Oracle® Fusion Middleware

Implementing Multi-Data Centers with Oracle Access Management

11*g* Release 2 (11.1.2.3) for All Platforms

E54424-04

September 2015

ORACLE®

Oracle Fusion Middleware Administrator's Guide for Oracle Access Management, 11*g* Release 2 (11.1.2.3) for All Platforms

E54424-04

Primary Author: Michael Teger

Contributing Author: Vinaye Misra, Kevin Kessler, Cathy Tenga, Serge Pomorski

Contributor: Vadim Lander, Vamsi Motokuru, Damien Carru, Peter Povinec, Weifang Xie, Satish Madawand, Neelima Jadhav, Charles Wesley, Harshal X Shaw, Jeremy Banford, Rey Ong, Ramana Turlapati, Deepak Ramakrishnan, David Goldsmith, Vishal Parashar, Carlos Subi, Patricia Fuzesy

Content

18 Configuring Multi-Data Centers

19 Synchronizing Data In A Multi-Data Center

20 Setting Up the Multi-Data Center: A Sequence

Preface

This extract from the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management* provides information on administration and configuration tasks for implementing Multi-Data Centers using Oracle Access Management.

Audience

This document is intended for Administrators who are familiar with:

- Oracle WebLogic Server concepts and administration
- LDAP server concepts and administration
- Database concepts and administration (for policy and session management data)
- Web server concepts and administration
- WebGate and mod_osso agents
- Auditing, logging, and monitoring concepts
- Security token concepts
- Integration of the policy store, identity store, and familiarity with Oracle Identity Management and OIS might be required

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

This Preface is for the Implementing Multi-Data Centers extract from the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*. It explains configuration and sequences for setting up Multi-Data Centers using Oracle Access Management. For more information, see the following guides in the Oracle Fusion Middleware 11*g* Release 2 (11.1.2.3) documentation.

- Oracle Access Management *11g* Release 2 (11.1.2.3) *Release Notes*
- *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*—Explains how to use the Oracle Universal Installer and the WebLogic Configuration Wizard for initial Access Manager 11g deployment. Installing 11g WebGates for Access Manager is also covered.
- *Oracle Fusion Middleware Developer's Guide for Oracle Access Management*—Explains how to write custom applications and plug-ins to functions programmatically, to create custom Access Clients that protect non-Web-based resources.
- *Oracle Fusion Middleware Upgrade Guide for Java EE*—For information about the types of Java EE environments available in 10g and instructions for upgrading those environments to Oracle Fusion Middleware 11g.
- *Oracle Fusion Middleware Upgrade Guide for Oracle Identity and Access Management*
- *Oracle Fusion Middleware Migration Guide for Oracle Identity and Access Management*
- *Oracle Fusion Middleware Performance and Tuning Guide*
- *Oracle Fusion Middleware Administrator's Guide*—Describes how to manage a secure Oracle Fusion Middleware environment, including how to change ports, deploy applications, and how to back up and recover Oracle Fusion Middleware. This guide also explains how to move data from a test to a production environment.
- *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*—For a step-by-step guide to deployment.
- *Oracle Fusion Middleware High Availability Guide*—For high availability conceptual information as well as administration and configuration procedures for Administrators, developers, and others whose role is to deploy and manage Oracle Fusion Middleware with high availability requirements.
- *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference for Identity and Access Management*—Provides details on customized Identity and Access Management WLST commands.
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*—Describes how to administer and secure Web services.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Understanding Multi-Data Centers

Oracle Access Manager allows for distribution of identical copies of directory service data across more than one data center. These multiple data centers (referred to as *multi-data centers*) provide a scalable deployment model to support access management requirements for millions of users.

The Access Manager Multi-Data Center topology scales horizontally - within a single data center by clustering multiple nodes, or across multiple data centers. This model provides for load balancing as well as failover capabilities in the case that one of the nodes or data centers goes down. This chapter contains introductory details.

1.1 Introducing the Multi-Data Center

Large organizations using Access Manager 11g typically deploy their applications across multi-data centers to distribute load as well as address disaster recovery. Deploying Access Manager in multi-data centers allows for the transfer of user session details transparently after configuration of single sign-on (SSO) between them. The scope of a data center comprises protected applications, WebGate agents, Access Manager servers and other infrastructure entities including identity stores and databases.

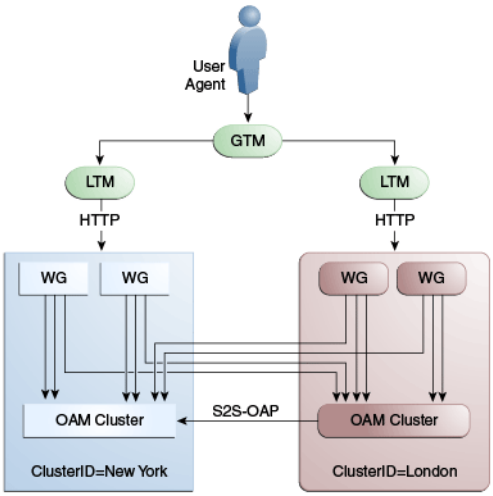
> **Note:** Access Manager 11g supports scenarios where applications are distributed across two or more data centers.

The Multi-Data Center approach supported by Access Manager is a Master-Clone deployment in which the first data center is specified as the Master and one or more Clone data centers mirror it. (Master and Clone data centers can also be referred to as Supplier and Consumer data centers.) A Master Data Center is duplicated using Test-to-Production (T2P) tools to create one or more cloned Data Centers. See *Oracle Fusion Middleware Administrator's Guide* for information on T2P.

During setup of the Multi-Data Center, session adoption policies are configured to determine where a request would be sent if the Master Data Center is down. Following the setup, a manner of replicating data from the Master to the Clone(s) will be designated. This can be done using the Automated Policy Sync (APS) Replication Service or it can be done manually. See Chapter 2, "Configuring Multi-Data Centers" and Chapter 3, "Synchronizing Data In A Multi-Data Center" for details on the setup and synchronization process.

A data center may include applications, data stores, load balancers and the like. Each data center includes a full Access Manager installation. The WebLogic Server domain in which the instance of Access Manager is installed will not span data centers. Additionally, data centers maintain user to data center affinity. Figure 1–1 illustrates the Multi-Data Center system architecture.

Figure 1–1 Multi-Data Center System Architecture



Multi-Data Center System Architecture

****

Note: Global load balancers are configured to route HTTP traffic to the geographically closest data center. No load balancers are used to manage Oracle Access Protocol traffic.

All applications are protected by WebGate agents configured against Access Manager clusters in the respective local data centers. Every WebGate has a primary server and one or more secondary servers; WebGate agents in each data center have Access Manager server nodes from the same data center in the primary list and nodes from other data centers in the secondary list. Thus, it is possible for a user request to be routed to a different data center when:

- A local data center goes down.
- There is a load spike causing redistribution of traffic.
- Certain applications are deployed in only one data center.
- WebGates are configured to load balance within one data center but failover across data centers.

The following sections contain more information on how the Multi-Data Center solution works.

- Understanding Cookies for Multi-Data Center
- Understanding Session Adoption During Authorization
- Understanding Session Indexing
- Supported Multi-Data Center Topologies

1.1.1 Understanding Cookies for Multi-Data Center

The following sections contain information on the SSO cookies enhanced and used by the Multi-Data Centers.

- OAM_ID Cookie
- OAMAuthn / ObSSO WebGate Cookies
- OAM_GITO (Global Inactivity Time Out) Cookie

1.1.1.1 OAM_ID Cookie

The OAM_ID cookie is the SSO cookie for Access Manager and holds the attributes required to enable the MDC behavior across all Data Centers. If a subsequent request from a user in the same SSO session is routed to a different Data Center in the Multi-Data Center topology, session adoption is triggered per the configured session adoption policies. *Session adoption* refers to the action of a Data Center creating a local user session based on the submission of a valid authentication cookie (OAM_ID) that indicates a session for the user exists in another other Data Center in the topology. (It may or may not involve re-authentication of the user.) When a user session is created in a Data Center, the OAM_ID cookie will be augmented/updated with the `clusterid` of the Data Center, a `sessionid` and the `latest_visited_clusterid`.

In Multi-Data Center deployments, OAM_ID is a host-scoped cookie. Its domain parameter is set to a virtual host name which is a singleton across data centers and is mapped by the global load balancer to the Access Manager servers in the Access Manager data center based on the load balancer level user traffic routing rules (for example, based on geographical affinity). The OAM_ID cookie is not accessible to applications other than the Access Manager servers.

1.1.1.2 OAMAuthn / ObSSO WebGate Cookies

OAMAuthn is the WebGate cookie for 11g and ObSSO is the WebGate cookie for 10g. On successful authentication and authorization, a user will be granted access to a protected resource. At that point, the browser will have a valid WebGate cookie with the `clusterid:sessionid` of the authenticating Data Center. If authentication followed by authorization spans across multiple Data Centers, the Data Center authorizing the user request will trigger session adoption by retrieving the session's originating `clusterid` from the WebGate cookie. (WebGates need to have the same host name in each data center due to host scoping of the WebGate cookies.) After adopting the session, a new session will be created in the current Data Center with the synced session details.

Note: The WebGate cookie cannot be updated during authorization hence the newly created `sessionid` cannot be persisted for future authorization references. In this case, the remote sessionid and the local sessionids are linked through `session indexing`. During a subsequent authorization call to a Data Center, a new session will be created when:

- MDC is enabled.
- A session matching the sessionid in the WebGate cookie is not present in the local Data Center.
- There is no session with a Session Index that matches the sessionid in the WebGate cookie.
- A valid session exists in the remote Data Center (based on the MDC SessionSync Policy).

