RDFox v7.4 Migration Guide

This guide describes functional changes in RDFox v7.4 that might prevent scripts, configuration, or code that work with RDFox v7.3, 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 v7.4.

RDFox v7.4 is persistence-incompatible with RDFox v7.3, so it is necessary to upgrade the server directory or transcribe the server content before restarting existing servers with the new version. As always, before attempting any upgrade, ensure that you have an up-to-date backup of the complete server directory.

1. Changes affecting servers

1.1. The `channel` endpoint parameter must now be specified when initializing a server

Prior to RDFox v7.4, the default value for the `channel` endpoint parameter was `unsecure`. In the interests of providing secure defaults, the `init` mode of the RDFox executable now requires that this parameter be specified explicitly to ensure that operators are at least conscious of this choice.

Required actions

Ensure that all scripts or configuration that invoke the RDFox executable in `init` mode specify the `channel` endpoint parameter explicitly. Wherever possible, this should be set to `ssl` and a private key and certificate supplied via the `credentials` or `credentials-file` endpoint parameters.

1.2. Extended server information will omit information on data stores that can't be locked

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. Previously the underlying operation to obtain this information would wait, for the applicable lock timeout period, to obtain an exclusive lock on every data store in the server. If any of these waits timed out, the entire operation would return an error.

As of v7.4, data stores that cannot be locked immediately are skipped. The final server info output will contain a message showing that the data store is present but that it could not be locked.

No change has been made to the operation that retrieves extended diagnostic information for a single data store which previously and still waits to lock the data store for the applicable lock timeout and returns an error if the timeout is reached.

Required actions

It is unlikely that any action is needed in response to this change. If your monitoring setup for RDFox relies on obtaining extended diagnostic information for a data store but currently accesses this as part of extended server information, consider fetching the data store information only.

1.3. The default strategy for loading new snapshots in `file-sequence` persistence has changed

For a persisted data store, RDFox's compaction operation appends a new snapshot record to the log, with the exact same logical content as the pre-compacted data store. In a HA setup, new snapshots are created by one replica (whichever receives the compaction command) and then restored (loaded) by the other replicas.

In RDFox v7.3 and lower versions, the strategy for restoring new snapshots was as follows:

1. acquire an exclusive lock on the data store,
2. reinitialize the existing in-memory data store object to an empty state,
3. load the snapshot directly into the existing in-memory data store object,
4. release the lock.

This strategy has the benefit that it does not need to allocate any additional memory because the pre-compacted state is dropped before the compacted state is loaded. As a result, this strategy is referred to as

the *memory-efficient* strategy. The downside to the memory-efficient approach is that the data store is locked exclusively while the new snapshot is loaded from disk (step 3), making it unavailable even for reads. Depending on the size of the data, this could amount to a considerable period of being unavailable. This problem is compounded by the fact that all the non-compacting replicas in a cluster are likely to do this simultaneously.

RDFox v7.4 introduces two new strategies for loading snapshots in file-sequence persistence and adopts one these as its new default behavior.

The first new strategy is named *highly-available*. It works as follows:

1. load the new snapshot into a *new* data store object in memory,
2. acquire an exclusive lock on the data store,
3. update all internal references to the data store to point to the new data store object,
4. release the lock,
5. delete the old in-memory data store object.

This option should be used in deployments where availability is critical as it holds the data store lock only momentarily while references are updated to point to the new version. It has substantially higher memory requirements than the memory-efficient strategy as both the pre-compacted and compacted data store states are temporarily present in memory simultaneously. Operators using this strategy should ensure that all replicas in their HA cluster, when at a steady-state (no compaction operation is underway) have sufficient free memory to load a second copy of the largest data store in their server.

The second new strategy is called *adaptive*. It attempts the highly-available strategy, falling back to memory-efficient if an allocation fails. It is a good choice for environments where availability is important but can be sacrificed to avoid requiring too much memory.

The choice of strategy is controlled by the new `persistence.snapshot-restore-mode` server parameter, which accepts values `memory-efficient`, `highly-available`, and `adaptive`. The default value is `adaptive`.

