



Corticon Upgrade Guide

Copyright

Visit the following page online to see Progress Software Corporation's current Product Documentation Copyright Notice/Trademark Legend: <https://www.progress.com/legal/documentation-copyright>.

Last updated with new content: Corticon 7.1

Updated: 2025/03/17

Table of Contents

Overview of Corticon upgrades.....7

Dimensions of Corticon architecture.....9

Planning your upgrade.....11

What was removed in Corticon 6.....15

Procedures for upgrading Corticon installations.....19

Overview of Corticon upgrades

This Corticon upgrade guide describes the preparations and tasks to successfully upgrade an existing deployment of Corticon. Before you start,

- Always backup all assets and workspaces.
- If you are unsure how to proceed with an upgrade, consult with Progress Corticon Support or your Progress representative to discuss your migration plan.

Upgrade Considerations

Progress strives to provide backwards compatibility with all new releases but sometimes changes are necessary to reflect new best practices, standards, or capabilities. The probability of non-backwards compatible changes increases the older your current version of Corticon is. When planning a Corticon upgrade, be sure to consider each of the following as part of your upgrade plan:

- **Release**—At any time, there are multiple Corticon releases available. These include both LTS (Long Term Support) and non-LTS releases. When planning your upgrade decide if you want to upgrade to an LTS release or if you want access to the latest features available in a non-LTS release. Whichever you choose, be sure to start with the latest update available for that release.
- **Components**—Corticon Studio, Corticon Server, Corticon Web Console, and Corticon Utilities are setup with the separate installers. Java 17 and an appserver are not included in Corticon download packages. Identify what Corticon components are in use and need to be upgraded.
- **Security**—Progress strives to keep Corticon current with all necessary updates to address vulnerabilities reported in third party components. Updating to the most recent release of Corticon will best ensure you have all available security updates. Consult with your IT department to confirm the version of Corticon you are upgrading to meets their mandatory security requirements.
- **Type**—Corticon Server may be deployed as a web service, running in an application server, or in-process, running inside a custom application. When deployed as a web service, you need to be aware of any changes

to the Corticon REST or SOAP APIs. When deploying as in-process, you need to be aware of any changes to the Corticon Server In-Process API.

- **Clients**—When Corticon Server is deployed as a web service, there may be changes to Corticon REST or SOAP APIs which will impact the clients calling Corticon.
- **Platforms**—Corticon supports multiple versions of Windows and Linux operating systems, applications servers, databases and Java. The versions you currently use may not be supported by the version of Corticon you are upgrading to. Check the Corticon Supported Platforms Matrix to see if the versions you use are supported by the version you are upgrading to.
- **DevOps**—How you build, test and deploy decision services is a critical part of your Corticon deployment. Check to see if any of your current practices need to change as part of your Corticon upgrade or if Corticon has introduced new capabilities which will improve your DevOps practice.
- **Eclipse**—Corticon Studio is a set of plug-ins integrated with the Eclipse IDE. The Eclipse version used by Corticon Studio may have changes in the upgrade version of Corticon. If you have added third-party plug-ins to Eclipse, such as for accesses a source control system, check to see if versions of these plug-ins are available for the new version of Eclipse.

Note: Cannot install Studio into an existing Eclipse— Corticon Studio ships with Java 11 or higher, which introduces changes in how JAR files are handled: Java no longer supports the direct addition of custom JARs to the classpath as earlier versions did. As a result, the ability to drop extension JARs into the Eclipse plugin folder is no longer functional. The recommended method for adding extension JARs is in an installed Corticon Studio either:

- For Eclipse tools, choose **Help > Install New Software**, and then typically choosing tools at <https://marketplace.eclipse.org/>.
 - For your extensions, such as extended operators and service callouts, in a Rule Project, choose **Properties > Corticon Extensions**.
-

Dimensions of Corticon architecture

A Corticon implementation has several dimensions. Here are some general aspects to upgrades:

- **Release:** *Major, minor, service pack, hotfix*—To get the best of Corticon features, install the latest version. All your installations must be at the same major.minor release. More than one major.minor release can co-exist on one machine but you should strive to get upgrades and fallback positions finished and then uninstall the older version. A patch can always overlay existing installations.
- **Components:** *Studio, Server, WebConsole, Browser (WebConsole), clientApps*—Corticon's installers provide the complete Eclipse development environment and the complete deployment environment. The Web Console is installed with the server and users provide the browser for the client application runtime. Client Applications are provided strictly as samples. Customers are responsible for their apps, which typically do not need upgrades.
- **Platform:** *Windows/Progress App Server (Java), Windows/IIS(.NET), Linux(Java), AppServer-WAR(Java)*—The typical installation is on Windows with the Tomcat appserver. The utilities that transform a Java Server to .NET Server and then deploy to an IIS server are provided. Linux and App Server WAR installations are produced in every release so that they stay in sync with other installations.
- **Eclipse and Java:** *Provider/version from/to*—The Eclipse and Java version/provider are installed with the product. See the Supported platforms page.
- **Deployment strategy:** *CDD, EDS*—While the historic technique was to move all the rule assets to the server for compilation and deployment, due to security considerations, now version 6.x does not support server-side compilation. As such, CDDs are listings of EDS files in contrast with directly deploying an EDS.
- **Type:** *Web Services, In-Process*—Business planning determines which strategy is better. Both are offered and the techniques for successful deployment on either are documented. Note however that strategies for widely dispersed in-process servers can create slowdowns when upgrades are offered.