In these instances, a new session is created in the local Data Center with a Session Index that refers to the sessionid in the WebGate cookie.

1.1.1.3 OAM_GITO (Global Inactivity Time Out) Cookie

OAM_GITO is a domain cookie set as an authorization response. The session details of the authentication process will be recorded in the OAM_ID cookie. If the authorization hops to a different Data Center, session adoption will occur by creating a new session in the Data Center servicing the authorization request and setting the session index of the new session as the incoming `sessionid`. Since subsequent authentication requests will only be aware of the clusterid:sessionid mapping available in the OAM_ID cookie, a session hop to a different Data Center for authorization will go unnoticed during the authentication request. To address this gap, an OAM_GITO cookie (which also facilitates timeout tracking across WebGate agents) is introduced.

During authorization, the OAM_GITO cookie is set as a domain cookie. For subsequent authentication requests, the contents of the OAM_GITO cookie will be read to determine the latest session information and the inactivity/idle time out values. The OAM_GITO cookie contains the following data.

- Data Center Identifier
- Session Identifier
- User Identifier
- Last Access Time
- Token Creation Time

Note: For the OAM_GITO cookie, all WebGates and Access Manager servers should share a common domain hierarchy. For example, if the server domain is `.us.example.com` then all WebGates must have (at least) `.example.com` as a common domain hierarchy; this enables the OAM_GITO cookie to be set with the `.example.com` domain.

1.1.2 Understanding Session Adoption During Authorization

Multi-Data Center session adoption is supported during the authorization flow. After successful authentication, the OAMAuthn cookie will be augmented with the cluster

ID details of the Data Center where authentication has taken place. During authorization, if the request is routed to a different Data Center, Access Manager runtime checks to determine whether it is a Multi-Data Center scenario by looking for a valid remote session. If one is located, the Multi-Data Center session adoption process is triggered per the session adoption policy.

The session adoption policy can be configured so that the clone Access Manager cluster would make a back-end request for session details from the master Access Manager cluster using the Oracle Access Protocol (OAP). The session adoption policy can also be configured to invalidate the previous session so the user has a session only in one data center at a given time. Following the session adoption process, a new session will be created in the Data Center servicing the authorization request.

Note: Since OAMAuthn cookie updates are not supported during authorization, the newly created session's session index will be set to that of the incoming session ID. See Understanding Session Indexing.

More details on session adoption can be found in the Section 1.2, "Understanding Multi-Data Center Deployments."

1.1.3 Understanding Session Indexing

During an authorization call to a Data Center, a new session will be created in the local Data Center with a Session Index that refers to the session identifier in the OAMAuth/ObSSO cookie. This will occur if all of the following conditions are met:

- Session matching Session ID in the OAMAuth/ObSSO cookie is not present in the local Data Center.
- MDC is enabled.
- No session with Session Index matching Session ID in the OAMAuth/ObSSO cookie.
- Valid Session exists in the remote Data Center based on the MDC SessionSync Policy.

1.1.4 Supported Multi-Data Center Topologies

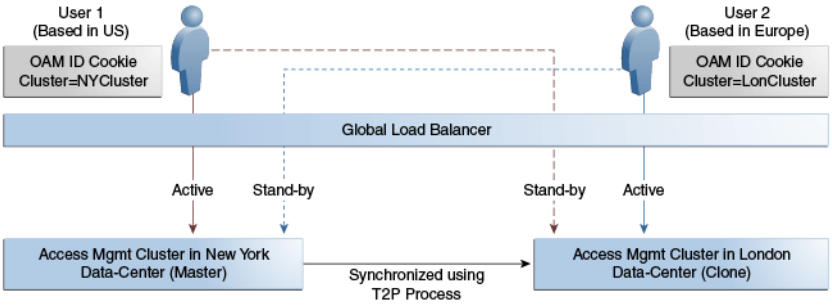
Access Manager supports several Multi-Data Center topologies. The following sections contain details on these modes.

- The MDC Active-Active Mode
- The MDC Active-Passive Mode
- The MDC Active-Hot Standby Mode

1.1.4.1 The MDC Active-Active Mode

An Active-Active topology is when Master and Clone data centers are exact replicas and active at the same time. They cater to different sets of users based on defined criteria; geography, for example. A load balancer routes traffic to the appropriate Data Center. Figure 1–2 illustrates a Multi-Data Center set up in Active-Active mode during normal operations.

Figure 1–2 Active-Active Deployment Mode



Active-Active Deployment Mode flow illustration

****

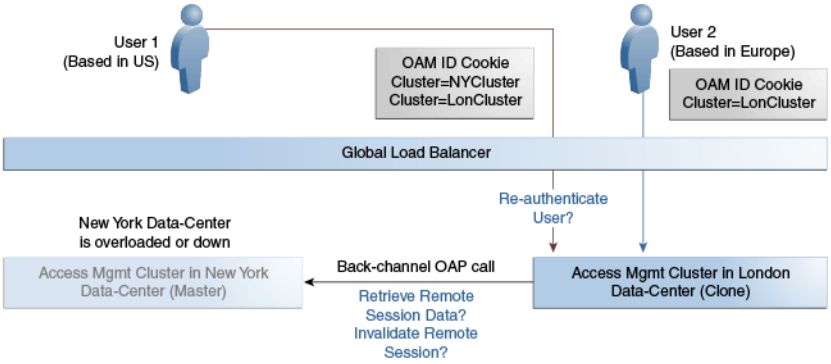
In Figure 1–2, the New York Data Center is designated as the Master and all policy and configuration changes are restricted to it. The London Data Center is designated as a Clone and uses T2P tooling and utilities to periodically synchronize data with the New York Data Center. The global load balancer is configured to route users in different geographical locations (US and Europe) to the appropriate data centers (New York or London) based on proximity to the data center (as opposed to proximity of the application being accessed). For example, all requests from US-based User 1 will be routed to the New York Data Center (NYDC) and all requests from Europe-based User 2 will be routed to the London Data Center (LDC).

> **Note:** The Access Manager clusters in Figure 1–2 are independent and not part of the same Oracle WebLogic domain. WebLogic domains are not recommended to span across data centers.

In this example, if NYDC was overloaded with requests, the global load balancer would start routing User 1 requests to the clone Access Manager cluster in LDC. The clone Access Manager cluster can tell (from the user's OAM_ID cookie) that there is a valid session in the master cluster and would therefore create a new session without prompting for authentication or re-authentication. Further, the session adoption policy can be configured such that the clone Access Manager cluster would make a back-end request for session details from the master Access Manager cluster using the Oracle Access Protocol (OAP). The session adoption policy can also be configured to invalidate the remote session (the session in NYDC) so the user has a session only in one data center at a given time.

Figure 1–3 illustrates how a user might be rerouted if the Master cluster is overloaded or down. If the Master Access Manager cluster were to go completely down, the clone Access Manager cluster would try to obtain the session details of User 1 but since the latter would be completely inaccessible, User 1 would be forced to re-authenticate and establish a new session in the clone Access Manager cluster. In this case, any information stored in the previous session is lost.

Figure 1–3 Active-Active Mode Failover



Active-Active Mode Failover flow illustration

****

> **Note:** An Active-Active topology with agent failover is when an agent has Access Manager servers in one Data Center configured as primary and Access Manager servers in the other Data Centers configured as secondary to aid failover scenarios.

More details on an Active-Active topology can be found in "Deploying Active-Active Multi-Data Center Topology."

1.1.4.2 The MDC Active-Passive Mode

An Active-Passive topology is when the primary Data Center is operable but the clone Data Center is not. In this topology, the clone can be brought up within a reasonable time in cases when the primary Data Center fails. Thus, in the Active-Passive Mode one of the data centers is passive and services are not started. In this use case, the data center does not have to be brought up immediately but within a reasonable amount of time in cases when the primary data center fails. There is no need to do an MDC setup although policy data will be kept in sync.

1.1.4.3 The MDC Active-Hot Standby Mode

Active–Hot Standby is when one of the Data Centers is in *hot standby* mode. In this case, traffic will not be routed to the Hot Standby Data Center unless the Active Data Center goes down. In this use case, you do not need additional data centers for traffic on a daily basis but only keep one ready. Follow the Active-Active Mode steps to deploy in Active-Hot Standby Mode but do not route traffic to the center defined as Hot Standby. The Hot Standby center will continue to sync data but will only be used when traffic is directed there by the load balancer or by an administrator.

1.2 Understanding Multi-Data Center Deployments

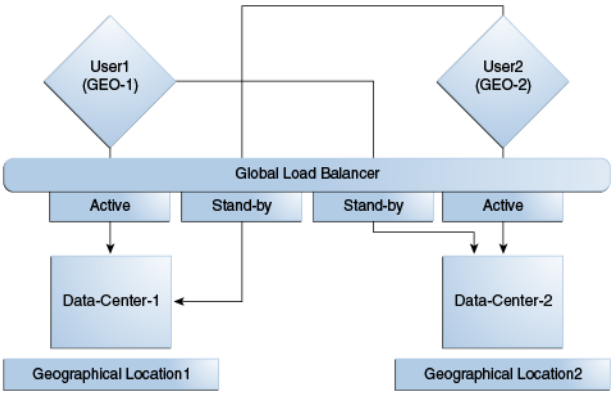
In a Multi-Data Center deployment, each data center will include a full Access Manager installation; WebLogic Server domains will not span the Data Centers. Global load balancers will maintain user to Data Center affinity although a user request may be routed to a different Data Center when:

- The data center goes down.

