



Get Started with Corticon.js

Copyright

© 2020 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.

Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, Icenium, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. Analytics360, AppServer, BusinessEdge, DataDirect Autonomous REST Connector, DataDirect Spy, SupportLink, DevCraft, Fiddler, iMail, JustAssembly, JustDecompile, JustMock, NativeScript Sidekick, OpenAccess, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, and WebClient 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.

Last updated with new content: Corticon 6.1.1

Updated: 2020/10/07

Table of Contents

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

Install Corticon.js Studio.....9

Model rules with Corticon.js.....15

Package rules with Corticon.js21

Set preferred language for Corticon.js Studio.....25

Uninstall Corticon.js Studio.....27


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.

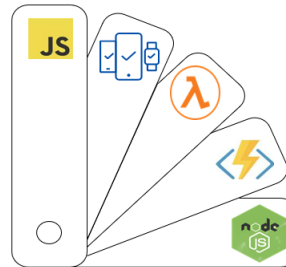
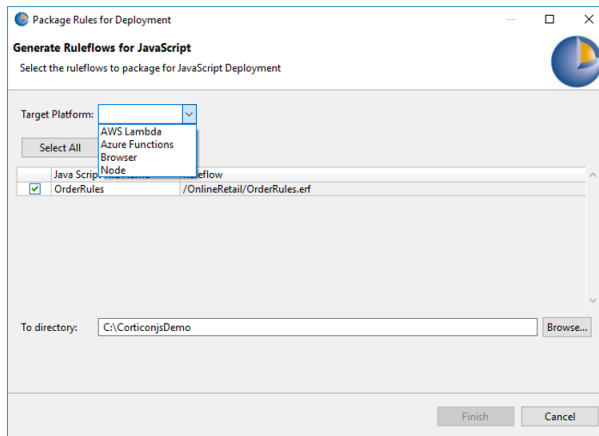
Corticon.js frees you from dependence on your IT department to write code for your automated business decisions.

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

Install Corticon.js Studio

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

Requirements

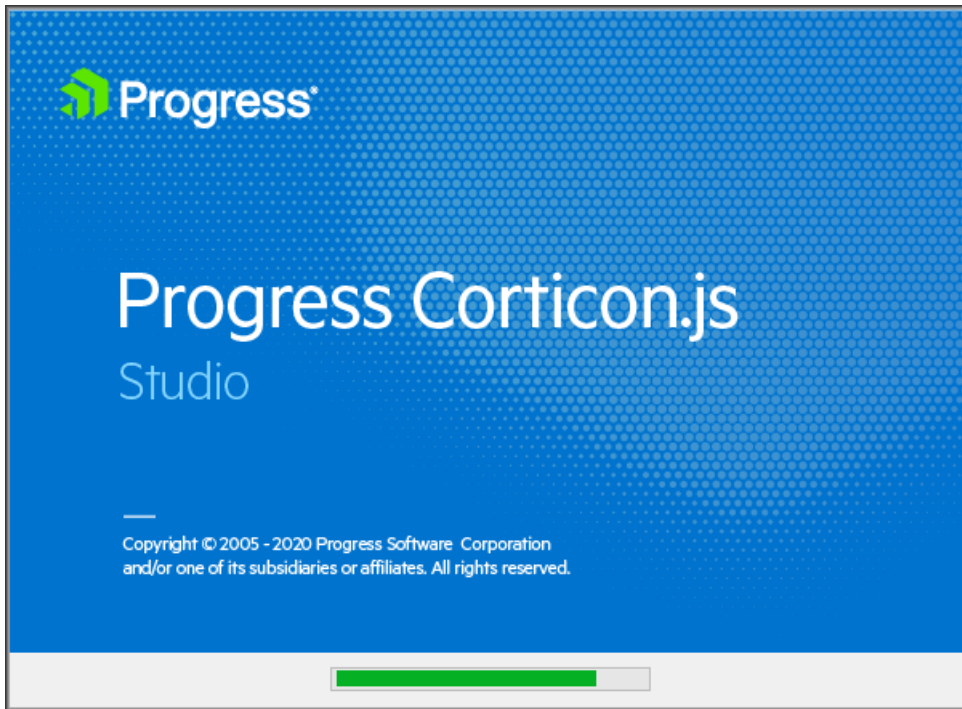
Before installing Corticon.js, confirm that you have the following:

- **System requirements**
 - Supported version of Microsoft Windows. For more information, see [Corticon.js platform 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** — Disable all protection applications.

