server console lets you choose a functional path, as shown:

Figure 2: Server console functions



4. Click **Decision Services** to view the list of deployed Web services, which might look like this:

Business Rules Server Console -> Home
 Deployed Decision Services

Service Name	Version	Live	Effective Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	<input checked="" type="checkbox"/>		Yes	Yes	0	0	Clear
Candidates	1.14	<input checked="" type="checkbox"/>		Yes	Yes	0	0	Clear
ProcessOrder	1.10	<input checked="" type="checkbox"/>		Yes	No	0	0	Clear



Corticon Java Server files and API tools

Corticon Server is provided in two installation sets: Corticon Server for Java, and Corticon Server for .NET.

Corticon Servers implement web services for business rules defined in Corticon Studios.

- The **Corticon Server for deploying web services with Java** -- the product documented here -- is supported on various application servers, databases, and client web browsers. After installation on a supported Windows platform, that server installation's deployment artifacts can be redeployed on various UNIX and Linux web service platforms. See the Progress Software web page [Progress Corticon 5.3 - Supported Platforms Matrix](#) for more information.
- The **Corticon Server for deploying web services with .NET** facilitates deployment on Windows .NET framework 4.0 and Microsoft Internet Information Services (IIS) that are packaged in the supported operating systems. The .NET server has its own installer and documentation. See *Deploying Web Service with .NET* for more information.

For details, see the following topics:

- [Basic server classes](#)
- [Setting up Corticon Server use cases](#)
- [The Corticon home and work directories](#)
- [The Corticon Server Sandbox](#)
- [Testing the installed Corticon Server](#)

Basic server classes

At its most basic level, Corticon Server is simply a set of Java classes, packaged in Java archive, or `.jar`, files. The minimum set of jars needed to deploy and call Corticon Server is listed below:

- `CcServer.jar` – The main Corticon Server JAR, containing the core engine logic.
- `CcConfig.jar` – Contains a set of text `.property` files that list and set all the configuration properties needed by Corticon Server.
- `CcLicense.jar` – An encrypted file containing the licensing information required to activate Corticon Server.
- `CcThirdPartyJars.jar` – Contains third-party software such as XML parsers and JDOM. Corticon warrants Corticon Server operation **ONLY** with the specific set of classes inside this jar.
- `CcI18nBundles.jar` – Contains the English and other language templates used by Corticon Server.
- `ant-launcher.jar` – Supports Corticon Server's deployment-time Decision Service compilation capability
- `CcExtensions.jar` – **Optional**. – Necessary only if you have added new rule language operators as "Extended Operators." (See the *Rule Language Guide* for details.)

Setting up Corticon Server use cases

In most production deployments, Corticon Server JARs are bundled and given a J2EE interface class or classes. The interface class is often called a "helper" or "wrapper" class because its purpose is to receive the client application's invocation, translate it (if necessary) into a call which uses Corticon Server's native API, and then forwards the call to Corticon Server's classes. The type of interface class depends on the J2EE container where you intend to deploy the Corticon Server.

Corticon Studio makes in-process calls to the same Corticon Server classes (although packaged differently) when Ruletests are executed. This ensures that Ruleflows behave exactly the same way when executed in Studio Ruletests as they do when executed by Corticon Server, no matter how Corticon Server is installed.

Installing Corticon Server as a J2EE SOAP servlet

If **Installation Option 1 (Web Services)** -- as described in Chapter 1 of the *Integration and Deployment guide* -- was chosen, then a SOAP Servlet interface will be used for production deployments. Install Corticon Server into the Servlet container of a J2EE web or application server such as WebLogic, Oracle, or WebSphere. Commercial application servers offer the greatest degree of manageability, security and reliability, but open-source J2EE web servers such as Apache Tomcat are also excellent, production-quality options. One advantage of the wrapper or helper class approach to installation is that variations in the web server environment (such as SOAP version) may be addressed in the wrapper class itself, rather than in the set of Corticon Server classes. The base set of Corticon Server jars remains the same irrespective of deployment environment – only the wrapper class changes.

The industry-standard method of deploying a Servlet into a J2EE web server's Servlet container is via a "web archive", or `.war`, file. This file contains everything required to deploy a fully functional Servlet, including all classes, configuration files, and interfaces. Corticon provides a complete sample `.war` file, along with all source code, with the standard default Corticon Server installation. In addition to the base set of Corticon JARs, the provided `.war` file also contains Apache Axis SOAP messaging framework, which is supported by most commercial J2EE web and application servers, including Apache Tomcat web server. Your web server documentation will include instructions for installing or loading a `.war` file.

This `.war` file, named `axis.war`, is located in your Corticon working directory in the `[CORTICON_WORK_DIR]\Samples\Server\Containers\WAR` directory.

Important: The `.war` file provided is intended to be a sample wrapper for instructional purposes. Source code is provided in the `[CORTICON_HOME]Server\Tomcat\CcServer\src` directory so you can adapt the wrapper to your environment and platform. **The sample `.war` file is not certified for use on any specific web server and is not warranted by Corticon .**

This `.war` file contains the default evaluation license, named `CcLicense.jar`. If you are a Corticon customer, you will be provided with a permanent version of this file. You must insert it into the `.war` (replacing the original) in order to remove the evaluation license limitations.

For a quick start, Corticon Server ships with Apache Tomcat web server and the Apache Axis SOAP messaging infrastructure, and is preconfigured to work with Corticon Server. Apache is installed by default (unless deselected in a custom installation) by the Corticon Server Installer.

Installing Corticon Server as a J2EE enterprise Java bean (EJB)

If **Installation Option 2 (Java Services with XML Payloads)** or **Installation Option 3 (Java Services with Java Object Payloads)** -- as described in Chapter 1 of the *Integration and Deployment guide* -- was chosen above, then an EJB interface will be used for production deployments. Install Corticon Server into the EJB container of a J2EE application server such as WebLogic, Oracle, or WebSphere. Commercial application servers offer the greatest degree of manageability, security and reliability, but open-source J2EE application servers such as JBOSS are also available. One advantage of the wrapper or helper class approach to installation is that variations in the application server environment may be addressed in the wrapper class itself, rather than in the set of Corticon Server classes. The base set of Corticon Server JARs remains the same regardless of the deployment environment – only the wrapper class changes.

The industry-standard method of deploying an EJB into a J2EE application server's EJB container is via an "enterprise archive", or `.ear`, file. This file contains everything required to deploy a fully functional Session EJB (stateless), including all classes, configuration files and interfaces. Corticon includes a complete sample `.ear` file along with all source code with the standard default Corticon Server installation. Your application server documentation will include instructions for installing an `.ear` file.

This `.ear` file, `CcServer.ear`, is located in the Corticon working directory `[CORTICON_WORK_DIR] \Samples\Server\Containers\EAR` directory.

Important: The `.ear` file provided is intended to be a sample wrapper for instructional purposes. **The sample `.ear` file is not certified for use on any specific application server and is not warranted by Progress Corticon.**

This `.ear` file contains the default evaluation license, named `CcLicense.jar`. Progress Corticon customers are provided with a permanent version of this file. You must insert it into the `.ear` (replacing the original) in order to remove the evaluation license limitations.

The sample `.ear` file also contains `axis.war`, enabling you to deploy both a Servlet and an EJB version of Corticon Server simultaneously. This allows you to expose both SOAP and Java interfaces to the same Decision Service, making your Decision Services even easier to use throughout your enterprise infrastructure.

Installing Corticon Server as Java classes in-process or in a custom Java container

If you choose to manage Corticon Server in-process via your client application or via a custom container, you are taking responsibility for many of the tasks that are normally performed by a J2EE web or application server. But by doing it in your own code, you can optimize your environment and eliminate unneeded overhead. This can result in much smaller footprint installations and faster performance.

Because Corticon Server is a set of Java classes, it can easily be deployed in-process in a JVM. When deployed in-process, the following tasks are the responsibility of the client application:

- Management of Java classpaths, ensuring the base set of Corticon Server classes is properly referenced.
- JVM lifecycle management, including startup/shutdown
- Concurrency & Thread management
- Security (if needed)
- Transaction management, including invocation of Corticon Server using the standard Java API set.

Corticon Server can also be installed into a custom container within any application. It has a small footprint and thus can be installed into client applications including browser-based applications, laptops and mobile devices.

For step-by-step instructions on using the Installer to gain access to Corticon Server's core jar files, see [Using the Corticon Server for Java installer](#) on page 17 in this guide.

Installation in-process or in a custom container involves these basic steps:

1. Place the following Corticon Server JAR files in a location accessible by the surrounding Java container:

- CcServer.jar
 - CcConfig.jar
 - CcLicense.jar
 - CcThirdPartyJars.jar
 - CcI18nBundles.jar
 - ant_launcher.jar
 - CcExtensions.jar [optional – only necessary if you have added custom rule language operators to the Operator Vocabulary as "Extended Operators." (See *Rule Language Guide* for details.)]
2. Configure the Java classpath to include the JAR files listed above.
 3. Change the `logPath` property inside `CcCommon.properties` to an explicit path to a directory for the Corticon Server Log file.
 4. Write code that:
 - Initializes Corticon Server
 - Sets environment variables such as `CORTICON_HOME` (see [The Corticon home and work directories](#) on page 33)
 - Deploys the Decision Services into Corticon Server
 - Requests a decision by marshaling the data payload and then invoking the relevant Corticon Decision Service
 - Processes the response from the Decision Service.

Sample code is provided that demonstrates an in-process deployment of Corticon Server. This code, named `CcServerApiTest.bat`, is located in the `src` directory of your `[CORTICON_HOME]` directory.

The Corticon home and work directories

As a Corticon installation completes, it tailors two properties that define its global environment. As you could have elected to specify preferred locations for the installation directory and for the work directory, the installer sets `CORTICON_HOME` to the **Server Install Directory**, (referred to within the installer as `%CORTICON_INSTALL_DIR%`), and `CORTICON_WORK_DIR` to the **Server Work Directory** you specified. These variables are used throughout the product to determine the relative location of other files.

Configurations that use global environment settings

CORTICON_HOME

The batch file `startServer.bat`, located in the Server installation's `Server\Tomcat` directory of your Corticon installation directory, `CORTICON_INSTALL_DIR`, calls on the environment setting file `..\..\bin\corticon_env.bat` to determine `CORTICON_HOME`.

The form `./` means "current directory", so the path Corticon Server uses in a default installation would be `C:\Program Files (x86)\Progress\Corticon 5.3\Server\Tomcat\CcServer`. Then, the call to the startup script is `./bin/startup.bat`. That launches the file `C:\Program Files (x86)\Progress\Corticon 5.3\Server\Tomcat\bin\startup.bat` yet leaves the `CcServer` directory as the current location.

CORTICON_WORK_DIR

The logs should be stored in the Corticon Server work directory. That target subdirectory is defined by the `logPath` property defined in `CcCommon.properties` (inside `CcConfig.jar`) as:

```
logpath=%CORTICON_WORK_DIR%/logs
```

Therefore, when Corticon Server is launched, the `logPath` location will be, in a default installation, `C:\Users\{username}\Progress\Corticon 5.3\logs`.

It is a good practice to use global environment settings

Many file paths and locations are determined by the `CORTICON_HOME` and `CORTICON_WORK_DIR` variables. Be sure to call `corticon_env.bat`, and then use these variables in your scripts and wrapper classes so that they are portable to deployments that might have different install paths.

The Corticon Server Sandbox

When Corticon Server starts up, it checks for the existence of a "sandbox" directory. This Sandbox is a directory structure used by Corticon Server to manage its state and deployment code.

The location of the Sandbox is controlled by `com.corticon.ccserver.sandboxDir` inside `CcServer.properties` (inside `CcConfig.jar`).

This configuration setting is defined by the `CORTICON_WORK_DIR` variable, in this case:

```
com.corticon.ccserver.sandboxDir=%CORTICON_WORK_DIR%/CORTICON_SETTING%/CcServerSandbox
```

The result for this in a default Windows installation is

`C:\Users\{username}\Progress\CorticonWork_5.3\SER\CcServerSandbox`. In other words, in the `SER` subdirectory of the `CORTICON_WORK_DIR`. This directory is created (as well as peer directories, `logs` and `output`) during the first launch of Corticon Server.

Note: If the location specified by `com.corticon.ccserver.sandboxDir` cannot be found or is not available, the Sandbox location defaults to the JVM's directory as it is typically the location that initiated the call.

Testing the installed Corticon Server

With Corticon Server installed in the environment and container of your choice, it is useful to test the installation to ensure Corticon Server is running and listening. At this point, no Decision Services have been deployed, so Corticon Server is not ready to process transactions. However, the Corticon Server API set contains administrative methods that interrogate it and return status information. Several tools are provided to help you perform this test.

Testing the installed Corticon Server as a J2EE SOAP servlet

To test that Corticon Server deployed as a SOAP Servlet is running correctly, all you need is a SOAP client or the sample batch file provided and described below.

Testing the Servlet installation here assumes you have already installed and started Corticon Server as a Web Service in the bundled Apache Tomcat or using the `.war` file in another web server.

Because a SOAP Servlet is listening for SOAP calls, we need a way to invoke an API method via a SOAP message then send that message to Corticon Server using a SOAP client. In the sample code supplied in the default installation, Corticon provides an easy way to make API calls to it via a SOAP message.

The included batch file, `testServerAxis.bat`, will help ensure that Corticon Server is installed properly and listening for calls. `testServerAxis.bat`, located in the `[CORTICON_HOME]\Server\Tomcat` directory, provides a menu of available Corticon Server methods to call into the SOAP Servlet. Running `testServerAxis.bat` (or `.sh` in a UNIX environment) causes the following actions to occur:

- Sets classpaths needed by the `CcServerAxisTest` class, which is acting as our menu-driven SOAP client. The source code (`.java`) is included in the `[CORTICON_HOME]\Server\Tomcat\CcServer\src` directory.
- Defines variables for web server location and port. Port changes that may be necessary depending on the type of application or web server you are using to host the Servlet. Notice that the defaults are for the bundled Apache Tomcat web server, which uses `localhost` and `8082` as the application server's location and port setting.
- Starts a JVM for our SOAP client class, `CcServerAxisTest`, to run inside.
- Calls the `CcServerAxisTest` class (our simple SOAP client) with arguments for web server location and port. Notice that the rest of the URI has been hard-coded in this batch file.

When executed, the `testServerAxisTomcat.bat` opens a Windows console and displays the API menu, as shown below:

Figure 3: Portion of the testServerAxisTomcat.bat Windows console

```

C:\Program Files (x86)\Progress\Corticon 5.3\Server\Tomcat>echo off
Defining Corticon 5.3 runtime environment.

C:\Program Files (x86)\Progress\Corticon 5.3\Server\Tomcat>CALL "..\JRE\bin\ja
F\lib\commons-discovery.jar;..\webapps\axis\WEB-INF\lib\commons-logging.jar;..\w
R_JAVA" -DCORTICON_WORK_DIR="C:\Users\XXXXXXXX\Progress\CorticonWork_5.3 com.c

-----
Axis Servlet Location : http://localhost:8082
Execute Servlet (Message Style) : http://localhost:8082/axis/services/Corticon
Execute Servlet (RPC Style)      : http://localhost:8082/axis/services/Corticon
Admin Servlet                   : http://localhost:8082/axis/services/Corticon
Output Directory                 : C:\Users\XXXXXXXX\Progress\CorticonWork_5.3
  
```

The menu displayed in the Windows console is too large to fit on a single printed page, so it has been divided into two screenshots here. In the upper portion of the Windows console, shown in the figure above, the classpath definition process is visible. Once all classes are loaded, the Corticon Server starts up in the JRE, which is needed by our simple SOAP client class.

Figure 4: testServerAxis.bat 100 commands

```

C:\Windows\system32\cmd.exe

--- Current Apache Axis Location: http://localhost:8082

Transactions:
-1 - Exit Server Api Test

0 - Change Connection Parameters

101 - Add a Decision Service (3 parameters)
102 - Add a Decision Service (6 parameters)
103 - Add a Decision Service (9 parameters)

110 - Load CcServer with .cdd file
111 - Load CcServer files from directory

112 - Reload Decision Service
113 - Reload Decision Service (by specific Decision Service Major Version)
114 - Reload Decision Service (by specific Decision Service Major and Minor Ve

115 - Remove Decision Service
116 - Remove Decision Service (by specific Decision Service Major Version)
117 - Remove Decision Service (by specific Decision Service Major and Minor Ve

118 - Clear All Non-Cdd Decision Services

120 - Get Decision Service Names
121 - Get CcServer current info

130 - Execute SOAP Document Style (CorticonRequest Document)
131 - Execute SOAP RPC Style (CorticonRequest String)

150 - Precompile a Ruleflow into a .eds file
151 - Precompile a Ruleflow into a Database Access optimized .eds file

100 - Switch menu to Common Functions
200 - Switch menu to Decision Service Functions
300 - Switch menu to Monitoring Functions
400 - Switch menu to CcServer Functions

Enter transaction number:

```

In the lower portion of the Windows console, shown in the figure above, we see the available API methods of the **Common Functions** (the 100 series) listed by number. You can list the commands in the other series by entering their series number:

- Enter 200 to list the Decision Service Functions command set
- Enter 300 to list the Monitoring Functions command set
- Enter 400 to list the CcServer Functions command set
- Enter 100 to again list the Common Functions command set

Note: After you enter a transaction, the result is displayed followed by a restating of the current command set. You might need to scroll back a bit to see your results.

Since we have not deployed any Ruleflows yet, we will use an administrative method to test if Corticon Server is correctly installed as a SOAP Servlet inside our web server. A good administrative method to call is transaction #121, **Get CcServer current info**. This choice corresponds directly to the Java API method `getCcServerInfo()`, described in complete detail in the *JavaDocs* provided in the standard Corticon Server installation.

To try this, confirm that Tomcat is running, and then enter 121 in the `testServerAxisTomcat` command window. The `CcServerAxisTest` class makes a call to the server's SOAP Servlet. It asks for a list of configuration parameters and returns them to the Windows console. The results of the call are shown in the following figure:

Figure 5: testServerAxisTomcat Response to command 121

```

C:\Windows\system32\cmd.exe

Enter transaction number:
121
<?xml version="1.0" encoding="UTF-8"?>
<CcServerInfo CcLicensePath="C:\Program Files (x86)\Progress\Corticon 5.3\License\CcLice
  <LicenseInfo>
    <LicenseProperty name="LICENSE_OEM_NAME" value="Evaluation - Database Access" />
    <LicenseProperty name="LICENSE_DATE_GRANTED" value="" />
    <LicenseProperty name="LICENSE_MAX_POOLS" value="10" />
    <LicenseProperty name="LICENSE_MAX_REACTORS" value="5" />
    <LicenseProperty name="LICENSE_MAX_NUMBER_OF_RULES" value="200" />
    <LicenseProperty name="LICENSE_DEACTIVATE_DATE" value="6/1/2013" />
    <LicenseProperty name="LICENSE_DEACTIVATE_DATE_OVERRIDE" value="NO" />
    <LicenseProperty name="LICENSE_SINGLE_EXECUTION_ENABLED" value="YES" />
    <LicenseProperty name="LICENSE_BATCHPROCESSING_ENABLED" value="YES" />
    <LicenseProperty name="LICENSE_USAGE_ENFORCED" value="false" />
    <LicenseProperty name="LICENSE_USAGE_NAME" value="Corticon" />
    <LicenseProperty name="LICENSE_SERVER_IP" value="0.0.0.0" />
    <LicenseProperty name="LICENSE_INCREMENT_IP" value="0" />
    <LicenseProperty name="LICENSE_IP_OVERRIDE" value="YES" />
    <LicenseProperty name="LICENSE_SERVER_TRANSACTIONS_TIME_PERIOD" value="60000" />
    <LicenseProperty name="LICENSE_SERVER_TRANSACTIONS_MAX_EXECUTIONS" value="100" />
    <LicenseProperty name="LICENSE_SERVER_TRANSACTION_OVERRIDE" value="YES" />
  </LicenseInfo>
  <CDDs />
  <NonCDDs />
  <RegisteredTrackingAttributes />
  <ServiceStartups />
  <LoggingStartups />
</CcServerInfo>

Transaction completed.
  
