Progress®
Corticon®

Get Started with Corticon.js

Copyright

Last updated with new content: Corticon.js 2.4

Updated: 2025/11/19

Table of Contents

What is Corticon.js?.....7

Install Corticon.js Studio.....9

How to upgrade Corticon.js.....10

Model rules with Corticon.js.....11

Package rules with Corticon.js 17

Set preferred language for Corticon.js Studio.....21

Uninstall Corticon.js Studio.....23

1

What is Corticon.js?

Corticon.js Studio enables business users, domain experts, and JavaScript developers to define rules to automate critical business decisions. Automating business decisions ensures that you consistently apply business policies and comply with regulations.

Automate decisions with Corticon.js Studio

Corticon.js brings the capabilities of the proven Corticon business rules engine to JavaScript. With Corticon.js, you can define rules and package them into fully, self-contained JavaScript bundles that can be deployed to any compatible JavaScript platform.

Every business has rules, policies, and regulations that must be enforced. Whether it is deciding which loan terms to offer on a mortgage application or determining if a benefit claim should be approved, decisions must be made accurately and consistently. Corticon.js provides the ability to automate business decisions.

Data	Rules	Decisions
Mortgage Application		Loan Offer
College Transcript		Course Recommendations
Vehicle Lease		Terms & Conditions
Medical Diagnostics		Health Report
Insurance Claims		Benefits Paid

With Corticon.js, you can define your rules once and deploy them to where decisions need to be made.

Corticon.js Studio provides the tools to:

- Define the data model, or vocabulary for your rules.
- Model rules as Rulesheets in an intuitive spreadsheet-like user interface.
- Validate Rulesheets to ensure that there are no conflicts and all conditions are included.
- Arrange Rulesheets into Ruleflows for complex decisions.
- Test Rulesheets and Ruleflows using the integrated Testsheets.
- Package rules for JavaScript deployment.

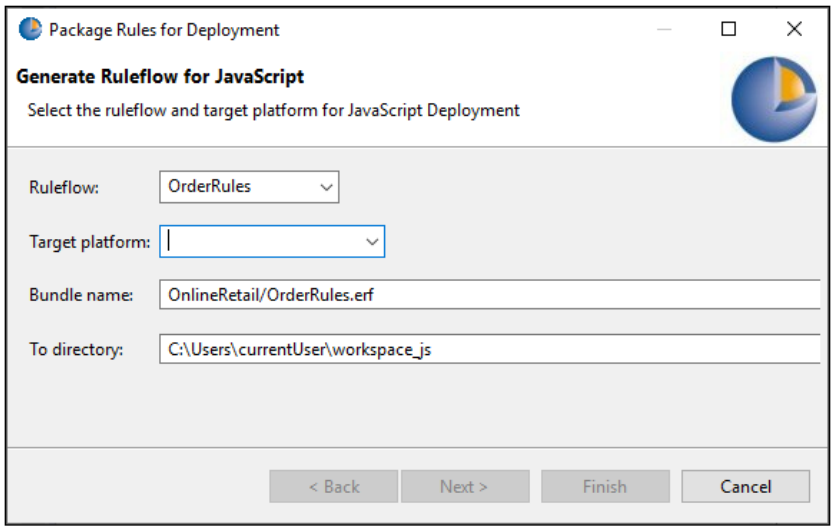
Corticon.js frees you from dependence on your IT department to write code for your automated business decisions.

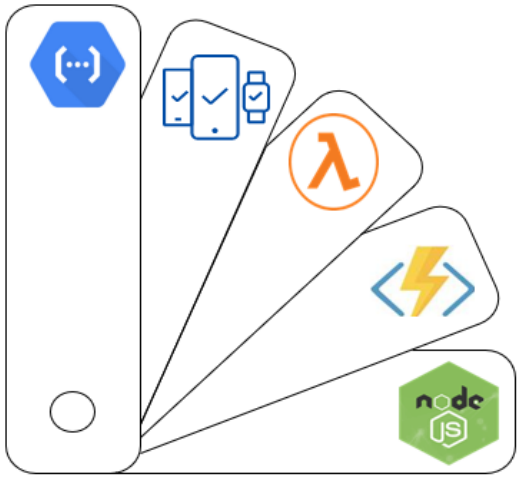
Deploy rules where decisions are made

With Corticon.js Studio, you can deploy rules to any platform that supports a compatible version of JavaScript. With this flexibility, you automate decisions where the data originates from or where the decisions have the most effect. Deployment opportunities with Corticon.js include:

- Rules deployed as serverless functions on AWS Lambda or Microsoft Azure Functions
- Rules integrated into a cloud work flow such as AWS Step Functions or Microsoft Flow
- Rules run on your own back end as part of your Node.js platform
- Rules bundled in a mobile app created with Xamarin, React, Vue.js, or other toolkits
- Rules executed in a browser as part of a web application

Note: The list is a sample of the available options. With Corticon.js, rules are compatible with platforms using the compatible JavaScript version.