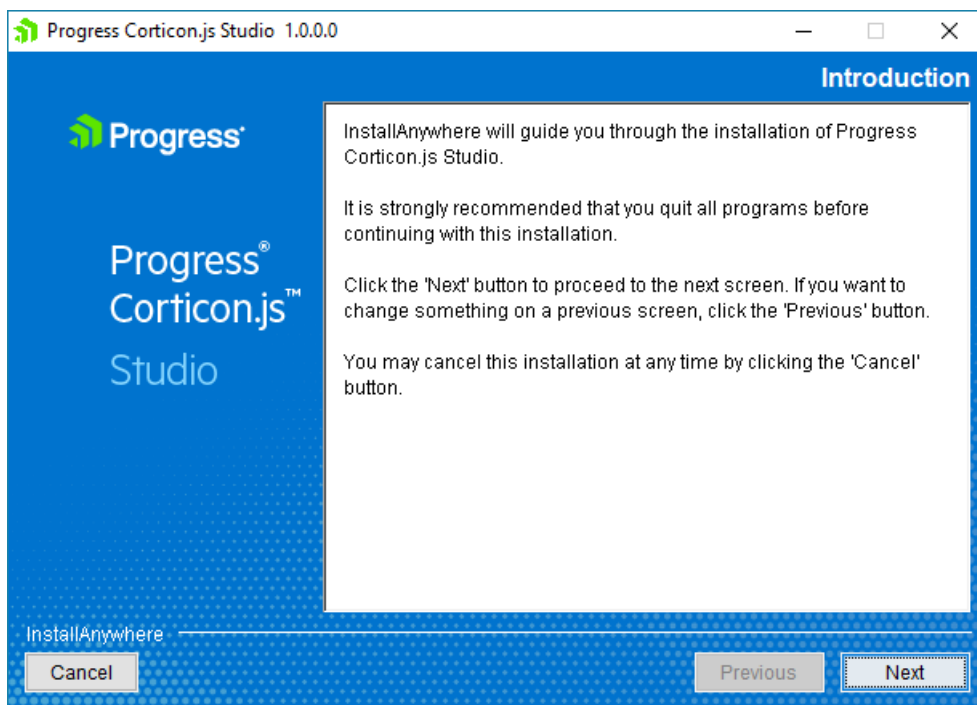
Install Corticon.js Studio

1. Download the Corticon.js Studio installer:

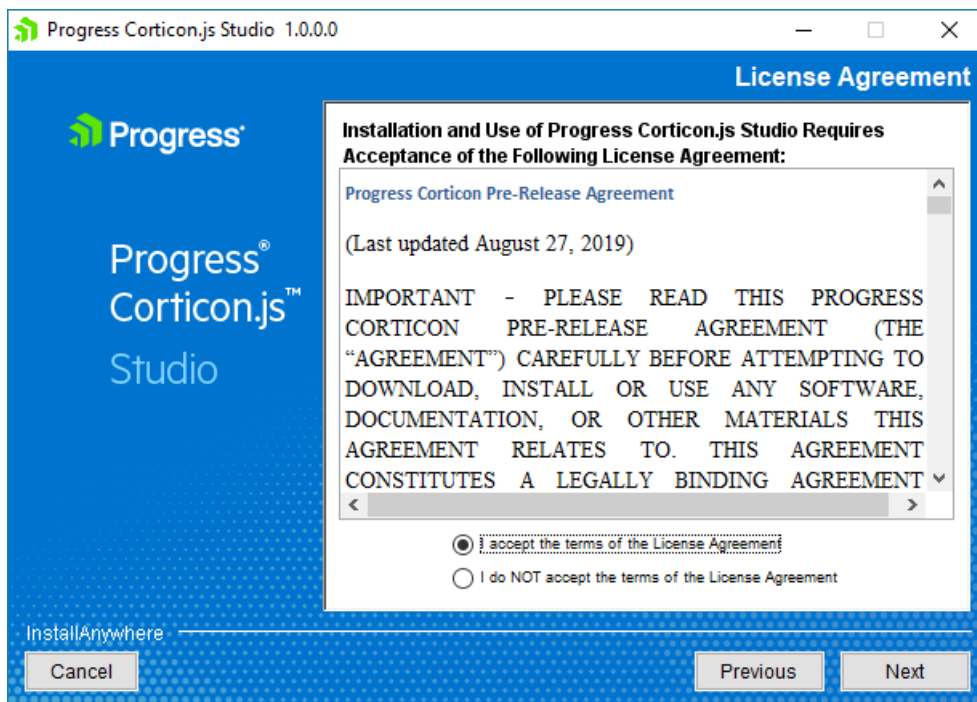
- a. Get credentials to access and download packages on the Progress Software Electronic Software Download (ESD) site.
 - b. [Go to the ESD](#), log in, and then navigate to the Corticon.js 1.0 page.
 - c. Locate, download, and save the required installer, `PROGRESS_CORTICON_1.0_JS_STUDIO_WIN_64.exe`, to a temporary location accessible by the target machine.
2. Double-click `PROGRESS_CORTICON_1.0_JS_STUDIO_WIN_64.exe` to run the installer.



