



Corticon 6.3 Release Enablement

Corticon 6.3 Release Enablement



Corticon 6.3.4 rolls up the various patch releases for 6.3.3 as well as:

- **Updated Web Console**—The Web Console for 6.3.4 introduces the OneWebConsole, on par with the 7.1 Web Console, hardened against many CVEs, and offering new features for deployment visibility and management. The updated Web Console can manage deployed servers and their decision services from version 6.3 and version 7.1. Corticon 6.3.4 and later bundle the Web Console from Corticon 7.1. To keep documentation streamlined, all Web Console documentation lives in the 7.1 library—no separate “6.3.4” Web Console guide is available. The benefits are:

- Updated UI with improved behavior and security
- Ability to manage all 6.3 and later Corticon Servers from one console. Wherever you see “Web Console” in any 7.1 topic, it applies equally to your 6.3 environment.

For Corticon 7.1 Web Console documentation, see [the 7.1 archive](#).

For Corticon 6.3.3 Web Console documentation, see [the 6.3 archive item for 6.3.3](#)

- **Log filter changes**—A new filter is available, PAYLOAD, that will put the request payload and the response payload in the log. The filter INVOCATION has been dropped.
- **New server property: Turn off disabling of thread underutilization allocation** -- A new server property has been added to control how execution threads are allocated when the Corticon Server is underutilized:

```
com.corticon.ccsrver.allocation.disable.underutilization.algorithm  
=false
```

See *"Server Execution Properties" in the Server Guide* for details.

Corticon 6.3.3 provides additional functionality to 6.3 with:

- [Corticon Server now enforces version check on EDS deployment](#) on page 6
- [Swagger is optional](#) on page 7
- [Tomcat 8.5 is not supported](#) on page 7

Corticon 6.3.2 provides additional functionality to 6.3 with:

- Updates the Corticon Web Console's LDAP authentication to support OpenLDAP
- Improved runtime performance of rulesheets containing filters that use entities not also used in a rule condition or action.
- Improved decimal precision comparison in rule tests.
- Logging configurations in Web Console for trace logging are disabled.

Corticon 6.3.1 provides additional functionality to 6.3 with:

- [ADC Insert Primary Key handling](#) on page 7

Corticon 6.3.0 delivers updates and enhancements in the following product areas:

- [Rule trace viewer](#) on page 8
- [OAuth2 and POST support in REST connection driver](#) on page 9
- [Server registration updates in Web Console](#) on page 9
- [Simplified JSON metadata](#) on page 9
- [Deploy Corticon on Docker](#) on page 11

For details, see the following topics:

- [Corticon Server now enforces version check on EDS deployment](#)
- [Swagger is optional](#)
- [Tomcat 8.5 is not supported](#)
- [ADC Insert Primary Key handling](#)
- [Rule trace viewer](#)
- [OAuth2 and POST support in REST connection driver](#)
- [Server registration updates in Web Console](#)
- [Simplified JSON metadata](#)
- [Deploy Corticon on Docker](#)

Corticon Server now enforces version check on EDS deployment

When an EDS is deployed, Corticon Server will inspect the EDS to determine which version of Corticon it was generated with. If this version is not the same major.minor version of Corticon Server, the EDS will not be deployed. Examples:

- - 6.3.0.0 EDS deployed to 6.3.3.0 Corticon Server: Allowed
- - 6.1.0.0 EDS deployed to 6.3.3.0 Corticon Server: Disallowed

Enforcing this version check ensures compatibility between the EDS and Corticon Server. The generated EDS code is dependent on internal Corticon Server APIs which may change across major.minor releases. This could result in an EDS failing to execute despite initially reporting successful deployment.

Swagger is optional

Corticon Server no longer bundles Swagger for exploring and testing the Corticon Server REST API. Swagger is now optional and can be added to an instance of Corticon Server. See *"The REST API Swagger documentation" in the Server Guide* for details. The use of Swagger is best limited to development deployments of Corticon Server. It is typically not needed, or wanted, in production deployments.

Tomcat 8.5 is not supported

Apache Tomcat 8.5.x will reach [end of life](#) in March 2024. Customers are encouraged to update to Tomcat 9.0 to ensure access to Tomcat security updates. Corticon Server bundles Tomcat 9.0.