- A load spike causes redistribution of traffic.
- Each Data Center is not a mirror of the other. For example, certain applications may only be deployed in a single Data Center.
- WebGates are configured to load balance within the Data Center and failover across Data Centers.

Figure 1–4 illustrates a basic Multi-Data Center deployment.

Figure 1–4 Multi-Data Center Deployment



Multi-Data Center Deployment flow illustration

****

The following sections describe several deployment scenarios.

- Understanding Session Adoption Without Re-authentication, Session Invalidation or Session Data Retrieval
- Understanding Session Adoption Without Re-authentication But With Session Invalidation & Session Data Retrieval
- Understanding Session Adoption Without Re-authentication & Session Invalidation But With On-demand Session Data Retrieval
- Understanding Authentication & Authorization Requests Served By Different Data Centers
- Understanding Logout and Session Invalidation
- Understanding Stretch Cluster Deployments

> **Note:** The OAP connection used for back channel communication does not support load balancing or failover so a load balancer needs to be used.

1.2.1 Understanding Session Adoption Without Re-authentication, Session Invalidation or Session Data Retrieval

The following scenario illustrates the flow when the Session Adoption Policy is configured without re-authentication, remote session invalidation and remote session data retrieval. It is assumed the user has affinity with DC1.

1. User is authenticated by DC1.

   On successful authentication, the OAM_ID cookie is augmented with a unique data center identifier referencing DC1 and the user can access applications protected by Access Manager in DC1.

2. Upon accessing an application deployed in DC2, the user is routed to DC2 by a global load balancer.

3. Access Manager in DC2 is presented with the augmented OAM_ID cookie issued by DC1.

   On successful validation, Access Manager in DC2 knows that this user has been routed from the remote DC1.

4. Access Manager in DC2 looks up the Session Adoption Policy.

   The Session Adoption Policy is configured without reauthentication, remote session invalidation or remote session data retrieval.

5. Access Manager in DC2 creates a local user session using the information present in the DC1 OAM_ID cookie (lifetime, user) and re-initializes the static session information ($user responses).

6. Access Manager in DC2 updates the OAM_ID cookie with its data center identifier.

   Data center chaining is also recorded in the OAM_ID cookie.

7. User then accesses an application protected by Access Manager in DC1 and is routed back to DC1 by the global load balancer.

8. Access Manager in DC1 is presented with the OAM_ID cookie issued by itself and updated by DC2.

   On successful validation, Access Manager in DC1 knows that this user has sessions in both DC1 and DC2.

9. Access Manager in DC1 attempts to locate the session referenced in the OAM_ID cookie.

   - If found, the session in DC1 is updated.
   - If not found, Access Manager in DC1 looks up the Session Adoption Policy (also) configured without reauthentication, remote session invalidation and remote session data retrieval.

10. Access Manager in DC1 updates the OAM_ID cookie with its data center identifier and records data center chaining as previously in DC2.

1.2.2 Understanding Session Adoption Without Re-authentication But With Session Invalidation & Session Data Retrieval

The following scenario illustrates the flow when the Session Adoption Policy is configured without re-authentication but with remote session invalidation and remote session data retrieval. It is assumed the user has affinity with DC1.

1. User is authenticated by DC1.

   On successful authentication, the OAM_ID cookie is augmented with a unique data center identifier referencing DC1.

2. Upon accessing an application deployed in DC2, the user is routed to DC2 by a global load balancer.

3. Access Manager in DC2 is presented with the augmented OAM_ID cookie issued by DC1.

   On successful validation, Access Manager in DC2 knows that this user has been routed from the remote DC1.

4. Access Manager in DC2 looks up the Session Adoption Policy.

   The Session Adoption Policy is configured without reauthentication but with remote session invalidation and remote session data retrieval.

5. Access Manager in DC2 makes a back-channel (OAP) call (containing the session identifier) to Access Manager in DC1 to retrieve session data.

   The session on DC1 is terminated following data retrieval. If this step fails due to a bad session reference, a local session is created as documented in Section 1.2.1, "Understanding Session Adoption Without Re-authentication, Session Invalidation or Session Data Retrieval."

6. Access Manager in DC2 creates a local user session using the information present in the OAM_ID cookie (lifetime, user) and re-initializes the static session information ($user responses).

7. Access Manager in DC2 rewrites the OAM_ID cookie with its own data center identifier.

8. The user then accesses an application protected by Access Manager in DC1 and is routed to DC1 by the global load balancer.

9. Access Manager in DC1 is presented with the OAM_ID cookie issued by DC2.

   On successful validation, Access Manager in DC1 knows that this user has sessions in DC2.

10. Access Manager in DC1 makes a back-channel (OAP) call (containing the session identifier) to Access Manager in DC2 to retrieve session data.

    If the session is found, a session is created using the retrieved data. If it is not found, the OAM Server in DC1 creates a new session. The session on DC2 is terminated following data retrieval.

1.2.3 Understanding Session Adoption Without Re-authentication & Session Invalidation But With On-demand Session Data Retrieval

Multi-Data Center supports session adoption without re-authentication except that the non-local session are not terminated and the local session is created using session data retrieved from the remote DC. Note that the OAM_ID cookie is updated to include an attribute that indicates which data center is currently being accessed.

1.2.4 Understanding Authentication & Authorization Requests Served By Different Data Centers

Consider a scenario where an authentication request is served by the New York Data Center (NYDC) but the authorization request is presented to the London Data Center (LDC) because of user affinity. If Remote Session Termination is enabled, the scenario requires a combination of the OAM_ID cookie, the OamAuthn/ObSSO authorization cookie and the GITO cookie to perform the seamless Multi-Data Center operations. This flow (and Figure 1–5 following it) illustrates this. It is assumed that the user has affinity with NYDC.

1. Upon accessing APP1, a user is authenticated by NYDC.

On successful authentication, the OAM_ID cookie is augmented with a unique data center identifier referencing NYDC. The subsequent authorization call will be served by the primary server for the accessed resource, NYDC. Authorization generates the authorization cookie with the NYDC identifier (cluster-id) in it and the user is granted access to the APP1.

2. User attempts to access APP2 in LDC.
3. The WebGate for APP2 finds no valid session in LDC and initiates authentication.

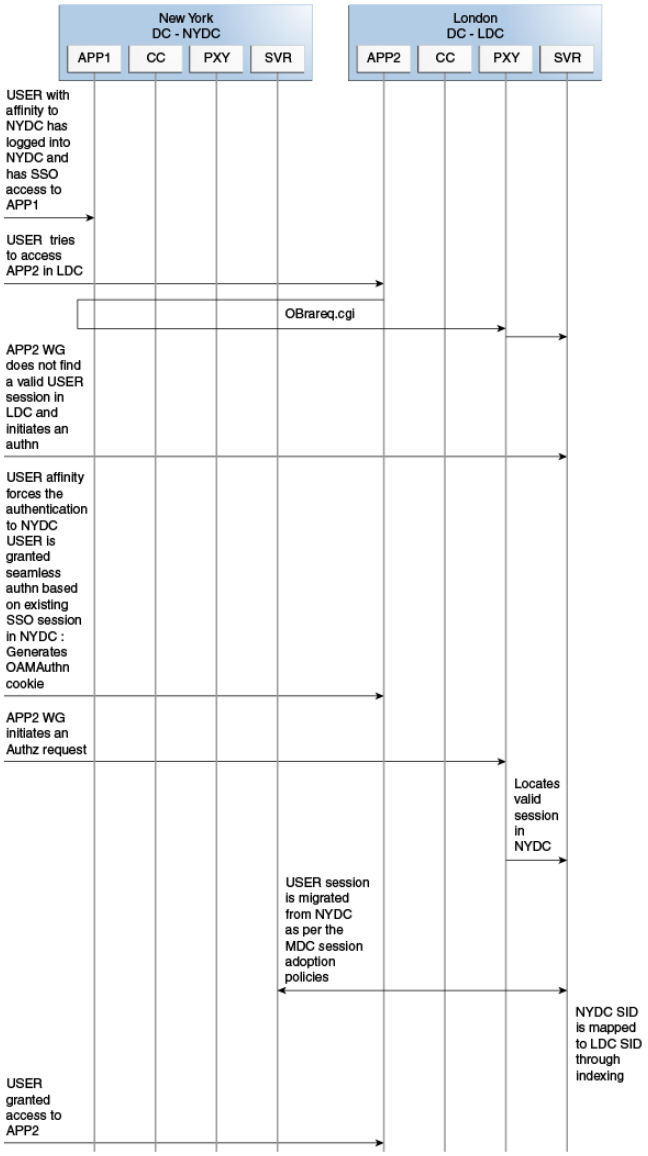
   Due to user affinity, the authentication request is routed to NYDC where seamless authentication occurs. The OamAuthn cookie contents are generated and shared with the APP2 WebGate.
4. The APP2 WebGate forwards the subsequent authorization request to APP2's primary server, LDC with the authorization cookie previously generated.

   During authorization, LDC will determine that this is a Multi-Data Center scenario and a valid session is present in NYDC. In this case, authorization is accomplished by syncing the remote session as per the configured session adoption policies.
5. A new session is created in LDC during authorization and the incoming session id is set as the new session's index.

   Subsequent authorization calls are honored as long as the session search by index returns a valid session in LDC. Each authorization will update the GITO cookie with the cluster-id, session-id and access time. The GITO cookie will be re-written as an authorization response each time.

   lf a subsequent authentication request from the same user hits NYDC, it will use the information in the OAM_ID and GITO cookies to determine which Data Center has the most current session for the user. The Multi-Data Center flows are triggered seamlessly based on the configured Session Adoption policies.