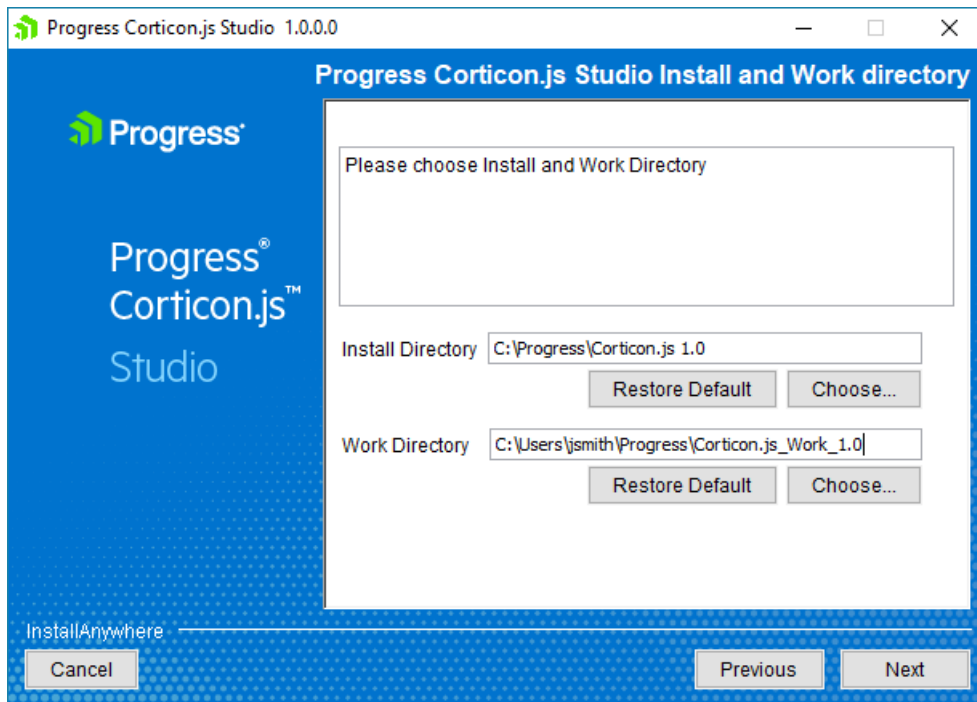
3. Review the information on the **Introduction** page and click **Next**.



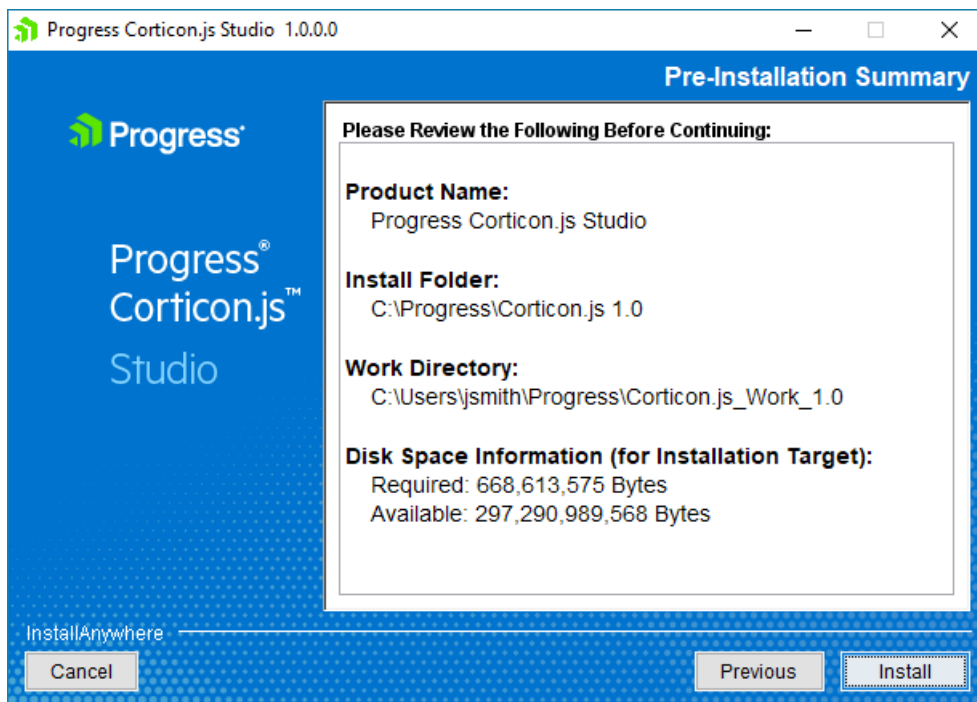
4. Review and accept the license agreement, and click **Next**.



