Progress®
Corticon®

What's new in Corticon.js

Copyright

Last updated with new content: Corticon.js 2.3

Updated: 2025/04/23

1

Learn about Corticon.js releases

Corticon.js 2.3

In the Corticon.js 2.3 release:

- **Corticon.js AI Assistant improvements** include:
  - AI Assistant has access to all constructs in a ruleflow including branches and service callouts. This allows you to use AI to explore any aspect of a ruleflow.
  - AI Assistant can be used on rule assets containing errors. This allows you to use AI to explore rule assets even if they have unresolved errors.

  See *"Corticon.js AI Assistant" in the Corticon.js Quick Reference Guide* for details.

  Note: AI Assistant is visible by default in all new project workspaces. If you do not use the AI Assistant, you can close it.

- **New Utility for Custom Reporting**—The `corticonJS` command now allows you to create a JSON representation of the contents of any ruleflow. The utility is useful for customer reporting solutions capturing the specifics of a decision service to be deployed or other use cases where a simplified format detailing the contents of a decisionService is needed. See the Integration Guide topic *"How to use the Corticon.js utilities"* for details.
- **Improved native JSON Support in Tester**—The tester now supports importing native JSON containing arrays of scalar values into the Input. This enables customers to work seamlessly with JSON test data as supported by the runtime.

Corticon.js 2.2

New in Corticon.js 2.2, the Corticon AI Assistant enables the use of artificial intelligence (AI) in Corticon.js Studio to explore assets in your Corticon.js rule projects by asking natural language questions and viewing the AI-generated responses. See *"Corticon AI Assistant" in the Corticon.js Quick Reference Guide* for details.

Corticon.js 2.1.1

Corticon.js 2.1.1 contains new Service Callout API's that make it easy for SCO's to add JSON data retrieved from an external service to the working memory. See *"Use APIs to have the payload update entities and associations" in the Corticon.js Extensions guide* for details on how to use it. You can also easily retrieve a segment of the payload in JSON format to be passed to an external service to be used or persisted. See *"Use APIs to access entities and associations" in the Corticon.js Extensions guide* for details.

Corticon.js 2.1

Corticon.js 2.1 contains the following new features:

- **MarkLogic integration**—Generate decision services compatible with MarkLogic. See the blog "No-Code Business Logic Development for MarkLogic Database"
- **Support for variable number of parameters and mixed types**—Invocation can be done with any number of parameters and the parameters can be of any type. See *"How to use multiple variables in Get and Set operators" in the Integration guide*.
- **Support for JSON arrays**— See *"How to use multiple variables in Get and Set operators" in the Integration guide*.
- **Report on complexity metrics for Vocabularies and Ruleflows**—

  Generate a `csv` file with metrics that can be analyzed in Excel with values that include:

  - For vocabularies, the number of entities, attributes, associations, and more.
  - For ruleflows, the number of rules, conditions, actions, filters, rule messages, and the number of nodes, branches, callouts, and more.

  The reports contains only the count of different elements, with no other details of the vocabulary or rules so that reports can be shared with Progress without concerns about revealing confidential information.

- **License changes**—While running the installer, if you chose to provide no license or an invalid one, you will get an alert when attempting to start which provides a way to specify a valid license. You are completely shut out of Studio until you have a license.

Corticon.js 2.0

Corticon.js 2.0 introduced several new features to accelerate development of your rule projects and to simplify the integration of deployed decisions with your applications. Specific features include:

- **Service callouts and Extensions**—Enrich your JSON payload from external data. The feature is a general mechanism to execute custom JavaScript code that can change the payload by adding, updating, or deleting from it by doing various computations and data manipulations, or by accessing external data within the context of rules. You can define static runtime properties to your service callouts that will be passed in as a JSON object to the service callout. For more information, see the *Corticon.js Extensions Guide*.
- **Support for Date data type**—The Corticon.js Vocabulary supports the Date data type as well as the DateTime operator `toDate`. Date can be specified either in ISO 8601 date format, or as a long value that represents the milliseconds since the epoch. The only Date operators that are not available as compared to the Java product's Date data types are `nextDay`, and `toString`.

Note: The date/time/and dateTime formats in the Java product and the JavaScript product have the same results in different formats.

Figure 1: JavaScript

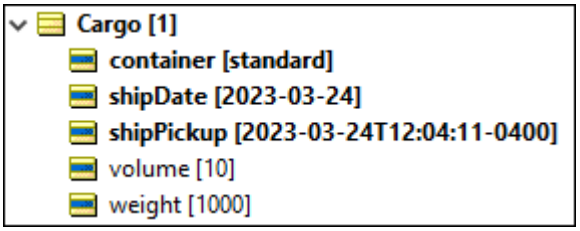
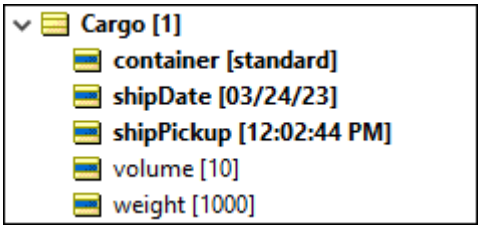
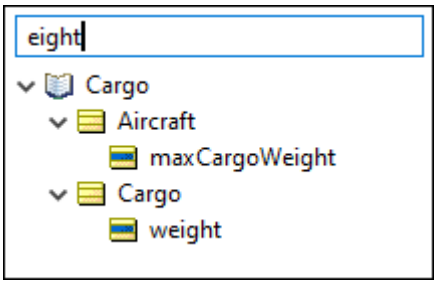