Packaged as a JavaScript bundle of your choosing

2

Install Corticon.js Studio

Corticon.js Studio provides the tooling to automate rules and package them for deployment.

Requirements

Before installing Corticon.js, confirm that you have the following:

- **System requirements**
  - Supported version of Microsoft Windows. For more information, see Corticon.js Supported Platforms Matrix.
  - 8 GB RAM (minimum of 2 GB available RAM)
  - 670 MB disk space
- **Administrator permissions on the target machine**—See your system administrator to obtain these rights.
- **Read and write permission for the work directory**—Corticon.js uses the work directory to write logs and other files produced by Corticon.js.
- **Anti-virus, anti-malware, and anti-spyware** — You might need to disable your protection applications temporarily.

Install Corticon.js Studio

1. Download the Corticon.js Studio installer:

   a. Get credentials to access and download packages on the Progress Software Electronic Software Download (ESD) site.
   b. Go to the ESD, log in, and then navigate to the Corticon.js 2.x page.
   c. Locate, download, and save the required installer, `PROGRESS_CORTICON_2.x_JS_STUDIO_WIN_64.exe`, to a temporary location accessible by the target machine.

2. Double-click `PROGRESS_CORTICON_2.x_JS_STUDIO_WIN_64.exe` to run the installer.
3. Review the information on the **Introduction** page.
4. Click **Next**.
5. Click **Next**.
6. Set your **Install Directory** and **Work Directory** to your preferred locations.
7. If you have a Corticon.js 2.x license staged, choose to add it the installation
   - If you select **Add Corticon license?** on this panel, the entry area for the **License File** lets you enter or choose the path where you have staged your license file on the local machine.
   - If you skip the selection, a normal installation will proceed. However, the Studio will not open for your use until you add a license file from a staging location. You will then need to open Studio, choose **Preferences > Progress Corticon** , and then enter or browse to the license staging location.
8. Click **Next**.
9. Review the information on the **Pre-Installation Summary** page, and then click **Install**.
10. When the install process completes, click **Done** to close the installer.
11. On the **Start** menu, choose **Progress > Corticon.js Studio 2.x** to launch the Studio workbench.

For details, see the following topics:

- How to upgrade Corticon.js

How to upgrade Corticon.js

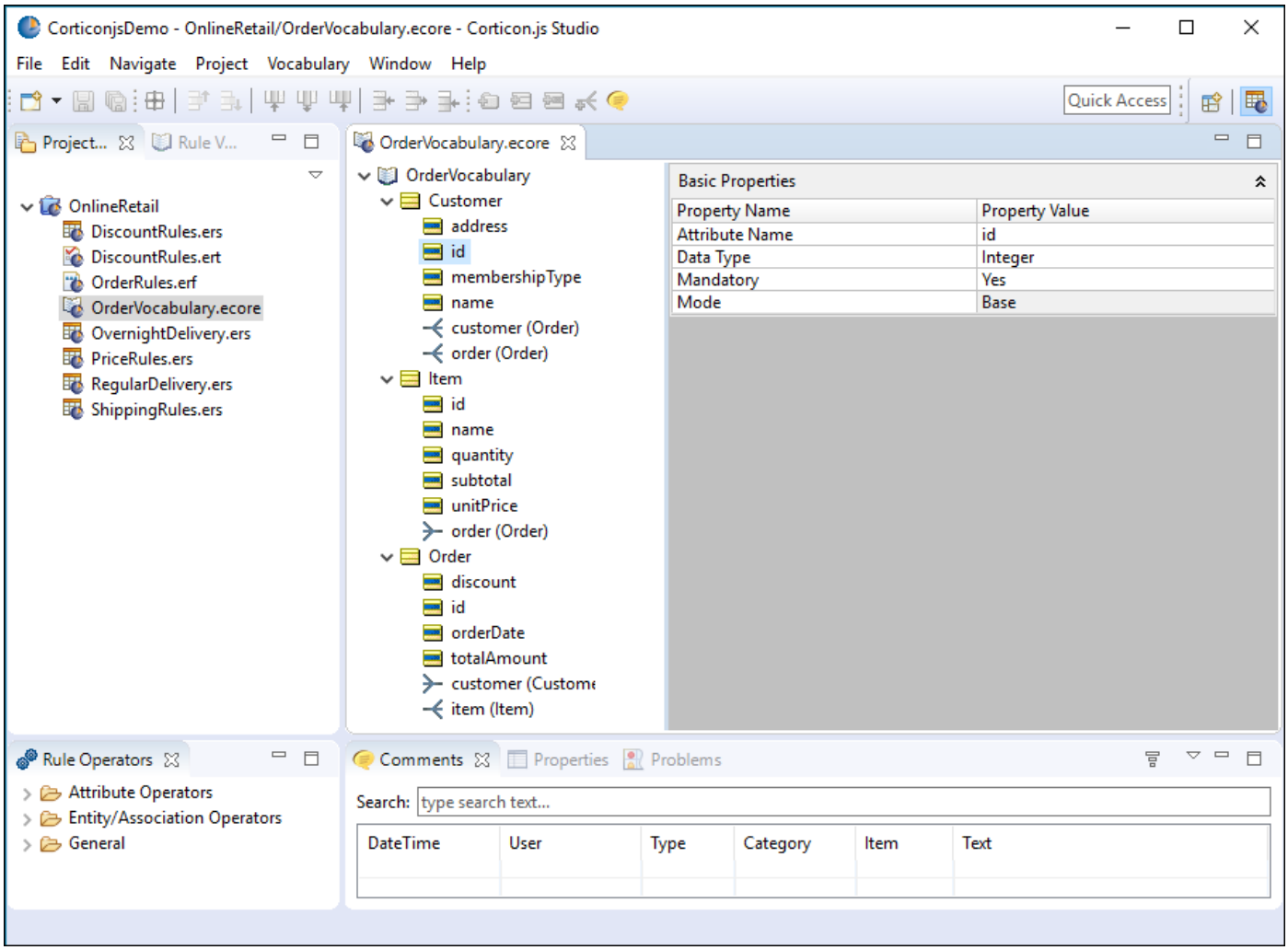
Corticon.js is easily upgraded:

- You can install the Corticon.js 2.2 release and installation on the same system as an earlier installation.
- In the 2.2 Studio, copy the project from the earlier release.
  - Select the project. Right-click to choose **Upgrade Rule Assets**. Then right-click to choose **Validate Project**.
- As the services are serverless, you do not have to "undeploy" the existing decision services.

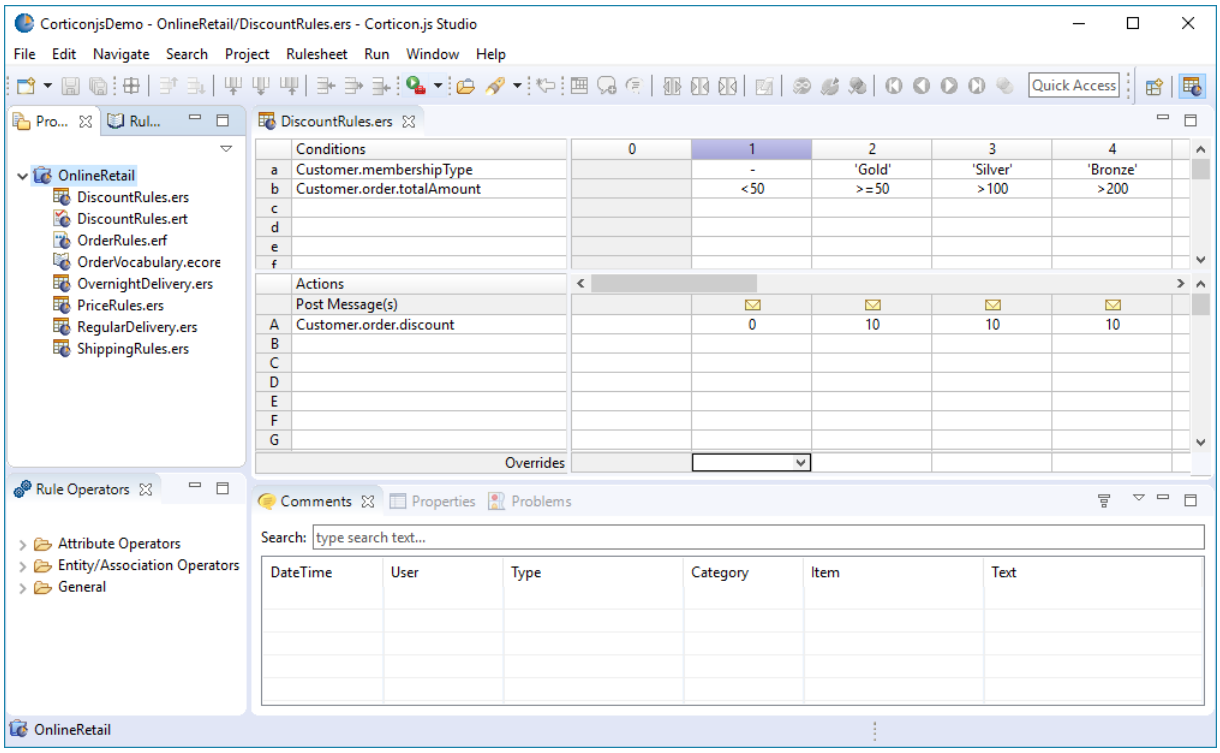
3

Model rules with Corticon.js

Every Corticon.js rule project starts with a vocabulary. The vocabulary is where you specify the data model your rules operate on by defining the entities, attributes, and associations between entities.