```

Note: Your output might vary when sample services have been deployed.

The response verifies that our Corticon Server is running correctly as a SOAP Servlet and is listening for, and responding to, calls. At this stage in the deployment, this is all we want to verify.

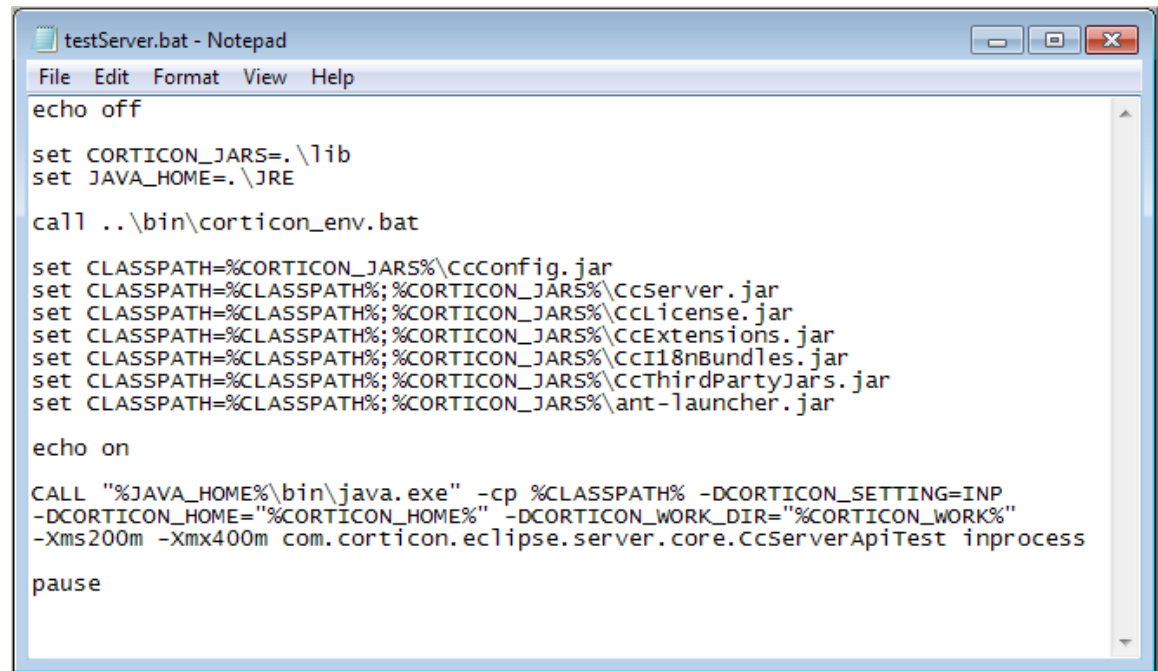
As in-process Java classes

The batch file `testServer.bat` is located in the `[CORTICON_HOME]\Server` where `CORTICON_HOME` is your installation location, and is designed to perform the basic in-process initialization of a Corticon Server, and then present a menu of API methods you can invoke from within a Windows console.

Viewing `testServer.bat` with a text editor, you can see how it sets classpaths, starts the JVM and then invokes the `CcServerApiTest` class. The source code (`.java`) for this class is provided in the `[CORTICON_HOME]\Server\src` directory. It is a good reference to use when you want to see exactly how the Corticon Server API set is used in your own code.

In addition to the several base Corticon Server JARs listed in [Installing Corticon Server as Java classes in-process or in a custom java container](#) section, the batch file also loads some hard-coded Java business objects for use with the Java Object Messaging commands 132-141 in [Bottom Portion of the testServer.bat Windows console](#)). These hard-coded classes are included in `CcServer.jar` so as to ensure their inclusion on the JVM's classpath whenever `CcServer.jar` is loaded. The hard-coded Java objects are intended for use when invoking the `OrderProcessing.erf` Decision Service included in the default Corticon Server installation.

Figure 6: testServer.bat



```
testServer.bat - Notepad
File Edit Format View Help
echo off

set CORTICON_JARS=.\lib
set JAVA_HOME=.\JRE

call ..\bin\corticon_env.bat

set CLASSPATH=%CORTICON_JARS%\CcConfig.jar
set CLASSPATH=%CLASSPATH%;%CORTICON_JARS%\CcServer.jar
set CLASSPATH=%CLASSPATH%;%CORTICON_JARS%\CcLicense.jar
set CLASSPATH=%CLASSPATH%;%CORTICON_JARS%\CcExtensions.jar
set CLASSPATH=%CLASSPATH%;%CORTICON_JARS%\Cci18nBundles.jar
set CLASSPATH=%CLASSPATH%;%CORTICON_JARS%\CcThirdPartyJars.jar
set CLASSPATH=%CLASSPATH%;%CORTICON_JARS%\ant-launcher.jar

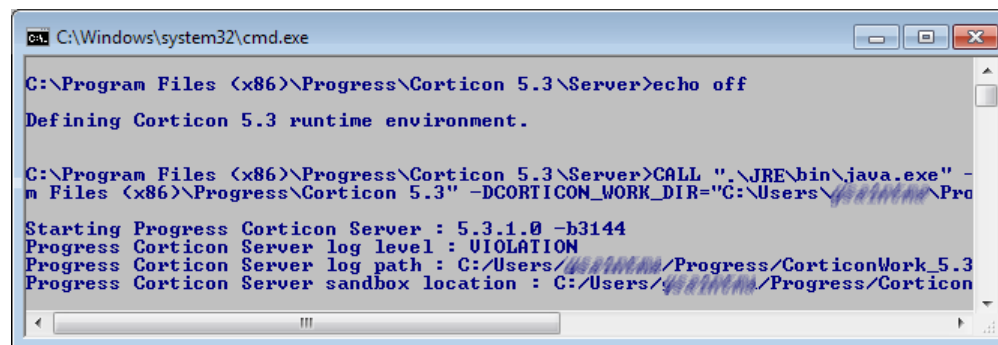
echo on

CALL "%JAVA_HOME%\bin\java.exe" -cp %CLASSPATH% -DCORTICON_SETTING=INP
-DCORTICON_HOME="%CORTICON_HOME%" -DCORTICON_WORK_DIR="%CORTICON_WORK%"
-Xms200m -Xmx400m com.corticon.eclipse.server.core.CcServerApiTest inprocess

pause
```

When executed, the `testServer.bat` opens a Windows console and displays the API menu, as shown below:

Figure 7: Top Portion of the testServer.bat Windows console



```
C:\Windows\system32\cmd.exe

C:\Program Files (x86)\Progress\Corticon 5.3\Server>echo off
Defining Corticon 5.3 runtime environment.

C:\Program Files (x86)\Progress\Corticon 5.3\Server>CALL ".\JRE\bin\java.exe" -
m Files (x86)\Progress\Corticon 5.3" -DCORTICON_WORK_DIR="C:\Users\%username%\Pro

Starting Progress Corticon Server : 5.3.1.0 -b3144
Progress Corticon Server log level : VIOLATION
Progress Corticon Server log path : C:/Users/%username%/Progress/CorticonWork_5.3
Progress Corticon Server sandbox location : C:/Users/%username%/Progress/Corticon
```

The menu displayed in the Windows console is too large to fit on a single printed page, so it has been divided into two screenshots here. In the upper portion of the Windows console, shown in the figure above, the class loading process is visible. Once all classes are loaded, Corticon Server starts up in the JVM.

Figure 8: Bottom Portion of the testServer.bat Windows console

```

C:\Windows\system32\cmd.exe

Transactions:
-1 - Exit Server Api Test

-----
101 - Add a Decision Service <3 parameters>
102 - Add a Decision Service <6 parameters>
103 - Add a Decision Service <9 parameters>
-----
110 - Load CcServer with .cdd file
111 - Load CcServer files from directory
-----
112 - Reload Decision Service
113 - Reload Decision Service <by specific Decision Service Major Version>
114 - Reload Decision Service <by specific Decision Service Major and Minor Version>
-----
115 - Remove Decision Service
116 - Remove Decision Service <by specific Decision Service Major Version>
117 - Remove Decision Service <by specific Decision Service Major and Minor Version>
-----
118 - Clear All Non-Cdd Decision Services
-----
120 - Get Decision Service Names
121 - Get CcServer current info
-----
130 - Execute using a JDOM Document <CorticonRequest Document>
131 - Execute using a XML String <CorticonRequest String>
-----
132 - Execute using a hard-coded set of Business Objects <Collection>
133 - Execute using a hard-coded set of Business Objects <Collection> <by specific Decision Ser
134 - Execute using a hard-coded set of Business Objects <Collection> <by specific Decision Ser
135 - Execute using a hard-coded set of Business Objects <Collection> <by specific execution Da
136 - Execute using a hard-coded set of Business Objects <Collection> <by specific execution Da
-----
137 - Execute using a hard-coded set of Business Objects <HashMap>
138 - Execute using a hard-coded set of Business Objects <HashMap> <by specific Decision Servic
139 - Execute using a hard-coded set of Business Objects <HashMap> <by specific Decision Servic
140 - Execute using a hard-coded set of Business Objects <HashMap> <by specific execution Date>
141 - Execute using a hard-coded set of Business Objects <HashMap> <by specific execution Date
-----
150 - Precompile a Ruleflow into a .eds file
151 - Precompile a Ruleflow into a Database Access optimized .eds file
-----
100 - Switch menu to Common Functions
200 - Switch menu to Decision Service Functions
300 - Switch menu to Monitoring Functions
400 - Switch menu to CcServer Functions
-----

Enter transaction number:

```

In the lower portion of the Windows console, shown in the figure above, we see the available API methods of the **Common Functions** (the 100 series) listed by number. You can list the commands in the other series by entering their series number:

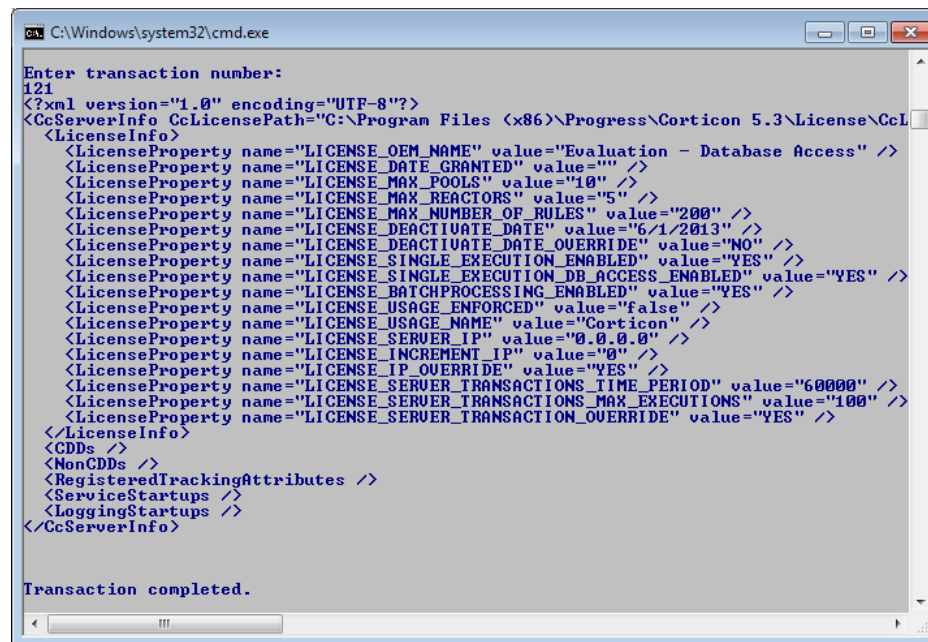
- Enter 200 to list the Decision Service Functions command set
- Enter 300 to list the Monitoring Functions command set
- Enter 400 to list the CcServer Functions command set
- Enter 100 to again list the Common Functions command set

Note: After you enter a transaction, the result is displayed followed a restating of the current command set. You might need to scroll back a bit to see your results.

Since we have not deployed any Rulesheets yet, we will need to use an administrative method to test if Corticon Server is loaded in-process correctly. A good administrative method to call is option #121, **Get CcServer Info**. This choice corresponds directly to the Java API method `getCcServerInfo()` described in complete detail in the *JavaDoc*.

To try this, enter 121 in the `testServerTomcat` command window. The `CcServerApiTest` class makes a call to the Corticon Server running in-process. It asks for a list of configuration parameters and returns them to the Windows console. The results of the call are shown in the following figure:

Figure 9: testServer.bat Response



```

C:\Windows\system32\cmd.exe
Enter transaction number:
121
<?xml version="1.0" encoding="UTF-8"?>
<CcServerInfo CcLicensePath="C:\Program Files (x86)\Progress\Corticon 5.3\License\CcL
  <LicenseInfo>
    <LicenseProperty name="LICENSE_OEM_NAME" value="Evaluation - Database Access" />
    <LicenseProperty name="LICENSE_DATE_GRANTED" value="" />
    <LicenseProperty name="LICENSE_MAX_POOLS" value="10" />
    <LicenseProperty name="LICENSE_MAX_REACTORS" value="5" />
    <LicenseProperty name="LICENSE_MAX_NUMBER_OF_RULES" value="200" />
    <LicenseProperty name="LICENSE_DEACTIVATE_DATE" value="6/1/2013" />
    <LicenseProperty name="LICENSE_DEACTIVATE_DATE_OVERRIDE" value="NO" />
    <LicenseProperty name="LICENSE_SINGLE_EXECUTION_ENABLED" value="YES" />
    <LicenseProperty name="LICENSE_SINGLE_EXECUTION_DB_ACCESS_ENABLED" value="YES" />
    <LicenseProperty name="LICENSE_BATCHPROCESSING_ENABLED" value="YES" />
    <LicenseProperty name="LICENSE_USAGE_ENFORCED" value="false" />
    <LicenseProperty name="LICENSE_USAGE_NAME" value="Corticon" />
    <LicenseProperty name="LICENSE_SERVER_IP" value="0.0.0.0" />
    <LicenseProperty name="LICENSE_INCREMENT_IP" value="0" />
    <LicenseProperty name="LICENSE_IP_OVERRIDE" value="YES" />
    <LicenseProperty name="LICENSE_SERVER_TRANSACTIONS_TIME_PERIOD" value="60000" />
    <LicenseProperty name="LICENSE_SERVER_TRANSACTIONS_MAX_EXECUTIONS" value="100" />
    <LicenseProperty name="LICENSE_SERVER_TRANSACTION_OVERRIDE" value="YES" />
  </LicenseInfo>
  <CDDs />
  <NonCDDs />
  <RegisteredTrackingAttributes />
  <ServiceStartups />
  <LoggingStartups />
</CcServerInfo>

Transaction completed.

```

Note: Your output might vary when sample services have been deployed.

`testServer.bat` didn't load any Decision Services, so Corticon Server is basically replying with an empty status message. But the important thing is that we have verified that Corticon Server is running correctly in-process and is listening for, and responding to, calls. At this stage in the deployment, this is all we want to verify.

Deploying a Ruleflow to the Corticon Server

Just because the Corticon Server is running does not mean it is ready to process transactions. It must still be "loaded" with one or more Ruleflows. Once a Ruleflow has been loaded, or deployed, to the Corticon Server we call it a Decision Service because it is a service ready and able to make decisions for any external application or process ("client") that requests the service properly.

Loading the Corticon Server with Ruleflows can be accomplished in four ways:

- **Publish Wizard** - The easiest method, introduced in Corticon 5.3. It is discussed in detail in the *Publish and Download Wizards* section of the *Deploying Corticon Ruleflows* chapter of the *Integration & Deployment Guide*.
- **Deployment Descriptor files** - An easy method, and the one we will use in this Tutorial.
- **Deployment using the Server Web Console** - Another easy way to load Decision Services. It is discussed in detail in the Monitoring Server chapter of the *Integration & Deployment Guide*.
- **Java APIs** - This method requires more knowledge of the Java programming language, and is not discussed in this Tutorial.

All four methods are described more thoroughly in the *Server Integration & Deployment Guide*.

For details, see the following topics:

- [Creating a Ruleflow](#)
- [Creating and installing a Deployment Descriptor file](#)

Creating a Ruleflow

You created a Ruleflow suitable for deployment in the *Corticon Studio Tutorial: Advanced Rule Modeling*, that is ready for deployment to the Corticon Server.

You could also use a simple one (a single Rulesheet) that was installed in the *Cargo* tutorial files, `tutorial_example.erf` located in `[CORTICON_WORK_DIR]\Samples\Rule Projects\Tutorial\Tutorial-Done`.

To create a new Ruleflow, see the topics *"Ruleflows" in the Quick Reference Guide*.

Creating and installing a Deployment Descriptor file

A Deployment Descriptor file tells the Corticon Server which Ruleflows to load and how to handle transaction requests for those Ruleflows. A Deployment Descriptor file has the suffix `.cdd`, and we will often simply refer to it as a `.cdd` file.

Important: The `.cdd` file "points" at the Ruleflow via a path name - a name can contain spaces but it is a good practice to avoid spaces and special characters except for underscore (`_`) character.

Deployment Descriptors are easily created using the Deployment Console, which is installed only by the Corticon Server installers.

Using the Deployment Console tool's Decision Services on the Java Server

The Corticon Deployment Console is started, as follows:

- **Java Server:** On the Windows Start menu, choose **Programs > Progress > Corticon n.n > Deployment Console** to launch the script file `\Server\deployConsole.bat`.

The Deployment Console is divided into two sections. Because the Deployment Console is a rather wide window, its columns are shown as two screen captures in the following figures. The **red** identifiers are the topics listed below.

