RDFox v7.5 Migration Guide

This guide describes functional changes in RDFox v7.5 that might prevent scripts, configuration, or code that work with RDFox v7.4, from being used directly with the newer version. Each change is described in its own section with clear actions to take as part of the upgrade. Users should read this guide in conjunction with the release notes for v7.5.

RDFox v7.5 is persistence-compatible with RDFox v7.4, so it is not necessary to upgrade the server directory or transcribe the server content before restarting existing servers with the new version.

1. Changes affecting servers

1.1. Server parameter `allowed-schemes-on-load` has been replaced by `allowed-url-schemes-on-import`

In previous releases, the server parameter `allowed-schemes-on-load` determined which URL schemes could be used in SPARQL `LOAD` updates. From v7.5 onwards, the set of allowed URL schemes is instead controlled by the `allowed-url-schemes-on-import` server parameter, which applies to *all* import operations.

For example, with the default value for the old parameter (`"https rdfox"`), previous releases would have blocked the following SPARQL LOAD request because the http scheme is not amongst the allowed values:

```
LOAD <http://localhost:8080/my-turtle-file.ttl>
```

but they would have allowed the following shell command:

```
import <http://localhost:8080/my-turtle-file.ttl>
```

In RDFox v7.5 and onwards, both of the above operations will be blocked with the default value of `allowed-url-schemes-on-import` (also `"https rdfox"`).

To give users time to respond to this change, RDFox v7.5 will still accept the old parameter name as an alias of the new parameter, but this will be removed at a later date, so users are advised to take the actions listed below to upgrade now.

Required actions

- Search all RDFox-related configuration, scripts, or code for the string `allowed-schemes-on-load` in and replace it with `allowed-url-schemes-on-import`.
- Ensure that the value set for the new parameter covers the URL schemes used for all imports the RDFox server should allow.

1.2. Extended Server Information has been restructured/reorganized.

Extended diagnostic information on the state of a server and many of its constituent parts can be obtained with the `serverinfo extended` command and equivalent APIs. RDFox v7.5 introduces new response formats for returning this information in REST API calls, making it easier for clients to parse and utilize this data for debugging and monitoring.

To ensure all formats provide consistent information, there has been some renaming and restructuring of data within the component info responses of the existing APIs e.g. Components report their memory consumption using the key “Memory consumed (bytes)” rather than simply “Size”, All the Data stores are collected under a single “DataStores” component rather than directly connecting to the “Server” Component etc.

Required actions

- If your monitoring setup for RDFox relies on obtaining extended diagnostic information in the previous response format (e.g., specific property or component names), you will need to update your code to accommodate the new structure. Failure to do so may result in errors or missing data.

2. Changes affecting the REST API

2.1. Data store transactions no longer have their own URL

Each data store connection in RDFox can support one transaction at a time. In RDFox v7.4 and earlier, the transaction belonging to a connection was represented as a separate resource in the REST API with a path like `/datastores/<DSTRNAME>/connections/<DSCONN>/transaction`.

From v7.5 onwards, the functionality previously exposed via the `transaction` resource has been moved to the connection resource that contains it. Calling the API with paths ending `…/transaction` is now deprecated but still recognized by RDFox v7.5. This backwards compatibility will be removed entirely in a future release, so clients are encouraged to adopt the new resource path now.

Required actions

- Identify all code and scripts that build or use URIs which match the pattern `/datastores/<DSTRNAME>/connections/<DSCONN>/transaction` and delete the final segment (`/transaction`) from each match.

3. Changes affecting the Endpoint

3.1. The default fields reported by the RDFox HTTP endpoint `elf` logger have changed

RDFox can be configured to use various endpoint logging formats. If the extended log format (`elf`) request logger is used, it can further be configured using the `elf-logger-fields` parameter, which allows the set of reported fields to be specified.