Planning your upgrade

Deciding when to upgrade:

- **Stay on a supported version**—Do you want LTS or non-LTS? See [Corticon Life Cycle](#)
- **Support for latest platforms**—Are your platforms supported? See [Corticon Supported Platforms Matrix](#)
- **Access to latest security and bug fixes**—Are there new features you want to access? See [What's New in Corticon](#)

What changed in 7.1.1

The service pack release 7.1.1 extends AI development, and provides some customer-requested features, such as a new utility for Custom Reporting, an updated REST Datasource, and improved JSON Support in Tester.

What changed in 7.1

Consider the impact of an upgrade to 7.1:

- **Corticon Installers** – Changed in 7.1, Corticon now consists of 5 separate installers, providing you greater control over installations and updates. See the *Corticon Installation Guide* for details.
 - **Corticon Studio**– Provides a complete platform for modeling rules for Corticon.
 - **Corticon Server** – Provides the Corticon Server for Java for deployment to supported J2EE application servers and Java applications.
 - **Corticon Server .NET** – Provides the Corticon Server .NET for deployment to Microsoft IIS application server and .NET applications.
 - **Corticon Web Console** – Provides a Web UI for managing Corticon Server and Server .NET deployments.
 - **Corticon Utilities** – Provides command-line utilities for automating the packaging, testing, and deployment of Corticon decision services.

Note: A Corticon Server install provides the Docker configuration file, `Dockerfile`, and the `.war` files for Corticon Server and Corticon Web Console. See *"How to deploy Corticon on Docker" in the Deployment Guide*.

- **Java Requirement** – Changed in 7.1 is the requirement for Java 17 and the packaging of Java. See “The Corticon Supported Platforms” for details on supported Java distributions.
 - **Java 17 Not Bundled** – Java is no longer packaged with Corticon Server, Server .NET, Web Console, or Utilities. You need to download and install Java 17 to use any of these Corticon installations.
 - **Java 17 Bundled with Corticon Studio** – Java is bundled with Corticon Studio, enabling you to use Studio without a separate Java 17 installation. The distribution of Java bundled with Studio should not be used with any other Corticon components.
- The new **AI Assistant**, empowering you to explore and optimize your rule assets. Automatically generate project documentation, detect rule similarities, identify potential problems, and ask the AI Assistant any questions to improve your rule projects.
- **Generate unit tests** for individual Rulesheets from existing systems tests. This granular testing approach speeds up regression detection and ensures your rule logic continues to perform as intended.
- **Manage multiple Corticon Server versions** seamlessly from a single instance of the Corticon Web Console. This unified management approach simplifies upgrades and provides full visibility across all deployments.
- **Complexity Report** is a new report option that provides a statistical summary of the complexity of a ruleflow.

What changed in prior LTS releases

If you are advancing from an older installation, also evaluate the impact of an upgrade based on those changes to the technologies and the tools:

- **No longer provides EAR download**—Corticon does not provide EAR deployment (and EJB) since 6.x and only WAR deployment is available. However, Corticon does support EAR files as a packaging for your enterprise application where you want to deploy multiple applications. You could add multiple WARs and your applications to an EAR file and deploy one EAR file into a supported application server that then deploys each of the WARs individually.
- **Both LTS versions provide the same Web Console**—The Corticon Web Console is a distinct Corticon Server installation, `Corticon.war`, that provides browser access to administrative features for many Corticon Servers, and enables many powerful features such as deployment across all Servers in a group, staging of deployment packages from Studio, batch processing, and analysis of Server performance.

Note: Corticon no longer supports the older Server Console that was distributed as a subset of the Server download in 5.6 and 5.7.

- **.NET Server changed from cross-compiling to a proxy**—Starting at 6.0, Corticon .NET Server no longer cross-compiles Corticon into .NET code so that your code could call directly into the Server. Now, the .NET code can connect to a proxy that forwards the call to a in-memory Java JVM where Corticon Server is running. This reduces the complexity and amount of extra processing to deal with cross-compiled code. There are limitations to the proxy implementation:
 - Business objects are supported in Corticon, but not on .NET Servers. If you are using Business Objects, convert your payloads from business objects to JSON or XML, and then pass JSON or XML to the proxy.
 - If you have implemented .NET extended operators or service callouts for Corticon Server .NET and cannot convert them to Java, contact Progress Support for options on using existing .NET extensions with Corticon 6.0 and later.