Required actions

Operators of HA RDFox clusters (using `file-sequence` persistence) should review the options described above carefully and decide which is right for their deployment of RDFox. If it is important that only the `memory-efficient` or `highly-available` strategies should be used, update your RDFox command line or server parameters file to set the `persistence.snapshot-restore-mode` explicitly.

No action is required for the operators of non-HA RDFox instances (using `file` persistence or no persistence).

2. Changes affecting data stores

2.1. The data store parameter `init-resource-capacity` has been removed

The data store parameter `init-resource-capacity` which allowed users to give RDFox a hint about how much space should be allocated for storing resources at data store creation time, has been removed.

Required actions

Search all code, scripts, and configuration for references to the above parameter and remove any that are found.

2.2. Pre-v7.3 tuple table names are no longer recognized

RDFox v7.3 renamed for the `quad-table-lg` and `quad-table-sg` tuple table types to `quad-table-lg-fi` and `quad-table-sg-fi` respectively. For backwards compatibility, v7.3 accepted the older names however v7.4 no longer recognizes them.

This change was also described in section 2.1 of the RDFox v7.3 Migration Guide.

Required actions

Identify any scripts or code which create tuple tables of type `quad-table-lg` or `quad-table-sg`. And replace the current value with the new `quad-table-lg-fi` or `quad-table-sg-fi` type.

2.3. The output of the STR built-in function has changed for some inputs

The results of the STR built-in function on literals of type `xsd:decimal`, `xsd:float`, and `xsd:double` was changed so that the result clearly indicates what kind of literal the input was. Specifically, this function now returns the *canonical* lexical representation as per the XML Schema Datatypes v2 standard (https://www.w3.org/TR/xmlschema-2/). This also means that the literals of type mentioned above are serialized in all formats in the same way. Prior to this change, the Turtle form and the form in, say, JSON query output could in some cases be different. The following table illustrates some of the differences.

Literal	STR output in v7.3	STR output in v7.4
2 decimal	2	2.0
2 double	2	2.0E0
2 float	2	2.0E0

As one can see from this table, the output of STR in RDFox v7.4 distinguishes 2 decimal from 2 double (but not from 2 float because the canonical form of `xsd:float` is the same as for `xsd:double`).

Required actions

This change is unlikely to have any material effect in most deployments of RDFox. A situation where this change might be relevant is if STR is used to construct IRIs or unique IDs of resources in rules or in INSERT statements. For example, if the following rule is matched to the literals of type mentioned above, the derived

triples in RDFox v7.4 will be different from the triples derived by RDFox v7.3. This, in turn, might prevent incremental reasoning and can lead to inconsistencies in the data.

```
[?x, :p2, ?z] :- [?x, :p1, ?y], BIND(IRI(CONCAT("http:/test.com/", STR(?y)) AS ?z) .
```

Clients should thus review their usage of the `STR` function in their rules and `INSERT` statements and check whether the function is used to generate new IRIs or IDs in a way illustrated above. If so, one option is to drop all rules before upgrading to v7.4, perform the upgrade, and then reinsert such rules. Another option is to simply upgrade and then perform a "from scratch" materialization using RDFox's API.

2.4. The output of the `STREX` built-in function has changed for some inputs

The `STREX` function is a proprietary RDFox that is applicable to resources of all types (including blank nodes). This function was modified in the same way as the `STR` function (see Section 2.3). In addition, the function's output was also modified on literals with language tags. Namely, RDFox uses the `rdf:PlainLiteral` specification (https://www.w3.org/TR/rdf-plain-literal/) to unify the representation of all literals in the system. According to this specification, a Turtle literal of the form `"abc"@en` is actually just an abbreviation for a typed literal `"abc@en"^^rdf:PlainLiteral` – that is, the lexical form of the typed literal is the string `"abc@en"`. Now, in RDFox v7.3, STREX(`"abc"@en`) returns `"abc"`, which agrees with the definition of `STR` – that is, `STR("abc"@en)` also returns `"abc"`. However, this introduces a discrepancy: `STREX` returns in v7.3 the lexical form of the literal for all literals apart from literals of type `rdf:PlainLiteral`.