Figure 1–5 Requests Served By Different Data Centers



Flow diagram illustrating requests served by different Data Centers

****

1.2.5 Understanding Logout and Session Invalidation

In Multi-Data Center scenarios, logout ensures that all server side sessions across Data Centers and all authentication cookies are cleared out. For session invalidation, termination of a session artifact over the back-channel will not remove the session cookie and state information maintained in the WebGates. However, the lack of a server session will result in an Authorization failure which will result in re-authentication. In the case of no session invalidation, the logout clears all server side sessions that are part of the current SSO session across Data Centers. This flow (and Figure 1–6 following it) illustrates logout. It is assumed that the user has affinity with NYDC.

1. User with affinity to NYDC gets access to APP1 after successful authentication with NYDC.

2. User attempts to access APP2.

   At this point there is a user session in NYDC as well as LDC (refer to section 2.4.5) as part of SSO.

3. User logs out from APP1.

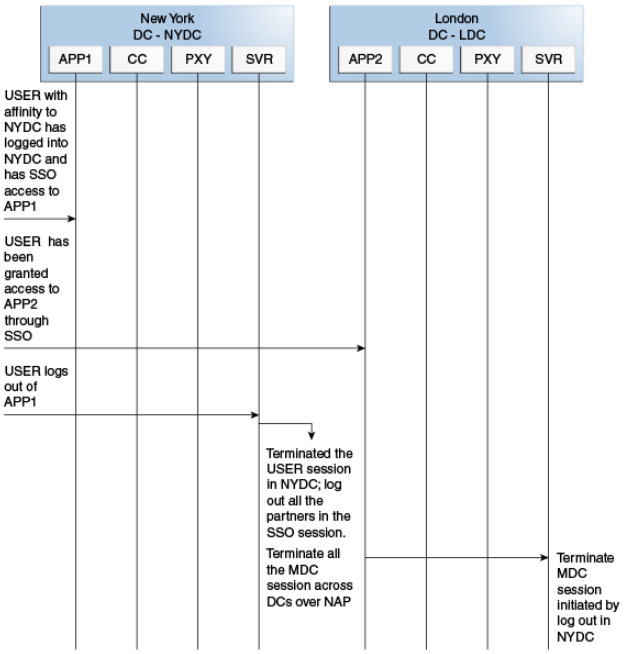
   Due to affinity, the logout request will reach NYDC.

4. The NYDC server terminates the user's SSO session and logs out from all the SSO partners.

5. The NYDC server sends an OAP terminate session request to all relevant Data Centers associated with the SSO session - including LDC.

   This results in clearing all user sessions associated with the SSO across Data Centers.

Figure 1–6 Logout and Session Invalidation



Flow diagram illustrating logout and session invalidation

****

1.2.6 Understanding Stretch Cluster Deployments

For data centers that are geographically very close and have a guaranteed latency of less than 5 milliseconds, customers can choose a Stretch Cluster deployment. In this case, unlike the traditional Multi Data Center deployment described in the preceding sections, a single OAM cluster is stretched across multiple data centers; there are some OAM nodes in one Data Center and the remaining nodes in another Data Center. Though the deployment is spread across two data centers, Access Manager treats this as a single cluster. The policy database would reside in one of the Data Centers. The following limitations apply to a Stretch Cluster Deployment.

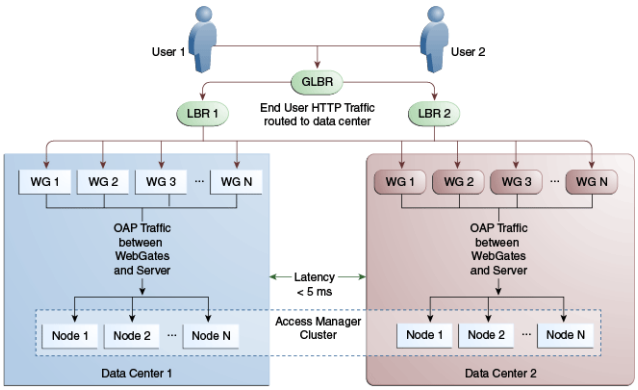
- Access Manager depends on the underlying WebLogic and Coherence layers to keep the nodes in sync. The latency between Data Centers must be less than 5

milliseconds at all times. Any spike in the latency may cause instability and unpredictable behavior.

- The cross data center chatter at runtime in a Stretch Cluster deployment is relatively more than that in the traditional Multi Data Center deployment. In case of the latter, the runtime communication between Data Centers is restricted to use-cases where a session has to be adopted across Data Centers.
- Since it is a single cluster across Data Centers, it does not offer the same level of reliability/availability as a traditional multi data center deployment. The policy database can become a single point of failure. In a traditional Multi Data Center deployment, each Data Center is self-sufficient and operates independent of the other Data Center which provides far better reliability.

Figure 1–7 illustrates a Stretch Cluster deployment while Figure 1–8 below it illustrates a traditional MDC deployment. Oracle does recommend a traditional multi data center deployment over a Stretch Cluster deployment. See also Section 1.1, "Introducing the Multi-Data Center" for other topology illustrations.

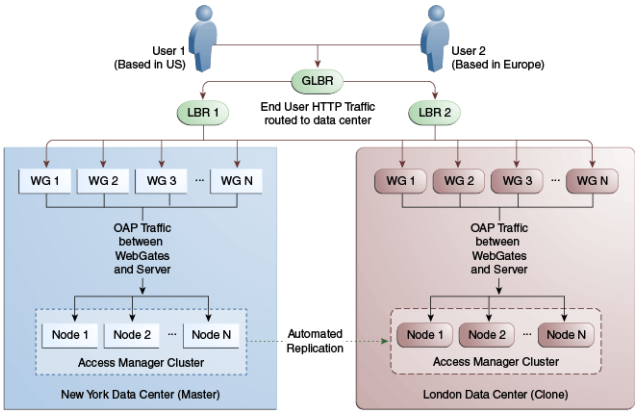
Figure 1–7 Stretch Cluster Deployment



Stretch Cluster Deployment

****

Figure 1–8 Traditional MDC Deployment



Traditional MDC Deployment

****

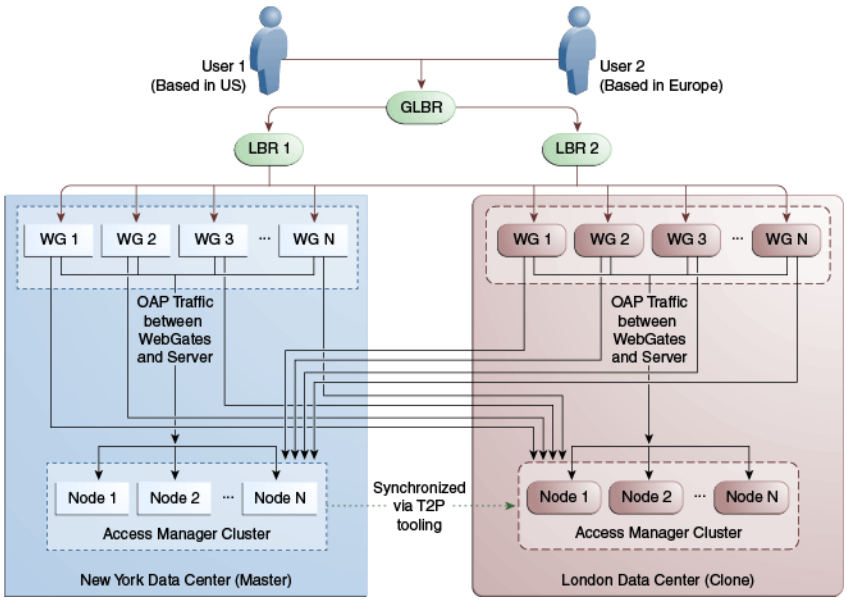
1.3 Deploying Active-Active Multi-Data Center Topology

An Active-Active topology is when Master and Clone Data Centers are exact replicas of each other (including applications, data stores and the like). They are active at the same time and cater to different sets of users based on defined criteria - geography, for example. A load balancer routes traffic to the appropriate Data Center. Identical Access Manager clusters are deployed in both locales with New York designated as the Master and London as the Clone.

> **Note:** An Active-Active topology with agent failover is when an agent has Access Manager servers in one Data Center configured as primary and Access Manager servers in the other Data Centers configured as secondary to aid failover scenarios.

Figure 1–9 illustrates the topology for a Multi-Data Center deployment in Active-Active mode. The New York Data Center is designated as the Master and all policy and configuration changes are restricted to it. The London Data Center is designated as a Clone and uses T2P tooling and utilities to periodically synchronize data with the New York Data Center. The global load balancer is configured to route users in different geographical locations (US and Europe) to the appropriate data centers (New York or Europe) based on proximity to the data center (as opposed to proximity of the application being accessed). For example, all requests from US-based User 1 will be routed to the New York Data Center (NYDC) and all requests from Europe-based User 2 will be routed to the London Data Center (LDC).