Figure 10: Left Portion of Deployment Console, with Deployment Descriptor File Settings Numbered

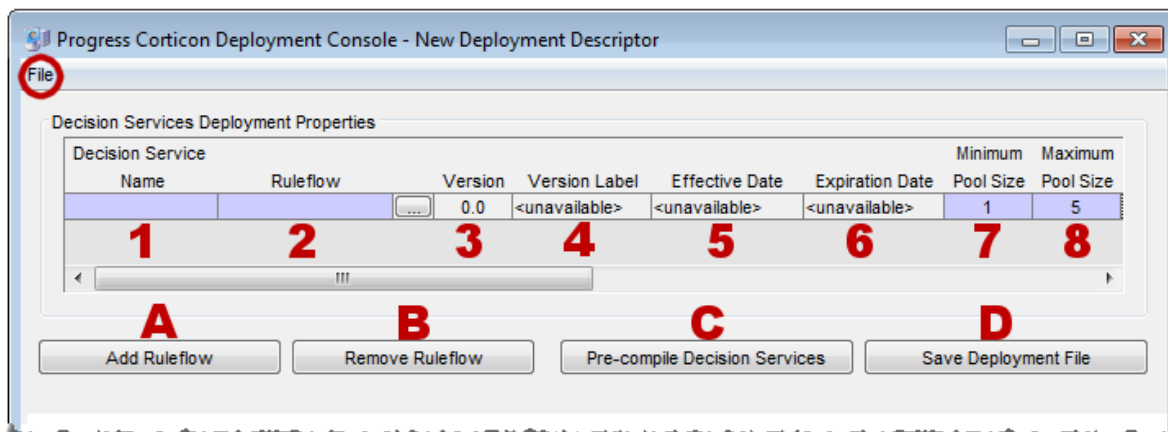
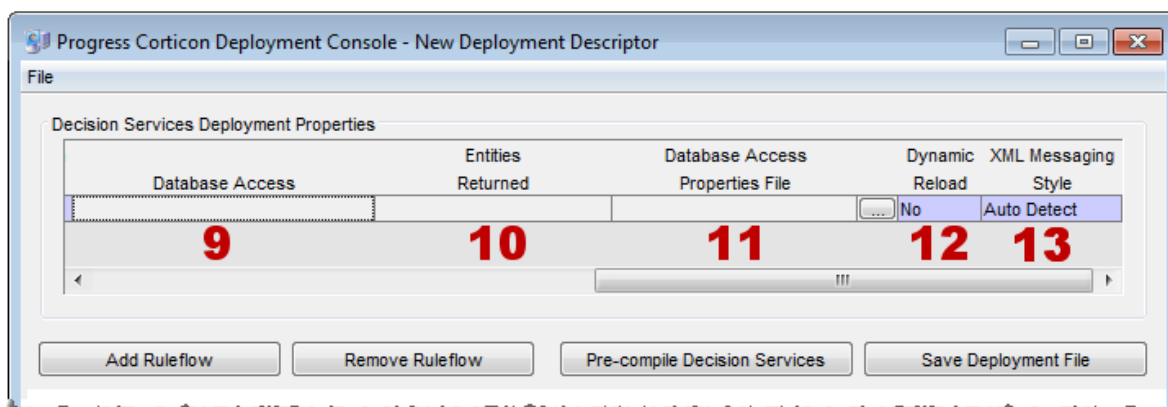


Figure 11: Right Portion of Deployment Console, with Deployment Descriptor File Settings Numbered




The name of the open Deployment Descriptor file is displayed in the Deployment Console's title bar.

The **File** menu, circled in the top figure, enables management of Deployment Descriptor files:

- To save the current file, choose (**File > Save**).
- To open an existing .cdd, choose (**File > Open**).
- To save a .cdd under a different name, choose (**File > Save As**).

The marked steps below correspond to the Deployment Console columns for each line in the Deployment Descriptor.

1. **Decision Service Name** - A unique identifier or label for the Decision Service. It is used when invoking the Decision Service, either via an API call or a SOAP request message. See [Invoking Corticon Server](#) for usage details.
2. **Ruleflow** - All Ruleflows listed in this section are part of this Deployment Descriptor file. Deployment properties are specified on each Ruleflow. Each row represents one Ruleflow. Use the  button to navigate to a Ruleflow file and select it for inclusion in this Deployment


Descriptor file. Note that Ruleflow *absolute* pathnames are shown in this section, but *relative* pathnames are included in the actual `.cdd` file.

The term "deploy", as we use it here, means to "inform" the Corticon Server that you intend to load the Ruleflow and make it available as a Decision Service. It does **not** require actual physical movement of the `.erf` file from a design-time location to a runtime location, although you may do that if you choose – just be sure the file's path is up-to-date in the Deployment Descriptor file. But movement isn't required – you can save your `.erf` file to any location in a file system, and also deploy it from the same place *as long as the running Corticon Server can access the path*.

3. **Version** - the version number assigned to the Ruleflow in the **Ruleflow > Properties** window of Corticon Studio. Note that this entry is editable only in Corticon Studio and not in the Deployment Console. A discussion of how Corticon Server processes this information is found in the topics *"Decision Service Versioning and Effective Dating" of the Integration and Deployment Guide*. Also see the *Quick Reference Guide* for a brief description of the Ruleflow Properties window and the *Rule Modeling Guide* for details on using the Ruleflow versioning feature. It is displayed in the Deployment Console simply as a convenience to the Ruleflow deployer.
4. **Version Label** - the version label assigned to the Ruleflow in the **Ruleflow > Properties** window of Corticon Studio. Note that this entry is editable only in Corticon Studio and not in the Deployment Console. See the **Quick Reference Guide** for a brief description of the Ruleflow Properties window and the purpose of the Ruleflow versioning feature.
5. **Effective Date** - The effective date assigned to the Ruleflow in the **Ruleflow > Properties** window of Corticon Studio. Note that this entry is editable only in Corticon Studio and not in the Deployment Console. A discussion of how Corticon Server processes this information is found in the topics *"Decision Service Versioning and Effective Dating" of the Integration and Deployment Guide*. Also see the *Quick Reference Guide* for a brief description of the Ruleflow Properties window and the purpose of the Ruleflow effective dating feature.
6. **Expiration Date** - The expiration date assigned to the Ruleflow in the **Ruleflow > Properties** window of Corticon Studio. Note that this entry is editable only in Corticon Studio and not in the Deployment Console. A discussion of how Corticon Server processes this information is found in the topics *"Decision Service Versioning and Effective Dating" of the Integration and Deployment Guide*. Also see the *Quick Reference Guide* for a brief description of the Ruleflow Properties window and the purpose of the Ruleflow expiration dating feature.
7. **Minimum Pool Size** - The minimum number of instances or 'copies' created for a Decision Service when it is loaded by Corticon Server. Instances of a Decision Service are known as **Reactors** - These Reactors are placed in a pool, where they wait for assignment by Corticon Server to an incoming request, or they expire due to inactivity. The larger the pool size, the greater the concurrency (but greater the memory usage). The default value is **1**, which means that even under no load (no incoming requests) Corticon Server will always maintain one Reactor in the pool for this Decision Service.
8. **Maximum Pool Size** - The maximum number of Reactors Corticon Server can put into the pool for this Decision Service. Therefore, the number of Reactors that can execute concurrently is determined by the max pool size. If additional requests for the Decision Service arrive when all Reactors are busy, the new requests queue until Corticon Server can allocate a Reactor to the new transaction (usually right after a Reactor is finished with its current transaction). The more Reactors in the pool, the greater the concurrency (and the greater the memory usage). See [Performance and Tuning](#) chapter for more guidance on Pool configuration. The default value is **5**.

Note: Functions 9, 10, and 11 are active only if your Corticon license enables EDC, and you have registered its location in tool.

Note: If you are evaluating Corticon, your license requires that you set the parameter to 1.

9. **Database Access** - Controls whether the deployed Rule Set has direct access to a database, and if so, whether it will be read-only or read-write access.
10. **Entities Returned** - Determines whether the Corticon Server response message should include all data used by the rules including data retrieved from a database (**All Instances**), or only data provided in the request and created by the rules themselves (**Incoming/New Instances**).
11. **Database Access Properties File** - The path and filename of the database access properties file (that was typically created in Corticon Studio) to be used by Corticon Server during runtime database access. Use the adjacent  button to navigate to a database access properties file.
12. **Dynamic Reload** - If **Yes**, then Corticon Server will periodically look to see if a Deployment Descriptor file, or any of the Decision Service entries in that file, has changed since the `.cdd` was last loaded. If so, it will be automatically reloaded. The time interval between checks is defined by property `com.corticon.ccserver.serviceIntervals` in [CcServer.properties](#). Even if **No**, Corticon Server will still use the most recent Ruleflow when it adds new Reactors into the pool.
13. **XML Messaging Style** - Determines whether request messages for this Decision Service should contain a flat (**Flat**) or hierarchical (**Hier**) payload structure. The [Decision Service Contract Structures](#) section of the Integration chapter provides samples of each. If set to **Auto Detect**, then Corticon Server will accept either style and respond in the same way.

The indicated buttons at the bottom of the Decision Service Deployment Properties section provide the following functions:

- **(A) Add Ruleflow** - Creates a new line in the Decision Service Deployment Properties list. There is no limit to the number of Ruleflows that can be included in a single Deployment Descriptor file.
- **(B) Remove Ruleflow** - Removes the selected row in the Decision Service Deployment Properties list.
- **(C) Pre-compile Decision Services** - Compiles the Decision Service before deployment, and then puts the `.eds` file (which contains the compiled executable code) at the location you specify. (By default, Corticon Server does not compile Ruleflows *until* they are deployed to Corticon Server. Here, you choose to pre-compile Ruleflows in advance of deployment.) The `.cdd` file will contain reference to the `.eds` instead of the usual `.erf` file. Be aware that setting the EDC properties will optimize the Decision Service for EDC.
- **(D) Save Deployment File** - Saves the `.cdd` file. (Same as the menu **File > Save** command.)

Installing the Deployment Descriptor file

Once Corticon Server has been installed and deployed to Tomcat, the included startup scripts ensure the following sequence occurs upon launching Tomcat:

Note: If you are using an evaluation license, the Maximum Pool Size in the Ruleflow you are deploying must be exactly 1. Any other value will issue an alert that you have exceeded the allowed number of reactors.

1. The Tomcat web server starts up.
2. Corticon Server starts up as a web service in Tomcat's Servlet container.
3. Corticon Server looks for Deployment Descriptor files in a specific directory.
4. Corticon Server loads into memory the Ruleflow(s) referenced by the Deployment Descriptor files, and creates Reactors for each according to their minimum pool size settings. At this stage, we say that the Ruleflows have become Decision Services because they are now usable by external applications and clients.

In order for the Corticon Server to find Deployment Descriptor files when it looks in step 3, we must ensure that the `.cdd` files are moved to the appropriate location. In the default installation used in this Tutorial, that location is the `[CORTICON_WORK_DIR]\cdd` directory. In the future, when creating `.cdd` files, you may want to save them straight to this directory so they become immediately accessible to the default Corticon Server deployed in this Tutorial.

Of course, this location is fully configurable. See the Deploying Corticon Ruleflows chapter of the *Server Integration & Deployment Guide* for more details.

Now, when the startup sequence reaches step 3 above, Corticon Server will "know" where all Ruleflows are located because `.cdd` files contain their pathnames.

Hot re-deploying Deployment Descriptor files and Ruleflows

Changes to a Deployment Descriptor file or any of the Ruleflows it references do **not** require restarting Tomcat. A maintenance thread in the Corticon Server watches for additions, deletions, and changes and updates appropriately. A Ruleflow can be modified in Corticon Studio even while it is also simultaneously deployed as a Decision Service and involved in a transaction - Corticon Server can be configured to update the Decision Service dynamically for the very next transaction.

Dynamic updating of deployed Ruleflows is not normally used in production environments because standard IT change control processes require a more disciplined and controlled deployment process. But in development or testing environments, it can be convenient to allow dynamic updates so that Ruleflow changes can be deployed more quickly.

Having selected `No` for the **Dynamic Reload** setting earlier, our `tutorial_example` Decision Service will not update automatically when the `.erf` file is changed. To enable this automatic refresh, choose `Yes` for the **Dynamic Reload** setting.

Consuming a decision service

Let's review what we have accomplished so far:

1. We have installed Corticon Server for Java onto the target machine.
2. We have deployed Corticon Server as a web service onto the bundled Tomcat web server.
3. We have used the **Deployment Console** to generate a Deployment Descriptor file for our sample *Ruleflow*.
4. We have installed the Deployment Descriptor file in the location where Corticon Server looks when it starts.

Now we are ready to consume this Decision Service by sending a real XML/SOAP "request" message and inspecting the "response" message it returns.

For details, see the following topics:

- [Integrating and testing a Decision Service](#)
- [Path 1- Using Corticon Studio as a SOAP client to consume a Decision Service](#)
- [Path 2 - Using bundled sample code to consume a Decision Service](#)
- [Path 3 - Using SOAP client to consume a Decision Service](#)
- [Troubleshooting](#)

Integrating and testing a Decision Service

In order to use a Decision Service in a process or application, it is necessary to understand the Decision Service's service contract, also known as its interface. A service contract describes in precise terms the kind of input a Decision Service is expecting, and the kind of output it returns following processing. In other words, a service contract describes how to *integrate* with a Decision Service.

When an external process or application sends a request message to a Decision Service that complies with its service contract, the Decision Service receives the request, processes the included data, and sends a response message. When a Decision Service is used in this manner, we say that the external application or process has successfully "consumed" the Decision Service.

This Tutorial describes three paths for consuming a Decision Service:

- [Path 1](#)

Use Corticon as a SOAP client to send and receive SOAP messages to a Decision Service running on a remote Corticon Server - This is different from testing Ruleflows in Corticon "locally." This path is the easiest method to use and requires the least amount of technical knowledge to successfully complete. If you have already installed Corticon Studio, then you have all necessary components to complete this path. If not but want to follow this path, we recommend completing the *Corticon Studio: Installation Guide* and the *Corticon Studio Tutorial: Basic Rule Modeling* before continuing on this path.

- [Path 2](#)

Manually integrate and test a Decision Service - In this path, we will use bundled sample code (a command file) to send a request message built in Corticon Studio's Tester, and display the results. This path requires more technical knowledge and confidence to complete, but illustrates some aspects of the software which may be interesting to a more technical audience. If you have already installed Corticon Studio, then you have all necessary components to complete this path. If not but want to follow this path, we recommend completing the *Studio Installation Guide* and the *Basic Tutorial for Corticon Studio – Modeling Rules* before continuing on this path.

- [Path 3](#)

Use a commercially available SOAP client to integrate with and test a Decision Service - In other words, this SOAP client will read a web-services-standard service contract (discussed below), generate a request message from it, send it to the Corticon Server and display the response message. Progress Corticon does not include such an application, so the reader must obtain one in order to complete this path.

Path 1- Using Corticon Studio as a SOAP client to consume a Decision Service

In this path, we will use Corticon Studio as a SOAP client to execute Decision Services running on a remote Corticon Server.

Creating a new Java server test in Corticon Studio

Return to Corticon Studio, or reopen it if closed. Open `Cargo.ecore` and then, *without opening any Ruleflows*, open a new Test by selecting **File>New>Ruletest** from the Corticon Studio menubar.

For the Ruletest creation process outlined below, see also [Requesting List of Remote Decision Services](#):

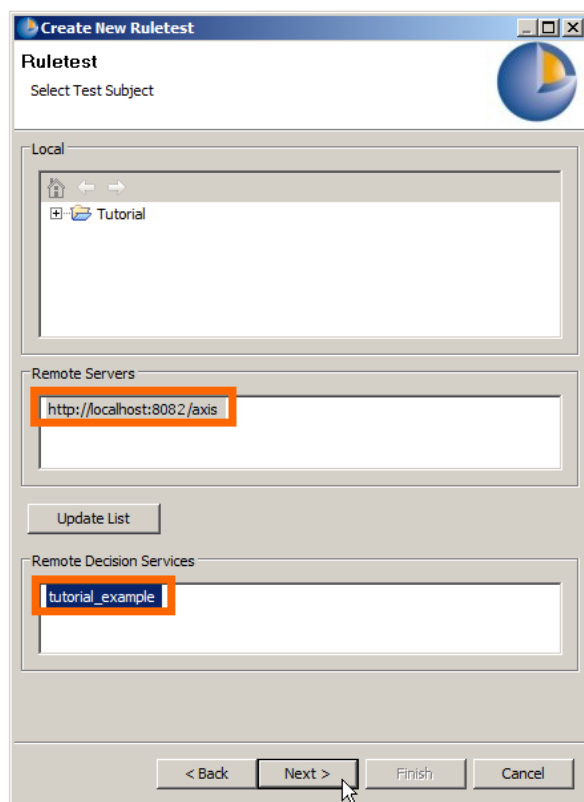
1. Specify a filename for your Ruletest, and then click **Next**.
2. You will be asked to **Select Test Subject**. Be sure to select the <http://localhost:8082/axis> in the **Remote Servers** box.
3. Select **Update List**. Corticon Studio will attempt to contact a Corticon Server instance at the location specified above. If a Corticon Server instance is running, it will respond with a list of available Decision Services, and display that list in the **Remote Decision Services** window.
4. Choose the Decision Service to invoke. In this Tutorial, we want `tutorial_example`.
5. Click **Next>**
6. Select the Vocabulary to use, as per normal Ruletest creation procedure.

Important: Remember, even though we are using Corticon Studio to test, we are using its *remote* testing feature, which executes a Decision Service running on Corticon Server ("remotely"), not a Ruleflow open in Corticon Studio ("locally").

To keep this distinction clear, we are **not** going to open `tutorial_example.erf` in Corticon Studio – it is not necessary since we're really testing the Decision Service running on Corticon Server.

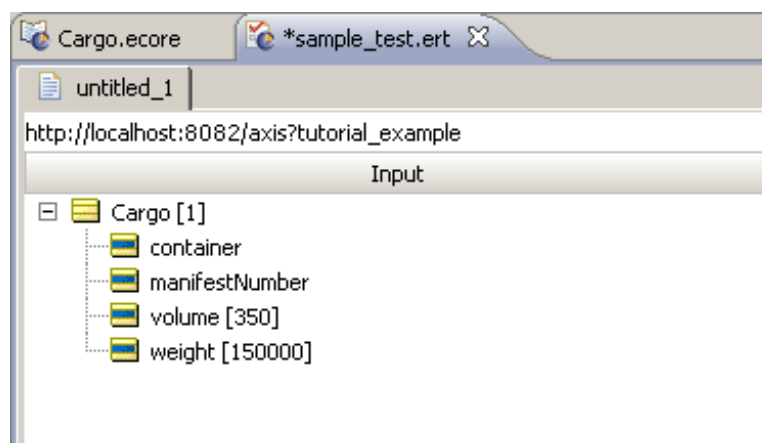
In step 1, you selected the default URL: Corticon Server running on `localhost`. If you want to change the URL to another address, see "Designer properties & settings" in *Server Integration & Deployment Guide* for more information about configuring Corticon Studio properties.

Figure 12: Requesting List of Remote Decision Services



Now, drag a `Cargo` entity from the Vocabulary to the Input pane. Enter sample data as shown:

Figure 13: Sample Data in a Studio Remote Ruletest

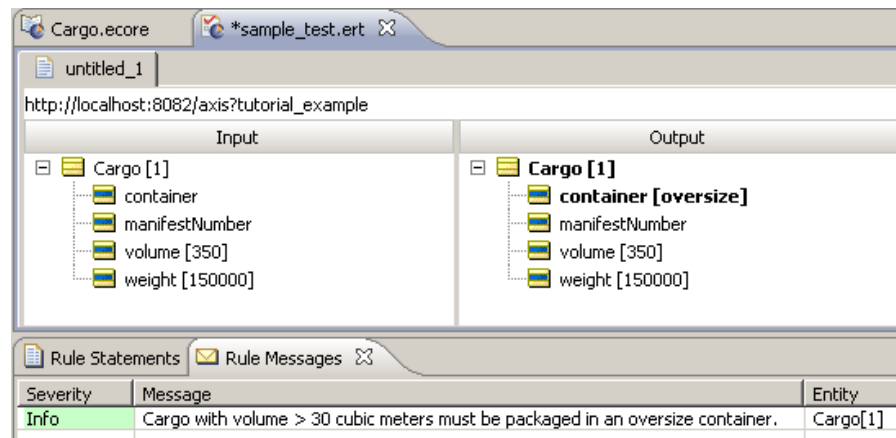


Executing the remote test

Execute the Test by selecting **Ruletest > Testsheet > Run Test** from the Corticon Studio menubar or  from the toolbar.