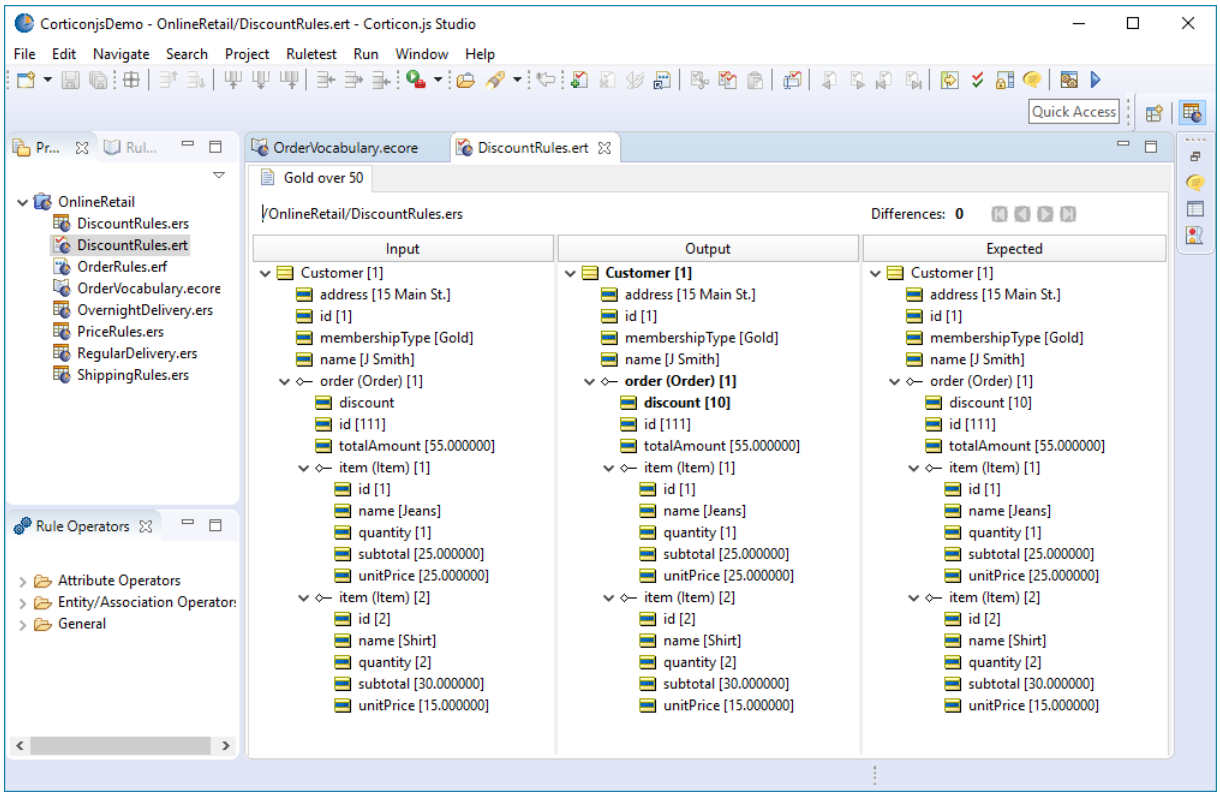


Next, you create Rulesheets that contain the rules for your application. A Rulesheet contains one or more rules where each rule includes conditions that, if true, result in the corresponding action being performed. In this example, the conditions are to check the membership type and the total amount of the order. If membership type is equal to Gold and the total amount of the order is greater than or equal to $50, then the action sets the discount equal to 10 percent.