Note: IMPORTANT You must request a Corticon license specifically for evaluation and runtime of the Corticon .NET Server. No evaluation license is included when you install Corticon Server for .NET. Hence, it will not run without a proper license.

- **WSDL**—Decision Service compilation no longer includes the WSDL and the report in the EDS file by default. If you use WSDL or reports in the Decision Service, add the following lines to the `brms.properties` file of the Studio that is producing the EDS:

```
com.corticon.server.compile.add.wsdl=true
com.corticon.server.compile.add.report=true.
```

For more information, see the Deployment Guide topic ["Properties that impact Decision Service compilation in the Corticon Deployment Guide"](#). See also the Web Console topic, ["WSDL" in the Corticon Web Console Guide](#)

- **Metadata tags**—Metadata tags, a technique for annotating data to identify the entity and the instance, are only required on the root entity in a request. You will notice that metadata tags are given when you choose to export the request to JSON. Both are valid. However, metadata tags are required when you use `#ref_id` tags. See ["About creating a JSON request message for a Decision Service"](#) and ["Sample JSON request and response messages" in the Deployment Guide](#)
- **Corticon Extension JARs are now built-in**—When ADC was introduced in 5.7.2, and REST in 6.0 was introduced, you had to specify the `CcADCSCO.jar` and `CcRESTSCO.jar` as Corticon Extensions. At 6.2, those JARs were added to the core Corticon Jars, so you no longer need to 1) add the JARs as Studio project extensions, 2) include the JARs in the `.eds` files, or 3) include the JARs in the deployed `axis.war` on an application server. However, you do need them if you choose to upgrade to 6.1.

Note: The process to upgrade assets that had explicit specifications of these JARs will not have those references removed. You will see warnings after upgrading a project. The condition is benign. The files can be removed from the Corticon Extensions section of the Project properties. However, the packaging and testing techniques in Studio will ignore the listed JARs.

- **Database access properties melded with Datasource Configuration File**—Starting at 5.7.2, the Datasource Configuration File `datasource.xml` replaced the `database.properties` file previously used with EDC Decision Services. While you can still deploy using the `database.properties` file, you are encouraged to move to the `datasource.xml`.
- **Corticon Server registration in Corticon Web Console**—When new servers want to be managed in the Web Console or just to change an IP address, you can add properties to the server's `brms.properties`

file that will connect and authenticate on the Web Console. For details, see ["Server registration with Web Console" in the Server Guide](#)

- **Swagger config file**—The version of Swagger in the product is, as of 7.0, downloaded with the server installation, but it is no longer implemented by default. Further back, the `config` file changed from `open-api` in 6.1 to `swagger` in 6.3. See [REST API Swagger documentation](#) and [Test the installed Corticon Server](#) for details.
- **Custom extensions and Service Callouts**—You might encounter conflicts if you have extended operators or service callouts that include 3rd party JARs that conflict with Corticon JARs. To resolve such conflicts, remove the conflicting JARs from your extension, or upgrade them to versions compatible with the version of Corticon.

Getting the new version

A new version requires that you have appropriate licenses and access to the Progress Electronic Software Download (ESD) site, accessed at <https://www.progress.com/esd>. While you are in the ESD, it is a good idea to download the End User Legal Agreement (EULA) and save it locally, as the EULA precisely describes the license terms.

1. **Are you under maintenance?**—If so, have your license holder contact Progress Customer Order Management at com-global2@progress.com to get access to a new version of Corticon.
2. **Can you use your existing license?**—Generally, no. All Corticon Studios, Corticon Servers, assets in projects, and deployed Decision Services must be at the same version. Check your ESD account to see if new licenses have been provided for you by Progress.
3. **How many cores does each server CPU have?**—Corticon Server licenses can be restricted to the number of cores available for serving concurrent execution requests. For example, if you have a 4 core license, only 4 concurrent executions will be allowed regardless of the number of cores on the system.
4. **How long can you run the old and new in parallel?** As stated in the End User Legal Agreement: *Each Update to an On-Premise Product replaces part or all of the On-Premise Product (or earlier Update) previously licensed to you ("Replaced Product") and will terminate such previously licensed Replaced Product to the extent replaced by the Update; provided, however, that you may continue to operate the Replaced Product for up to ninety (90) days from delivery of the Update to allow you to complete your implementation of the Update. You must cease all use of the Replaced Product at the end of the ninety (90) day period.*

Note: Load balancing—If you have a load balancing architecture, traffic can be redirected when a new Corticon Server version is installed adjacent to a previous version. It is good practice to backup all workspaces before proceeding. If you are using the Corticon Web Console to administer Servers, remove one of the Servers from its Server group. Then, this architectural setup will incur no down time. Leave the previous version untouched until the new version is installed. Then, delete the other Corticon Server and re-install it as the newer version. When you add the Server to its Server group, all the updated Decision Services will be deployed. Note that the Web Console server itself should not be load balanced.