We should see an Output pane similar to the following:

Figure 14: Response from Remote Decision Service



The Output pane of the Testsheet shown above displays the response message returned by the Corticon Server. This confirms that our Decision Service has processed the data contained in the request and sent back a response containing new data (the `container` attribute and the message).

Path 2 - Using bundled sample code to consume a Decision Service

Creating a Request Message for a Decision Service

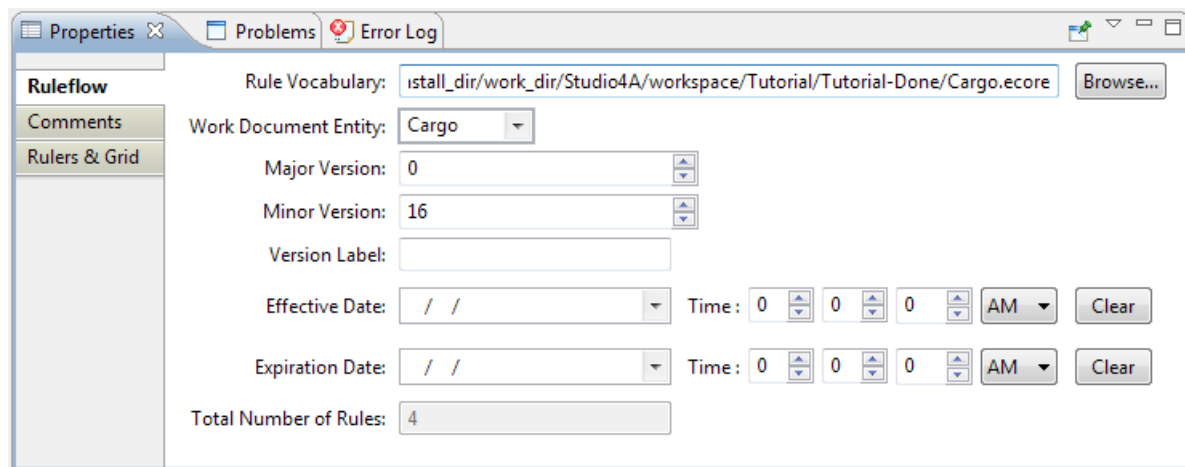
In this path, we will use a feature of Corticon Studio to generate a request message directly. The steps to accomplish this are:

Note: This section uses Service Contracts and Work Document Entities. For more on these functions, see *"Service contracts" in the Integration and Deployment Guide*.

1. Open Corticon Studio.
2. Open the Ruleflow you have deployed as a Decision Service. If you are using the Tutorial example, this is `tutorial_example.erf`.

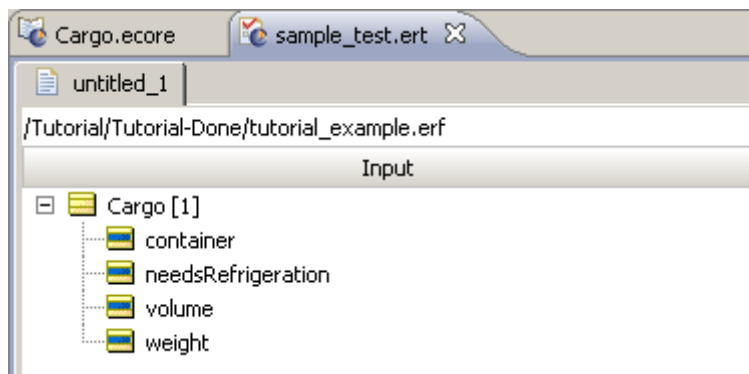
- Set a work document entity (WDE) for the Ruleflow. The WDE defines the "root" element in the XML document. Choose the menu command **Ruleflow > Properties**. The **Properties** tab typically opens in the bottom center of the Studio. If you are using `tutorial_example.erf`, the appropriate WDE is `Cargo`, as shown:

Figure 15:



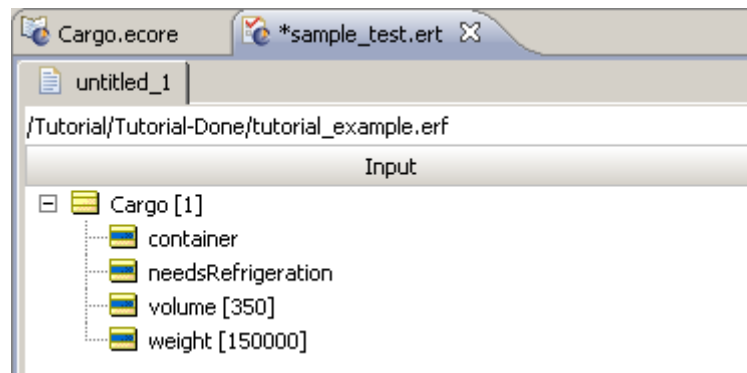
- Create a new Ruletest by following the procedure outlined in Option 1 above. For Test Subject, you can choose either your local or remote `tutorial_example.erf`.
- Create an Input test tree manually as in Option 1 above, or use menubar option **Ruletest>Testsheet>Data>Input>Generate Data Tree**, which produces the structure of a request message in the Input pane. One `Cargo` entity should appear in the Input pane, as shown below:

Figure 16: A New Test



- Enter data into the Input Testsheet as you did when testing the Ruleflow in the *Corticon Studio Tutorial: Basic Rule Modeling*. Your Input Testsheet will now look similar to the following:

Figure 17: Test with Data



- Use Corticon Studio's menubar option **Ruletest>Testsheet>Data>Input>Export Request XML** to export this Input pane as an XML document. We will use this exported XML document as the body or "payload" of our request message. Give the export file a name and remember where you save it. We will assign a filename of `sample.xml` for purposes of this Tutorial.
- Open `sample.xml` in any text editor. It should look very similar to the following figure:

Figure 18: Sample XML File Exported from a Studio Test

```

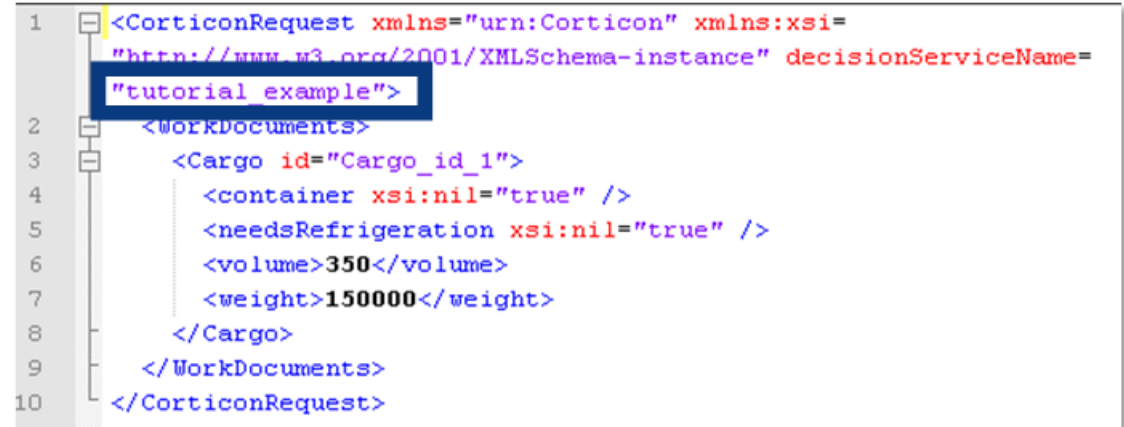
1  <?xml version="1.0" encoding="UTF-8"?>
2  <CorticonRequest xmlns="urn:Corticon" xmlns:xsi=
   "http://www.w3.org/2001/XMLSchema-instance" decisionServiceName=
   "InsertDecisionServiceName">
3    <WorkDocuments>
4      <Cargo id="Cargo_id_1">
5        <container xsi:nil="true" />
6        <needsRefrigeration xsi:nil="true" />
7        <volume>350</volume>
8        <weight>150000</weight>
9      </Cargo>
10   </WorkDocuments>
11 </CorticonRequest>

```

- Modify `sample.xml` by deleting the `<?XML version="1.0" encoding="UTF-8"?>` tag from the very top (this will be added automatically by the bundled sample code we will use to send this as a request message to the Decision Service). This tag is shown above, enclosed in an **orange box**.

10. Change the `decisionServiceName` attribute value in the `CorticonRequest` element from `InsertDecisionServiceName` to the service name of the Decision Service as it was defined in your deployed `.cdd` file. In our example, this name is `tutorial_example`. This piece is shown in the figures above and below (before and after the changes) enclosed in a **blue box**. Your `sample.xml` file should now look like this:

Figure 19: Sample XML File with Changes Made



```
1 <CorticonRequest xmlns="urn:Corticon" xmlns:xsi="
  "http://www.w3.org/2001/XMLSchema-instance" decisionServiceName=
  "tutorial_example">
2   <WorkDocuments>
3     <Cargo id="Cargo_id_1">
4       <container xsi:nil="true" />
5       <needsRefrigeration xsi:nil="true" />
6       <volume>350</volume>
7       <weight>150000</weight>
8     </Cargo>
9   </WorkDocuments>
10 </CorticonRequest>
```

11. **Save** your changes to the XML file and exit your text editor.

Sending a request message to Corticon Server

We include a basic command script in the default Corticon Server installation. The command script causes a test XML file to be enclosed ("wrapped") in a SOAP envelope and passed to a target Decision Service on the Corticon Server. The SOAP response from the Decision Service (that is, the Decision Service's response message) is then displayed to the user in the Command Prompt window where the test was launched. The steps required for using this command script are described below.

1. Open the `TestDS.cmd` file, found in `[CORTICON_HOME]\Server\Tomcat\CcServer\cddinit`, in a text editor. It will be similar to the following:

Figure 20: TestDS Command File

```
1  echo off
2
3  SET AXIS_JARS=..\..\webapps\axis\WEB-INF\lib
4  SET JAVA_HOME=..\..\..\JRE
5
6  SET CLASSPATH=%AXIS_JARS%\CcConfig.jar
7  SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\CcI18nBundles.jar
8  SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\CcServer.jar
9  SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\CcThirdPartyJars.jar
10 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\axis.jar
11 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\commons-discovery.jar
12 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\commons-logging.jar
13 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\jaxrpc.jar
14 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\saaj.jar
15 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\activation-1.2.jar
16 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\mail-1.2.jar
17 SET CLASSPATH=%CLASSPATH%;%AXIS_JARS%\soap-2.3.1.jar
18
19 echo on
20
21 "%JAVA_HOME%\bin\java.exe" -cp %CLASSPATH% com
   .corticon.soap.client.util.CcMessageHandler "EXECUTE"
   "http://localhost:8080/axis/services/Corticon"
   "OrderProcessingPayload.xml" true"
22
23 PAUSE
```

2. Replace `OrderProcessingPayload.xml` (shown in **orange box** in the figure above) with the full path of your test XML file. In this Tutorial, our test XML file is named `sample.xml`.
3. Save the changes to `TestDS.cmd`.

4. Double-click on the command file to run it.
5. You should see the following response from your Decision Service:

Figure 21: Corticon Response Message Displayed in Command Prompt Window

```

Command Prompt - testds
RESPONSE:
-----
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:
s:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSc
hema-instance">
  <soapenv:Body>
    <ns1:CorticonResponse xmlns:ns1="urn:Corticon" xmlns="urn:Corticon" decision
ServiceName="tutorial_example">
      <ns1:WorkDocuments>
        <ns1:Cargo id="Cargo_id_1">
          <ns1:container>oversize</ns1:container>
          <ns1:needsRefrigeration xsi:nil="true" />
          <ns1:volume>350</ns1:volume>
          <ns1:weight>150000</ns1:weight>
        </ns1:Cargo>
      </ns1:WorkDocuments>
      <ns1:Messages version="0.12">
        <ns1:Message>
          <ns1:severity>Info</ns1:severity>
          <ns1:text>Cargo with volume > 30 cubic meters must be packaged in a
n oversize container.</ns1:text>
          <ns1:entityReference href="#Cargo_id_1" />
        </ns1:Message>
      </ns1:Messages>
    </ns1:CorticonResponse>
  </soapenv:Body>
</soapenv:Envelope>

C:\Program Files\Corticon\Server\Tomcat\CcServer\cddinit>PAUSE
Press any key to continue . . . _

```

The response message shown in this figure is exactly what the Corticon Server's output looks like when it is returned to the consuming application. There are several things to note in this response.

- The `container` attribute has been assigned a value of `oversize`.
- The input data `volume` and `weight` have been returned in the response message unchanged. If your rules do not change input data, it will be returned exactly as sent.
- The Studio Ruletest assigned unique `id` values to our terms during the XML export. However, if the Server receives a request message *without* `id` values, it will assign them automatically to ensure resulting terms remain associated properly.
- The `Messages` section of the response has been populated with a posted message. Notice that the contents of the `severity` and `text` tags match those used in the rules.
- The `entityReference` tag in the `Messages` section uses an `href` to "link" or "point" to the associated element's `id` value. In this case, we see that the posted message links to (or is associated with) `Cargo` with `id` equal to `Cargo_id_1`. In this case, there is only one `Cargo` it *could* link to, but a production request message may contain hundreds or thousands of `Cargo` entities. In those cases, maintaining `href` association between entities and their message(s) is critical.
- The SOAP "wrapper" or envelope tags were added by the bundled sample code to ensure the request message was sent in accordance with web service standards.

Other details of the Corticon Server response message are described in the *Corticon Server: Integration & Deployment Guide*.

Path 3 - Using SOAP client to consume a Decision Service

Web Services Service Contracts

Web Services has two main ways of describing a service contract:

1. An XML schema document, also known by its file suffix XSD.
2. A Web Services Description Language document, or WSDL (often pronounced "wiz-dull").

Many commercial SOAP and web services development tools have the ability to import an XSD or WSDL service contract and generate a compliant request message directly from it. This path assumes you have access to such a tool and want to use it to consume a Decision Service.

The Corticon Deployment Console can produce both XSD and WSDL documents. The *Server Integration & Deployment Guide* contains more information about these documents, including detailed descriptions of their structure and elements. However, if you have chosen this path, we assume you are already familiar enough with service contracts to be able to use them correctly once generated.

Web services messaging styles

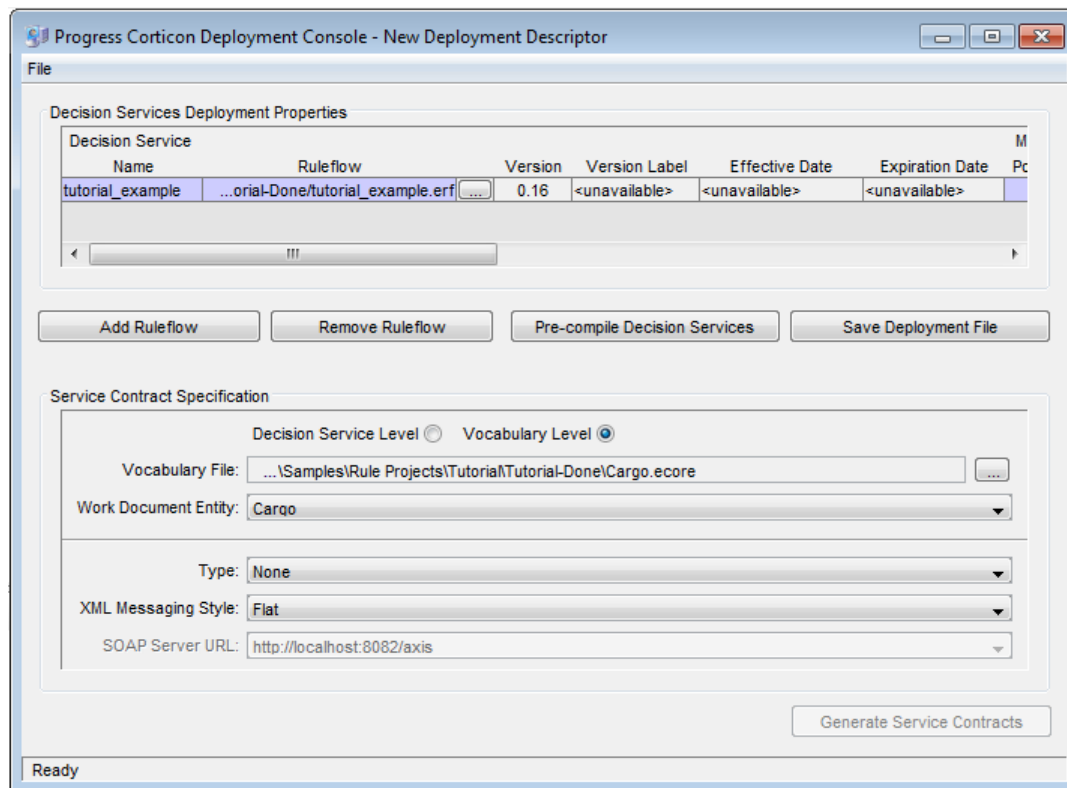
There are also two types of messaging styles commonly used in web services:

1. RPC-style, which is a simpler, less-capable messaging style generally used to send smaller messages and receive single variable answers. All of the administrative methods in Corticon Server's SOAP API use RPC-style messaging.
2. Document-style, which is more complex, but allows for richer content, both in request and response messages. The Corticon Server rule execution (**execute**) interface supports both document-style and RPC-style messaging.

Important: Any SOAP client or SOAP-capable application used to consume a Decision Service deployed to the Corticon Server typically uses document-style messaging. See the *Integration & Deployment Guide* for complete details on proper structure of a compliant request message.

Creating a service contract using the Deployment Console

Figure 22: Deployment Console's Service Contract Specification Window



Launch the **Deployment Console** as before and follow the instructions below to generate a service contract. All **Deployment Console** options below are also described in detail in the *Corticon Server: Integration & Deployment Guide*.

1. **Decision Service Level / Vocabulary Level.** These radio buttons determine whether one service contract is generated per listed Ruleflow, or if a single "master" service contract is generated from the entire Vocabulary. A Decision Service-level service contract is usable only for a specific Decision Service, whereas a Vocabulary-level service contract can be used for all Decision Services that were built using that Vocabulary. Choose the option that is most compatible with your SOAP tool.
2. **Vocabulary File.** If generating a Vocabulary-level service contract, enter the Vocabulary file name (.ecore) here. If generating a Decision Service-level contract, this field is read-only and shows the Vocabulary associated with the currently highlighted Ruleflow row above.
3. **Select Work Document Entity.** If generating a Vocabulary-level service contract, enter the entity name from the list that is the "root" of the service contract. For more details on the concept of work document entity (WDE), refer also to the *Rule Modeling Guide*. If generating a Decision Service-level contract, the field shows the WDE associated with the currently highlighted Ruleflow. If no WDE has yet been specified, or if you want to change it, select one from the list. If you are using the `tutorial_example.erf` Ruleflow, the appropriate WDE is `Cargo`.
4. **Type.** This is the service contract type: WSDL, XML Schema, or none. Note, that the **Generate Service Contracts** button will be enabled only when Type is set to either WSDL or XML Schema.
5. **XML Messaging Style.** Describes the message style, flat or hierarchical, in which the WSDL will be structured.

