



## What's new in Corticon.js



# Copyright

---

© 2021 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.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, 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, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, 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.js 1.3

**Updated:** 2021/11/08



---

# Learn about Corticon.js releases

---

## Corticon.js 1.3

Corticon.js 1.3 introduces several new features to accelerate development of your rule projects and to simplify the integration of deployed decisions with your applications. Specific features include:

- **Branching**—In a Ruleflow, you might have steps that should only process an entity with a specific attribute value. You can accomplish this by defining a branch within a Ruleflow. See *"Conditional Branching in Ruleflows" in the Corticon.js Rule Modeling Guide*.
- **Transient attributes**—Corticon.js supports attributes for in-process values created, derived, or assigned by rules in Studio. See *Build a Vocabulary by hand: "Step 6 Modeling the Vocabulary in Corticon.js Studio" in the Corticon.js Rule Modeling Guide*.
- **Clone**—Corticon.js supports a `clone` operator so you can create a new instance of `<Entity>` with the same attributes and their respective values. See *"Clone" in the Corticon.js Rule Language Guide*.
- **New Unique**—Corticon.js support the `newUnique` operator so you can create a new instance of `<Entity>` only if the instance created is unique as defined by an optional `<Expression1`. See *"New unique" in the Corticon.js Rule Language Guide*.
- **License enforcement**—When evaluating Corticon.js, every Decision Service that you package has an embedded expiration date based on your evaluation license.
- **Kinvey**—Progress Kinvey is now a supported platform. See *"Kinvey platform" in the Corticon.js Integration Guide*.
- **Use JSON import on different domains**—The Vocabulary generation function now lets you populate additional domains through JSON import. See the section "Integrating multiple sources into a Vocabulary" in the topic *"Use JSON to generate a Vocabulary" in the Corticon.js Rule Modeling Guide*.
- **Custom data access operators for SET**—The companion to the GET operators let you set a value for the specified data type. See *"Customized data access operators" in the Corticon.js Integration Guide*.

## Corticon.js 1.2

Corticon.js 1.2 introduces several new features to accelerate development of your rule projects and to simplify the integration of deployed decisions with your applications. Specific features include:

- **Vocabulary generation**—Every rule project starts with a rule vocabulary. New in 1.2 is the ability to generate your vocabulary from examples of the JSON your rules will process when deployed. This makes it easy to create your vocabulary and accelerates your rule project development. It also helps ensure that your vocabulary is in sync with the payloads that will be processed when deployed, helping to avoid potential problems. You can even generate your rule vocabulary from a JSON schema file. See *"Generate a Vocabulary" in the Corticon.js Rule Modeling Guide*.
- **Simplified integration**—With 1.2 you no longer need to annotate your JSON payload with Corticon metadata tags identifying the entities within the payload. Corticon.js 1.2 now processes JSON payloads in the native format used by other components of your application. This allows you to integrate your decisions without the need to add special code to format JSON before passing it to your decisions. This accelerates project development and opens new opportunities for decision integration. See *"JSON payloads and results in Corticon.js" in the Corticon.js Integration Guide*. Related changes include:
  - **Modified ruletest import/export behavior**—The ruletest import and export data features have been modified to support import and export of JSON without metadata. See *"How to import a JSON document to a testsheet"* and *"How to export a testsheet as a JSON document"* in the Corticon.js Quick Reference Guide.
  - **Simplified result payload**—The changes to eliminate metadata tags are reflected in the result, or output, payload of your decisions.
  - **New deployment configuration option**—A new config option has been added to the Corticon.js integration API to configure aspects of the result payload. See *"Config options" in the Corticon.js Integration Guide*.
- **Custom data access operators**—It is common for rules to need access application data such as session data in a browser or AWS Lambda function. New in 1.2 is a set of extensible data access operators where you provide the JavaScript implementation of the function. This feature allows you to extend the capabilities of Corticon.js and more tightly integrate your decisions with your application. See *"Customized data access operators" in the Corticon.js Integration Guide*.
- **CI/CD support**—Corticon.js 1.2 enables CI/CD best practices by adding the ability to script the packaging and testing of your decisions. New in 1.2 are scriptable utilities to package your rules for deployment to your platform of choice, automate the execution of your ruletests, and even generate rule reports. You can now integrate deployment of your JavaScript decisions with your existing CI/CD practices. See *"How to use the Corticon.js utilities" in the Corticon.js Integration Guide*.
- **Google Cloud Functions**—Serverless function platform support is expanded to include Google Cloud Functions. This adds to the existing AWS Lambda and Azure Functions serverless functions support, providing built-in serverless function support for the major cloud platforms. See *"Google Cloud Functions platform" in the Corticon.js Integration Guide*.
- **Expanded rule language capabilities**—The depth of the Corticon.js rule language was expanded in multiple ways to provide even greater flexibility for rule development. These enhancements include:
  - **Enhanced rulesheet filters**—Rulesheet filters were enhanced to support *limiting filters* — filters that apply to elements in a collection that will keep matching entities even if none of the children survive the filter. See *"Limiting filters" in the JS Rule Modeling Guide*.
  - **Rulesheet preconditions**—Precondition behavior of a filter ensures that execution of a rulesheet stops unless at least one piece of data survives the filter. See *"What is a precondition" in the JS Rule Modeling Guide*.
  - **Collection operators in filters**—Rulesheet collection operators can now be used in the filters for a rulesheet. See *"How to use collection operators in a filter" in the JS Rule Modeling Guide*.

- 
- **New "in" operator for membership tests**—The "in" operator can be used in Rulesheets to test whether a value is in a list of values. See *"In LIST" in the JS Rule Language Guide*.
  - **New DateTime operators**—New operators were added to allow rules to compare just the date or time portion of DateTime attribute values. See *"DateTime" in the JS Language Guide*
  - **Multiple decision in a single browser app**—It is now easier to deploy multiple decisions in a single browser app. See *"Browser platform" in the JS Integration Guide*.

## Corticon.js 1.1

Corticon.js 1.1 added the following features:

- **Rule Statements**—Document why a decision was made. Rule Modelers can attach Rule Statements to any rule to document the rule was fired. For example, you might use rule statements to communicate the reason for approval or denial for a loan application. Rule Statements can be used to create an audit trail of the rules that fired when processing a request. They can also be used to troubleshoot rule behavior.
- **Collection operators**—Efficiently process collections of associated entities in your rules when you use `forAll`, `exists`, `size`, `empty`, and `notEmpty`. For example, you could use the `exists` operator to filter the data being processed to just those customers being processed with a prior purchase greater than \$100. The new collection operators expand the set of complex rule problems which can be efficiently addressed in Corticon.js.
- **Export JSON to Clipboard**—Use the option in Corticon.js Studio tester that makes it easy to take the input data of any of your ruletests and use it to test your deployed Corticon.js rules.

## Corticon.js 1.0

Corticon.js 1.0 introduced a way to define rules and package them into fully, self-contained JavaScript bundles that can be deployed to any compatible JavaScript platform.

For more information, see: *What is Corticon.js?*