Figure 1–9 Active-Active Topology



Active-Active Topology flow diagram

****

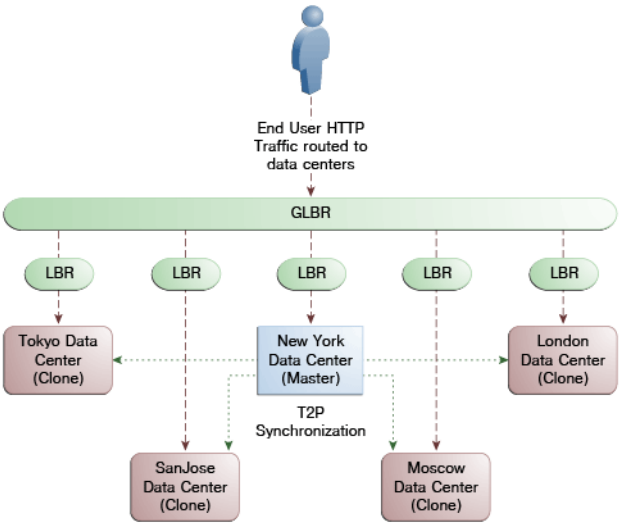
The Global Load Balancer is configured for session stickiness so once a user has been assigned to a particular data center, all subsequent requests from that user would be routed to the same data center. In this example, User 1 will always be routed to the New York Data Center and User 2 to the London Data Center.

User requests in the respective data centers are intercepted by different WebGates depending on the application being accessed. Each WebGate has the various nodes of the Access Manager cluster within the same data center configured as its primary servers. In this case, the WebGates load balance and failover the local data center.

> **Note:** Administrators have the flexibility to configure the primary servers for every WebGate in different orders based on load characteristics. Running monitoring scripts in each data center will detect if any of the Access Manager components – the WebGates or the servers – are unresponsive so administrators can reconfigure the load balancers to direct user traffic to a different data center.

Any number of Clone data centers can be configured to distribute the load across the globe. The only condition is that all Clone data centers are synchronized from a single Master using T2P. Figure 1–10 below depicts an Active-Active Multi-Data Center deployment across five data centers.

Figure 1–10 Active-Active Topology Across Multiple Data Centers



Active-Active Topology Across MDC flow diagram

****

1.4 Load Balancing Between Access Management Components

The topology described earlier shows global and local load balancers for routing the end user HTTP traffic to various data centers. Additionally, customers can choose to deploy load balancers between the access manager components to simplify the configuration of the access manager components by using virtual host names. For example, instead of configuring the primary servers in each WebGate in the NYDC as ssonode1.ny.acme.com, ssonode2.ny.acme.com and so on, they can all point to a single

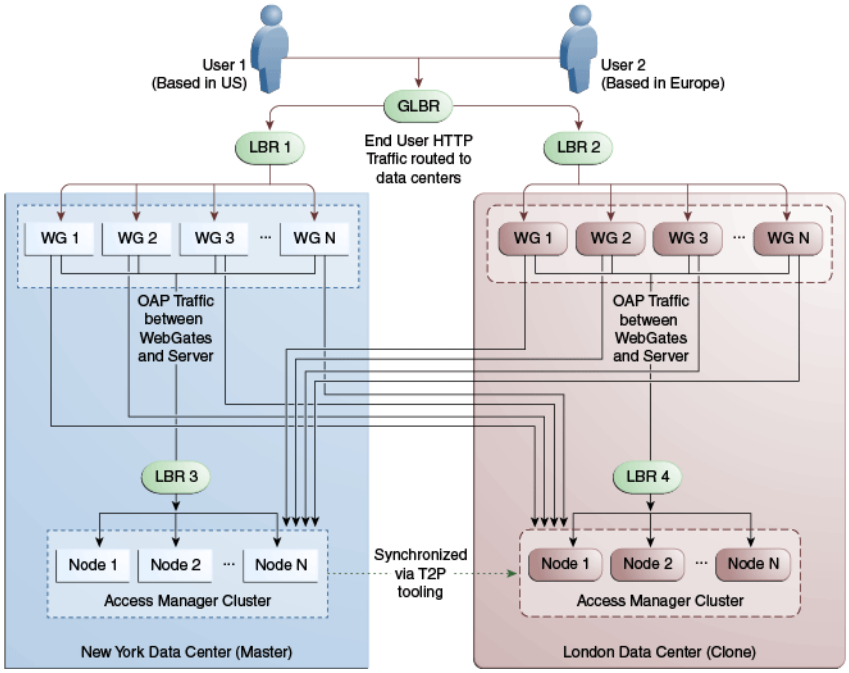
virtual hostname like sso.ny.acme.com and the load balancer will resolve the DNS to direct them to various nodes of the cluster. However, while introducing a load balancer between Access Manager components, there are a few constraining requirements to keep in mind.

- OAP connections are persistent and need to be kept open for a configurable duration even while idle.
- The WebGates need to be configured to recycle their connections proactively prior to the Load Balancer terminating the connections, unless the Load Balancer is capable of sending TCP resets to both the Webgate and the server ensuring clean connection cleanup.
- The Load Balancer should distribute the OAP connection uniformly across the active Access Manager Servers for each WebGate (distributing the OAP connections according the source IP), otherwise a load imbalance may occur.

Figure 1–11 illustrates a variation of the deployment topology with local load balancers (LBR 3 and LBR 4) front ending the clusters in each data center. These local load balancers can be Oracle HTTP Servers (OHS) with `mod_wl_ohs`. The OAP traffic still flows between the WebGates and the Access Manager clusters within the data center but the load balancers perform the DNS routing to facilitate the use of virtual host names.

Note: For information on monitoring Access Manager server health with a load balancer in use, see Section 11.6, "Monitoring the Health of an Access Manager Server."

Figure 1–11 Load Balancing Access Manager Components

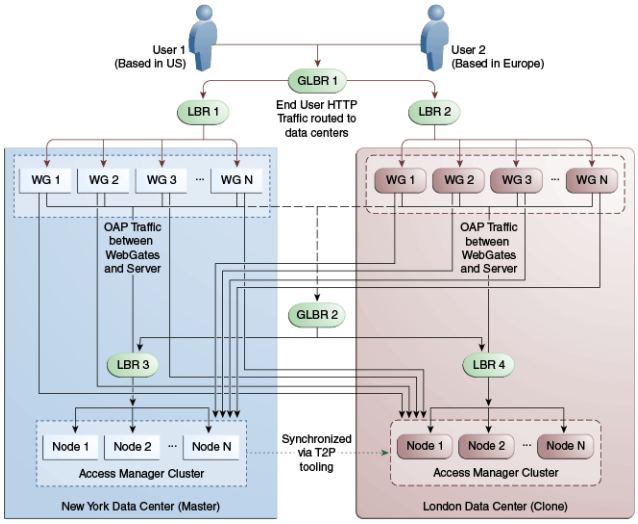


Load Balancing OAM components flow diagram

****

Figure 1–12 illustrates a second variation of the deployment topology with the introduction of a global load balancer (GLBR2) to front end local load balancers (LBR3 and LBR4). In this case, the host names can be virtualized not just within the data center but across the data centers. The WebGates in each data center would be configured to load balance locally but fail over remotely. One key benefit of this topology is that it guarantees high availability at all layers of the stack. Even if the entire Access Manager cluster in a data center were to go down, the WebGates in that data center would fail over to the Access Manager cluster in the other data center.

Figure 1–12 Global Load Balancer Front Ends Local Load Balancer



Global Load Balancer Front Ends Local Load Balancer

****

1.5 Understanding Time Outs and Session Syncs

The following sections contain information on how the Multi-Data Center deals with session time outs and syncs.

- Ensuring Maximum Session Constraints
- Configuring Policies for Idle Timeout
- Expiring Multi-Data Center Sessions
- Synchronizing Sessions and Multi-Data Center Fail Over

1.5.1 Ensuring Maximum Session Constraints

Credential Collector user affinity ensures that maximum session constraints per user are honored. There is no Multi-Data Center session store to validate allowed maximum sessions per user.

1.5.2 Configuring Policies for Idle Timeout

The OAM_ID and OAM_GITO cookies are used to calculate and enforce idle (inactivity) timeouts. The OAM_GITO cookie, though, can be set only if there is a common sub-domain across WebGates. Thus, Multi-Data Center policies should be

configured based on whether or not the OAM_GITO cookie is set. Table 1–1 documents the policy configurations.

Table 1–1 Multi-Data Center Policy Configurations for Idle Timeout

OAM_GITO Set	Multi-Data Center Policies
Yes	SessionMustBeAnchoredToDataCenterServicingUser=<true/false>
Idle timeout will be calculated from the latest OAM_GITO cookie	SessionDataRetrievalOnDemand=true
	Reauthenticate=false
	SessionDataRetrievalOnDemandMax_retry_attempts=<number>
	SessionDataRetrievalOnDemandMax_conn_wait_time=<milliseconds>
	SessionContinuationOnSyncFailure=<true/false>
	MDCGitoCookieDomain=<sub domain>
No	SessionMustBeAnchoredToDataCenterServicingUser=false
Idle time out will be calculated from the OAM_ID cookie because OAM_GITO is not available	SessionDataRetrievalOnDemand=true
	Reauthenticate=false
	SessionDataRetrievalOnDemandMax_retry_attempts=<number>
	SessionDataRetrievalOnDemandMax_conn_wait_time=<milliseconds>
	SessionContinuationOnSyncFailure=<true/false>
	#MDCGitoCookieDomain= This setting should be commented or removed

1.5.3 Expiring Multi-Data Center Sessions

Session expiration will be managed by the Data Center with which the user has affinity. Users have affinity to a particular Data Center based on the global trffic manager/load balancer.

1.5.4 Synchronizing Sessions and Multi-Data Center Fail Over

Access Manager server side sessions are created and maintained based on single sign-on (SSO) credentials. The attributes stored in the session include (but are not limited to) the user identifier, an identity store reference, subject, custom attributes, partner data, client IP address and authentication level. SSO will be granted if the server can locate a valid session corresponding to the user's request.

