Orchestration Server Deployment Guide

Persistence

4/4/2025

Contents

Persistence

To achieve persistence, ORS leverages the capability to store and retrieve information for SCXML sessions and SCXML documents. This includes the SCXML session data and the SCXML session state, as well as scheduled activities (such as start and stop).

Whenever required, these stored sessions or documents that have been persisted are fully recovered and used as needed. In some cases, extended sessions are utilized for long durations of time, up to several months or more. ORS 8.1 and later releases use persistence storage exclusively in conjunction with third-party Apache Cassandra (NoSQL Datastore).

The key function of persistence is supporting the retrieval of SCXML session information to restore the session context, as required. Session context is restored, for example, upon:

- a restart of the ORS component.
- the arrival of an event for an SCXML session, which was previously deactivated to reduce the memory required.

Storage and Retrieval Design and Configuration

The key principles of ORS persistence storage, retrieval design, and configuration are as follows:

- During configuration, all ORS instances within a single cluster must be configured to work with the same persistent storage, so that each Orchestration instance has access to all session information.
- The persistence layer allows storage and retrieval of data related to active Orchestration sessions. The amount of stored data is sufficient to restore an Orchestration session fully, and continue its execution.
- SCXML sessions are persisted upon the request by ORS application and only when the last queued event is flushed.
- Additional information is also persisted for ORS operations:
- A list of actions scheduled to be executed at specific times.
- Information that identifies the current SCXML session being handled by an ORS instance.

Persistent Storage Connection

ORS 8.1.2 and earlier releases required a connection to Cassandra. If ORS detects a lost connection, it exited. ORS 8.1.3 and later do not require a connection to Cassandra. If a connection is not present, an alarm is raised.

Be aware that if a connection is not present, the following functionality is not available:

- Session recovery is not available.

- Sending events between sessions may not be available.
- Working with HTTP interface may be unavailable depending on the configuration of the cluster and destination of the message.
- Scheduled session functionality will also not be available.

Persistent Storage Operation

Apache Cassandra provides a decentralized, fault tolerant, elastic, durable, and highly scalable distributed database system. Using Cassandra gives ORS a simpler deployment and installation, and assures scalability and performance of the back-end datastore for high-availability.

Document Persistence

SCXML documents are the basis for session construction. Many sessions may refer to the same document when persisted. The SCXML engine:

- Retrieves the requested document from the specified location. For example, in the Annex tab of a RoutePoint configuration, there would be an Orchestration section with the path to the application, such as: http://test52/Queues_1.scxml.
- Requests the persistence component to store the document information in the persistent storage when the document has been verified and compiled.

The compiled document is serialized and a request to store this document is made to the specified Cassandra cluster.

SCXML Session State and Data-Model Persistence

SCXML sessions are validated and compiled based on an SCXML document, and the session information including the full data model, is stored in the persistent storage. SCXML sessions are persisted at specific points of time to optimize session information integrity as well as system performance. When all events in the event queue have been processed, the SCXML session including the data model is serialized and stored in the persistence storage.

Persistence configuration options are listed in the persistence section, ORS Application object.

Note: Active sessions means that the sessions as defined in the SCXML document have not reached the final state. If this state is reached, the session information is removed from persistent storage.

Persistence Scheduling

SCXML sessions can be started in the future, or hung sessions can be terminated, through the Schedule component.

When ORS receives a request to schedule a session, if the requested time has passed, the session action is processed immediately.

Note: The requested time past is defined as the requested time plus the latency in processing the request. Currently five seconds is allowed for the latency period.

The scheduled session information is serialized and stored in Cassandra within the Keyspace as specified in configuration, in the Schedule SuperColumn.

Session High Availability

For effective High Availability of a session, the Cassandra data model provides persistence by maintaining the following information:

- ORS node currently responsible for processing the session.
- sessionID of the session.