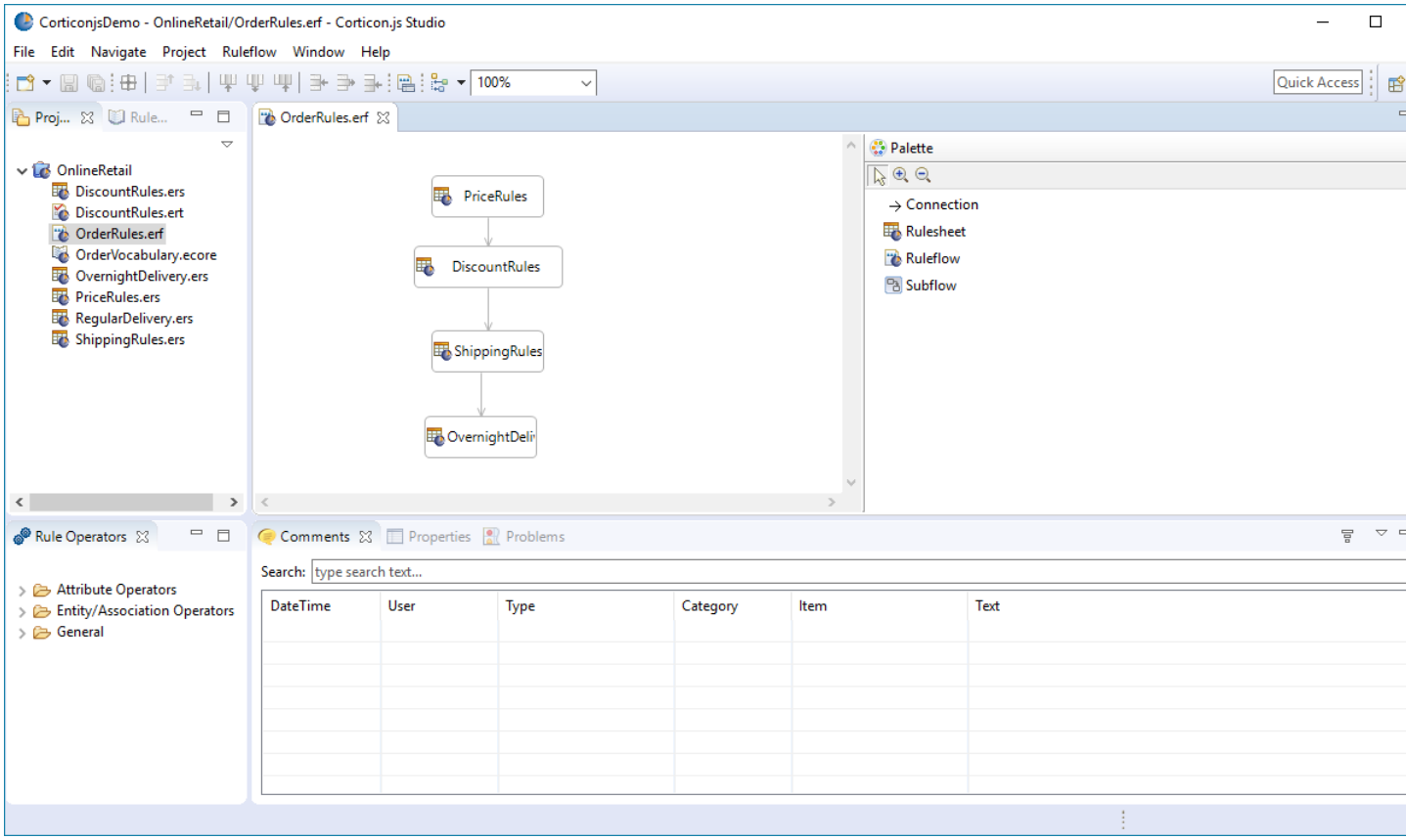


With your Rulesheets created, you are ready to test them. In Corticon.js Studio, you define Testsheets. Each Testsheet defines test input data and the expected output. Running Testsheets confirms that the expected results are produced and protects against regressions when future rule changes are made.

In this example, the Testsheet input order has no value assigned to the `discount` attribute. The expected discount is 10 percent, based on the input values for `membershipType` and `order.totalAmount`. When the test is run, the output is generated, and the expected discount can be compared to the output `discount` value assigned by the rule that fired.



Rulesheets are then added to a Ruleflow. A Ruleflow automates complex decisions by dividing the rules across multiple Rulesheets. This modularity simplifies development and testing, and enables reuse. You can even use Ruleflows within other Ruleflows. You can add a Testsheet for a Ruleflow to test the output for the Ruleflow.



With the rule modeling complete, the next step is to package the rule for deployment on the JavaScript platform of your choice. For more information on rule modeling, see *"Introduction to Corticon.js rule modeling" in the Corticon.js Rule Modeling Guide*.

4

Package rules with Corticon.js

After the rules are tested, you package the rules for deployment to any of the supported JavaScript platforms. Packaging rules is easy.