ADC Insert Primary Key handling

When the primary key of an inserted record is generated by the connected database, Corticon retrieves this generated value and adds it to working memory for that Entity. This will allow follow-up database updates on that Entity to occur, and also allows associated Entities that are dependent on that primary key value to be stored as a foreign key value in the associated Entity.

For more information about ADC writes, see *"How to configure ADC writes" in the Corticon Data Integration Guide*.

How to: Use Rule Trace Viewer

The viewer is a simple option for running any Ruletest:

- In a Ruletest window, choose **Ruletest > Testsheet > Run Test with Rule Trace**.
- On the **Rule Trace** tab, click any line to see its highlighted element in the Ruletest.
- Double-click on any line to open the Rulesheet with the Action line and Rule column highlighted.
- Click any column header to sort the data in that column.
- Change any rule, and then rerun the test to see the changes immediately.

Troubleshooting

- If the results are overwhelming, try changing the test subject to just one Rulesheet or disabling some rules.
- The Rule Trace Viewer is based on JSON. If you have the Studio property `com.corticon.testers.ccservice.execute.format` set to XML (instead of the default, JSON), the Rule Trace Viewer function is inoperative.

Rule trace viewer

You can now get a dynamic view of all the actions and rules in a Ruletest by simply running the test with rule trace. It reveals every rule that fired in a way that lets you sort, highlight, and locate every action and rule. You can even make changes to Rulesheets and then rerun the tests. Here is an example based on the `checkout` testsheet in the

Rule Statements Rule Messages Rule Trace						
Sequence	Action	Element	Old Value	New Value	Assoc...	Location
1	Update Attribute	Item (Item) [3]/department		285		checks : A0
2	Update Attribute	Item (Item) [2]/department		300		checks : A0
3	Update Attribute	Item (Item) [1]/department		280		checks : A0
4	Update Attribute	ShoppingCart (ShoppingCart) [1]/totalAmount		82.490000		checks : D0
5	Update Attribute	Customer [1]/isPreferredMember		true		:
6	Update Attribute	ShoppingCart (ShoppingCart) [1]/cashBackEarned		1.649800		coupons : A0
7	Add Entity	Coupon [1]				coupons : 3
8	Update Attribute	Coupon [1]/expirationDate		05/07/22		coupons : 3
9	Update Attribute	Coupon [1]/description		10% off next gas ...		coupons : 3
10	Update Attribute	preferredCard (PreferredAccount) [1]/cumulativeCashBack	9.240000	10.889800		coupons : B0
11	Update Attribute	ShoppingCart (ShoppingCart) [1]/totalAmount	82.490000	71.600200		use_cashBack : 1
12	Update Attribute	ShoppingCart (ShoppingCart) [1]/savings		10.889800		use_cashBack : 1
13	Update Attribute	preferredCard (PreferredAccount) [1]/cumulativeCashBack	10.889800	0.000000		use_cashBack : 1

The results of a rule trace are dynamic:

- **Highlight**—Click anywhere on a line to highlight that element in the Testsheet output. Click on any item in the Ruletest to see all the rules related to that element highlighted in the Rule Trace Viewer.
- **Locate**—Double-click on any line to open the related Rulesheet positioned at the Action line and rule. The Rulesheet is in editable form so you can quickly make changes and rerun to see the change effect.
- **Sort**—Click any column header on the **Rule Trace** tab to sort the tab content in ascending order. Click again to sort into descending order.

This dynamic feature is best presented in this video:

<https://youtu.be/H7S8U0gll6k>

For more information, see *"Trace rule execution" in the Corticon Rule Modeling Guide*. The topic adds this feature to the Rule message metadata technique, which should now be obsoleted for that troubleshooting technique.

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](#).

How to: Use Rule Trace Viewer

The viewer is a simple option for running any Ruletest:

- In a Ruletest window, choose **Ruletest > Testsheet > Run Test with Rule Trace**.
- On the **Rule Trace** tab, click any line to see its highlighted element in the Ruletest.
- Double-click on any line to open the Rulesheet with the Action line and Rule column highlighted.
- Click any column header to sort the data in that column.
- Change any rule, and then rerun the test to see the changes immediately.

Troubleshooting

- If the results are overwhelming, try changing the test subject to just one Rulesheet or disabling some rules.
- The Rule Trace Viewer is based on JSON. If you have the Studio property `com.corticon.tester.ccserver.execute.format` set to XML (instead of the default, JSON), the Rule Trace Viewer function is inoperative.