6. **SOAP Server URL.** URL for the SOAP node that is bound to the Corticon Server. Enabled for WSDL service contracts only. The default URL <http://localhost:8082/axis/services/Corticon> makes a Decision Service available to the default Corticon Server installation performed earlier. Note: this URL can be changed and additional URLs can be added to the drop-down list. See see "Designer properties and settings" in *Server Integration & Deployment Guide* for details.
7. **Generate Service Contracts.** Use this button to generate either the WSDL or XML Schema service contracts into the output directory. If you select Decision Service-level contracts, one service contract per Ruleflow listed at top will be created. If you select Vocabulary-level, only one contract is created per Vocabulary file.

Creating a request message for a Decision Service

Once your SOAP development tool has imported the WSDL or XSD service contract, it should be able to generate an instance of a request message that complies with the service contract. It should also provide you with a way of entering sample data to be included in the request message when it is sent to the Decision Service.

Important:

Most commercial SOAP development tools accurately read service contracts generated by the Deployment Console, ensuring well-formed request messages are composed.

One occasional problem, however, involves the Decision Service Name, which was entered in field 3 of the Deployment Console's Deployment Descriptor section. Even though all service contracts list `decisionServiceName` as a mandatory element, many SOAP tools do not automatically insert the Decision Service Name attribute into the request message's `decisionServiceName` element. Be sure to check this before sending the request message. If the request message is sent without a `decisionServiceName`, the Server will not know which Decision Service is being requested, and will return an error message.

Corticon Server also offers a mode of WSDL generation that's more compatible with Microsoft's Windows Communications Framework. See the *Server Integration & Deployment Guide* for more details.

Enter all required data into the request message. The `tutorial_example.erf` example produces its best results when you enter positive integer values such as:

- `cargo.weight 200000`
- `cargo.volume 1000`

Sending a request message to Corticon Server

Make sure the Tomcat server is running and your Deployment Descriptor file is in the correct location as described earlier. Now, use your SOAP tool to send the request message to the Corticon Server.

Your SOAP tool should display the response from the Corticon Server. Are the results what you expected? If not, or if the response contains an error, proceed to the [Troubleshooting](#) section of this tutorial.

Troubleshooting

The *Rule Modeling Guide* contains an extensive chapter on *Troubleshooting*, including tips for troubleshooting rules in Corticon Studio, as well as problems encountered during Decision Service consumption. Refer to the *Rule Modeling Guide* for more details.

Many users will find other installations of Tomcat already present on their machines. Very often, the default port settings for these installations conflict with the default port setting of Corticon's Tomcat installation, which is 8082. Consult with your system administrator to identify alternate ports you might use.

Modeling and managing rules using Server Console

The Progress Corticon Server Console enabled rule modelers to make some changes to Rulesheets. The Server Console is accessed through a browser connecting to the URL (for a Java Web Service) `http://{server_hostname}:8082/axis`, and then using either of the default modeler credentials:

- Username `modeler1` with its password `modeler1`
- Username `modeler2` with its password `modeler2`

Note: See the next set of topics, [Using the Corticon Server Console](#) on page 65 for detailed information about all the Server Console panels, tabs, and functions. Consult with your administrator for other credentials that provide more (or less) rights with the Server Console.

For details, see the following topics:

- [Lifecycle management Through the Server Console](#)
- [Creating a new decision service version](#)
- [Opening the new version](#)
- [Modifying the new version](#)
- [Promoting the new version to live](#)
- [Removing the version from live](#)
- [Telling the server where to find Deployment Descriptor files](#)

Lifecycle management Through the Server Console

Modifying rules within "live" Decision Services already deployed to Corticon Server requires more considerations than just updating rules in Progress Corticon Studio. In Progress Corticon Studio, the rules in Rulesheets are still just design-time assets, perhaps not even tested yet or packaged in a Ruleflow, let alone deployed to a Server.

But rules in a live Decision Service are available for invocation *right at that moment*, so we need to take a few extra precautions to ensure we do not interfere with clients trying to use our deployed Decision Services.

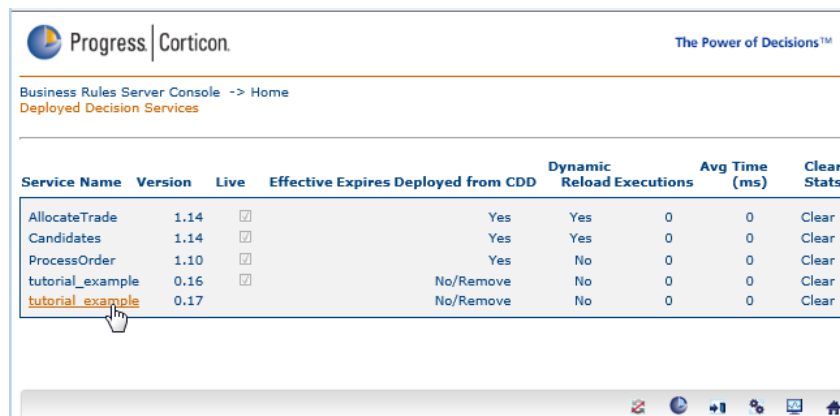
Creating a new decision service version

Before any rule changes can be made from within Server Console, a new version of the Decision Service must be created first. A new version can be created for a Decision Service deployed via a .cdd, or for a Decision Service deployed via the Server Console.

To create a new Decision Service version:

1. Login to Server Console
2. Select Decision Services from the Server Console Main Menu
3. Click the name of the Decision Service you want to modify. Each Decision Service listed in the Service Name column is a hyperlink.
4. Select **Create New Version** button at the top of the page. See the Decision Service Versioning and Effective Dating chapter of the *Server Integration & Deployment Guide* for more information about versions and how to use them during Decision Service invocation.
5. You will return to the Decision Services page, where you should see an additional Decision Service listed in the **Service Name** column. The Version should be incremented by 1. In the figure shown below, a new version of `tutorial_example` has been created.

Figure 23: Server Console with a new version of `tutorial_example` shown



Service Name	Version	Live	Effective Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	<input checked="" type="checkbox"/>		Yes	Yes	0	0	Clear
Candidates	1.14	<input checked="" type="checkbox"/>		Yes	Yes	0	0	Clear
ProcessOrder	1.10	<input checked="" type="checkbox"/>		Yes	No	0	0	Clear
tutorial_example	0.16	<input checked="" type="checkbox"/>		No/Remove	No	0	0	Clear
tutorial_example	0.17	<input type="checkbox"/>		No/Remove	No	0	0	Clear

Notice in the figure that `tutorial_example` version .17 is not yet "live" (the **Live** column checkbox is unchecked). This means we can make changes to it.

Opening the new version

To make changes to the new version (or any other "non-live" Decision Service):

1. Click on the Decision Service you want in the Service Name column. The **Overview** tab opens.
2. Under the **General Settings** header, select the **(Edit)** hyperlink following the `.erf` or `.ers` you want to modify:
 - a) Selecting the **(Edit)** hyperlink following the `.erf` opens a page that lets you modify effective/expiration dates of the Decision Service, or increase its Version number.
 - b) Selecting the **(Edit)** hyperlink following a `.ers` opens a page that lets you modify that Rulesheet.

Modifying the new version

Server Console allows the following types of changes to a Rulesheet:

- Changing values in a Condition or Action cell, including creating new values in the existing set
- Adding/removing/changing Overrides for rule columns
- Changing Rule Statements for those rule columns that have them, including Text and Serverity changes.
- Re-applying the Conflict Checker following any changes to ensure new no new conflicts have been introduced by your changes

Server Console does not allow the following types of changes to a Rulesheet:

- Adding/removing/changing any Scope used by any rules
- Adding/removing/changing any Filters used by the Rulesheet
- Adding/removing/changing Condition or Action expressions
- Adding/removing rule columns to the decision table
- Adding/removing Rule Statements
- Adding/removing/changing and looping or advanced Inferencing features in a Rulesheet
- Enabling or disabling any Rulesheet rows or columns

The Rulesheet change page also includes a **Business View/Technical View** toggle button that shows/hides Rulesheet Scope and Filters, if used.

Once you have updated the Rulesheet with any changes, click the **Save** button at the bottom of the page. A message will display the updated Decision Service timestamp and advise that a slight delay may occur before the Decision Service is ready to be promoted to "live" status. This delay is due to the time required to compile the Decision Service "on the fly" by Corticon Server. For more information about Decision Service compilation, see the *Server Integration & Deployment Guide*.

Promoting the new version to live

After saving the new version of the Rulesheet (and after making any other changes to the Ruleflow or other Rulesheets), return to the Decision Services:Overview tab for the new version.

Clicking the **Promote to Live** button causes the new version to deploy to "live" status, and will update the the **Live** checkbox on the Decision Services page.

Removing the version from live

To remove a new Decision Service version (or any other Decision Service deployed via the Server Console):

1. Connect to the Server Console with administrative user rights.
 - Username `admin` with its password `admin`
 - Username `administrator` with its password `changeme`
2. Click the **No/Remove** hyperlink in the **Deployed from CDD** column of the Decision Services page.

Telling the server where to find Deployment Descriptor files

You can do direct deployment of Decision Services without having to use a `.cdd` or code a Java method call. This method of deployment can only be used with pre-compiled `.eds` files, not with uncompiled `.erf` files.

Using the Corticon Server Console

Corticon's Server Console is a browser-based administration tool for monitoring and managing installations of Corticon Server for Java.

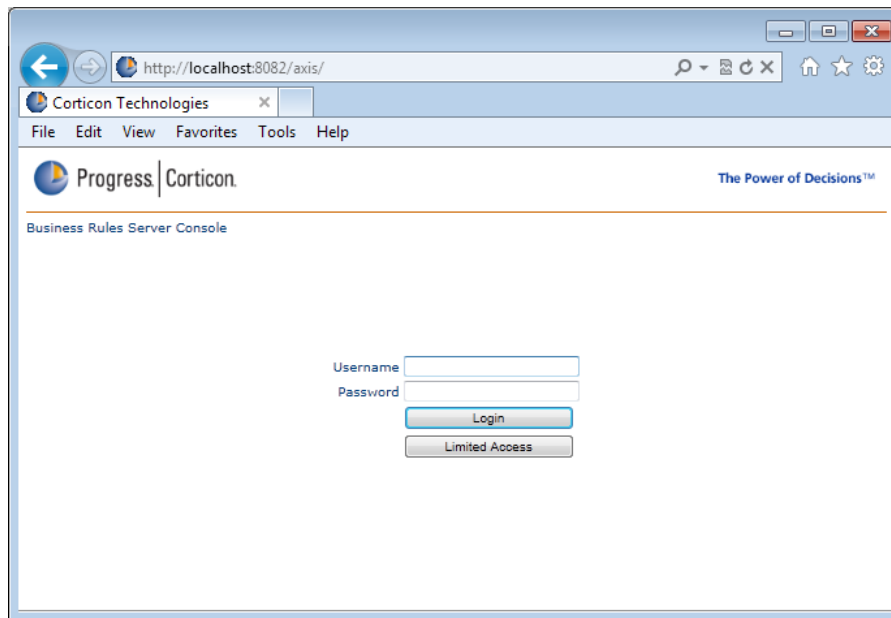
For details, see the following topics:

- [Launching and logging in to Corticon Server Console](#)
- [Using LDAP authentication in Server Console](#)
- [Console main menu page](#)
- [Decision Services page](#)
- [Deploy Decision Service page](#)
- [Configure rule server](#)
- [Monitor rule server page](#)
- [WSDLs](#)

Launching and logging in to Corticon Server Console

When your Corticon Java Server is running, open a web browser to the URL `http://<yourServerURL>:<yourPort>/axis` in a web browser. By default, `axis.war` is installed in the bundled Tomcat, and the default URL is `http://localhost:8082/axis`. The Corticon Server Console login page opens, as shown:

Figure 24: Server Console Login Page



As installed, the Corticon Server has six Server Console login usernames defined, listed here in descending order of strength:

Table 1: Server Console Default Credentials

Username	Password
admin	admin
administrator	changeme
modeler2	modeler2
modeler1	modeler1
tester	tester
<empty>	<empty>

The **rights** assigned to a username specify the view, read, and write functions that are enabled for the user, as follows:

Figure 25: Server Console User Rights

	guest	tester	modeler_limited	modeler_full	admin
Decision Services List					
Display "Clear Stats" Column	N	N	N	N	Y
Allow user to remove non-CDD Decision Service	N	N	N	N	Y
Allow user to view Results Distribution Page	Y	Y	Y	Y	Y
Deploy Decision Service					
Allow user to deploy a new Decision Service	N	N	N	N	Y
Configure Rules Server					
Allow user to "View"/"Edit" Logging settings	N	N	N	N	Y
Allow user to "View"/"Edit" Deployment Directory settings	N	N	N	N	Y
Allow user to "View"/"Edit" Decision Service Options settings	N	N	N	N	Y
Allow user to "View"/"Edit" License information	N	N	N	N	Y
Monitor Rules Server					
View contents of Rules Server	N	N	N	N	Y
WSDL					
View WSDL	N	N	N	N	Y
Decision Service Details					
Allow user to create Test DS	N	N	Y	Y	Y
Allow user to promote Test DS to Live	N	N	Y	Y	Y
Allow user to "View" Ruleflow values	N	Y	Y	Y	Y
Allow user to "Edit" Ruleflow values	N	N	Y	Y	Y
Allow user to "View" Rulesheet contents	N	Y	Y	Y	Y
Allow user to "Edit" Rulesheet contents	N	N	Y	Y	Y
View basic Decision Service details	Y	Y	Y	Y	Y
Allow user to edit Decision Service Pool sizes, Ruleflow URL, ect.	N	N	N	N	Y
Allow user to run Report for a Decision Service	N	N	Y	Y	Y
Test using CorticonRequest against ICcServer.execute(Document)	N	Y	Y	Y	Y
Ruleflow Editor					
Allow user to "View" Ruleflow contents	N	Y	Y	Y	Y
Allow user to "Edit" Ruleflow contents	N	N	Y	Y	Y
Allow user to update the Model using the Update Button	N	N	Y	Y	Y
Rulesheet Editor					
Allow user to "View" Rulesheet contents	N	Y	Y	Y	Y
Allow user to "Edit" Rulesheet contents	N	N	Y	Y	Y
Allow user to use the "Edit" feature in the Drop Down List Boxes	N	N	N	Y	Y
Allow user to update the Model using the Update Button	N	N	Y	Y	Y

The XML file that maintains these credentials is the following:

Figure 26: CcUsernamePassword.xml

```

CcUsernamePassword.xml x
1  <?xml version='1.0' encoding='utf-8'?>
2  <serverconsole-users>
3    <user username="" password="" rights="guest"/>
4    <user username="tester" password="tester" rights="tester"/>
5    <user username="modeler1" password="modeler1" rights="modeler_limited"/>
6    <user username="modeler2" password="modeler2" rights="modeler_full"/>
7    <user username="admin" password="admin" rights="admin"/>
8    <user username="administrator" password="changeme" rights="admin"/>
9  </serverconsole-users>

```

To add, modify, or delete Server Console users, access `CcConfig.jar` in your installation directory, either at `\Server\lib` (in-process use) or `\Server\Tomcat\webapps\axis\WEB-INF\lib` (Tomcat use.) Edit the file `CcUsernamePassword.xml` and update it to suit your preferences. Save the updated file in its `CcConfig.jar`.

Note: The bundled Tomcat configuration is not intended for product use. For production, you should configure LDAP or use an alternate application server. If you expose the bundled Tomcat outside development, it is highly recommended that you at least change all the user passwords.

If you login using the `<empty>` account, or if you choose the **Limited Access** login option, your use of the Server Console will be limited to read-only mode. In this mode, you will not be able to deploy Decision Services or configure or monitor the Server. If you try to view the Deploy Decision Service, Configure Rules Server, or Monitor Rules Server pages (see below), you get appropriate alerts that "User does not have rights to..."

Try logging in as `admin` or `administrator` to expose the complete set of features and functionality that are documented in this chapter. Or, follow the instructions in the following topic to move your authentication to your LDAP domains, and then login as a user that has `admin` rights.

Using LDAP authentication in Server Console

Instead of the built-in user and rights definitions, Corticon Server Console lets you choose to use Lightweight Directory Access Protocol (LDAP) domains for role-based authentication, so that you can control access to Corticon Server Console and define roles in your current user management systems, such as Microsoft's Active Directory.

Note: The interface to LDAP evaluates all credentials (username and password) and properties (name and value) as case-sensitive.

Properties for LDAP

The name-value lines in a well-formed `ldap.properties` file are as follows:

Table 2: LDAP properties for Server Console authentication

Property	Description
<code>ldap.base.provider.url</code>	URL for connection to a directory service.
<code>ldap.base.dn</code>	Parent Domain for the specified directory.
<code>ldap.security.principal</code>	Qualified username or e-mail address of admin user.
<code>ldap.security.credentials</code>	Password of the user
<code>ldap.user.base</code>	Location to search for users.
<code>ldap.group.base</code>	Location to search for groups.
<code>ldap.user.search</code>	Search field for user.

Property	Description
<code>ldap.group.search</code>	Search field for group.
<code>ldap.group.mappings</code>	Mapping of Corticon Server Console Roles with Active Directory groups.
<code>ldap.check.all.directories</code>	Property to authenticate user against all specified directories in <code>ldap.server.ids</code>
<code>ldap.server.ids</code>	List all directory services.

Configuration Examples

The following three configurations cover most use cases:

- Lookup in one directory service
- Lookup in multiple specified directory services
- Lookup across a list of directory services

The following topics detail each of these use cases.