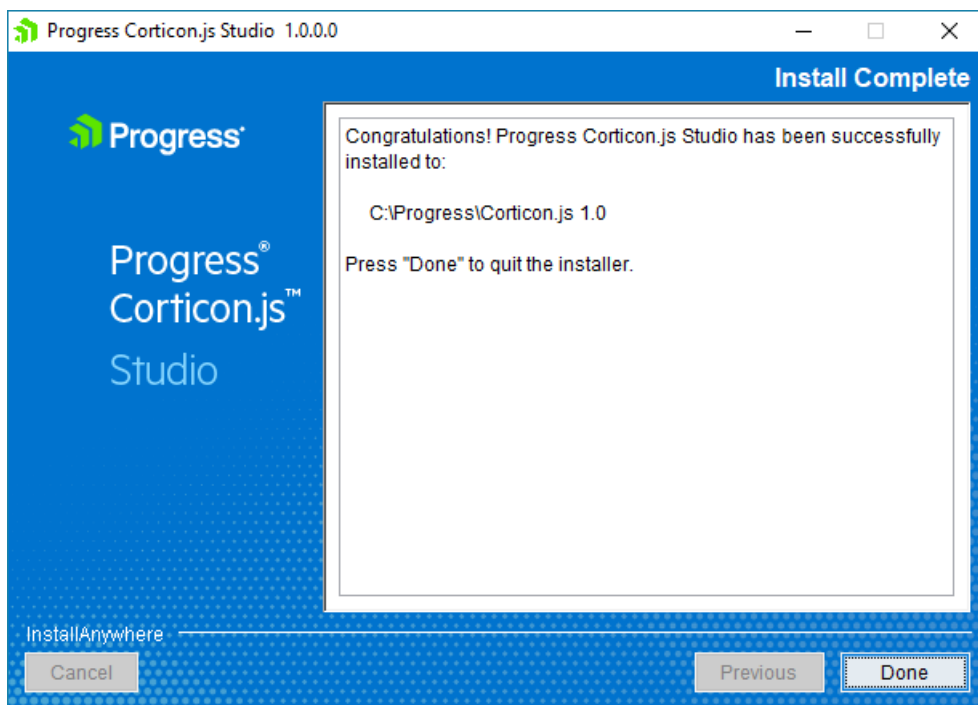
5. Set your **Install Directory** and **Work Directory** to your preferred locations, and click **Next**.



6. Review the information on the **Pre-Installation Summary** page, and click **Install**.



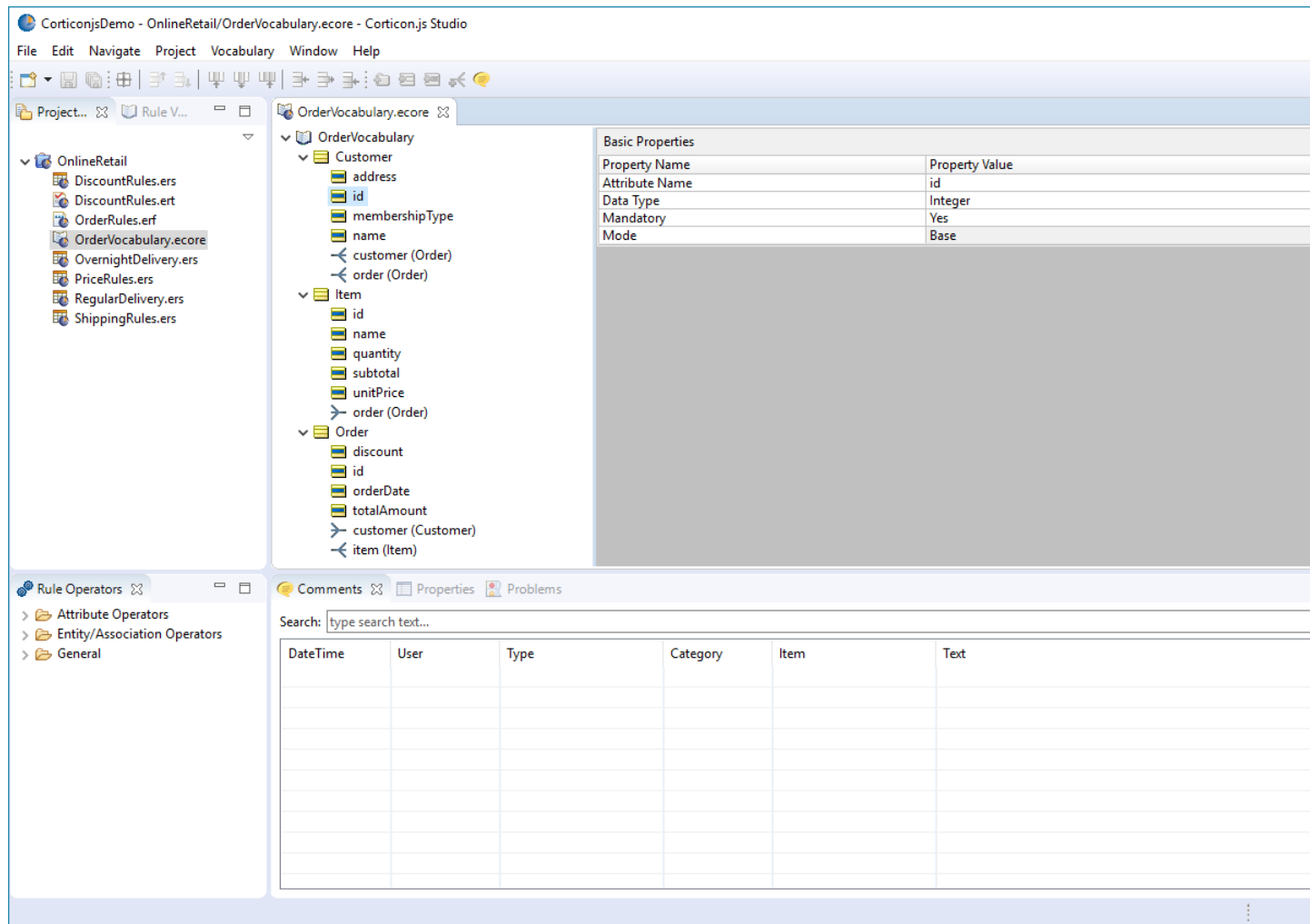
7. Click **Done** to close the installer and begin using Corticon.js Studio.



