RDFox v7.6 Migration Guide

This guide describes functional changes in RDFox v7.6 that could prevent scripts, configuration, or code that work with RDFox v7.5, 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 v7.6.

RDFox v7.6 is persistence-compatible with RDFox v7.4 and v7.5, so it is not necessary to upgrade the server directory or transcribe the server content before restarting existing servers persisted by those versions with the new version. One exception to this for servers with Lucene data sources registered is described in Section 3.1.

Users upgrading from RDFox v7.4 should also read the RDFox v7.5 Migration Guide.

1. Changes affecting the Shell

1.1. Various shell directory variables have been removed

The shell variables `dir.dlog`, `dir.facts`, `dir.queries`, `dir.scripts` and `dir.stores` have been removed.

Relative paths that would have been resolved against any of these variables are now resolved against the `dir.root` variable.

Required actions

Search any RDFox shell scripts for the above variables (Likely as part of a `set` command). If any exist, you can resolve the issue with one of the following actions:

1. Remove the `set` commands and move all existing files to be in the correct place relative to the `dir.root`.
2. Remove the `set` commands and update relevant file paths in the shell script to be relative to the `dir.root`.
3. Keep the `set` commands, but add `$(var)` to the start of all the relevant file paths, where `var` is the variable being `set`.

1.2. Various times are printed with higher precision in the shell

In general, RDFox's shell output should not be used as a programming interface

2. Changes affecting query evaluation

2.1. SELECT WHERE ... LIMIT 1 and ASK queries

In previous releases, SPARQL `ASK WHERE {<query_body>}` queries were mapped internally to `SELECT DISTINCT WHERE {<query_body>}` queries (with no answer variables). This required result deduplication, which could be expensive on large datasets.

In RDFox v7.6, `ASK` queries are mapped to `SELECT WHERE {<query_body>} LIMIT 1`. Since `ASK` only needs to determine whether any result exists, execution now stops immediately after the first match is found, without deduplication overhead. The semantics are identical: both produce 0 or 1 empty tuples, but `LIMIT 1` is significantly faster.

Because the internal representation of `ASK` queries has changed, two edge-case `SELECT` query forms now produce **different response formats**. Search your RDFox-related configuration, scripts, and code for any usage of these patterns:

All examples below assume **no answer variables** (empty selection).

Query	RDFox v7.5 and earlier	RDFox v7.6 onwards
`ASK WHERE {...}`	Mapped to `SELECT DISTINCT WHERE {...}` → classified as `ASK` query → **boolean** response ({ "boolean": true })	Mapped to `SELECT WHERE {...} LIMIT 1` → classified as `ASK` query → **boolean** response ({ "boolean": true })
`SELECT DISTINCT WHERE {...}`	Classified as `ASK` query → **boolean** response ({ "boolean": true })	Classified as `SELECT` query → **tabular** response ({ "results": { "bindings": [...] } })
`SELECT WHERE {...} LIMIT 1`	Classified as `SELECT` query → **tabular** response ({ "results": { "bindings": [...] } })	Classified as `ASK` query → **boolean** response ({ "boolean": true })

Required actions

1. **Search for `SELECT DISTINCT WHERE {...}` with no answer variables**: If your code previously relied on this pattern being treated as an `ASK` query and returning a boolean response, it will now return a tabular response instead. Rewrite these as `ASK WHERE {...}` to get boolean output.
2. **Search for `SELECT WHERE {...} LIMIT 1` with no answer variables**: If your code uses this pattern and expects tabular results, be aware it is now internally classified as an `ASK` query and will return a boolean response instead. If you need tabular output, add an explicit answer variable (e.g., `SELECT ?x WHERE {...} LIMIT 1`) or use `SELECT DISTINCT WHERE {...}`.

3. Changes affecting data sources

3.1. Lucene data sources now have a sort parameter

The implementation of Lucene data sources has been updated to support geospatial queries. This change introduces a new parameter, `sort`, which allows the user to specify how answers should be ordered within Lucene before they are returned to RDFox.

Because the new parameter was not persisted by earlier releases, v7.6 cannot load servers persisted by lower versions if they have one or more registered Lucene data sources.

Required actions

With the server running using the v7.5 executable or library, perform the following steps:

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

You may now restart the server with v7.6, reregister the data sources deregistered in step 2, and reinstall the rules removed in step 1.

If for any reason, you need to restart the server with v7.5, you should follow the above procedure in reverse (i.e. deregistering and compacting in v7.6 and reregistering in v7.5).