To unify the definition of `STREX`, RDFox v7.4 now returns the canonical lexical form of any literal without any exception. The `rdf:PlainLiteral` specification explicitly says that the lexical form of literal `"abc"@en` is `"abc@en"`, and so `STREX` returns the latter in v7.4. Note that the behavior of `STR` is unchanged in RDFox v7.4 – that is, `STR("abc"@en)` returns `"abc"` in both v7.3 and v7.4 since this is dictated by the SPARQL 1.1 standard.

Required actions

The possible effects of this change and any remedial actions are the same as for changes to the STR function described in Section 2.3.

2.5. The ranges for Duration and DateTime data types have been reduced

In RDFox v7.4, the internal storage of values representing moments in time (such as `xsd:dateTime`) and durations of time (such as `xsd:duration`) has been optimized to significantly reduce memory consumption. As a result, the some of the supported ranges have changed when compared to previous versions of RDFox.

Moments in Time:

- Affected Data Types:
  - `xsd:dateTime`
  - `xsd:dateTimeStamp`
  - `xsd:date`
  - `xsd:gYear`
  - `xsd:gYearMonth`
- New Supported Range:

- Values which have XSD "Time on Timeline" values between -142713-08-11T07:30:29.504Z and 142714-05-24T16:29:30.495Z.

Durations:

- Affected Data Types:
  - `xsd:duration`
  - `xsd:dayTimeDuration`
  - `xsd:yearMonthDuration`
- New Supported Ranges:
  - Month-only durations (i.e. xsd:yearMonthDuration or `xsd:durations` with zero days, hours, seconds and milliseconds) support values between -P192153584101141162Y8M and P192153584101141162Y7M.
  - Second-only durations (i.e. xsd:dayTimeDuration or `xsd:durations` with zero years and months) support values between -P26687997791DT19H48M13.952S and P26687997791DT19H48M13.951S.
  - Month-second durations support values between -P10922Y8M203613DT6H20M44.416S and P10922Y7M203613DT6H20M44.415S.

Any value outside these ranges will not be representable in RDFox v7.4 and will result in an error.

Required actions

To ensure a smooth migration, please run this SPARQL query against your data stores in RDFox v7.2 or v7.3 before upgrading to RDFox v7.4, to identify any values that fall outside the new supported ranges. If the query returns one or more values, update your data store using RDFox v7.2 or v7.3 to remove them before upgrading to RDFox v7.4. If you are not certain how to do this, please reach out to OST support for assistance.

3. Changes affecting data sources

3.1. The name of the data source table for `delimitedFile` data sources has changed

Details of a registered data source can be viewed using the `dsource show <data-source-name>` shell command or equivalent APIs. The output of the shell command includes detail about each table within the data source, including the name of the table. By definition, `delimitedFile` data sources only have one table. Prior to v7.4, the name of this table contained the path to the file but from v7.4 onwards, the name field always contains the static value `records`. The path to the file can still be retrieved in the parameters for the data source itself.

Required actions

In the unlikely event that your application relies on locating the file backing a `delimitedFile` data source from the table `name` field of the data source table description, update the code to find this information in the parameters of the data source instead.

3.2. The `absent` setting for the `if-empty` data source tuple table parameter has a new meaning

RDFox's data sources allow query or rule writers to define queries or rules that join data in their knowledge graph with data in an external source such as a relational database or a CSV file. To enable this, data in the external system must be mapped to RDF-compatible values by defining one or more data source tuple tables. The values in these table are created by interpolating the raw values from the external source into a lexical form template, according to the specification given when the data source tuple table is created. A typical lexical form template might be `http://family.guy/{1}_{2}` where `{1}` and `{2}` are placeholders for values from the external source.

Many data sources allow for some values to be unspecified. For example, relational data sources have the value `NULL` and delimited files can allow cells to be empty. The specification for `delimitedFile` and `solr` data sources allows the user to specify a policy for how to handle these empty values (the `if-empty` parameter for each column). Prior to RDFox v7.4, empty values in a lexical form template were handled differently in different data source types. For `delimitedFile` and `solr` data sources with an `if-empty` policy of `absent`, the final resulting value was undefined only if *all* the values referenced in the lexical form template were empty. In contrast, relational data source types (`PostgreSQL` and `ODBC`) do not accept the `if-empty` parameter but would produce an undefined value if *any* of the values referenced in the lexical form template were empty.