In a Multi-Data Center scenario, when a user request hops across Data Centers, the Data Center servicing the request should validate for a legitimate session locally and across Data Centers. If a valid session for a given request exists in a remote Data Center, the remote session needs to be migrated to the current Data Center based on the MDC session synchronization policies. (See Section 1.2, "Understanding Multi-Data Center Deployments" for details.) During this session synchronization, all session attributes from the remote session are synced to the newly created session in the Data Center servicing the current request.

The Multi-Data Center also supports WebGate failover across Data Centers. When a WebGate fails over from one Data Center to a second, the session data can not be synchronized because the first Data Center servers are down. Thus, the second Data Center will decide whether or not to proceed with the session adoption based on the setting configured for `SessionContinuationOnSyncFailure`. When true, even if the OAP communication to the remote Data Center fails, the Data Center servicing the current request can proceed to create a new session locally based on the mandatory

attributes available in the cookie. This provides seamless access to the requested resource despite the synchronization failure. Table 1–2 summarizes prominent session synchronization and failover scenarios. The parameters in this table are explained in greater detail in Section 2–3, " partnerInfo.properties Properties."

Table 1–2 Session Synchronization and Failover Scenarios

MDC Deployment	MDC Policy	Validate Remote Session	Session Synchronized in DC Servicing User From Remote DC	Terminate Remote Session	User Challenged
Active-Active	SessionMustBeAnchoredToDataCenterServicingUser=true SessionDataRetrievalOnDemand=true Reauthenticate=false SessionDataRetrievalOnDemandMax_retry_attempts=<number> SessionDataRetrievalOnDemandMax_conn_wait_time=<milliseconds> SessionContinuationOnSyncFailure = false MDCGitoCookieDomain=<subdomain>	Yes	Yes	Yes	When a valid session could not be located in a remote DC

Table 1–2 (Cont.) Session Synchronization and Failover Scenarios

MDC Deployment	MDC Policy	Validate Remote Session	Session Synchronized in DC Servicing User From Remote DC	Terminate Remote Session	User Challenged
Active-Active	SessionMustBeAnchoredToDataCenterServicingUser=false SessionDataRetrievalOnDemand=true Reauthenticate=false SessionDataRetrievalOnDemandMax_retry_attempts=<number> SessionDataRetrievalOnDemandMax_conn_wait_time=<millisecon ds> SessionContinuationOnSyncFailure = false MDCGitoCookieDomain=<subdomain>	Yes	Yes	No	When a valid session could not be located in a remote DC
Active-Standby	SessionMustBeAnchoredToDataCenterServicingUser=true SessionDataRetrievalOnDemand=true Reauthenticate=false SessionDataRetrievalOnDemandMax_retry_attempts=<number> SessionDataRetrievalOnDemandMax_conn_wait_time=<millisecon ds> SessionContinuationOnSyncFailure = false MDCGitoCookieDomain=<subdomain>	Could not validate as the remote DC is down	No, since the remote DC is down	Could not terminate as the remote DC is down	Yes
Active-Standby	SessionMustBeAnchoredToDataCenterServicingUser=true SessionDataRetrievalOnDemand=true Reauthenticate=false SessionDataRetrievalOnDemandMax_retry_attempts=<number> SessionDataRetrievalOnDemandMax_conn_wait_time=<milliseconds> SessionContinuationOnSyncFailure = true MDCGitoCookieDomain=<subdomain>	Could not validate as the remote DC is down	No, since the remote DC is down	Could not terminate as the remote DC is down	No Provides seamless access by creating a local session from the details available in the valid cookie

1.6 Replicating a Multi-Data Center Environment

Data in the Multi-Data Center environment must be replicated from the Master (supplier) to the Clones (consumers) as part of the initial setup procedure. Following this initial replication, the data must be synced across data centers on a regular basis. The following sections have more details.

- Replicating Data Using the WLST
- Syncing Data Using Automated Policy Synchronization

The following artifacts must be replicated and synced regularly.

- WebGate Profiles

  While the WebGate profile is replicated to the Clone, the primary server list and logout URL details are updated with information about the Clone data center.
- Authentication Modules
- OAM Proxy Configurations
- Session Manager configurations
- Policy and partner data

For more details on replication and syncing data, see Chapter 3, "Synchronizing Data In A Multi-Data Center."

1.6.1 Replicating Data Using the WLST

Initial replication of data (when setting up the Multi-Data Center) must be done manually using the WLST. Following this initial replication, WLST commands or the Automated Policy Sync Replication Service (discussed in Syncing Data Using Automated Policy Synchronization) can be used to sync the already replicated data. When using the WLST, partner profiles and policies are exported from the Master Data Center and then imported to the Clone Data Center. Replication of data in a Multi-Data Center environment is a requirement and using WLST for this purpose is the minimum method for accomplishing this. For more details, see Section 3.1.3, "Manually Syncing Data in a Multi-Data Center."

1.6.2 Syncing Data Using Automated Policy Synchronization

Automated Policy Synchronization (APS, also referred to as the Replication Service) is a set of REST API used to automatically replicate data from the Master Data Center to Clone Data Centers. It can be configured to keep Access Manager data synchronized across multiple data centers. A valid replication agreement between the data centers must be present before APS can run. For more details, see Section 3.1, "Understanding the Multi-Data Center Sync."

Note: APS is not used to do a complete replication from scratch. You will first need to replicate data manually using the WLST to establish a base line. APS is only designed to keep data centers in sync.

1.7 Multi-Data Center Recommendations

This section contains recommendations regarding the Multi-Data Center functionality.

- Using a Common Domain

- Concerning the DCC and the OAM_GITO
- Using an External Load Balancer
- Honoring Maximum Sessions
- WebGate Cookie Cannot Be Refreshed During Authorization

1.7.1 Using a Common Domain

It is recommended that WebGates be domain-scoped in a manner that a common domain can be inferred across all WebGates and the OAM Server Credential Collectors. This allows for WebGates to set an encrypted GITO cookie to be shared with the OAM Server. For example, if WebGates are configured on applications.abc.com and the OAM Server Credential Collectors on server.abc.com, abc.com is the common domain used to set the GITO cookie. In scenarios where a common domain cannot be inferred, setting the GITO cookie is not practical as a given Data Center may not be aware of the latest user sessions in another Data Center. This would result in the Data Center computing session idle-timeout based on old session data and could result in re-authenticating the user even though a more active session lives elsewhere.

> **Note:** A similar issue occurs during server fail-over when the `SessionContinuationOnSyncFailure` property is set. The expectation is to retrieve the session from contents of the OAM_ID cookie. Since it's not possible to retrieve the actual inactivity time out value from the GITO cookie, a re-authentication could result.

When there is no common cookie domain across WebGates and OAM servers, make the following configuration changes to address idle time out issues.

- Run the `enableMultiDataCentreMode` WLST command after removing the MDCGitoCookieDomain property from the input properties file.
- Because a WebGate cookie cannot be refreshed during authorization, set the value of the WebGate cookie validity lower than the value of the session idle time out property. Consider a session idle time out value of 30 minutes and a WebGate cookie validity value of 15 minutes; in this case, every 15 minutes the session will be refreshed in the authenticating Data Center.

> **Note:** For 10G WebGates, since the token is not expired by the WebGate, the server will continue to honor a 10G WebGate cookie until the session in the base DC (authenticating DC) idles out.

1.7.2 Concerning the DCC and the OAM_GITO

The OAM_GITO cookie is not applicable when using the DCC. Because of this:

- The #MDCGitoCookieDomain= setting should be commented out.
- The SessionMustBeAnchoredToDataCenterServicingUser parameter must be set to false.
- The WebGate cookie expiration interval should be set as documented in "Using a Common Domain."

1.7.3 Using an External Load Balancer

Access Manager uses the 11g SDK API to retrieve session data but this API does not support SDK based load-balancing across the configured set of primary servers. Use an external TCP based load balancer to front-end the OAP endpoints of the Data Center nodes where high performance is expected.

> **Note:** Failover between primary and secondary OAM servers is supported in the current release of 11g SDK APIs.

1.7.4 Honoring Maximum Sessions

A typical Multi-Data Center scenario authenticates users against the Data Center with which the user geography has an affinity. In the rare scenarios where user authentication and session creation for a given user spans across member Data Centers (bypassing geographic affinity and load spike), the maximum sessions the user has in the whole Multi-Data Center topology would not be honored.

1.7.5 WebGate Cookie Cannot Be Refreshed During Authorization

Because a WebGate cookie cannot be refreshed during authorization, set the value of the WebGate cookie validity lower than the value of the session idle time out property. Consider a session idle time out value of 30 minutes and a WebGate cookie validity value of 15 minutes; in this case, every 15 minutes the session will be refreshed in the authenticating Data Center. Setting the WebGate cookie expiration to less than 2 minutes is the recommendation.

> **Note:** This will not work for 10G WebGates because the 10G WebGate token expiration is driven by the server and not the WebGate. The server will continue to honor a 10G WebGate cookie until the session in the base DC (authenticating DC) idles out. A logout will work by clearing browser cookies; the dangling server side session will continue to exist but is considered harmless.

2

Configuring Multi-Data Centers

The Multi-Data Center feature is disabled by default. This chapter contains details on how to enable and configure the Multi-Data Center functionality.

The following sections have details.

- Before Setting Up a Multi-Data Center
- Understanding the Primary Use Cases
- Setting Up a Multi-Data Center
- Adding A Second Clone to An Existing Multi-Data Center Setup
- Understanding Multi-Data Center Security Modes
- WLST Commands for Multi-Data Centers

2.1 Before Setting Up a Multi-Data Center

The following prerequisites must be satisfied before beginning the Multi-Data Center (MDC) configuration process documented in Setting Up a Multi-Data Center.