What was removed in Corticon 6

This section summarizes the features that are no longer supported and no longer documented as of Progress® Corticon® 6.0.0:

- **Deployment of Ruleflows** —Corticon Server no longer supports deployment of ERF files, only compiled EDS files can be deployed. Previously you could deploy Ruleflows and associated assets to Corticon Server so that the Server would compile the assets into runnable Decision Services. That was a performance hit every time that Decision Service needed to be recompiled and reloaded due to changes to the rules, which could lead to error situations in production if the Decision Service failed to compile because all dependent files were not uploaded to the server. You can generate EDS files with Corticon Studio, Corticon command line utilities, or Corticon ant macros. If deploying ERF files with Corticon Deployment Descriptors (CDD files), you will need to modify these to deploy EDS files. Eliminating ERF deployment helps enforce Continuous Integration/Continuous Deployment (CI/CD) best practices by encouraging the deployment of fully compiled and tested Decision Services. See the topics ["How to package and deploy Decision Services" in the Deployment Guide](#).
- **Corticon Deployment Console**—The Corticon Deployment Console bundled with Corticon Server provided a GUI mechanism for creating CDD files, compiling Decision Services, and generating WSDL files for Decision Services. These tasks can each be performed in other ways.

If you have been using the Deployment Console, its functionality is available as:

- **Command line interface for creating CDD files**—The CDD file format is a simple text file manifest describing a Decision Service. The most common practice is to copy a sample CDD file bundled with Corticon, and then make modifications to it in a text editor. The new `corticonmanagement` command `-cdd` as described in ["Create a CDD file" in the Corticon Deployment Guide](#), enables all the CDD options for a single Decision Service deployment description file in one command.
- **Compile Decision Services**—Multiple techniques in the Deployment guide perform compilation:
 - [Use Studio to compile and deploy Decision Services](#)
 - [Use the `corticonManagement` utility to automate packaging and testing of Decision Services](#)
 - [Use Server API to package and deploy Decision Services](#)
- **WSDLs are now generated from the Vocabulary and Ruleflow editors**—To create WSDL, see ["Generate Service Contracts in Corticon Studio" in the Corticon Deployment Guide](#).
- **Download of Decision Services**—Corticon Studio no longer allows download of Decision Services. As rule assets can no longer be staged for compilation on the Server, the best practice for managing your rule assets is to store them from Studio in a source code control system such as Git. Corticon Studio now includes plugins for using Git with your rule projects.
- **SOAP Management API (except execution)**—Corticon Server no longer provides SOAP API to deploy or manage Decision Services. You will need to migrate to using the equivalent REST APIs. Note that the ability to execute a Decision Service with a SOAP request will not be removed, only the management APIs are affected. See ["Test the installed Corticon Server on Java" in the Web Services Guide](#). See

Note: Corticon Server's REST APIs are accessible:

- For your deployed Corticon Server at <http://localhost:8850/axis/swagger> (use your appropriate host name and port number)
 - For your Corticon Server for .NET running on IIS, Swagger is not available. Instead, see [Corticon Server .NET APIs](#)
-

- **Enterprise JavaBeans deployment**—With the removal of `CcServer.ear`, Corticon no longer provides sample EJBs that call into a running Corticon Server..
- **.NET Business Objects**—Corticon no longer supports the use of .NET business objects for executing Decision Services in-process. The use of business objects for runtime execution was primarily done for performance. Now that Corticon uses technology for .NET executions your performance-critical in-process deployments can pass JSON or XML to Corticon, you should do that, not business objects. The performance of JSON is superior to that of business objects. If you have been using .NET business objects, contact your Progress representative for guidance in migration strategies.
- **Option `useForQueryService` in `datasource.xml`**—The `useForQueryService` option in a `datasource.xml` file for identifying the datasource to use for ADC queries has been removed. In its place, add a **Query** datasource to your Corticon vocabulary to identify the source of ADC and Batch queries. See ["Define and import queries for ADC" in the Data Integration Guide](#).
- **EDC database.properties file** —The option to specify EDC properties via a `database.properties` file has been removed. In its place, use a `datasource.xml` file. The `datasource.xml` file can define properties for multiple datasources. See the section "Importing Datasource and Database Access configurations" in ["Define the database connection for EDC" in the Data Integration Guide](#) for more information.