OAuth2 and POST support in REST connection driver

The REST Connector driver has been updated and now supports OAuth2 Authentication and POST requests:

- **REST data source support of OAUTH2**—Uses authorization tokens to prove an identity without giving away your password. See *"Authentication on REST Service connections" in the Corticon Data Integration Guide*.
- **REST data source support of POST**—A **Post** request does not include parameters as part of the URL, instead the parameters are data in the request document. See *"Parameters on REST Service connections" in the Corticon Data Integration Guide*.

Troubleshooting

Selecting the **Post** parameter type changes how REST connector makes its requests to the endpoint, so any specified URL parameters may be ignored.

Request formats add **Post** parameters on the request. Post parameters are sent as name/value pairs in JSON format in the request body.

Server registration updates in Web Console

When a Corticon Server self-registers with the Web Console, the Web Console now updates any existing registration for the serve by updating the IP address of the registered server. Previously, a new registration would be added resulting in the Web Console having server registrations for both the old and new IP addresses. Updating an existing registration better supports deployments where the IP address of a Corticon Server might change on each restart. The Web Console can then provide continuity of trend data across restarts.

Simplified JSON metadata

Some users find that their JSON requests have metadata only at the root, expecting that the decision service can infer the metadata for subordinate levels. This tactic is now supported, although the output provides the metadata at all levels.

For more information, see "Simplified JSON in requests" in the Web Console Guide.

How to: Use Simplified JSON

- Add the decision service name as `name` at the top.
- Delete or leave out the metadata for related entities.
- Retain the metadata for the primary entity.

For example:

coupons.json	coupons_Simplified.json
1 {	1 {
2 "__metadataRoot": {"#locale": ""},	2 "__metadataRoot": {"#locale": ""},
3 "Objects": [{	3 "name" : "MyAdvancedTutorial",
4 "preferredCard": {	4 "Objects": [{
5 "cumulativeCashBack": 9.24,	5 "preferredCard": {
6 "__metadata": {	6 "cumulativeCashBack": 9.24,
7 "#type": "PreferredAccount",	7 "cardNumber": "12"
8 "#id": "PreferredAccount_id_1"	8 },
9 },	9 "ShoppingCart": {
10 "cardNumber": "12"	10 "totalAmount": null,
11 },	11 "Item": [
12 "ShoppingCart": {	12 {
13 "totalAmount": null,	13 "price": 25,
14 "Item": [14 "name": "Miller Beer",
15 {	15 "department": null,
16 "price": 25,	16 "barCode": "39-291-1234"
17 "name": "Miller Beer",	17 },
18 "__metadata": {	18 {
19 "#type": "Item",	19 "price": 1.99,
20 "#id": "Item_id_1"	20 "name": "RedBull",
21 },	21 "department": null,
22 "department": null,	22 "barCode": "39-285-98765"
23 "barCode": "39-291-1234"	23 }
24 },	24],
25 {	25 "useCashBack": true,
26 "price": 1.99,	26 "savings": null
27 "name": "RedBull",	27 },
28 "__metadata": {	28 "__metadata": {
29 "#type": "Item",	29 "#type": "Customer",
30 "#id": "Item_id_2"	30 "#id": "Customer_id_1"
31 },	31 },
32 "department": null,	32 "Name": ""
33 "barCode": "39-285-98765"	33 }]
34 }	34 }
35 },	
36 "useCashBack": true,	
37 "savings": null,	
38 "__metadata": {	
39 "#type": "ShoppingCart",	
40 "#id": "ShoppingCart_id_1"	
41 }	
42 },	
43 "__metadata": {	
44 "#type": "Customer",	
45 "#id": "Customer_id_1"	
46 },	
47 "Name": ""	
48 }]	
49 }	

Troubleshooting

- In the example, Name at the bottom is an attribute of the root entity.
- Mind your commas, braces, and brackets if you are paring down a known-good request.

Deploy Corticon on Docker

Docker is a popular platform for building, sharing and running applications. Since the Docker container packages applications with all their libraries and dependencies, the application can run on any Docker host in any supported environment. Among the benefits of this strategy are:

- Docker containers are faster to create and quicker to start.
- Docker scales efficiently when demand for your application grows.
- Corticon Server deployed to Docker as a web application can be accessed as a REST or a SOAP service.

The following video takes you through various configurations of Docker:

https://youtu.be/TpTILR-8x_w

For more information, see "How to deploy Corticon on Docker" in the Corticon Deployment Guide.