- Ensure you have a fully functioning Oracle Access Management environment with all applicable WebGates configured.
- Partners (WebGates or agents) are anchored to a single Data Center thus, partner registration is done at the individual Data Centers.
- All Data Center clusters must be front ended by a single Load Balancer. The load balancer should send all requests in a user session consistently to the same backend server (persistence, stickiness) and it should be route traffic geographically (geo-affinity).
- Clocks on the machines in which Access Manager and agents are deployed must be in sync. Non-MDC Access Manager clusters require the clocks of WebGate agents be in sync with Access Manager servers. This requirement applies to the MDC as well. If the clocks are out of sync, token validations will not be consistent resulting in deviations from the expected behaviors regarding the token expiry interval, validity interval, timeouts and the like.
- The identity stores in a Multi-Data Center topology must have the same Name.
- WebLogic Server domains do not span Data Centers.
- Any firewall between Data Centers must allow communication over the Oracle Access Protocol (OAP) channel. This entails opening the necessary ports and taking into account the lifetime of the connection. In regards to the latter, the

MaxSessionTime parameter in the WebGate profile should be set to less than the firewall timeout value.

2.2 Understanding the Primary Use Cases

Table 2–1 lists the primary MDC use cases.

Table 2–1 MDC Use Cases

MDC Deployment	MDC Policy	Validate Remote Session	Session Synchronized in DC Servicing User From Remote DC	Terminate Remote Session	User Challenged
Active-Active	SessionMustBeAnchoredToDataCenterServicingUser=false SessionDataRetrievalOnDemand=true Reauthenticate=false SessionDataRetrievalOnDemandMax_retry_attempts=<number> SessionDataRetrievalOnDemandMax_conn_wait_time=<milliseconds> SessionContinuationOnSyncFailure= false MDCGitoCookieDomain=<sub domain>	Yes	Yes	No	When a valid session could not be located in a remote DC
Active-Standby	SessionMustBeAnchoredToDataCenterServicingUser=false SessionDataRetrievalOnDemand=true Reauthenticate=false SessionDataRetrievalOnDemandMax_retry_attempts=<number> SessionDataRetrievalOnDemandMax_conn_wait_time=<milliseconds> SessionContinuationOnSyncFailure= true MDCGitoCookieDomain=<sub domain>	Could not validate as the remote DC is down	No, since the remote DC is down	Could not terminate as the remote DC is down	No Provides seamless access by creating a local session from the details available in the valid cookie

2.3 Setting Up a Multi-Data Center

The MDC feature is disabled by default. To set up an Access Manager MDC, start with an Access Manager cluster, set all MDC global configurations and designate the cluster as the Master Data Center. Following this, set up the Clone Data Center.

Note: Before beginning this procedure, ensure that you have completed the points documented in "Before Setting Up a Multi-Data Center."

The following sections document the process for setting up an MDC. They include running the commands documented in Section 2.6, "WLST Commands for Multi-Data Centers."

- Enabling the Master Data Center
- Setting Up the Clone Data Center

2.3.1 Enabling the Master Data Center

The following procedure contains more details.

1. Set up the primary Access Manager Data Center and designate it as the Master.

   A Master Data Center can be an existing Access Manager cluster or a vanilla installation.

   a. Make note of the clusterId.

      The Access Manager bootstrap assigns a unique `clusterId` to the Access Manager cluster. To set a custom clusterId, use the `setMultiDataCentreClusterName` WLST command documented in Section 2.6.7, "setMultiDataCentreClusterName."

   b. Enable Multi-Data Center mode by running the `enableMultiDataCentreMode` WLST command.

      `enableMultiDataCentreMode` sets an Access Manager cluster as Master, by default, and applies the global configurations. See Section 2.6.1, "enableMultiDataCentreMode" for details on the command and a sample properties file used for input. See Understanding the Primary Use Cases for details on the primary MDC scenarios.

      Note: To explicitly set the DC type as Master or Clone, use the `setMultiDataCenterType` WLST command documented in Section 2.6.5, "setMultiDataCenterType."

   c. Validate the MDC configuration by running the `validateMDCConfig` WLST command.

      See Section 2.6.8, "validateMDCConfig."

   d. Restart the Admin server.

2. Register and seed Partner Profiles for the Multi Data Centers.

   Each Data Center will use an Oracle Access Protocol (OAP) channel to fetch sessions from a remote DC. This runtime session sync can be initiated by a Master or Clone and thus each Data Center requires a WebGate agent profile (created using the Oracle Access Management Console) to be registered as a Partner Profile in the cluster.

   Note: These registrations can be performed in the Master DC and the configurations will be applied to all clones created using T2P process.

   This procedure will register two profiles, DCMaster and DCClone1. These profiles are used for the back channel OAP communication mentioned above. A WebGate profile should also be defined for the Master and all Clone Data Centers.

   a. Log in to the Oracle Access Management Console as System Administrator and click **Application Security** at the top of the window.

b. In the Application Security console, click **SSO Agent Registration** in the Quick Start Wizards section.

The SSO Agent Registration tab opens.

c. Configure the DCMaster agent profile by entering the required details including Version, Name (DCMaster) and Security Mode.

The security mode of the MDC partner profile should match the security mode defined for the Access Manager server. See Understanding Multi-Data Center Security Modes for details on the property files for each security mode. These files will be used as input when the `addPartnerForMultiDataCentre` WLST command is executed below.

d. Uncheck Auto Create Policies and click **Finish**.

A DCMaster tab is displayed. Ensure that the **Allow Management Operation** option is selected in the newly created agent profile.

e. Repeat the steps to create the DCClone1 agent profile.

f. Execute the `addPartnerForMultiDataCentre` WLST command to seed the DCMaster and DCClone1 agent profiles to the Master Data Center.

The WLST command would be run twice on the Master Data Center as follows:

```
addPartnerForMultiDataCentre(propfile= "/path/DCMaster.properties")
addPartnerForMultiDataCentre(propfile= "/path/DCClone1.properties"
```

Example 2–1 and Example 2–2 illustrate the property files used as input. The `RESTEndpoint` property takes a value of the HTTP or HTTPS endpoint of the Access Manager AdminServer used to invoke replication related REST services. HTTPS is preferred.

Example 2–1 DCMaster.properties for Master

```
remoteDataCentreClusterId=DC1

#webgate profile for session fetch created earlier and its password in DC1

oamMdcAgentId=DCMaster
AccessClientPasswd=secret
PrimaryHostPort=dc1.us.example.com:5575
SecondaryHostPort=dc1.example.com:5576
oamMdcSecurityMode=OPEN
trustStorePath=NA
keyStorePath=NA
globalPassPhrase=NA
keystorePassword=NA
agentVersion=11g
RESTEndpoint=https://<DCMaster Admin Server>:<port>
```

Example 2–2 DCClone1.properties for Clone

```
remoteDataCentreClusterId=DC2

#webgate profile for session fetch created earlier and its password in DC2

oamMdcAgentId=DCClone1
AccessClientPasswd=secret
PrimaryHostPort=dc2.us.example.com:5575
```

```
SecondaryHostPort=dc2.example.com:5576
oamMdcSecurityMode=OPEN
trustStorePath=NA
keyStorePath=NA
globalPassPhrase=NA
keystorePassword=NA
agentVersion=11g
RESTEndpoint=https://<DCClone1 Admin Server:port>
```

After running the WLST commands, the agent profiles will be recognized by the Master as the profiles to use for runtime session sync. See addPartnerForMultiDataCentre for more details.

Note: At this point in the procedure, the clone DC has not been setup but it is assumed that the service host and port information are known so can be defined in the `DCClone1.properties` file. If the information is not known, you can register the clone DC partner after the T2P process. In this case, only DCMaster will be registered now so the WLST command for the second DC has to be executed first in the Master DC and then in the clone DC after the T2P process is completed.

2.3.2 Setting Up the Clone Data Center

The Data Center set up in Enabling the Master Data Center is designated as the Master and will be cloned using T2P tools for any additional Data Centers. All configuration and policy changes are propagated from the Master to a Clone using the WLST commands provided as part of T2P Tooling. The T2P process is explained in the following documents.

- See *Oracle Fusion Middleware Administrator's Guide* for information on T2P when using WebLogic Server.
- See *Oracle Fusion Middleware Third-Party Application Server Guide for Oracle Identity and Access Management* for information on T2P when using Websphere Server.