Figure 2: Java



- **Stronger licensing enforcement**—When your Corticon.js license expires, you are alerted at startup. Acknowledgement of the alert will exit the startup. There is no "read-only" mode. Contact your Progress Corticon representative to obtain an updated license file.
- **Filter Vocabulary**—Delimit the listed items to the entities where attributes and associations match the case-sensitive text, as shown in Cargo:



  You can use regular expressions to broaden your filter expression.
- **Expand and collapse Vocabulary views**—You can extend the Vocabulary view to show all its Entities, their Attributes, and their Associations. You can also drop the Vocabulary view down to just listing Entities.
- **Filter rule trace results, and export the results**—The Ruletest feature that enables Rule Trace to expose the sequence of actions in a sortable panel, now allows filtering of displayed data as well as exporting results to a CSV file.
- **Copy input from Ruletests and paste into another testsheet in the same project**—You can copy portions of the input column in a Testsheet, and then paste into a new Testsheet in the Ruletest.

Corticon.js 1.4

Corticon.js 1.4 introduced several new features to accelerate development of your rule projects and to simplify the integration of deployed decisions with your applications. Specific features include:

- **Rule trace viewer**—Troubleshooting the execution of rules just got easier. When running Ruletests you now have the option to gather rule trace data. Trace data identifies the sequence of rules triggered and the

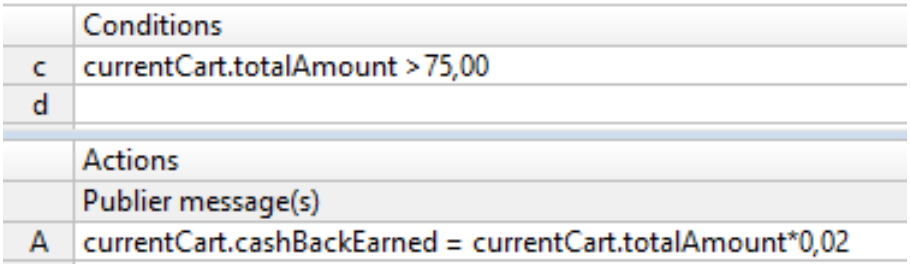
actions performed. When execution of your Ruletest completes, trace data is displayed in the Rule Trace Viewer where you can see each rule triggered, changes to attribute values and associations, and more. From the Rule Trace Viewer you can easily navigate to the Rulesheet for any rule show in the Rule Trace Viewer. This makes it easy for you to quickly make changes to your rules and see the results in your Ruletests. You no longer need to add rule messages on every rule to trace rule execution. See *"Trace rule execution" in the Corticon.js Rule Modeling Guide*.

Note: See how the rule trace view helped in a large project in the blog Fast Rules Diagnostics and Root Cause Analysis with the New Rule Trace Viewer.

- New operators in Corticon.js:
  - **toDateTime and toDecimal**—Corticon.js supports the `toDateTime` and `toDecimal` operators so you can convert String attribute values into valid `DateTime` or `Decimal` values.

    For more information, see "To dateTime" and "To decimal" in the Corticon.js Language Guide.
  - **sortedBy**—Corticon.js supports the `sortedBy` and `sortedByDesc` operators so you can sequence a collection based on an attribute which can then be selected by a sequence operator—`first`, `last`, or `at`.

    For more information, see "Sorted by" and "Sorted by descending" in the Corticon.js Language Guide.
- **Testsheet imports from clipboard**—You can import well-formed JSON test values from the clipboard. Right-click in the Testsheet's Input column or Expected column to choose Import from clipboard and replace the column's current content.
- **Node.js certifications**—Node.js 14 and 16 are certified for use in your deployments.
- **Locale awareness**—Corticon.js Studio now allows decimal values to be entered in Rulesheets and Ruletests using the decimal separator for your locale, as illustrated:



Note: The Corticon.js runtime requires that decimal values in payloads use the period character as the decimal separator. The locale handling for decimal separators is limited to editing Studio rulesheets and ruletests.

```
"ShoppingCart": {
    "totalAmount": 68.98,
    "Item": [
        {
            "price": 66.99,
```

- **Performance improvements**—Several operators and tester execution have improved performance.

- **Kinvey dropped**—Support for Progress Kinvey as a target platform for Corticon.js deployment has been dropped.

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 was introduced as a supported platform.
- **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 services in a single browser app**—It is now easier to deploy multiple decision services 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`, `isEmpty`, 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?*