Lookup in one directory service

The following block shows the content of an `ldap.properties` file with minimal required properties.

```
# Active Directory Configuration Details
ldap.base.provider.url=ldap://{hostname}:389/
ldap.base.dn=dc=test,dc=local
ldap.security.principal=admin@test.local
ldap.security.credentials=password

# Defaults search base to ldap.base.dn if user/group base is not specified
#ldap.user.base=cn=Users,dc=test,dc=local
#ldap.group.base=ou=Roles,dc=test,dc=local

ldap.user.search=sAMAccountName
ldap.group.search=sAMAccountName
ldap.group.mappings=admin\=group1,group3\nmodeler_full\=group2\ntester\=group4
```

The syntax for the username field on the login page is `username`. For example, `testuser`

Lookup in multiple directory services

In this authentication type, user has to give domain name and the username to authenticate against the specified directory. There are two regions in this example: `americas` and `apac`.

```
# Specify multiple domains(Active Directory) configurations

# Default Settings
ldap.base.provider.url=ldap://{hostname}:{port}/
ldap.base.dn=dc=test,dc=local
ldap.security.principal=Administrator@test.local
ldap.security.credentials=password
ldap.user.search=sAMAccountName
ldap.group.search=sAMAccountName
```

```
#ldap.user.base=cn=Users,dc=test,dc=local
#ldap.group.base=ou=Roles,dc=test,dc=local
ldap.group.mappings=admin\=group1,group3\nmodeler_full\=group2

# americas settings
ldap.base.provider.url.americas=ldap://{hostname}:{port}/
ldab.base.dn.americas=dc=sample,dc=local
ldap.security.principal.americas=Administrator@sample.local
ldap.security.credentials.americas=password
ldap.user.search.americas=sAMAccountName
ldap.group.search.americas=sAMAccountName
ldap.user.base.americas=cn=Users,dc=sample,dc=local
ldap.group.base.americas=ou=Roles,dc=sample,dc=local
#ldap.group.mappings.americas=admin=group1,group3\nmodeler_full=group2
```

The syntax for the username field on the login page is domainId\username. For example, apac\testuser

Lookup across a list directory services

In this authentication type, a user is authenticated against a list of directory service . Authentication process stops when the user is successfully authenticated against any specified directory.

```
# Configuration to check for user in all listed directories

# Check all directories
ldap.check.all.directories=true
# LDAP Server IDs
ldap.server.ids=apac,americas,default

# apac Settings
ldap.base.provider.url.apac=ldap://{hostname}:{port}/
ldap.base.dn.apac=dc=test1,dc=local
ldap.security.principal.apac=admin@test1.local
ldap.security.credentials.apac=password
ldap.user.search.apac=sAMAccountName
ldap.user.base.apac=cn=Users,dc=test1,dc=local
ldap.group.search.apac=sAMAccountName
ldap.group.base.apac=ou=Roles,dc=test1,dc=local
ldap.group.mappings.apac=admin\=group1,group3\nmodeler_full\=group2

# americas settings
ldap.base.provider.url.americas=ldap://{hostname}:{port}/
ldab.base.dn.americas=dc=test2,dc=local
ldap.security.principal.americas=Administrator@test2.local
ldap.security.credentials.americas=password
ldap.user.search.americas=sAMAccountName
ldap.user.base.americas=cn=Users,dc=test2,dc=local
ldap.group.search.americas=sAMAccountName
ldap.group.base.americas=ou=Roles,dc=test2,dc=local
#ldap.group.mappings.americas=admin=group1,group3\nmodeler_full=group2

# Default Settings
ldap.base.provider.url=ldap://{hostname}:{port}/
ldap.base.dn=dc=test2,dc=local
ldap.security.principal=Administrator@test2.local
ldap.security.credentials=password
#ldap.user.search=sAMAccountName
#ldap.user.base=cn=Users,dc=test2,dc=local
#ldap.group.search=sAMAccountName
#ldap.group.base=ou=Roles,dc=test2,dc=local
ldap.group.mappings=admin\=group1,group3\nmodeler_full\=group2
```

The syntax for the username field on the login page is `username`. For example, `testuser`. If a user provides `domainId\username` to login, authentication will look only in the specified domain.

Deploying LDAP authentication

Once you have defined your LDAP properties, save the file as `ldap.properties` locally, or - if you are using clusters of servers - a network-accessible location so that all cluster members can access it.

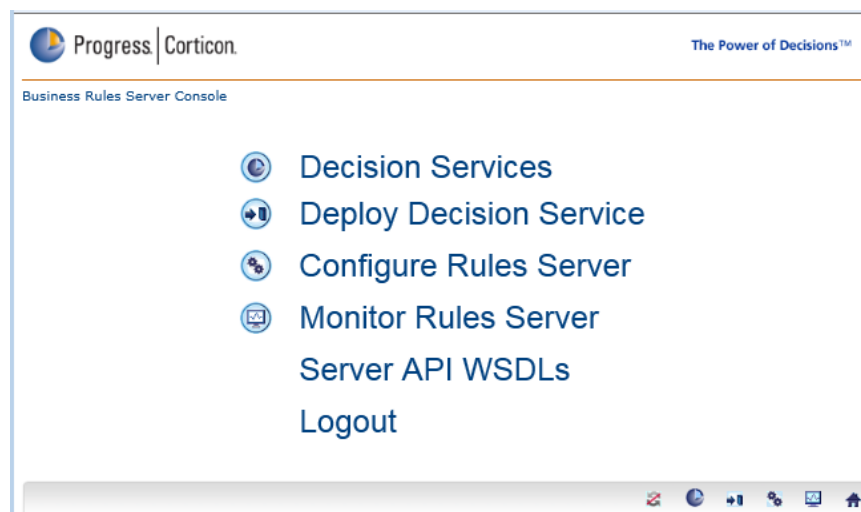
When you initiate or change the `ldap.properties` file, you need to restart the Corticon Server to implement the changes. Once the server restarts, user connections that were dropped will need to use LDAP credentials to connect their Server Console to the Corticon Server.

Note: You can revert to the non-LDAP authentication mechanisms by removing (or relocating) the `ldap.properties` file. Then, when you restart the Corticon Server to implement the changes, user connections that were dropped will need to use the local credentials (in `CcUsernamePassword.xml`) to connect their Server Console to the Corticon Server. See the preceding topic for more information.








Console main menu page

After a successful login, the Server Console opens to its **Home** page.

Figure 27: Server Console Home Page




The lower right corner of these pages provide buttons to navigate to other top-level pages:

Icon	Description
	Enable automatic Refresh of the Server Console display. When enabled, the Console updates every five seconds.
	Disable refresh of the Server Console display. It is a good practice to do this when you are changing settings to avoid losing unsaved entries.
	Show the Decision Services menu.
	Show the Deploy Decision Service menu.
	Show the Configure Rules Server menu.
	Show the Monitor Rules Server menu.
	Show the Home menu.

Decision Services page

Click **Decision Services** on the Home page (or  from the icon bar) to open a page that lists the Decision Services currently deployed to the Corticon Server instance monitored by Server Console.

Figure 28: the Decision Services Page



The Power of Decisions™

Business Rules Server Console -> Home

Deployed Decision Services

Service Name	Version	Live	Effective Expires	Deployed from CDD	Dynamic Reload	Executions	Avg Time (ms)	Clear Stats
AllocateTrade	1.14	<input checked="" type="checkbox"/>		Yes	Yes	0	0	Clear
Candidates	1.14	<input checked="" type="checkbox"/>		Yes	Yes	0	0	Clear
ProcessOrder	1.10	<input checked="" type="checkbox"/>		Yes	No	0	0	Clear
tutorial_example	0.16	<input checked="" type="checkbox"/>		No/Remove	No	0	0	Clear
<u>tutorial_example</u>	0.17			No/Remove	No	0	0	Clear

Each line item on the **Decision Services** page has these columns:

- **Service Name** - The name assigned to a deployed Decision Service, whether deployed by a .cdd file, or by data manually entered on the Deploy Decision Service page. When assigned via a .cdd file, the name is entered as the Decision Service Name in the Deployment Console. Each Service Name provides a link to the deployment properties of that Decision Service. Click on a Service Name to open its information tabs, as shown above.
- **Version** - The version identifier of the deployed Decision Service. A Decision Service's version number is set in the **Ruleflow > Properties** menu option of Corticon Studio for each *Ruleflow* (.erf file). See [Decision Service Versioning and Effective Dating](#) in this manual, and the *Rule Modeling Guide* for more information.
- **Live** - The deployment status of the Decision Service. When deployed and ready to receive invocations, the **Live** checkbox is checked. If a new version of a Decision Service has been created using the **Overview** tab, then it will not become live until promoted. This process is described in more detail in the *Rule Modeling Guide*.
- **Effective** - The date on which the selected Decision Service becomes active.
- **Expires** - The date on which the selected Decision Service ceases to be active.
- **Deployed from CDD** - Indicates whether a Decision Service has been deployed from a .cdd file (**Yes**) or from the [Deploy Decision Service](#) page (**No/Remove**). Note that **No/Remove** is hyperlinked, indicating that a Decision Service deployed using Server Console (from the [Deploy Decision Service](#) page) can also be removed from within Server Console by clicking the link. Decision Services deployed from a .cdd can only be removed by editing or removing the .cdd. Decision Services deployed via the Server Console update the `serverState.xml` file. Likewise, if a Decision Service is *removed* using the **No/Remove** hyperlink, then its corresponding entry is removed from `serverState.xml`.
- **Dynamic Reload** - Indicates whether Corticon Server's maintenance thread will watch for changes to the Decision Service and refresh the pool if updates are detected. For Decision Services deployed through a .cdd, this option is set in the [Dynamic Reload](#) field of the *Deployment Console*. If deployed via the [Deploy Decision Service](#) page of Server Console, Dynamic Reload is automatically **Yes**.
- **Executions** - The number of times the Decision Service has executed since statistics were last cleared. Click on its value to display its tabs and visualizations of execution average time and counts. This feature is enabled only when you turn **On** the optional **Decision Service Performance Monitoring Service**, located on the [Decision Service Options](#) tab on page 85.
- **Avg Time(ms)** - The amount of time, on average, required by the Decision Service to execute. This feature is enabled only when you turn **On** the optional **Decision Service Performance Monitoring Service**, located on the [Decision Service Options](#) tab on page 85.
- **Clear Stats** - Sets **Execution**, and **Average Time** -- as well as their **Performance** and **Distribution Chart** -- data to zero. You need administrator rights to perform the Clear Stats action.

Decision Service actions and information

The four buttons at the top of the page and the four tabs of each Decision Service's information page are discussed in this topic.

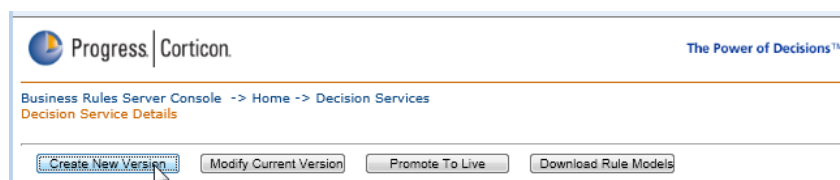
If the Decision Service was deployed by a .cdd file, then the contents of the .cdd are displayed on this page. If the Decision Service was deployed by the **Deploy Decision Service** page, then the information entered on that page (which is persisted in `serverstate.xml`) is displayed. The information displayed on the **Overview** page, shown in [Overview Tab](#), is collected from three sources:

- From the `.cdd` file or from the information entered in the **Deploy Decision Service** page (depending on how the Decision Service was deployed). In both cases, the information reflects the state of the Decision Service at the time of deployment.
- From the Ruleflow file (`.erf`) itself, either from file `DateTime` stamps or from properties set in the **Ruleflow > Properties** menu of Corticon Studio.
- Retrieved real-time/dynamically from the Server Console. For example **Pool Settings|Available/Total Instances in Pool** reflects the current Reactor pool status for that Decision Service. The number of Reactors available and total, will change as Corticon Server grows/shrinks each Decision Service's pool to handle demand. For more information about Reactor pools, see [Pool Size](#) and [Optimizing Pool Size](#).

BUTTONS

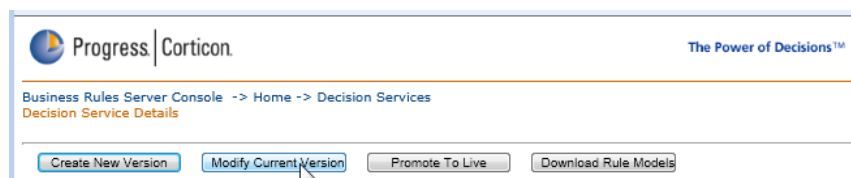
The four buttons at the top of a Decision Service information page let you perform the following actions:

- **Create New Version** - Click the button, as shown:



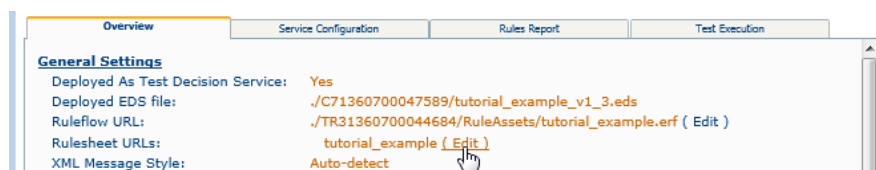
Opens a panel that defines a new major version of the Decision Service which you can edit.

- **Modify Current Version** - Click the button, as shown:



Opens a panel that defines a new minor version of the Decision Service which you can edit.

Click on an editable item to open its associated asset.



On the selected asset -- a Rulesheet in this example -- adjust selected parameters, click to check for conflicts, and see the alternative Business View.

Business Rules Server Console -> Home -> Decision Services -> Decision Service Details: tutorial_example (version 1.3)
Rulesheet Editor

Loaded Rulesheet: ./Cargo.ers

Scope Section

Cargo [load]
 container
 needsRefrigeration
 volume
 weight
 Aircraft [plane]
 maxCargoWeight
 tailNumber
 FlightPlan [plan]
 flightNumber

Rules

	Conditions	0	1
a	load.weight	-	<= 2000
b	load.volume	-	-
c	load.needsRefrigeration	-	-
d	load.weight -> sum > plane.maxCargoWeight	-	-

Actions

	Actions		
A	load.container	-	standard
B	plane.tailNumber	-	-

Rule Statements

Reference	Severity	Text
1	Info	Cargo weighing <= 20,000 kilos must be packaged in a standard container.
2	Info	Cargo with volume > 30 cubic meters must be packaged in an oversize container.
3	Info	Cargo weighing > 20,000 kilos, with volume <= 30 cubic meters, must be packaged in a heavyweight container.
4	Info	Cargo requiring refrigeration must be packaged in a reefer container.
5	Violation	plane.tailNumber must not be assigned to flightplan.flightNumber because the name weights too

- **Promote To Live** - Click the button, as shown to promote the changed service to live.:

Progress | Corticon. The Power of Decisions™

Business Rules Server Console -> Home -> Decision Services
Decision Service Details

- **Download Rule Models** - Click the button to package the decision service and its assets into a .zip archive file.

Do you want to open or save **tutorial_example_v1_3.zip** (608 KB) from localhost?

TABS

Overview tab

The selected decision service, `tutorial_example`, opens to its **Overview** tab, as shown:

Figure 29: Overview Tab

The screenshot shows the Progress Corticon Business Rules Server Console interface. At the top, the Progress Corticon logo and the tagline "The Power of Decisions™" are visible. Below the header, the breadcrumb navigation reads "Business Rules Server Console -> Home -> Decision Services". A link for "Decision Service Details" is present. A notification states "Decision Service has been updated:" with details: "Restrict Warning RuleMessages: No" and "Restrict Violation RuleMessages: Yes". Below this, four buttons are available: "Create New Version", "Modify Current Version", "Promote To Live", and "Download Rule Models". The main content area has four tabs: "Overview" (selected), "Service Configuration", "Rules Report", and "Test Execution". Under the "Overview" tab, there are four sections: "General Settings", "Pool Settings", "Database Access Settings", and "Monitored Attributes".

General Settings	
Deployed As Test Decision Service:	Yes
Deployed EDS file:	./C7_1391723118140.437168/ProcessOrder_v2_0.eds
Ruleflow URL:	./TR0_1391723114601.366515/RuleAssets/Order.erf (Edit)
Rulesheet URLs:	ProcessOrder (Edit)
XML Message Style:	Auto-detect
Execution Log Level:	RULETRACE
Execution Log Path:	C:/Corticon/logs/ProcessOrder_v2_0
Execution Log Per Thread:	Yes
Restrict Info RuleMessages:	(System Default)
Restrict Warning RuleMessages:	No
Restrict Violation RuleMessages:	Yes
Dynamic Reload:	Yes
Loaded from CDD file:	No
Deployment Timestamp:	Feb 24, 2014 1:08:33 PM
Ruleflow Timestamp:	Feb 6, 2014 4:45:14 PM
EDS Timestamp:	Feb 6, 2014 4:45:18 PM
Total Number of Rules Deployed:	6
Effective Date Start:	
Effective Date End:	
Total Execution Count:	0
Average Execution Time (ms):	0
Last Execution Timestamp:	Not Executed Yet

Pool Settings	
Minimum / Maximum Pool Size:	1 / 1
Available / Total Instances in Pool:	1 / 1

Database Access Settings	
Database Access Mode:	None

Monitored Attributes
No Monitored Attributes Registered

The **Overview** tab lets Server Console users with write permissions to perform limited management and updates to the deployed Ruleflows and constituent Rulesheets. See the *Rule Modeling Guide* for details on editing those Corticon assets.

Service Configuration tab

Click the **Service Configuration** tab to open that panel, as shown:

Figure 30: Service Configuration Tab

The screenshot displays the 'Service Configuration' tab within the Progress Corticon Business Rules Server Console. The interface includes a top navigation bar with the Progress Corticon logo and the tagline 'The Power of D'. Below the navigation bar, a breadcrumb trail shows the path: 'Business Rules Server Console -> Home -> Decision Services'.

The main content area features a tabbed interface with four tabs: 'Overview', 'Service Configuration' (which is the active tab), 'Rules Report', and 'Test Execution'. Above the tabs, there are four buttons: 'Create New Version', 'Modify Current Version', 'Promote To Live', and 'Download Rule Models'.

The 'Service Configuration' tab contains a list of configuration properties, each with a corresponding input field or dropdown menu. The properties and their values are as follows:

- Current Rule Asset URL: ./C7_1391723118140.437168/ProcessOrder_v2_0.eds
- Rule Asset URL: [Empty field] Browse...
- Minimum Pool Size: 1
- Maximum Pool Size: 1
- XML Message Style: Auto-detect
- Execution Log Level: (System Default)
- Execution Log Path: C:/Corticon/logs/ProcessOrder_v2_0
- Execution Log Per Thread: Yes
- Restrict Info RuleMessages: (System Default)
- Restrict Warning RuleMessages: No
- Restrict Violation RuleMessages: Yes
- Database Access Mode: None
- Database Access Entities Returned Mode: All Entities
- Current Properties File Location: [Empty field] Browse...
- Database Access Properties File: [Empty field] Browse...

Below the configuration properties, there is an 'Update' button. Underneath the 'Update' button, the text 'No Monitored Attributes Registered' is displayed.

At the bottom of the configuration section, there is a table with two columns: 'Monitored Attribute' and 'Analysis Bucket'. The table is currently empty, and there is an 'Add' button below it.

The bottom of the console features a status bar with several icons, including a red 'X' icon, a blue circular icon, a right-pointing arrow, and a gear icon.

This panel has two sections:

Update the Decision Service's Deployment Properties

If the Decision Service was deployed by the **Deploy Decision Service** page, then the deployment properties can be changed on the **Service Configuration** tab using the **Update** button. Enter your changes, and then click **Update** to re-deploy the changes to the Decision Service.

If the selected Decision Service was deployed by a .cdd file, then deployment properties cannot be changed on this tab – they must be changed in the .cdd file itself. See the [Deploying Corticon Ruleflows](#) chapter for more information about .cdd files and the Deployment Console. This function is accessible to admin users.

Monitored Attribute and Analysis Buckets

Server Console lets you monitor specific attributes in a deployed Decision Service. By choosing attributes to monitor, you can view the statistical breakdown of attribute values over the course of many Decision Service executions.