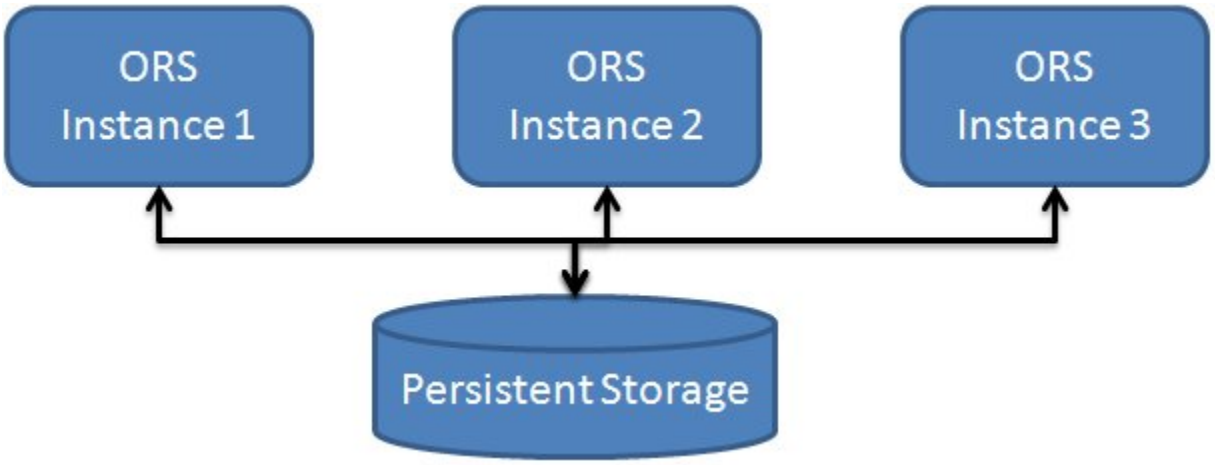
The ORS on which a session is created, persists the sessionID and its Orchestration Node ID, which is the dbid of the Orchestration application.

When the session reaches the state final, the entry for the session is removed from persistence.

The session-to-server node information is placed in the ColumnFamily SessionIDServerInfo, with ColumnName SessionServerInfo with keys of the sessionID. To facilitate the retrieval of the sessions for a given server node, the ColumnFamily SessionIDServerInfoRIndex is employed, with keys that are the string form of the Node ID and Columns that are the sessionids for that node.

Deploying Persistent Storage

In Figure 11, multiple instances of ORS are running and processing sessions and schedules at the same time. Refer to ORS Cluster Architecture for a description of multiple ORS instances configured as clusters. All ORS instances are active, and all work with persistent storage.



Configuring Persistent Storage

Configure persistent storage in the ORS Application object, Options tab, persistence section.

Option Name	Value
`cassandra-listenport`	Set to the value of the Cassandra port.
`cassandra-nodes`	Enter a semicolon-separated list of host names of the Cassandra nodes in the cluster.
`cassandra-keyspace-name`	Specify the name of the Cassandra Keyspace.
`cassandra-schema-version`	Enter the Cassandra schema version.
`cassandra-strategy-class`	Set to `SimpleStrategy` if Cassandra is deployed as a single cluster. Set to `NetworkTopologyStrategy` in the case of a Data Center Cassandra cluster deployment.
`cassandra-strategy-options`	Set the replication factor for a given Keyspace.

Examples

- If `SimpleStrategy`, cassandra-strategy-options = replication_factor:2
- If `NetworkTopologyStrategy`, cassandra-strategy-options = DC1:2;DC2:3

Enabling Persistence in the SCXML Application

Enabling persistence for an SCXML application is controlled with the `<scxml>` tag's attribute `_persist`, which has values of `true` and `false`. Starting with release 8.1.2, the default value for this attribute is `false`, therefore persistence for an SCXML application is disabled by default. A value of `true` activates persistence for an SCXML application.

Also see the `persistence-default` option in the scxml section.

When persistence for an SCXML application is enabled, ORS will regularly try to store the current session state into persistent storage.

Important: All nodes of ORS and Cassandra should have some time synchronization enabled and all should have the same time.

Function Behavior During Failover and Restoration

The information below relates to Orchestration persistence and how SCXML applications should be designed to handle failover and restoration.

ORS functions defined within SCXML documents (e.g., via <script> elements) may not behave as expected following an Orchestration failover and session restoration. Specifically, the scope in which

a function executes may change following a session restore. For more information, see the section on Functions and `<script>` Elements in the Extensions and Deviations section of the *SCXML Technical Reference*.