In RDFox v7.4, the definition of the `absent` policy for `delimitedFile` and `solr` data source tuple tables has changed to match the behavior of the relational data source types: if any of the of the input values referenced in the lexical form template is empty, the output value is undefined.

Review actions

No action is required if the server you are upgrading does not have a data source of type `delimitedFile` or `solr` in which at least one column:

- has its `if-empty` policy set to `absent` (note that this is not the default value)
- has a lexical form template with at least two value placeholders.

If your server does meet the above test please contact OST support for advice on how to check whether you are affected and how to proceed if so.

3.3. Solr data sources must be deregistered during upgrade

The implementation of Solr data sources has been updated to handle larger queries. This change introduces a persistence incompatibility that is not handled by RDFox's persistence upgrade functionality so additional steps must be taken with servers with Solr data source registrations.

Required actions

If you are performing an upgrade to v7.4 by transcribing the server's contents, no additional action is needed.

If you are performing an upgrade to v7.4 using the `upgrade` mode of the RDFox executable, you will need to take the following actions *for each data* store before upgrading the server:

1. Remove any rules that refer to a `solr` data source tuple table.
2. Deregister any `solr` data sources.
3. Compact the data store.

You may now perform the upgrade. Once it is finish, start the server and then reregister the data sources deregistered in step 2, and reinstall the rules removed in step 1.

4. Changes affecting the REST API

4.1. The REST API may now return HTTP response code 429

During the RDFox v7.4 release cycle, we became aware of a situation where all the HTTP worker threads for the endpoint could be simultaneously blocked waiting for a data store lock, which starved control requests to the `/health` route. This could lead to RDFox processes being presumed unhealthy and killed.

From RDFox v7.4 onwards, when the number of HTTP workers, as determined by the `num-http-workers` endpoint parameter, is 2 or greater, the endpoint will ensure that at least one thread is always available for observing and controlling the system. This includes the following:

- checking the endpoint health (`GET /health`),
- fetching information about the server (`GET /`),
- listing requests (`GET /requests`),
- cancelling requests (`DELETE /requests/{id}`)
- serving the console (`GET /console`),
- various other operations.

When servicing a new potentially long-running request would exceed the limit, the endpoint will now return 429 - Too many requests. Applications receiving this call can implement retrying logic with appropriate backoff behavior.

Required actions

Ensure that any code that calls the REST API can handle the 429 error code appropriately.

5. Changes affecting the Shell

5.1. Various shell variables have been changed to accept RDFox's standard duration specification format

Shell variables `log-frequency`, `query.profiler.sampling-frequency` and `reason.profiler.sampling-frequency` have been changed to use RDFox's standard duration specification format (see documentation Section 4.3.2). This is a breaking change for the latter two variables which previously accepted positive integers representing the frequency in milliseconds. When interpreted according to RDFox's standard duration specification format, a positive integer with no unit specifier would be interpreted as a number of seconds.

Required actions

Search any RDFox shell scripts for references to the above variables (probably as part of a `set` command). For any such commands, ensure that the specified value gives the desired duration when interpreted according to RDFox's duration specification format. For example, if the intention is for query profile sampling to occur every 10 seconds, the existing command would be:

```
set query.profiler.sampling-frequency 10000
```

which should be updated to either:

```
set query.profiler.sampling-frequency "10000 ms"
```

or:

```
set query.profiler.sampling-frequency 10
```

to give the desired frequency in RDFox v7.4.

6. Changes affecting the C and C++ APIs

6.1. Various changes to the C and C++ APIs

Various breaking changes were made to the C and C++ APIs to improve their ease of use. Prior to RDFox v7.4, these APIs were EXPERIMENTAL and undocumented but from this release, they are now supported for use in production and have minimal documentation.

Required actions

Developers with existing code that calls the C or C++ APIs should review the documentation of the C and C++ APIs to understand the new versions before attempting to work any compilation errors when rebuilding with the `CRDFox.h` file from v7.4. If additional assistance is needed, please contact OST support.