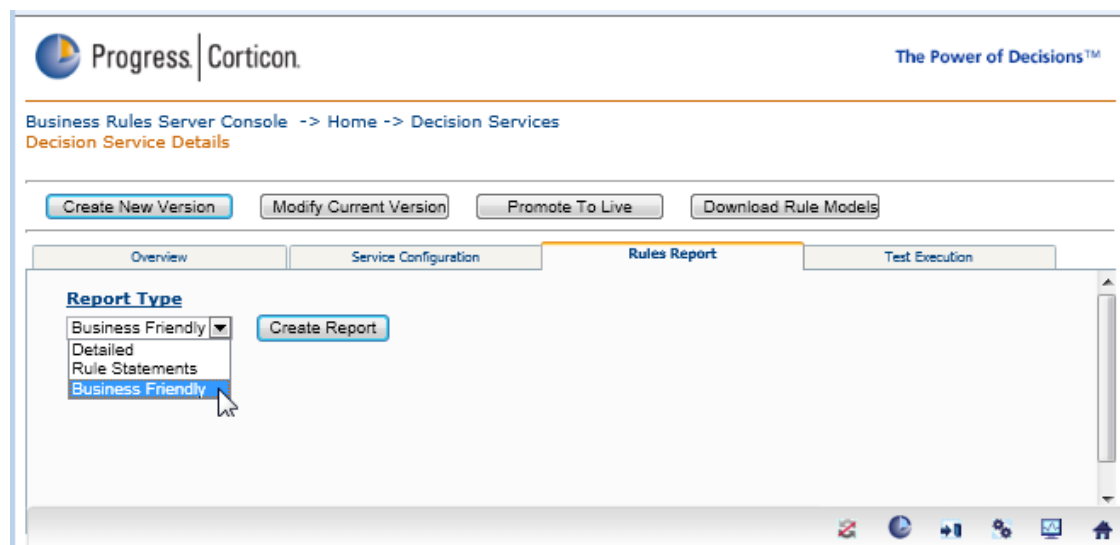
For example, the Ruleflow created in the *Basic Rule Modeling Guide* assigns values such as `oversize` and `reefer` to attribute `Cargo.container`. To monitor this attribute, enter the fully qualified attribute name (including the entity) in the **Monitored Attribute** box and enter the exact value or values in the **Analysis Buckets** box (separated by a comma).

After entering your attribute and its values, click **Add** to append the data to the monitored list. The results of attribute monitoring can be viewed in the [Distribution Chart](#) tab, described below.

Rules Report Tab

Click the **Rules Report** tab to open that panel, as shown:

Figure 31: Rules Report Tab



Rulesheet reports can be generated directly from this tab. See the *Rule Modeling Guide* for more information about Corticon's reporting framework. Its options are:

- **Detailed** reports display all elements of the Rulesheet, including Scope, Filters, and other sections.
- **Rule Statements** reports display only those Rule Statements entered for rules. If a rule has no Rule Statement, then that rule will not appear in this report style.
- **Business Friendly** reports display the Natural Language equivalents (if present) of the rules.

The following figure is a Business-Friendly report on the selected decision service:

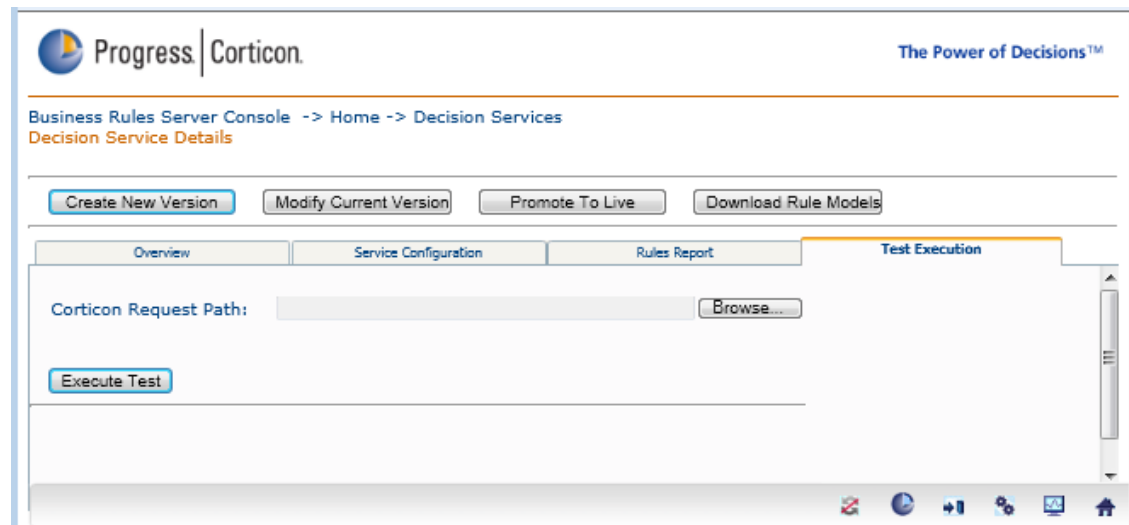
Figure 32: Business Friendly Report



Test Execution Tab

Click the **Test Execution** tab to open that panel, as shown:

Figure 33: Test Execution Tab



If you have a compliant Corticon Request XML message file, then you can use this tab to send it to Corticon Server, and then view its Response. Navigate to the XML message file using the **Browse...** button, then click **Execute Test** to invoke Corticon Server. To be compliant, your XML Request message must follow the same rules as described in Path 2 in the *Corticon Server: Deploying Web Services*.

Execution and Distribution Visualizations

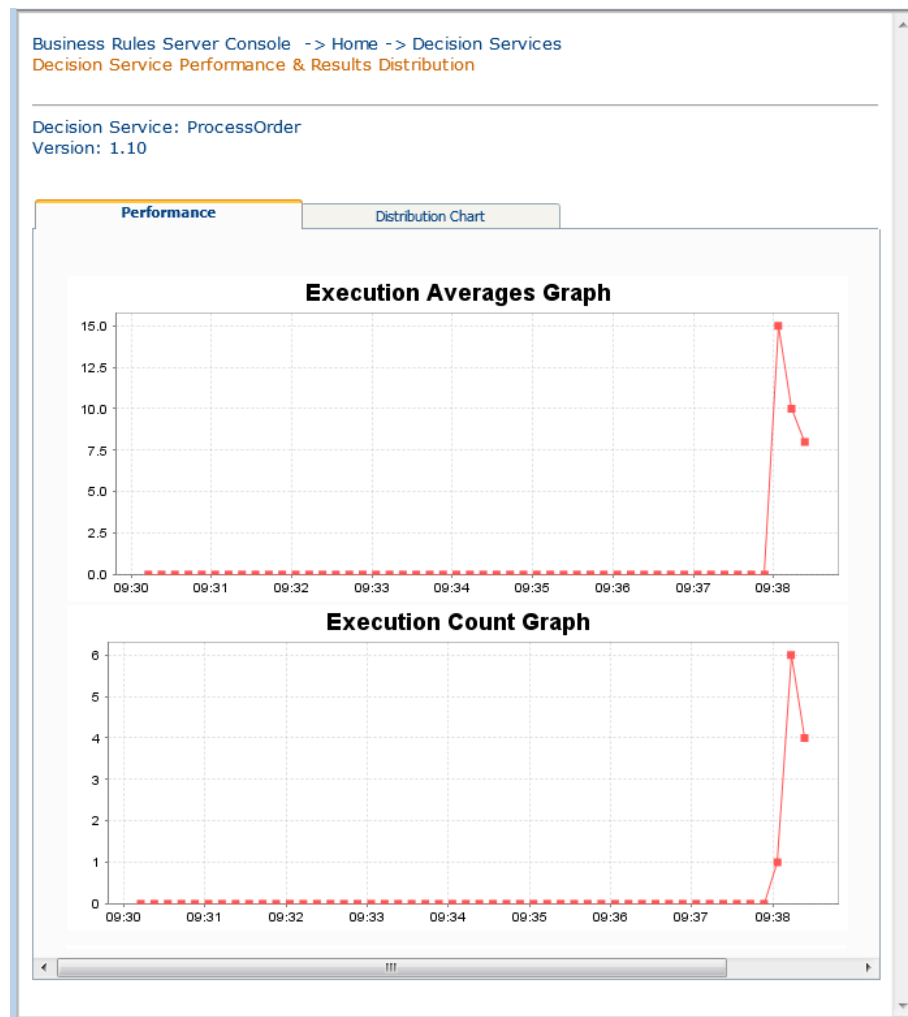
To access the visualizations of execution and distribution data, click on a decision service line's Execution or Avg Time value, as shown:

Executions	Avg Time (ms)
0	0
0	0
11	8

Performance Tab

The selected Decision Service records the number of executions since statistics were last cleared, and then computes the average time in milliseconds. The **Executions** tabs are:

Figure 34: Performance Tab: Execution Average Time & Count Graphs



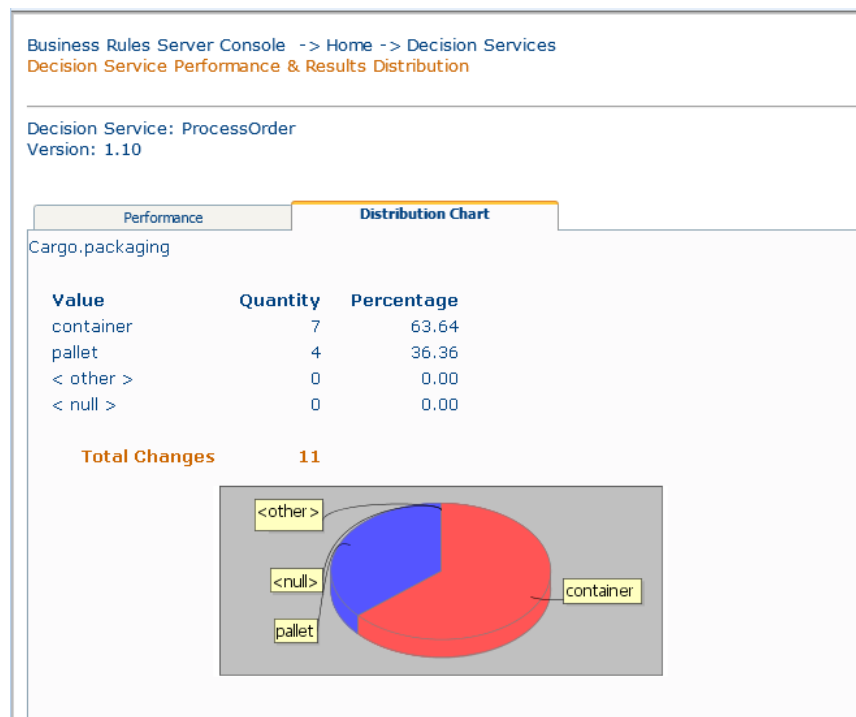
The [Decision Service Performance Monitoring Service](#) and [Decision Service Time Interval Performance Analysis Service](#) must be **On** for this feature to work. To turn these services **On**, go to the **Configure Decision Service** page, *Decision Service Options* tab.

The top graph displays average execution time for all executions performed within Server Console's recording interval. The lower graph displays the number of executions performed within Server Console's recording interval. Server Console's recording interval is 10 seconds by default, but can be changed in [CcServer.properties](#).

Distribution Chart Tab


This tab displays a pie chart of the distribution of values of attributes monitored in the [Service Configuration](#) tab.

Figure 35: Distribution Chart Tab: Results of Monitored Attributes



The [Decision Service Results Distribution Analysis Service](#) must be **On** for this feature to work. To turn **On**, go to the **Configure Decision Service** page, *Decision Service Options* tab.

Deploy Decision Service page

Click **Deploy Decision Service** on the Home page (or  on the icon bar) to open a page that enables you to deploy Decision Services directly from Server Console, instead of using the *Deployment Console* to create a `.cdd` file. Server Console uses standard Server APIs to perform the deployment.

To deploy Decision Services from the Server Console, you must use the pre-compile option in the *Deployment Console* to create an `.eds` file. Only an `.eds` file can be deployed using Server Console.

Important: Only an `.eds` file can be deployed using Server Console.

Server state

In Corticon Server, the deployment state of all Decision Services, regardless of how deployed, is persisted in a file named `ServerState.xml`, located in `{CORTICON_WORK_DIR}\SER\CcServerSandbox\DoNotDelete`.

The **Deploy Decision Service** page captures the same deployment information as a `.cdd`, and stores it in `ServerState.xml` file. These fields are described in the [Deploying Corticon Ruleflows](#) chapter above. One exception: to update a Decision Service, we assume you will change the Ruleflow's compiled `.eds` file and then redeploy using the [Service Configuration](#) tab's **Update** button. Therefore, a Dynamic Reload setting isn't necessary.

When Corticon Server restarts, it reads the `ServerState.xml` file and redeploys any Decision Services included in it.

Figure 36: Deploy Decision Service Page

To use this page, you must pre-compile your `.erf` file into an `.eds` file, and use the `.eds` file in the **Rule Set (path)** field above. You cannot directly deploy uncompiled `.erf` files via Server Console. Enter the required deployment information in the provided boxes, and click **Deploy**.

Once deployed, Corticon Server creates a "local" copy of the `.eds` file in its local `CcServerSandbox` directory structure. The location path can be seen in the `ServerState.xml` file in the following figure. It does this so that it always has access to the executable code (stored inside the `.eds` file) when it attempts to reload/redeploy the Decision Service.

If you want to change the Ruleflow, we recommend using the **Update** button on the [Service Configuration](#) tab of the **Decision Service** page. We do not recommend editing the local copy of the `.erf` file because the two files will not be identical. Using the `Update` method ensures the original and local copies are always the same.

Figure 37: ServerState.xml showing Local Storage of Deployed Decision Service


```
<?xml version="1.0" encoding="UTF-8"?>
<CcServerInfo CcLicensePath="">
  <CDDs>
    <CDD path="C:/Program Files/Corticon/eServer/Tomcat/CcServer/cdd/eBay.cdd"
timestamp="1295973604547">
      <DecisionService name="PaymentHold">
        <EDS path="C:/Program
Files/Corticon/eServer/Tomcat/CcServer/CcServerSandbox/DoNotDelete/DecisionServices
/C11295987098406/PaymentHold_v0.eds" />
        <ERF path="C:/Program
Files/Corticon/eServer/Tomcat/CcServer/studioAssets/PaymentHold.erf" timestamp=
"1295973743750" />
      </DecisionService>
    </CDD>
  </CDDs>
  <NonCDDs>
```

All other entries in the **Deploy Decision Service** page are visible above in the `<NonCDDs>` section of the document.

Note:

In Corticon Server 4.2 and earlier, Decision Services deployed via APIs remained deployed only for the duration of Corticon Server's session. If Corticon Server (or its host container) was stopped and restarted, Decision Services deployed previously by APIs would NOT redeploy automatically until those APIs were called again. Decision Services deployed via .cdd files, however, would redeploy automatically.

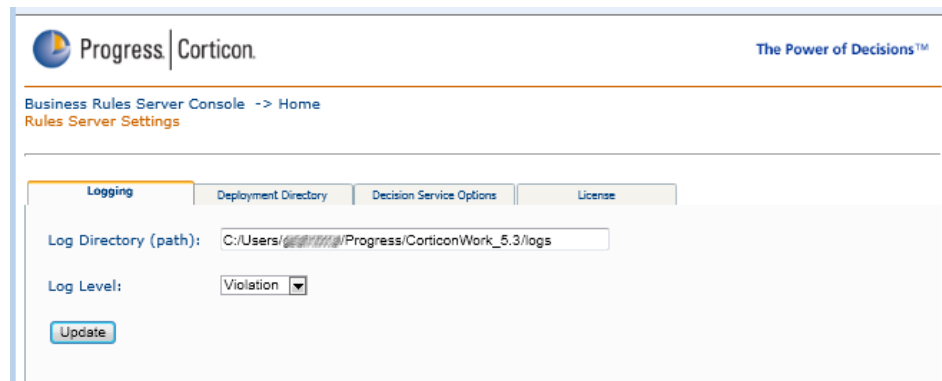
Configure rule server

Click **Configure Rule Server** on the Home page (or  from the icon bar) to open the configuration pages.

Logging tab

The **Logging** tab displays the current settings of `LogPath` and `LogLevel` properties, as shown:

Figure 38:

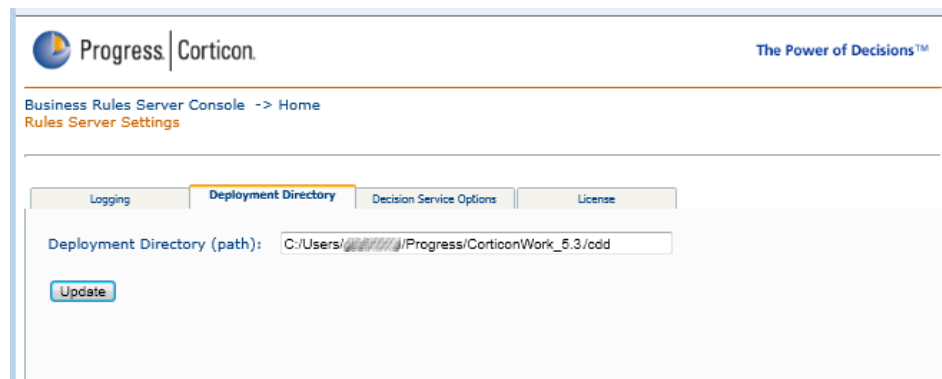


These properties are stored in `CcCommon.properties`, but can be overridden with values entered on this tab. The new values are written to `ServerState.xml` and take effect each time Corticon Server is started.

Deployment Directory tab

The **Deployment Directory** tab displays the current value of the `autoloadaddr` property, as shown:

Figure 39: Deployment Directory tab

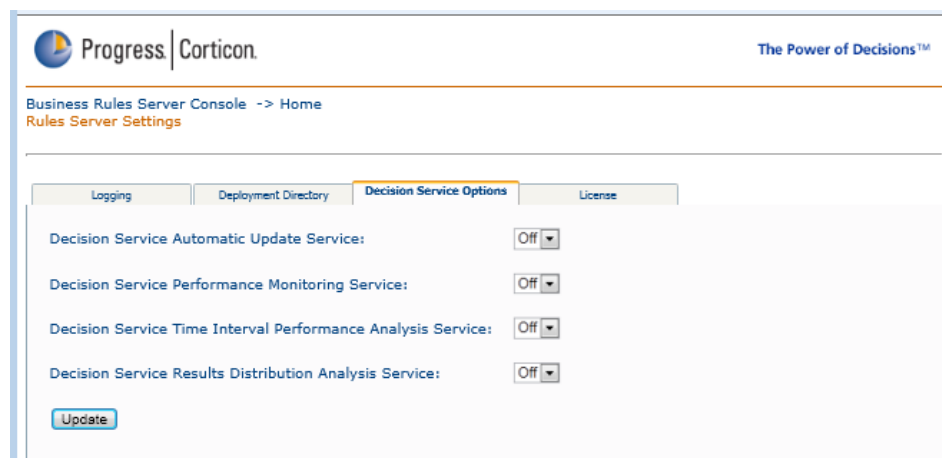


This property is stored in [CcServer.properties](#), but can be overridden with a path entered on this tab. The new value is written to `ServerState.xml` and takes effect each time Corticon Server is started.

Decision Service Options tab

Server Console lets you enable and disable four Corticon Server services on this tab, as shown:

Figure 40: Decision Service Options tab



By default, all these settings are **Off**. Set your preferred value (**On** or **Off**) for each feature, and then click **Update** to write the new values to `ServerState.xml`.

Decision Service Automatic Update Service

This service, also known as the Dynamic Update Monitoring Service, is performed by Corticon Server's maintenance thread. The

`com.corticon.ccserver.dynamicupdatemonitor.autoactivate` property in [CcServer.properties](#) maintains the permanent value of this property, but it can also be controlled during Corticon Server's session via this Server Console option. The option chosen is recorded in `ServerState.xml`.

Decision Service Performance Monitoring Service

This service is responsible for generating the [Executions](#) and [Avg Time \(ms\)](#) values displayed on the Decision Services page, as well as the Execution Averages and Execution Count graphs on the [Performance](#) tab. The option chosen is recorded in `ServerState.xml`.

Decision Service Time Interval Performance Analysis Service

This service is responsible for tracking and recording the time intervals across which other services operate. If this service is off, then the Performance Monitoring Service will not be able to display the data it collects because much of it is shown as averages over time. Generally, you will want to activate or deactivate both this and the Performance Monitoring service. The option chosen is recorded in `ServerState.xml`.