-
- **Passing REST API arguments in HTTP Header**—The Corticon Server REST API no longer supports passing arguments in the HTTP header. This mechanism did not fully support localization. In its place, pass arguments as query String parameters, as illustrated:



```
... "name": "Quote Policy Premiums",
... "majorVersion": "1",
... "minorVersion": "0",
... "effectiveTimeStamp": "4/7/2022 7:34:32 AM",
... "__metadataRoot": {
...   "#locale": ""
... },
```

- **Rule Execution Recording Service**—The Corticon Server rule execution recording service has been removed. In support of this, the Corticon Studio **Project > Create Execution Recording Schema** menu option has been removed.
- **WebConsole Monitoring of In-Process Corticon Servers**—The Corticon Server no longer provides an option to run an in-process Tomcat server for use when using Corticon in-process. This change results in in-process Corticon Servers not being monitorable with the Corticon Web Console. To monitor an in-process Corticon Server with the Corticon Web Console, your container application will need to proxy the Corticon REST API to the Corticon Server. Contact your Progress representative for guidance in migration strategies.
- **ICcServer API Changes**—Previously deprecated methods in the ICcServer API have been removed. In addition, methods present for support of the Java Server Console have been removed. Users running Corticon in-process may need to use alternate method signatures with 6.0. See the [Corticon Server JavaDoc](#) for current API methods.
- **Sample Extended Operators**—The SeMath and RandomGenerator extended operators have been removed. The RandomGenerator was replaced by the [Random](#) operator for Decimal and Integer data types. Be sure that your rules replace the deprecated ones with the new operators.

Procedures for upgrading Corticon installations

To synchronize your files and assets with the updated product, perform the tasks that optimize upgrades of Corticon Studios and Servers. These tasks are a checklist of good practices.

If your upgrade is:

- **Hot fix**— Provides fixes to one or more customer reported defects. Hot fixes are provided only to the customer which reported a defect. Hot fixes are inclusive of all prior fixes – it will include all fixes from prior hot fixes. When installing a hotfix, review the readme for the release for a summation of defects which are fixed. Upgrading to a new hotfix should be an easy overlay step with no changes in behavior.

Example:

- Upgrading from 7.1.0.0 to 7.1.0.1
- Upgrading from 7.1.0.0 to 7.1.0.4

- **Service pack**— Rolls up all fixed bugs, and is made available to all customers. When installing a service pack, review the readme for the release for a summation of defects which are fixed. Upgrading to a new service pack should be an easy overlay step with no changes in behavior.

Example:

- Upgrading from 7.1.0.0 to 7.1.1.0
- Upgrading from 7.1.0.0 to 7.1.1.4

• **Major or minor release**—Includes new features and may include changes to existing features and supported platforms. Although Progress prioritizes backwards compatibility, upgrading to a new major or minor release may require additional steps. When installing a major or minor release, review the What's New guide for the release to understand what has changed. Keeping your installation current with new Corticon release will simplify upgrades – the older the release you are upgrading from, the higher likelihood there will be additional steps required to complete the upgrade. A major minor release will be installed as a new product path and assets. The two versions can both run; however, if run concurrently you must avoid port conflicts.

Example:

- Upgrading from 7.0.0.0 to 7.1.0.0
- Upgrading from 5.7.4.0 to 7.1.0.0

IMPORTANT: Corticon Studio and Corticon Server versions must be consistent throughout your infrastructure.

Note: Do **not** copy a `.war` file from an older version installation to a new one.

In the order of action:

ON STUDIO MACHINES

- [Update Studio installations](#)
- [Update the Studio's brms.properties](#)
- [Update eclipse.ini and](#)
- [Update the Studio license file](#)
- [Upgrade assets](#)
- [Update desktop shortcut](#)
- [Repackage Decision Services and their Datasource Configuration file](#)

ON SERVER MACHINES

- [Undeploy the old Decision Services](#)
- [Install Corticon Web Console software](#)
- [Install Corticon Server software](#)
- [Update the Server license files](#)
- [Enable the Server's Swagger functionality](#)
- [Update the Server's brms.properties](#)
- [Clear browser caches that use the Corticon Web Console](#)
- [Deploy regenerated Decision Services](#)
- [Stop load balancers and start the new Server](#)

ON STUDIO MACHINES

Install Studio

Note: You can keep an installed prior version of Studio for 90 days. See your EULA for information about the "Replaced Product."

For the updated product, proceed with [Installing Corticon Studio](#).

Update the Studio's brms.properties file

Compare the `brms.properties` file that is currently in use. If any property adjustments you made in the past are still valid, copy them into the newer installation's `brms.properties` file. While a minor release installs into a separate location, a service pack overlays an existing installation and does not touch an existing `brms.properties` file.

For information about the `brms.properties`, see ["Studio Properties and settings" in the Rule Modeling Guide](#) and ["Server Properties and settings" in the Corticon Server Guide](#).