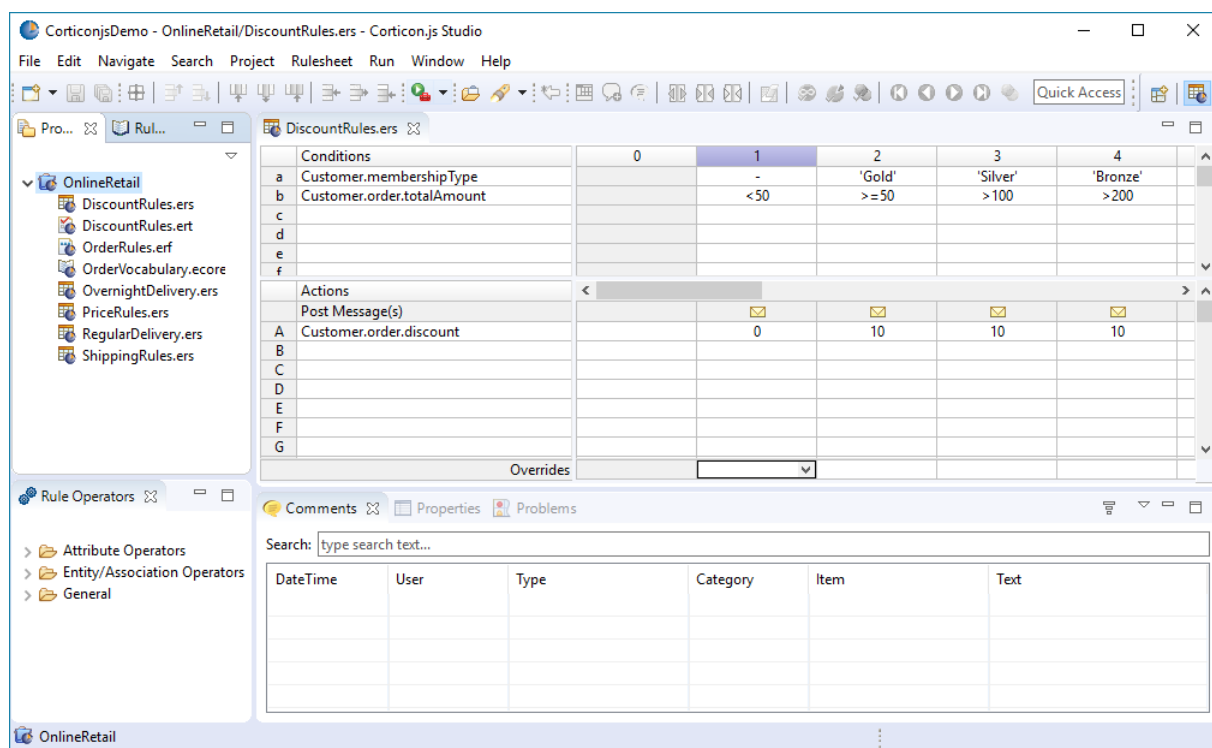
To learn more about Corticon.js Studio, see the [Corticon Information Hub](#).

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.