In RDFox v7.4 and earlier, the default value of the `elf-logger-fields` parameter was:

```
date time cs-method cs-uri-stem sc-status sc-bytes time-taken
```

In RDFox v7.5, the default value of the `elf-logger-fields` parameter is:

```
x-timestamp c-ip cs-method cs-uri-stem cs(RDFox-Request-ID) sc-status sc-bytes time-taken x-protocol-error
```

The `x-timestamp` field replaces the `date` and `time` fields in the default field list, providing further information (the time zone offset) and simpler parsing for log management systems e.g. OpenTelemetry. However, the `date` and `time` fields are still available and can be manually selected by setting the `elf-logger-fields` parameter. Pre-existing `c-ip`, `sc(RDFox-Request-ID)` and `x-protocol-error` options have also been added to the default list based on feedback from users.

Required actions

- If your RDFox instance is configured to use the `elf` request logger, either ensure the `elf-logger-fields` parameter is configured, or that any log management systems you use can tolerate this change (e.g. Note that the index of the `time-taken` field has changed).

4. Changes affecting the Java APIs

4.1. requiresIncrementalReasoning has been replaced by getDataStoreReasoningAlgorithm

The `requiresIncrementalReasoning` method of the `tech.oxfordsemantic.jrdfox.client.DataStoreConnection` interface, which returned a Boolean indicating whether the next update to a data store would use incremental reasoning, has been replaced by `getDataStoreReasoningAlgorithm`, which returns an Enum with values for from-scratch and incremental reasoning.

Required actions

- Search all RDFox-related Java code for the string "requiresIncrementalReasoning" to identify code locations that must be refactored.
- Replace code that used the return value of the former API without negation with:

  ```
  conn.getDataStoreReasoningAlgorithm() == ReasoningAlgorithm.INCREMENTAL
  ```

- Replace code that used the negated return value of the former API with:

  ```
  conn.getDataStoreReasoningAlgorithm() == ReasoningAlgorithm.MATERIALIZATION
  ```

4.2. Extended Server Information has been restructured.

In addition to the renaming/restructuring of the data returned for the server info APIs (see section 1.2), the programmatic type of the returned `ComponentInfo` has been updated to provide more information to the Java API.

Prior to RDFox v7.5, the Java `ComponentInfo` class exposed component information as a map from strings to Objects which was accessible via the `getPropertyValues()` function. This provided simple access to the data but made further analysis/aggregation difficult.

From RDFox v7.5 onwards, the Java `ComponentInfo` class will expose each datapoint of information as a `Property` object. These are uniquely identified by the combination of a string and `ComponentInfoMetric` Enum. The Enum allows measures of similar values to be programmatically collected/aggregated (e.g. Properties regarding RDFox's memory consumption all use Enum `ComponentInfoMetric.MEMORY_CONSUMED_IN_BYTES`, thus the total memory footprint of any component can be calculated by summing these values); while the string allows unique identification of a specific property within a component (e.g. the `Dictionary` component records multiple `ComponentInfoMetric.MEMORY_CONSUMED_IN_BYTES` metrics, individually recording the memory consumption of the dependency graph and bindings tables).

Required actions

- If your monitoring setup for RDFox uses the Java `ComponentInfo` class, then the application code must be updated to use the new `Property` APIs.

5. Changes affecting the C APIs

5.1. Extended Server Information has been restructured.

In addition to the renaming/restructuring of the data returned for the server info APIs (see section 1.2), the information contained within the returned `CComponentInfo` has been updated to provide more information to the C API.

Prior to RDFox v7.5, each property within a `CComponentInfo` struct was identified solely by a string identifier: the property "Name". This provided simple access to the data but made further analysis/aggregation difficult.

From RDFox v7.5 onwards, each property is identified by the combination of a string and `CComponentInfoMetric` Enum. The Enum allows measures of similar values to be programmatically collected/aggregated (e.g. Properties regarding RDFox's memory consumption all use Enum `CComponentInfoMetric.MEMORY_CONSUMED_IN_BYTES`, thus the total memory footprint of any component can be calculated by summing these values); while the string allows unique identification of a specific property within a component (e.g. the `Dictionary` component records multiple `CComponentInfoMetric.MEMORY_CONSUMED_IN_BYTES` metrics, individually recording the memory consumption of the dependency graph and bindings tables).

Required actions

- If your monitoring setup for RDFox uses the `CComponentInfo` struct, then the application code must be updated to use the new API definitions. Please contact OST support for more information.