Update eclipse.ini

Some Studio installations benefit from tweaking the `eclipse.ini`, often for Studio memory allocation. See ["Increase Corticon Studio memory allocation" in the Corticon Installation Guide](#). If you are pointing to your preferred , it might not be optimal.

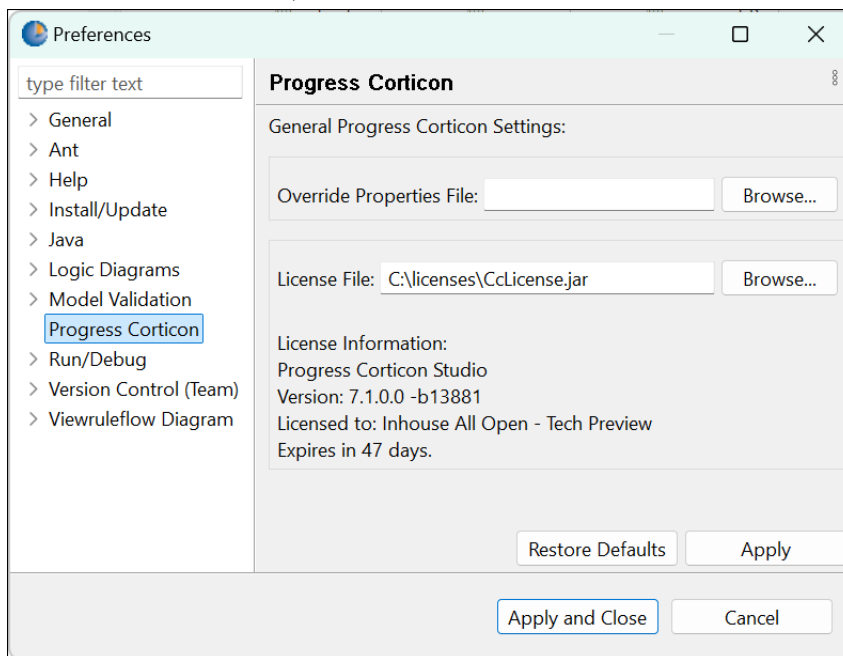
Note: Do not copy `eclipse.ini` from an older release. After you do an install, compare your old file to the new one to determine whether any lines are appropriate to migrate.

Update the Studio license file

A major and minor release requires a new license file. When you are provided a license for Corticon Studio and Server, you receive a JAR file, `CcLicense.jar`. Save the JAR file named `CcLicense.jar`, which you receive with the Corticon Studio and Server license, on each target machine. To avoid performance issues, Progress strongly recommends that you do not put your Studio license on a network drive, as that may cause network latency issues in Studio due to the frequent check of this license file when rule assets are edited. In this example, the license file that both Studio and Server use is placed at `C:\licenses`.

To update a Corticon Studio license:

1. Go to **Window > Preferences**, and expand the **Progress Corticon** group.
2. In the **License File** field, enter or browse the location of the license JAR:



3. Click **Apply**, and then click **OK**.

Upgrade assets

In Studio, be sure to upgrade all project assets. You can only upgrade one project at a time. The upgrade process separates the assets that did not need to be upgraded from those that did, and those that failed. A text report is placed at the root of the project.

It is a good practice to run any Ruletest or unit tests to see that the behaviors are as expected.

Note: If you are using a workspace that was used in a previous installation, you might need to refresh the workspace's links to **Welcome** screen topics. In Studio, click the **Home** button to refresh the **Welcome** links.

Update desktop shortcut

If you want to run your new Studio version in a different language, make changes to your desktop shortcut. In essence you are modifying the desktop shortcut to append `-nl ja`. For details and an example, see *"Set Studio to run in another language" under "Options after installing Corticon Studio" in the Install Guide*.

Repackage Decision Services and their Datasource Configuration file

As you cannot deploy the Decision Services until the Servers and Web Console are updated, choose the option **Package and save for later deployment** so that they are staged with their freshly generated Datasource Configuration file for deployment on the updated Server and Web Console installations .

ON SERVER MACHINES

Undeploy old Decision Services

Undeploy every decision service on the target Servers that were generated using a prior version.

Note: This might not be necessary with the updated implementation of the Web Console that supports multiple versions of decision services.

Install 17

Java is no longer installed with the Corticon runtime component download packages. See *"Java requirements" in the Installation Guide* for details.

Install an app server

Tomcat is no longer installed with the Corticon download packages. Instead the server installer provides the appropriate `.war` file for Docker and each supported app server at `C:\Progress\Corticon 7.1\Server\Deploy`.

Install Corticon Web Console software

If you want to use the Corticon Web Console, download and run the Corticon Web Console installer on a machine where an app server is already installed. You can run its `corticon.war` on a machine that collocates with a Corticon Server's `axis.war`.

After you install servers, you can have the servers register automatically into the Corticon Web Console. The properties and instructions are [Update the Server's brms.properties](#).