How to package rules for JavaScript deployment

When you package rules for deployment, you select the **target platform**. Corticon will generate a JavaScript bundle with your rules and a wrapper specific to the target platform. The available target platforms are:

- AWS Lambda
- Azure Functions
- Google Cloud Functions
- Browser
- Node.js

Selecting AWS Lambda, Azure Functions, or Google Cloud Functions as the target platform generates the rules and a wrapper ready for deployment as a serverless function. After the rules are deployed, you can use them on your cloud platform—for example, to expose the decision service through a REST endpoint, to respond to a database update, or to integrate into a cloud vendor workflow system such as AWS step functions.

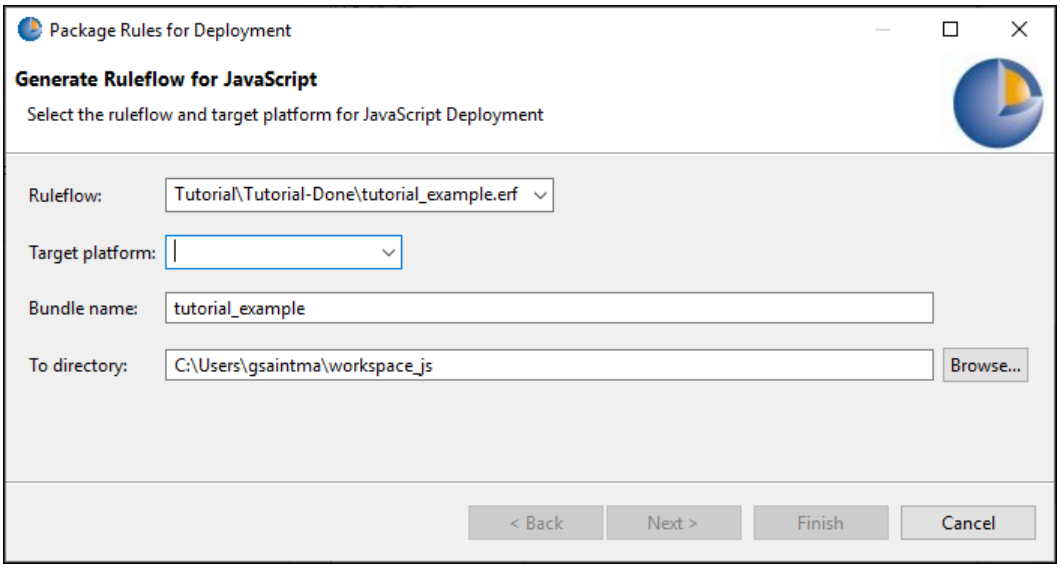
The Node.js target can be used for running decision services in Node server and as well to run them in Mobile applications created with NativeScript and ReactNative.

Selecting Browser generates the decision service and simple example code that demonstrates how to integrate the rules into your application.

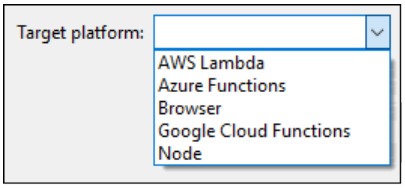
Note: See the Corticon.js Supported Platforms Matrix to review the supported JavaScript and web platforms, and mobile apps. You are not limited to these JavaScript platforms. The API for integrating rules in your JavaScript application allows you to develop your own wrapper or integration code that calls rules.

To create a Corticon JavaScript package:

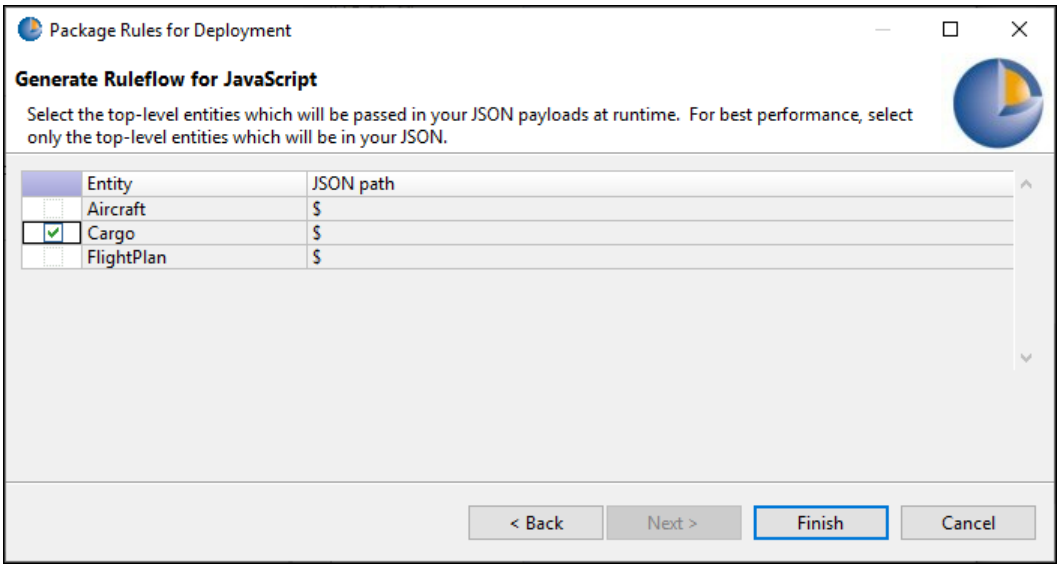
1. In Corticon.js Studio, select a project, and then select **Project > Package Rules for Deployment**. The **Package Rules for Deployment** dialog box opens:



2. Select the **Ruleflow** for your application.
3. Choose the **Target Platform**:



4. Enter a preferred **Bundle** name so that you produce your package in a distinctly named folder. Each Ruleflow selected generates a separate package.
5. Enter a preferred **to Directory** where your package will be placed.
6. Click **Next**. The second panel of the dialog opens:



7. Choose the top-level entities for this package. In our example, only the Cargo entity is referenced so just that entity is selected. The top-level entities are entities that will be expected (and required) in payloads executing the decision service. You could choose all entities or none, but that will mean that the decision service will have to evaluate each entity, which might be a lot of wasted cycles.

   Note: For more information, see *"JSON payloads and results in Corticon.js" in the Corticon.js Integration Guide*

8. Click **Finish** to generate the packaged JavaScript as a Decision Service plus everything you need to deploy it. The packaging process generates a single JavaScript bundle that is the entry point for the rules, which contain all the decision logic in its enclosed components for the selected Ruleflow. The generated bundle is compressed and obfuscated. It is valid, executable JavaScript, but it is not readable and cannot be edited.

Note: When evaluating Corticon.js, every Decision Service that you package has an embedded expiration date based on your evaluation license.

Integrate rules into applications

Rules are packaged into a self-contained JavaScript bundle, and a wrapper for the targeted platform is created.

- For AWS Lambda, Google Cloud Functions, and Azure Functions, the wrapper is ready to be deployed as a serverless function to the cloud. After the wrapper is deployed, you can make it accessible as a REST service, or integrate it into a cloud workflow.
- For Browser and Node, the wrapper provides an example of the code to integrate the rules into your own application.

Integrating rules into your own application can be done in three steps:

1. Include the generated rule bundle.

```
const decisionService = require('./decisionServiceBundle');
```

2. Get the JavaScript Object Notation (JSON) payload from your application.

```
payload = application.getPayload();
```

3. Execute the rules on the payload.

```
result = await decisionService.execute(payload);
```

The payload is a JSON document mapped to your rule vocabulary representing a loan application, benefits claim, vehicle configuration, or other business object required to make a decision.

The result returned is a JSON document including the input payload and any decisions made by the rules such as a risk rating, claim approval, vehicle quote, or other attributes set by the rules when processing the payload.

5

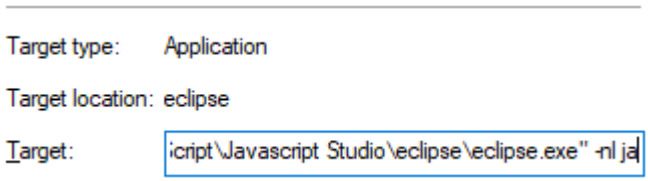
Set preferred language for Corticon.js Studio

You can specify a preferred language for Corticon.js Studio UI. The language options are:

- English (`en` - default)
- French (`fr`)
- Japanese (`ja`)
- Portuguese (Brazilian) (`pt_BR`)
- Spanish (`es`)

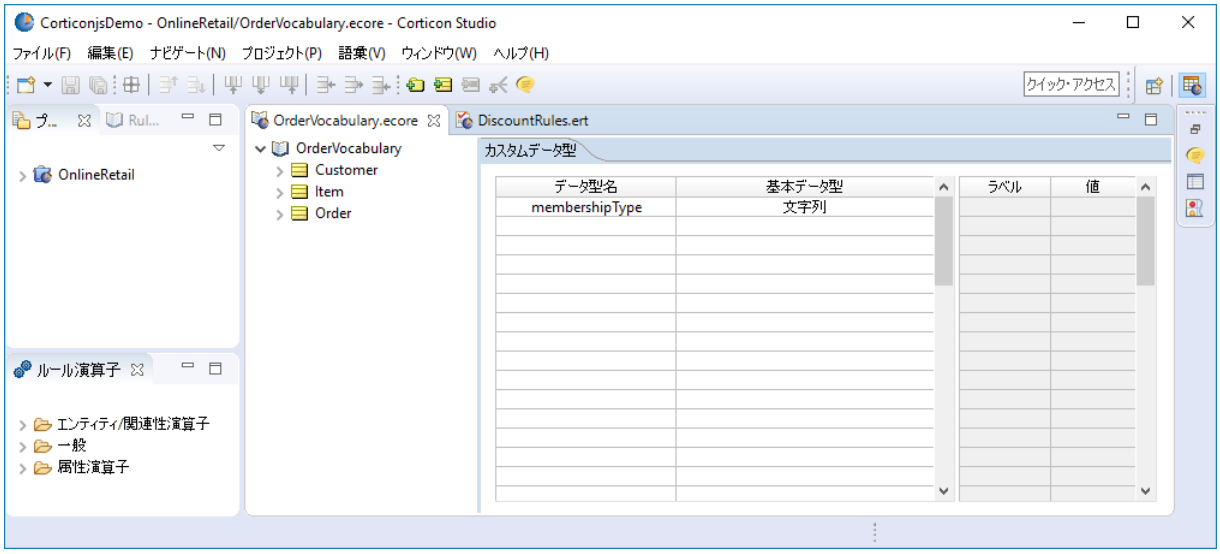
To set Corticon.js Studio to start in your preferred language:

1. On Windows 10, right-click on the **Start** menu item for the **Corticon.js Studio**, and then choose **More** > **Open File Location**.
2. Right-click on the shortcut **Corticon.js Studio**, and then choose **Properties**.
3. Choose the Shortcut tab, and add to the Target value the language option, `-nl`, followed by a language value, for example, `ja` for Japanese:



4. Click **Apply**, and then click **OK** to set the change.

When you start Corticon.js Studio, its labels are in Japanese, as shown:



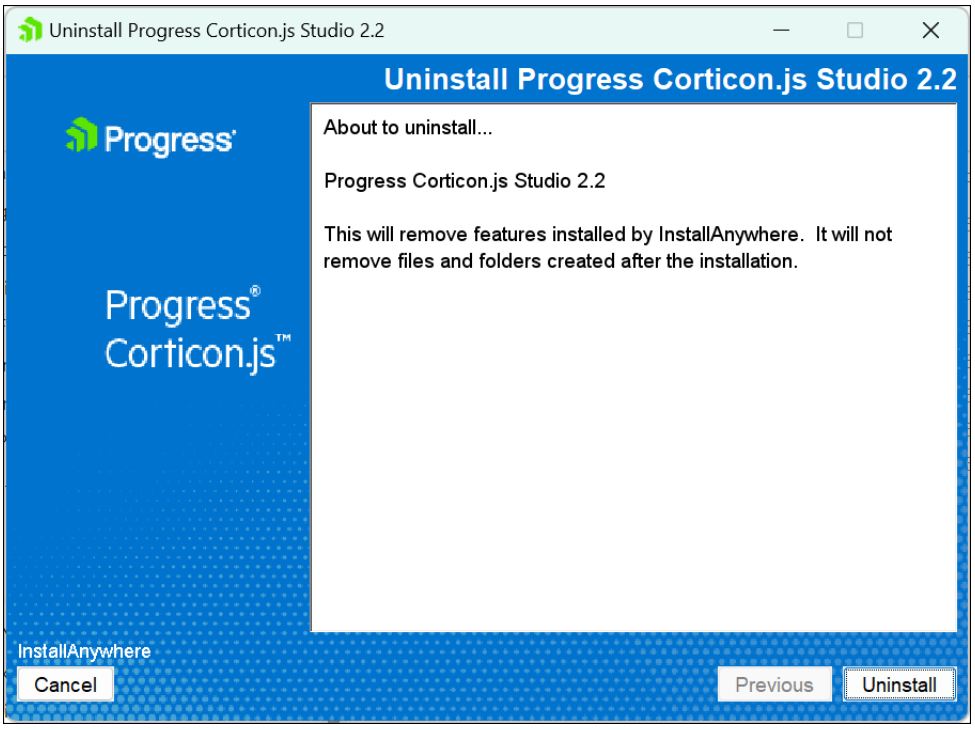
Note: A preferred user language might use different separator symbols than those documented for decimal values, list ranges, and dates. However, JSON payloads must represent decimal values using the period character as the decimal separator. The Corticon.js runtime does not support localized decimal separators.

6

Uninstall Corticon.js Studio

To remove a version of Corticon.js, do the following:

1. Close Corticon.js Studio.
2. Search for the **Add or Remove Programs** setting.
3. Double-click **Progress Corticon.js Studio**.
4. Click **Uninstall**.



The installed files in the Corticon.js Studio *[CORTICON_HOME]* are removed. Files you created (including the complete workspace) are **not** removed or replaced during this process.

If the Uninstaller program is unable to fully remove components, then it displays messages. Confirm that Corticon.js Studio was closed and run the uninstaller again.