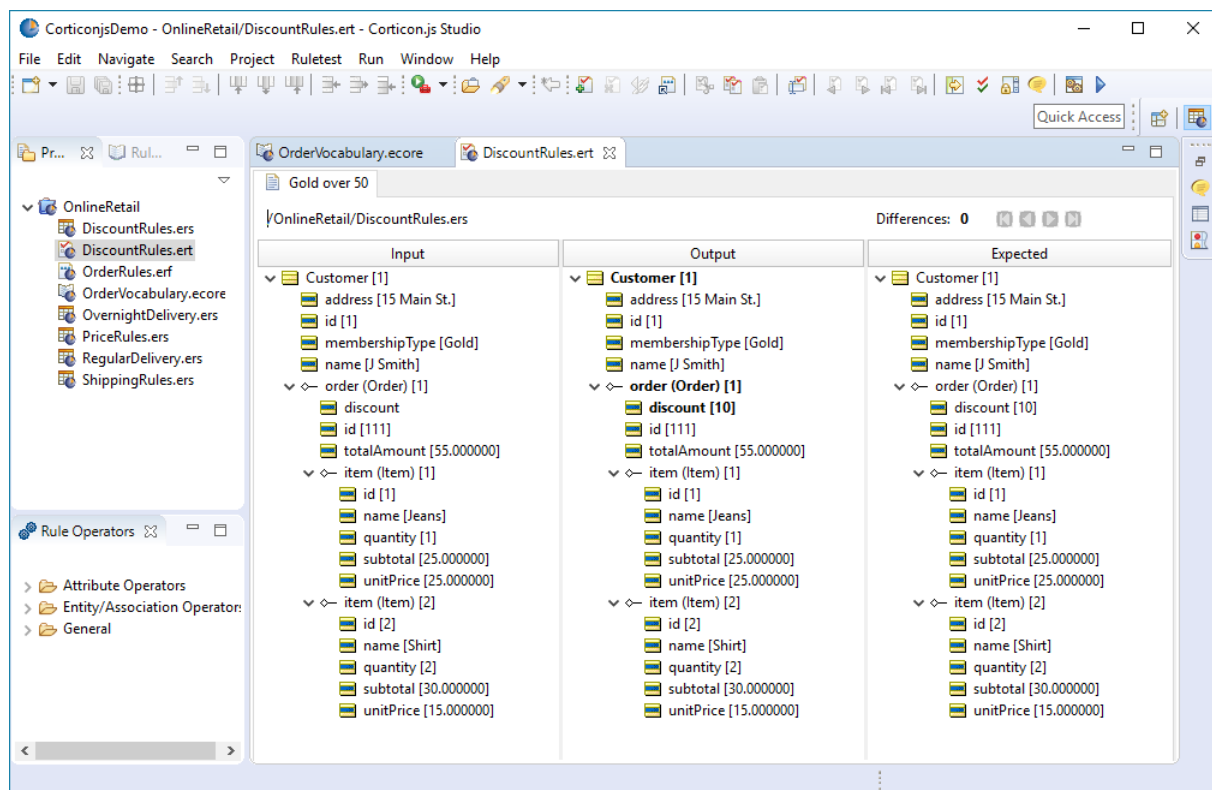


Next, you create Rulesheets that contain the rules for your application. A Rulesheet contains one or more rules where each rule includes conditions that, if true, result in the corresponding action being performed. In this example, the conditions are to check the membership type and the total amount 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.