Install Corticon Servers

You must install an app server on each machine that will host a Corticon Server.

Note: If you are using load balancers to carry the load during upgrade, take one of its peers offline, and then stop it.

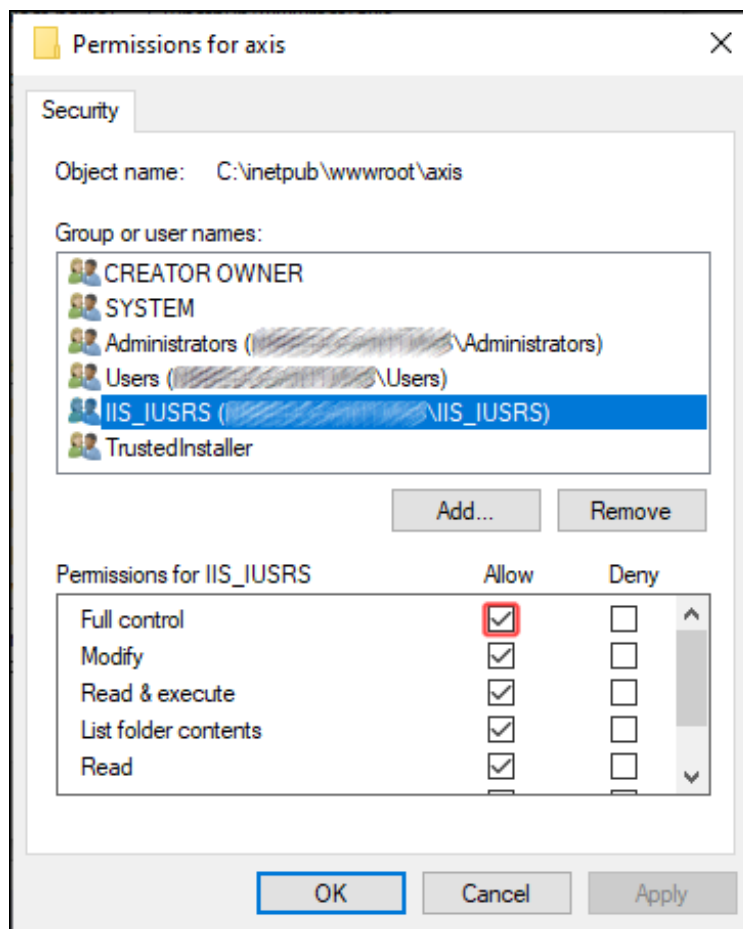
- **.NET server**—The Corticon Server installer option for .NET provides the `install.bat` script that will setup the Corticon application and the `axis` folder on the .NET IIS server.
-

Note: If you are upgrading into an existing Corticon .NET installation, you must stop and remove the current installation:

1. Backup the `axis` and `Corticon` directories under `c:/inetpub/wwwroot`.
 2. If you have a .NET license, put it aside as it will get added in to the new installation.
 3. Stop the IIS, then click on `Sites\Default Web Site`, and then right click on any Applications and choose **Remove**.
 4. In the File Explorer, delete the folders `C:\inetpub\wwwroot\axis` and `C:\inetpub\wwwroot\Corticon`.
 5. Close the IIS Manager, and then relaunch it.
-

To install Corticon for .NET to the IIS server:

1. Launch `C:\Progress\Corticon 7.x\Server .NET\IIS\install.bat` as **Administrator**. Press **Enter** to run the script.
2. After installation, immediately add your authorized .NET license file, `CcLicense.jar`, to the IIS server's `C:/inetpub/wwwroot/Corticon/lib`. Without this file the Corticon .NET server will not run.
3. Start the IIS Manager, and then do the following steps:
 - a. Expand `Sites` and `Default Web Site`.
 - b. Right-click on `axis`, and then choose **Convert to Application**. In the dialog, confirm or select **Application pool .NET v.4.5 Classic**, and then click **OK**
 - c. Right-click on `Corticon`, and then choose **Edit Permissions**. In the dialog, click the **Security** tab. Click **Edit**, and then choose **IIS_IUSRS**. Click the **Allow - Full control** box, as shown:



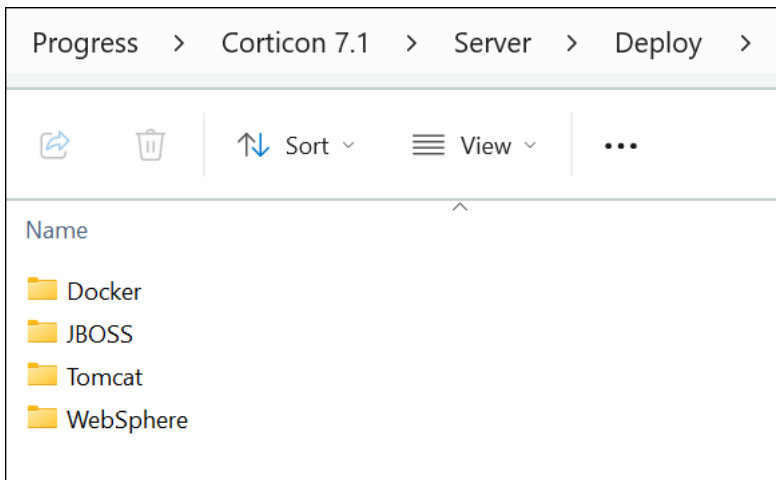
Click **Apply**, then **OK** to both dialogs.

