Clustering and Fail-over Considerations for Oracle Policy Automation 10.4

Oracle Technical Note – May 2014

Version 1.01 – Updated June 5th 2014

Oracle Policy Automation includes two high performance components that can be configured to run on an application server of your choice:

1. **OPA Web Determinations:** A web application for providing interactive guidance interviews to customers and to employees.
2. **OPA Determinations Server:** A web service that provides SOAP interfaces for (i) assessments, and (ii) custom development of Web Determinations-like interview experiences.

This document describes the clustering, fail-over and load-balancing options available for these applications.

OPA Determinations Server 10.4 Assess Service

OPA Determinations Server provides an Assess Service. Assess service SOAP operations are automatically provided for each policy model (rulebase) deployed to Determinations Server. For example, the policy model called "PersonalTaxAssessment" deployed to a Determinations Server with base URL https://my.com/opa-determinations-server would provide a generic WSDL at this URL: https://my.com/opa-determinations-server/assess/soap/generic/PersonalTaxAssessment?wsdl

The OPA Determinations Server Assess Service is a stateless service. Each request is made independently of any other, and does not require any session state to be managed in the application server.

A typical high performance, high availability configuration of the Assess Service involves two or more servers running OPA Determinations Server in an Active-Active configuration. Each server should be configured to have the same versions of the same policy models (rulebases). A hardware or software load-balancer can use any approach to direct incoming Assess Service requests to any of the available servers. This approach will work regardless of the selected application server (WebLogic, Tomcat, IIS etc.) or platform (Java or .NET). Server selection can be done by any load balancing algorithm such as round-robin or least connections, for example.

OPA Web Determinations 10.4

OPA Web Determinations is a web application that provides an HTML user interface for interactive guidance interviews. As provided, Web Determinations does not load or save data from any external data sources, but it can be configured to do so with the provided data adapters APIs.

A typical URL to start a web-determinations interview is https://my.com/opa-web-determinations/startsession/PersonalTaxAssessment, where PersonalTaxAssessment is the name of a deployed policy model (rulebase).

OPA Web Determinations is a state-based application. On each page a user typically enters information, which is added to the current interview session when the page is submitted. The OPA engine then determines whether a decision can be reached, or whether additional pages need to be displayed. The server maintains the session state of each interview that is currently in progress.

Because OPA Web Determinations maintains state, once an interview session is started, all subsequent requests for that session must be sent to a server that has the session state for that interview.

An affective high availability solution for OPA Web Determinations is to run an Active-Passive configuration, in which a single Active server is handling all incoming requests. If that server fails all requests are redirected to the formerly idle (Passive) server.

When OPA Web Determinations is running in an Active-Active clustered environment in which more than one instance is servicing the same end URL, the simplest approach to ensure correct operation is to use “sticky sessions” (also known as “affinity based clustering”). In this approach the load-balancing hardware or software must ensure that once an HTTP session is started by a particular Web Determinations instance in the cluster, every subsequent request on that session is directed to the same instance of Web Determinations. Sessions are typically identified by their Session ID which is transmitted via a cookie in the HTTP header. This approach will work regardless of the selected application server (WebLogic, Tomcat, IIS etc.) or platform (Java or .NET).

To avoid loss of all in-flight session data in the event of failure of an Active node, session persistence must be implemented. In some cases it may be appropriate for a user to simply restart their session, so session persistence may not be required. However, if complex data entry or long-running interviews are to be deployed using OPA Web Determinations, it may be desirable to implement some form of session persistence.

Session Persistence

To avoid loss of entered data when a server in a cluster fails, Web Determinations can be customized to automatically save every time the session is updated, and to automatically load case data when a session fails over to a different server in the cluster.

Customizing Web Determinations to do this comprises three parts:

1. A **data adaptor** that persists data entered by the user.
2. A **trigger** that will invoke the data adaptor’s save method every time the user submits data into the session.
3. A **data required** event handler that loads data when a case is not in memory on the current server.

The **data adaptor** is used by Web Determinations to save data collected during the interview. For example, a data adaptor may choose to save directly to an existing application using a provided web service, directly to a database or to an in-memory location such as memcache or Oracle Coherence. Information on Data Adaptors and implementation examples can be found at http://docs.oracle.com/html/E54502_01/toc.htm#Extensions/Extensions_DA_Overview.htm.

Every time a user submits a Web Determinations interview page (screen) that contains data, the interview engine's OnCommitEvent is fired. This notifies subscribers that data in the session have changed. The OnCommitEvent is therefore an ideal candidate for the save **trigger**. This event can be captured by creating and installing an OnCommitEventHandler plugin to the OPA Web Determinations application.

To ensure the session can be re-loaded in case of failure, a caseID needs to be set for the session which is done by passing it on the URL when the investigation is started. See http://docs.oracle.com/html/E54502_01/toc.htm#Extensions/Extensions_DA_Overview.htm.

The caseID must be passed to the OnCommitEvent so it can invoke the Data Adaptor's save method. To do this, the OnCommitEventHandler plugin can subscribe to the OnStartInvestigationEvent which provides access to the session's caseId. The recommended approach is to ensure a new OnCommitEventHandler instance is created for each session (i.e. not to have a singleton OnCommitEventHandler instance).

When a session fails over to a different server node, a plug-in that responds to the OnRequireSessionEvent is needed to load the session data. The request URL is passed to the event, and can be used to obtain the caseID. The handler for this event should then invoke the data adapter to load the data for that case.

For more information about OnCommitEvent, OnStartInvestigationEvent and OnRequireSessionEvent, including samples, see http://docs.oracle.com/html/E54502_01/toc.htm#Extensions/Events_and_Event_Handlers.htm.

OPA Determinations Server 10.4 Interview Service

Like OPA Web Determinations, the OPA Determinations Server Interview Service maintains session state, and so requires special consideration to support high-availability.

Unlike OPA Web Determinations, there is no opportunity to capture the OnRequireSessionEvent, so there is no ability to continue with the current interview session state should a node in the cluster fail.

Currently the best option available to implement high-availability for an application using the Interview Service is to:

1. Use affinity-based clustering (see explanation for Web Determinations)
2. Always call the Interview Service Load method to start a session (with a caseID)
3. Implement the auto-saving session persistence behavior as described for Web Determinations above.

In this case, the exact session state won't be preserved if a node in the cluster goes down, but no data will be lost. In a fail-over situation, the Load method will reload the saved case data, while information about the current screen the user was on will be lost. The user's experience would be to see the starting (summary screen) of the interview, and would be able to review their data to resume the interview where they left it off. In particular, if a flow goal is used for the interview, the full screen flow will be revisited. If a screen order is used, the interview with then resume at the first screen that needs to be shown to make progress towards the interview goal.

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com