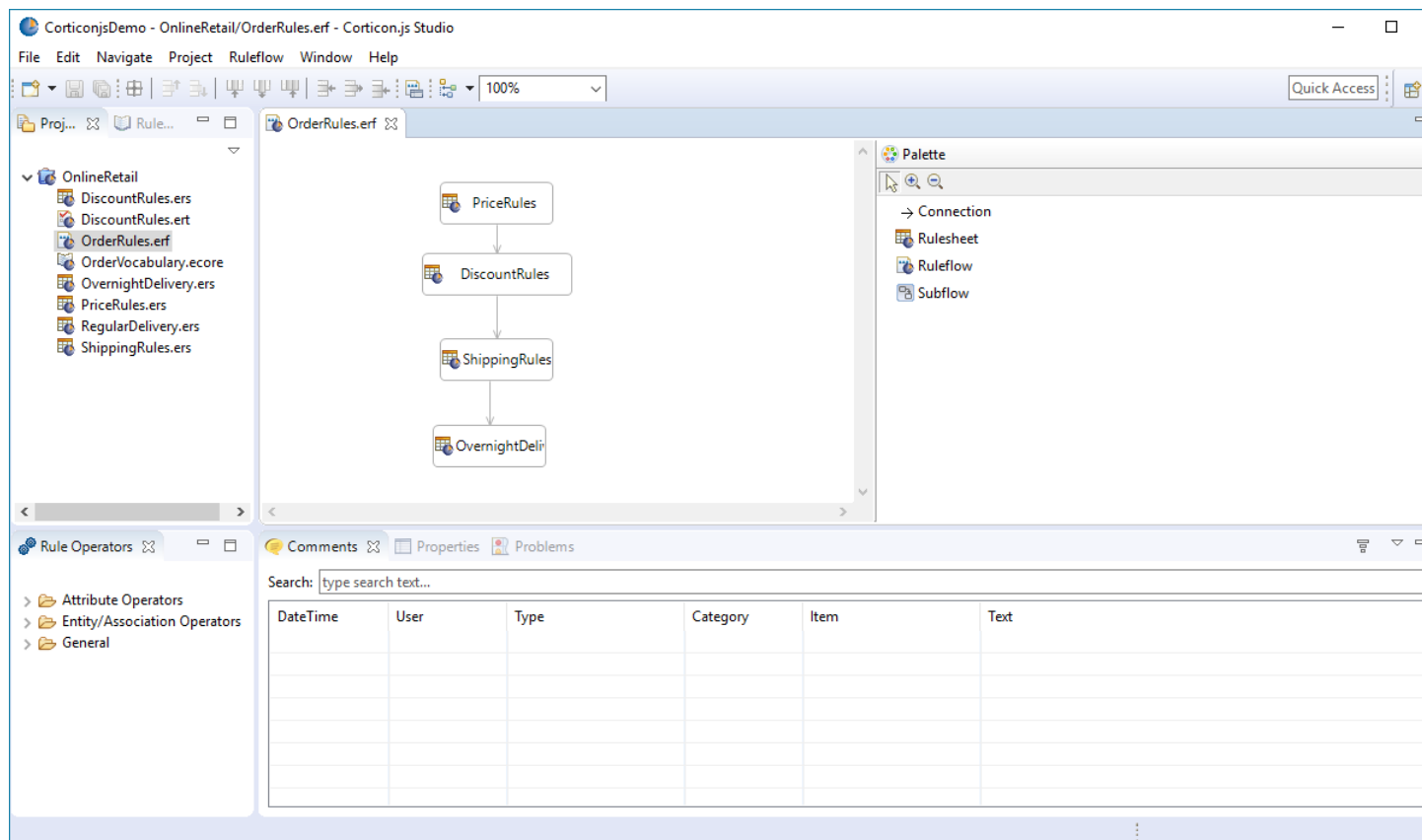


With your Rulesheets created, you are ready to test them. In Corticon.js Studio, you define Testsheets. Each Testsheet defines test input data and the expected output. Running Testsheets confirms that the expected results are produced and protects against regressions when future rule changes are made.

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.



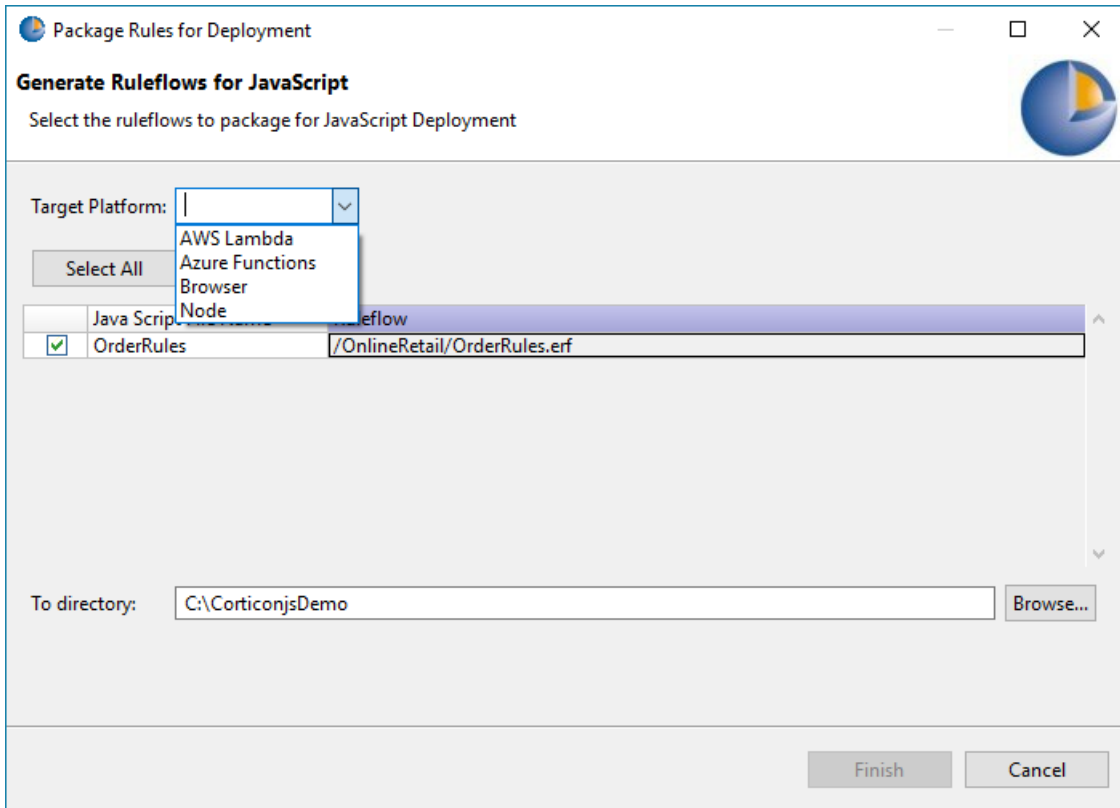
With the rule modeling complete, the next step is to package the rule for deployment on the JavaScript platform of your choice. For more information on rule modeling, see [Introduction to Corticon.js rule modeling](#).

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.

Package rules

1. In Corticon.js Studio, select **Project > Package Rules for Deployment**.
2. In the **Package Rules for Deployment** dialog box, select the Target Platform and Ruleflows for your application.



3. Click **Finish** to generate the packaged JavaScript.

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 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 = 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.

Set preferred language for Corticon.js Studio

Corticon.js Studio allows you to set your preferred language for Studio.