4. In a browser, enter <http://localhost/axis/corticon/server/ping>. The uptime is displayed.

Note: Multiple applications on one IIS Server—You can have several distinct applications on an IIS server that have separate logs and assets. See *"Set up Corticon Server for .NET for multiple applications"* in the *Corticon Web Services Guide*.

For more information on Corticon .NET server installation as well .NET deployment security, see ["Web services on .NET" in the Corticon Web Services Guide](#).

- **Application servers**—The Corticon Server .war files are installed with a Server installation in the **Server > Deploy** folder:



Note: The Web Console requires Tomcat 9.

Note: To review the currently supported UNIX/Linux platforms and brands of application servers, refer to [Corticon Supported Platforms Matrix](#). For detailed instructions on configuring Corticon Server on all supported platforms, see the Corticon KnowledgeBase entry [Corticon ServerCorticon Server sample WAR installation for different Application Servers](#).

Update Server licenses

Note: See the topic *"Corticon runtime component licensing" in the 7.1 Installation Guide* for details about licensing changes starting in 7.1.

The Server requires you to copy the `CcLicense.jar` file and then paste it to replace the existing file, as follows:

- **Server license:**
 - `[CORTICON_SERVER_HOME]\Server\lib\`—used by sample applications and the Corticon Management Utility.
 - When you install the `.war` file for deployment onto supported platforms and application servers, place the license file adjacent to the Corticon JARs. If you are managing a default Server from the Corticon Web Console, you can perform this task by referring to ["Edit Server groups and Servers" in the Corticon Web Console Guide](#).
- **.NET Server license:**

Note: You must have a license that enables .NET Server to work with Corticon. See [Knowledge Base article: Corticon licensing](#). The license in the installation where you setup the .NET server does not copy its license file `[CORTICON_HOME]\Server .NET\ThirdParty\lib\` to the IIS location.

Once you have a Corticon 7.1 Server license that supports .NET Server issued to you by Progress, copy the file to `C:\inetpub\wwwroot\Corticon\lib\`. If you are managing a .NET Server from the Corticon Web Console, you can perform this task by referring to ["Edit Server groups and Servers" in the Web Console Guide](#).

Enable the Server's Swagger functionality

The Swagger REST API documentation is installed with Corticon Server 7.x as `CcSwagger.jar`, but it is not located in the server's path, and its configuration is not enabled. In brief, copy the installation's `CcSwagger.jar` to the Tomcat `WEB_INF/lib`, and then edit

`web.xml` to set `corticon.swagger.enabled` to `true`. Restart the server. For details, see *"The REST API Swagger documentation" in the Corticon Server Guide*.

Update the Server's brms.properties

As with Studio, compare, update and tune the `brms.properties` files on each and .NET Server, considering that many properties are effective only on Servers. For more information see ["Server properties and settings" in the Server Guide](#)

Note: On .NET Servers, you can copy the `brms.properties` file from the Corticon Server root, paste it in the `Corticon` folder in the IIS location, and thereafter apply updates to it.

Corticon Server registration in Corticon Web Console—When new servers want to be managed in the Web Console, you can add properties to the server's `brms.properties` file that will connect and authenticate on the Web Console, and even put the server into a specified group so that the server automatically gets decision services deployed in the group. For details, see ["Server registration with Web Console" in the Server Guide](#)

Clear browser caches that use the Corticon Web Console

For administrators who use the Corticon Web Console, clear the browser caches.

Deploy the fresh Decision Services

Deploy the newly created Decision Services to the new Server. When using CDDs, be aware that you might have two types of data access configuration files. The legacy format used a `.properties` file while new format is a `datasources.xml` file.

- Where you have a legacy `.properties` file with EDC, use CDD Property :

```
PROPERTY_DATABASE_ACCESS_PROPERTIES_PATH
```

- Where you are have a new style `.xml` file, use CDD Property :

```
PROPERTY_DATASOURCE_CONFIG_FILE_PATH
```

Stop load balancers and start the new Server

Use Corticon Web Console (or other techniques) to deploy the upgraded EDS files and license to the new Server. Bring the other load balancers down, and then expose the upgraded Server to carry the load. Uninstall old Servers, and then upgrade and provision to be peers.