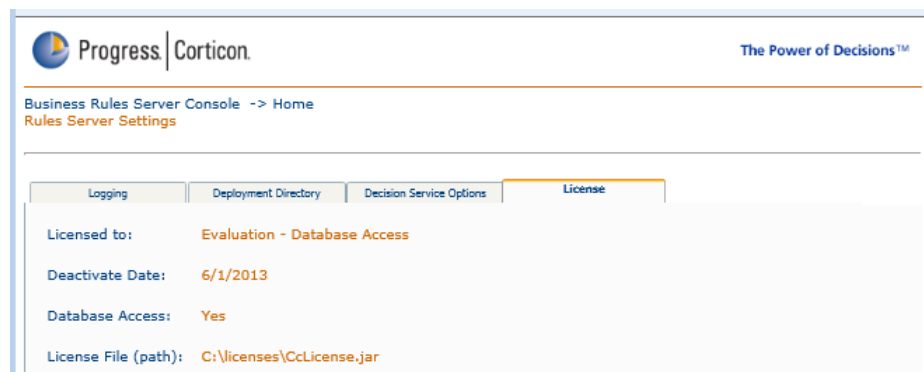
Decision Service Results Distribution Analysis Service

This service is responsible for tracking and recording the attribute values you designated in the **Decision Services** page's [Service Configuration](#) tab. The option chosen is recorded in `ServerState.xml`.

License tab


The Corticon license file at the specified License File (path) is decrypted from the `CcLicense.jar` currently in use by the running instance of Corticon Server. Three important fields are presented to the user: **Licensed To**, **Deactivate Date**, and **Database Access**, as shown:

Figure 41: License tab



This page is informational only.

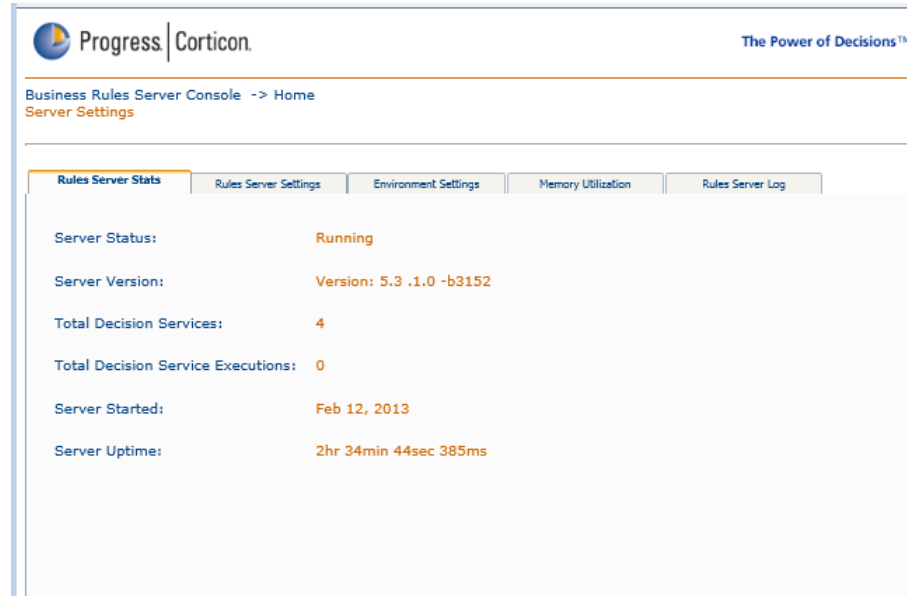
Monitor rule server page

Click **Monitor Rule Server** on the Home page (or  from the icon bar) to open the monitoring page.

Rules Server Stats tab

This tab displays a summary view of basic Corticon Server information, including version/build, total number of Decision Services deployed, session start time, and uptime duration, as shown:

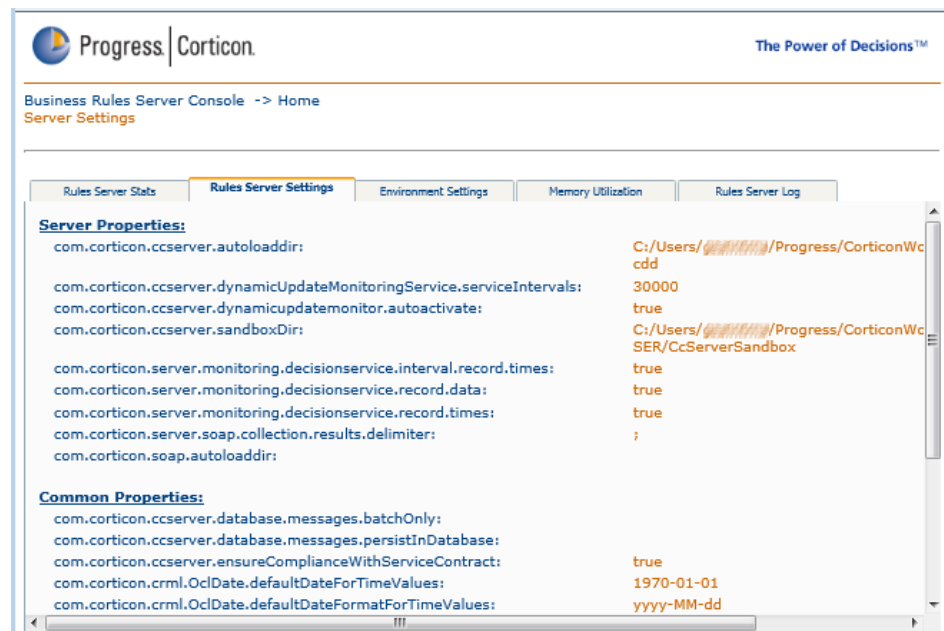
Figure 42: Rules Server Stats tab



Rules Server Settings tab

This tab displays a view of many of the property settings.

Figure 43: Rules Server Settings tab



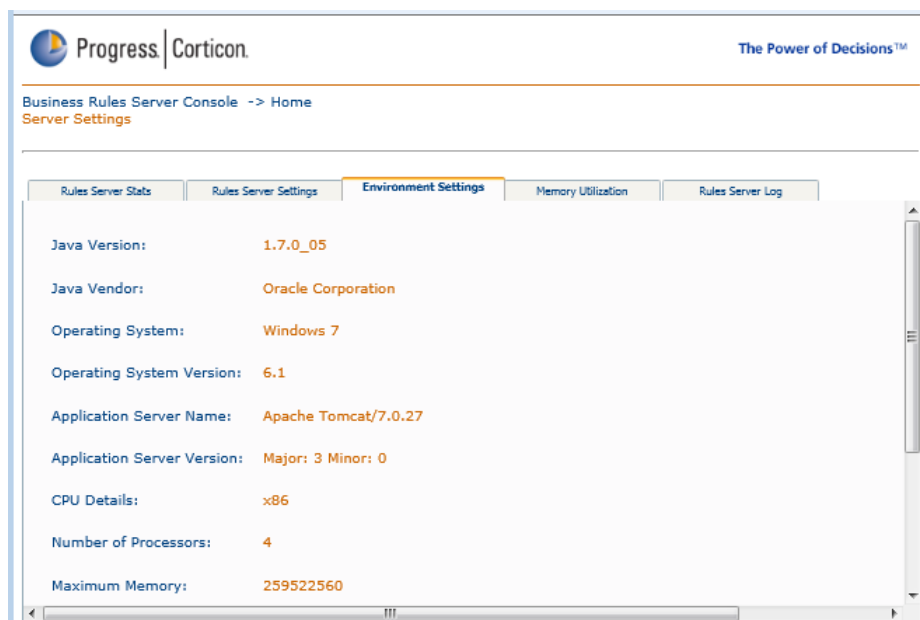
These settings are maintained in `CcConfig.jar` (see [Configuring Corticon properties and settings](#) for more information about this file). If a property setting (such as `logLevel`) has been overridden by a setting in Server Console (and persisted in `ServerState.xml`), then the Server Console value is displayed.

The list of properties displayed on this tab is controlled by `CcServerConsoleProperties.properties`, which is located in `Server\Tomcat\webapps\axis\WEB-INF` in your Corticon Server installation directory.

Environment Settings tab

This tab displays basic information about host operating system, application server, and Java virtual machine (JVM), as shown:

Figure 44: Environment Settings tab

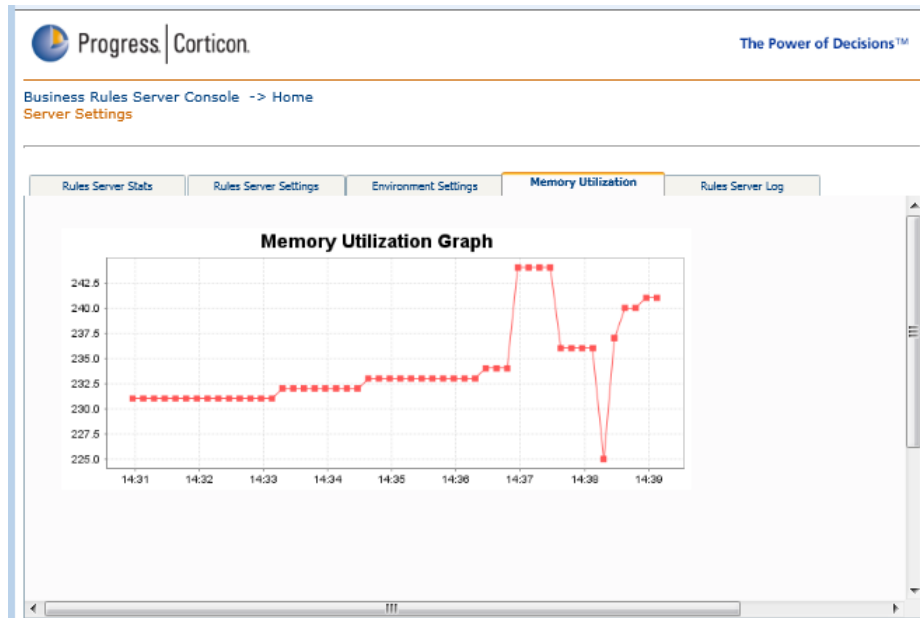


This information may be useful to Progress technical support in the event you need to contact us.

Memory Utilization tab

This tab displays a graph of JVM memory utilization, as shown:

Figure 45: Memory Utilization tab

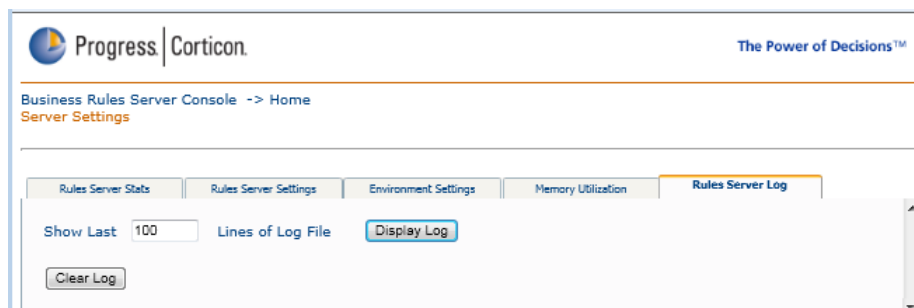


This graph will help you understand if Corticon Server's host environment is under-resourced, and may also be useful to Progress technical support in the event you need to contact us about memory problems.

Rules Server Log

This tab reads and displays lines from the current Corticon Server log file, as shown:

Figure 46: Rules Server Log

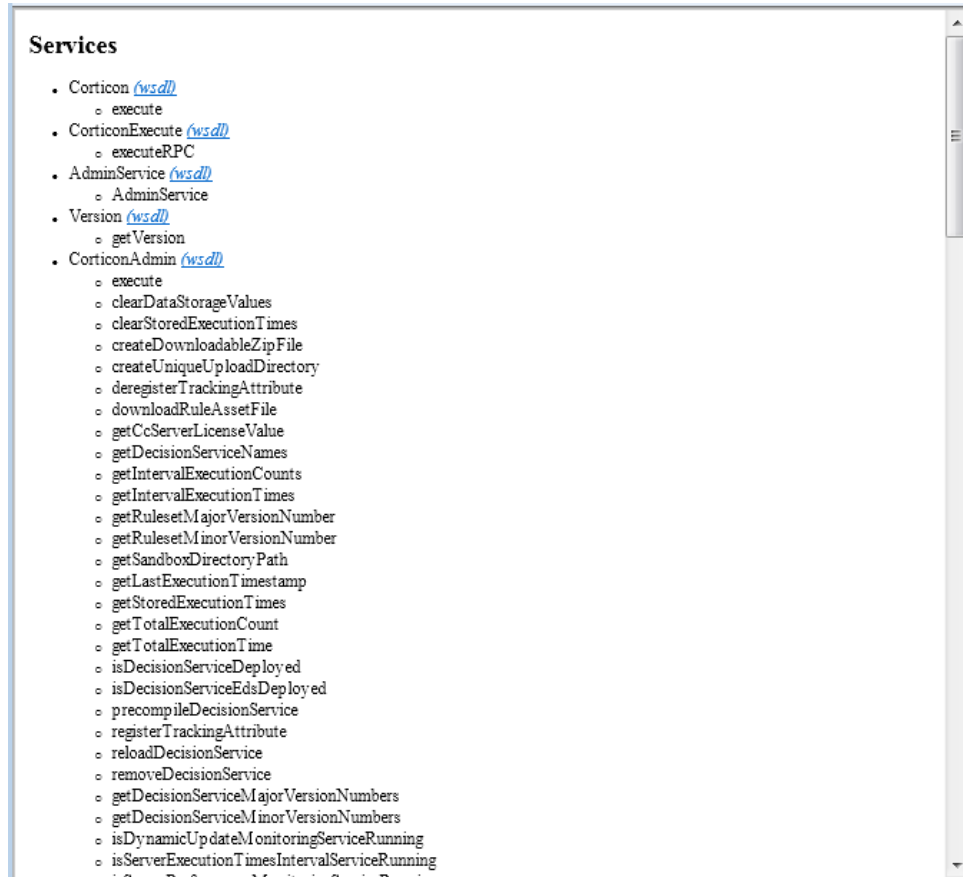


This tab allows you to read recent log entries without having direct access to the log file itself. You can also clear the log.

WSDLs

This page provides access to WSDL files for all Corticon APIs, as shown:

Figure 47:



This URL is the home address for Server Console. You also can access this page directly (external of the Console) by using this address:

`http://<yourServer>:<yourPort>/axis/servlet/AxisServlet`

Summary of Java samples

Corticon Server for Java contains several sample code files that are useful starting points when building your own integrations. The sample code files are self-documented and are located in the your Corticon installation.

Directory	Sample Code file	Usage
[CORTICON_HOME]\Server\src	CcServerApiTest.java	<p>Provides the source code used by testServer.bat to create a DOS menu interface on the Server API.</p> <hr/> <p>Note: uses Corticon Server in the same JVM as the Java test code and can not be executed against Corticon Server inside the Tomcat server (you can, of course, write your own Java client using these APIs to operate against any deployment, including a Tomcat deployment).</p> <hr/>
[CORTICON_HOME]\Server\src	CcDeployApiTest.java	Shows how to control the Deployment Console via Java API calls. This is the source code used by testDeployConsole.bat to create a Windows console interface for Server API.
[CORTICON_HOME]\Server\Tomcat\CcServer\src	CcMessageHandler.java CcServerAxisTest.java CcServerMessagingAxis.java CcSoapException.java CcSoapSendException.java CcSoapServerInit.java CcSoapUtilities.java	The source code used by axis.war, the sample SOAP wrapper shipped with Corticon Server.

Third party acknowledgments

One or more products in the Progress Corticon v5.3.4 release includes third party components covered by licenses that require that the following documentation notices be provided:

Progress Corticon v5.3.4 incorporates Apache Commons Discovery v0.2 from The Apache Software Foundation. Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 - Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgement: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).\" Alternately, this acknowledgement may appear in the software itself, if and wherever such third-party acknowledgements normally appear.
4. The names "The Jakarta Project", "Commons", and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache" nor may "Apache" appear in their names without prior written permission of the Apache Group.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <<http://www.apache.org/>>.

Progress Corticon v5.3.4 incorporates Apache SOAP v2.3.1 from The Apache Software Foundation. Such technology is subject to the following terms and conditions: The Apache Software License, Version 1.1 Copyright (c) 1999 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "SOAP" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see <<http://www.apache.org/>>.

Progress Corticon v5.3.4 incorporates DOM4J v1.6.1. Such technology is subject to the following terms and conditions: Project License BSD style license Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved.

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name "DOM4J" must not be used to endorse or promote products derived from this Software without prior written permission of MetaStuff, Ltd. For written permission, please contact dom4j-info@metastuff.com.

4. Products derived from this Software may not be called "DOM4J" nor may "DOM4J" appear in their names without prior written permission of MetaStuff, Ltd. DOM4J is a registered trademark of MetaStuff, Ltd.

5. Due credit should be given to the DOM4J Project - <http://www.dom4j.org>

THIS SOFTWARE IS PROVIDED BY METASTUFF, LTD. AND CONTRIBUTORS ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL METASTUFF, LTD. OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT,

INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Progress Corticon v5.3.4 incorporates Jaxen v1.0. Such technology is subject to the following terms and conditions: JAXEN License - \$Id: LICENSE,v 1.3 2002/04/22 11:38:45 jstrachan Exp \$ - Copyright (C) 2000-2002 bob mcwhirter and James Strachan. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.

3. The name "Jaxen" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jaxen.org.

4. Products derived from this software may not be called "Jaxen", nor may "Jaxen" appear in their name, without prior written permission from the Jaxen Project Management (pm@jaxen.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the Jaxen Project (<http://www.jaxen.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at

<http://www.jaxen.org/>. THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE Jaxen AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Jaxen Project and was originally created by bob mcwhirter <bob@werken.com> and James Strachan <jstrachan@apache.org>. For more information on the Jaxen Project, please see <<http://www.jaxen.org/>>.

Progress Corticon v5.3.4 incorporates JDOM v1.0 GA. Such technology is subject to the following terms and conditions: \$Id: LICENSE.txt,v 1.11 2004/02/06 09:32:57 jhunter Exp \$ - Copyright (C) 2000-2004 Jason Hunter and Brett McLaughlin. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact <request_AT_jdom_DOT_org>.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management <request_AT_jdom_DOT_org>.

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (<http://www.jdom.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.jdom.org/images/logos>. THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the JDOM Project and was originally created by Jason Hunter <jhunter_AT_jdom_DOT_org> and Brett McLaughlin <brett_AT_jdom_DOT_org>. For more information on the JDOM Project, please see <<http://www.jdom.org/>>.

Progress Corticon v5.3.4 incorporates Saxpath v1.0. Such technology is subject to the following terms and conditions: Copyright (C) 2000-2002 werken digital. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "SAXPath" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@saxpath.org.
4. Products derived from this software may not be called "SAXPath", nor may "SAXPath" appear in their name, without prior written permission from the SAXPath Project Management (pm@saxpath.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the SAXPath Project (<http://www.saxpath.org/>)."

Alternatively, the acknowledgment may be graphical using the logos available at <http://www.saxpath.org/> THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE SAXPath AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the SAXPath Project and was originally created by bob mcwhirter <bob@werken.com> and James Strachan <jstrachan@apache.org>. For more information on the SAXPath Project, please see <<http://www.saxpath.org/>>.