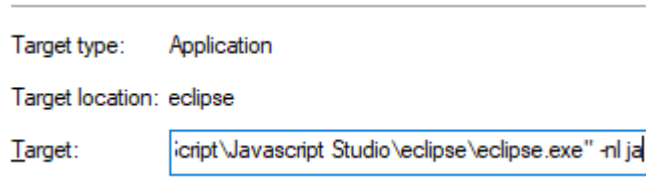
Use a different language

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:



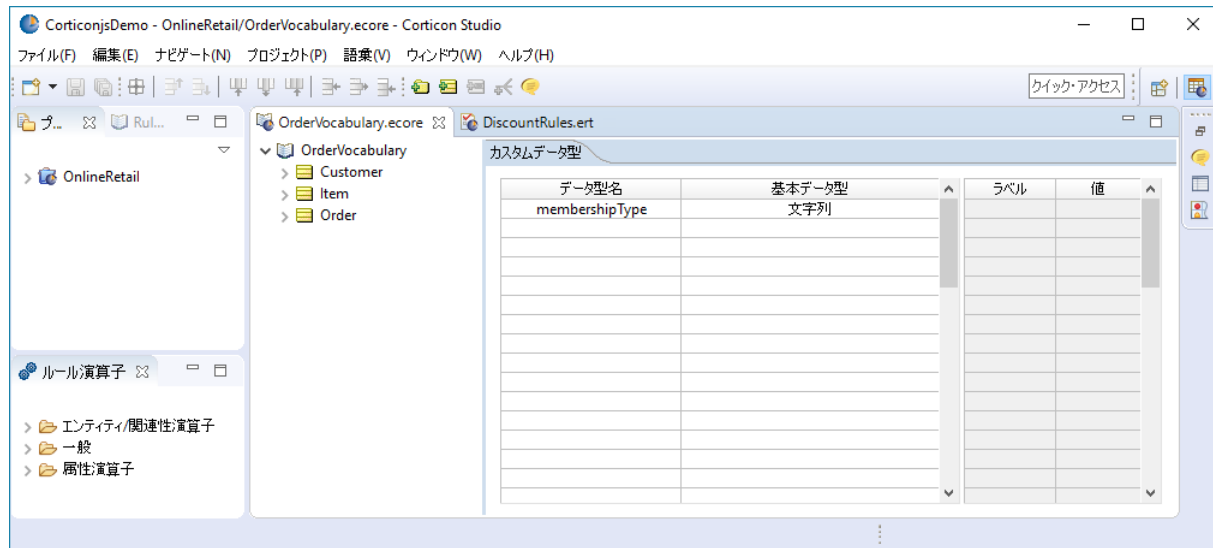
Target type: Application

Target location: eclipse

Target: "C:\Program Files\Javascript Studio\eclipse\ eclipse.exe" -nl ja

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

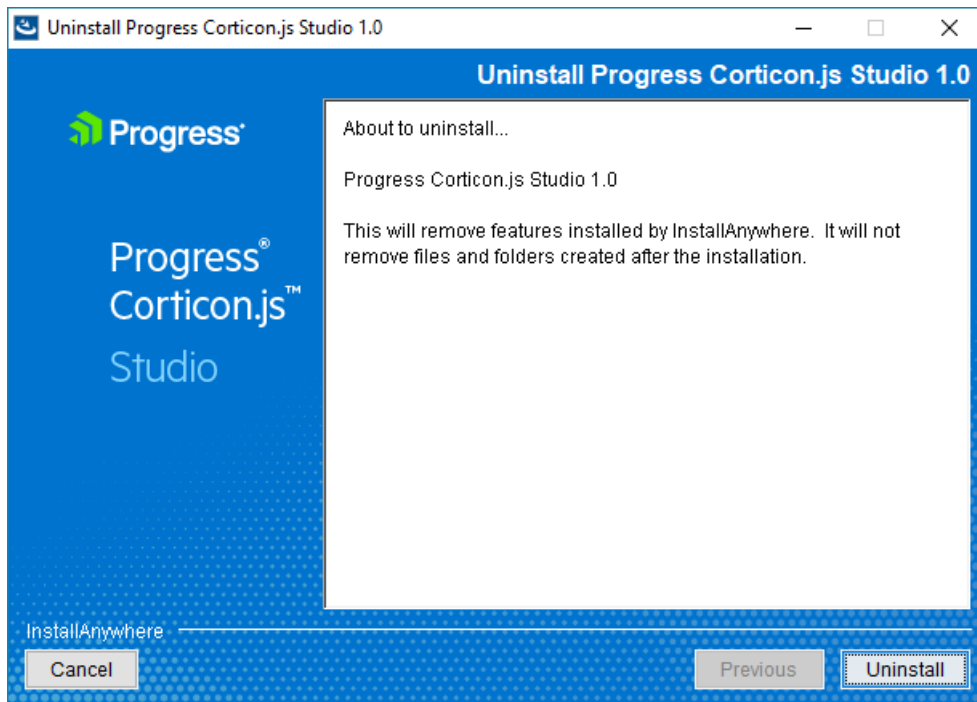
Progress Corticon.js: Get Started with Corticon.js: Version 1.0



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 **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.