1. Execute the following commands on the Master Data Center.

   Ensure that the AdminServer and all Managed Servers are running. The $T2P_HOME directory is just a location where all the artifacts of this process are saved.

```
$MIDDLEWARE_HOME/oracle_common/bin/copyBinary.sh -javaHome $JAVA_HOME
 -archiveLoc $T2P_HOME/oamt2pbin.jar
 -sourceMWHomeLoc $MIDDLEWARE_HOME
 -idw true
 -ipl $MIDDLEWARE_HOME/oracle_common/oraInst.loc
 -silent true
 -ldl $T2P_HOME/oam_cln_log;

$MIDDLEWARE_HOME/oracle_common/bin/copyConfig.sh -javaHome $JAVA_HOME
 -archiveLoc $T2P_HOME/oamt2pConfig.jar
 -sourceDomainLoc $DOMAIN_HOME
 -sourceMWHomeLoc $MIDDLEWARE_HOME
 -domainHostName admin-dc1.us.example.com
 -domainPortNum 7001
 -domainAdminUserName weblogic
 -domainAdminPassword $T2P_HOME/t2p_domain_pass.txt
 -silent true
 -ldl $T2P_HOME/oam_cln_log_config
```

```
 -opssDataExport true
 -debug true;
```

2. Copy the following files to the clone machine.

   The clone machine should not have any Oracle Access Management software installed on it.

```
$MIDDLEWARE_HOME/oracle_common/bin/pasteBinary.sh
$MIDDLEWARE_HOME/oracle_common/jlib/cloningclient.jar
$MIDDLEWARE_HOME/oracle_common/oraInst.loc
```

3. Execute the following commands on the Clone Data Center to copy all contents of $T2P_HOME directory from the master to the $T2P_HOME directory of the clone.

```
$T2P_HOME/pasteBinary.sh -javaHome $JAVA_HOME -al $T2P_HOME/oamt2pbin.jar
 -tmw $MIDDLEWARE_HOME -silent true -idw true -esp false
 -ipl $T2P_HOME/oraInst.loc -ldl $T2P_HOME/oam_cln_log_p
 -silent true

$MIDDLEWARE_HOME/oracle_common/bin/extractMovePlan.sh -javaHome $JAVA_HOME
 -al $T2P_HOME/oamt2pConfig.jar
 -planDirLoc $T2P_HOME/moveplan/
```

4. Edit the extracted Moveplan.xml on the Clone Data Center to provide relevant details.

   Note: Backup the original Moveplan.xml before editing.

```
cp $T2P_HOME/moveplan/moveplan.xml $T2P_
HOME/moveplan/moveplan.xml.org
```

   Each Clone Data Center will use a fresh set of OAM related schemas which need to be created using RCU in their respective databases. The new schema names and passwords need to be referenced in the moveplan.

```
$MIDDLEWARE_HOME/oracle_common/bin/pasteConfig.sh -javaHome $JAVA_HOME
 -archiveLoc $T2P_HOME/oamt2pConfig.jar
 -targetMWHomeLoc $MIDDLEWARE_HOME
 -targetDomainLoc $DOMAIN_HOME
 -movePlanLoc $T2P_HOME/moveplan/moveplan.xml
 -domainAdminPasswordFile $T2P_HOME/t2p_domain_pass.txt
 -ldl $T2P_HOME/oam_cln_log_paste_p
 -silent true
```

5. Use pack and unpack to copy managed servers on separate hosts.

   See *Oracle Fusion Middleware Creating Templates and Domains Using the Pack and Unpack Commands* for details.

6. Configure any or all Clone Data Centers as follows.

   a. Set a unique data center identifier for the clone DC using the `setMultiDataCentreClusterName` WLST command.

```
setMultiDataCentreClusterName(clusterName="DC2")
```

   Note: This step may be skipped if it was already done through the T2P process.

b. Set the type to Clone for the Clone DC.

```
setMultiDataCenterType(DataCenterType="Clone")
```

Optionally the configuration and policy updates can be disabled on the Clone by executing `setMultiDataCenterWrite(WriteEnabledFlag="false")`. After executing the command, the Clone becomes read only for policy and configuration artifacts. See setMultiDataCenterWrite for details.

7. Verify access to the Oracle Access Management Console and single sign-on between data centers.

To this point, one master and one clone are set up. Multiple clones can be set up similarly as required by repeating the cloning process. The above commands need to be executed for each of the clone DCs as per the topology using the appropriate cluster name each time.

2.4 Adding A Second Clone to An Existing Multi-Data Center Setup

To add another clone to an already existing MDC environment follow these steps. The specifics are documented in "Setting Up a Multi-Data Center."

1. Register a Partner Profile for the new Clone DC (DC3) on the Master DC.
2. Seed the data center partner on the Master DC.
3. Setup the new Clone DC.
4. Setup replication for the new Clone DC.
5. Customize with transformation rules if required.

2.5 Understanding Multi-Data Center Security Modes

A Multi-Data Center relies on the Oracle Access Protocol (OAP) channel for inter data center session management operations and back channel communication. The security mode of the MDC partner profile should match the security mode defined for the Access Manager server: OPEN, SIMPLE or CERT.

Note: An MDC partner profile is exposed by each data center and used by other data centers to communicate with it. Registering an MDC partner is a two step process. Consider an MDC with three data centers. In DC1, expose an MDC partner profile by creating a 10g or 11g WebGate (DC1_MDC_Partner). Then, register DC1_MDC_Partner in DC2 and DC3 using addPartnerForMultiDataCentre. See Section 2.6.3, "addPartnerForMultiDataCentre" for details.

The following sections have details about the security modes.

- OPEN Security Mode
- SIMPLE Security Mode
- CERT Security Mode

2.5.1 OPEN Security Mode

This is the default mode of the Access Manager deployment. No configuration is needed. The following is a sample input properties file for use with the `addPartnerForMultiDataCentre` WLST command.

```
remoteDataCentreClusterId=
 <CLUSTER ID OF REMOTE DC FOR WHICH THE AGENT IS BEING ADDED>
oamMdcAgentId=
 <AGENT ID OF THE REGISTERED PARTNER IN datacenter ABOVE>
PrimaryHostPort=<fully-qualified-host-name:OAM-port>
 for example:PrimaryHostPort=adc.example.com:5575
SecondaryHostPort=<fully-qualified-host-name:OAM-port>
 for example:SecondaryHostPort=adc.example.com:5577
AccessClientPasswd=<ACCESS CLIENT PASSWORD OF oamMdcAgentId IN datacenter>
oamMdcSecurityMode=OPEN
agentVersion=<WEBGATE AGENT VERSION 10g or 11g>
#NA ----> Not Applicable
trustStorePath=NA
keyStorePath=NA
globalPassPhrase=NA
keystorePassword=NA
```

2.5.2 SIMPLE Security Mode

Follow the instructions in Appendix C.5, "Configuring Simple Mode Communication with Access Manager" to set up the Access Manager servers in SIMPLE mode. In short, create an MDC partner profile in each of the member data centers in SIMPLE mode, and add it to each of the other data centers. The following is a sample input properties file for use with the `addPartnerForMultiDataCentre` WLST command.

```
remoteDataCentreClusterId=
 <CLUSTER ID OF REMOTE DC FOR WHICH THE AGENT IS BEING ADDED>
oamMdcAgentId=<AGENT ID OF THE REGISTERED PARTNER IN datacenter ABOVE>
PrimaryHostPort=<fully-qualified-host-name:OAM-port>
 for example:PrimaryHostPort=adc.example.com:5575
SecondaryHostPort=<fully-qualified-host-name:OAM-port>
 for example:SecondaryHostPort=adc.example.com:5577
AccessClientPasswd=<ACCESS CLIENT PASSWORD OF oamMdcAgentId IN datacenter>
oamMdcSecurityMode=SIMPLE
agentVersion=<WEBGATE AGENT VERSION 10g or 11g>

#Copy the oamclient-truststore.jks & oamclient-keystore.jks from
#<DOMAIN_HOME>/output/webgate-ssl/ from 'datacenter with cluster ID
#remoteDataCentreClusterId' above into the local DC say /scratch/MDCArtifacts/ and
#refer them in the below parameters

trustStorePath=</scratch/MDCArtifacts/oamclient-truststore.jks>
keyStorePath=</scratch/MDCArtifacts/oamclient-keystore.jks>

#Use the online WLST command displaySimpleModeGlobalPassphrase() to list
#the global passphrase in SIMPLE mode. Admins can also update this in the UI
#@ System Configuration-->Access Manager-->Access Manager Settings-->
#Access Protocol-->Simple Mode Configuration-->Global Passphrase.
#globalPassPhrase & keystorePassword are the same for SIMPLE mode

globalPassPhrase=<passphrase resulted in using the above steps>
keystorePassword=<same as globalPassPhrase>
```

2.5.3 CERT Security Mode

Follow the instructions in Appendix C.4, "Configuring Cert Mode Communication for Access Manager" to set up the Access Manager servers in CERT mode. In short, create an MDC partner in each of the member data centers in CERT mode, and generate the 'clientTrustStore.jks' and 'clientKeyStore.jks' keystores to be used by the MDC partner using the following procedure.

1. Run the following openssl command from a Linux command prompt to generate aaa_key.pem & aaa_req.pem.

   ```
   openssl req -new -keyout aaa_key.pem -out aaa_req.pem -utf8
   ```

   Use the certreq command to generate the certificate and chain.

2. Create aaa_cert.pem using the following procedure.

   a. Open aaa_req.pem in a text editor and copy the contents.

      Exclude the trailing spaces from your selection.

   b. Paste the copied text into Signcsr.

      Include [-----BEGIN CERTIFICATE REQUEST----- and -----END CERTIFICATE REQUEST-----].

   c. Copy the output into a text editor and save it as aaa_cert.pem.

3. Create aaa_chain using the following procedure.

   a. Open certreq.

   b. Click on chain.pem and copy/paste the contents into a text editor and save it as aaa_chain.pem.

      Excluding traiing and leading spaces from your selection.

4. Encrypt the private key (aaa_key.pem) using the following command.

   ```
   openssl rsa -in aaa_key.pem -passin pass: -out aaa_key.pem -passout
   pass:Welcome1 -des
   ```

   The password used in this command must be defined as the access client password or agent key password while registering the MDC partner.

5. Copy aaa_key.pem, aaa_cert.pem and aaa_chain.pem to a temporary location.

   For example, /tmp/clientCertArtifacts/

6. Convert aaa_cert.pem and aaa_key.pem into DER format using one of the following commands.

   ```
   -openssl x509 -in /tmp/clientCertArtifatcs/aaa_cert.pem -inform PEM -out
    /tmp/clientCertArtifatcs/aaa_cert.der -outform DER;

   -openssl pkcs8 -topk8 -nocrypt -in /tmp/clientCertArtifatcs/aaa_key.pem
    -inform PEM -out /tmp/clientCertArtifatcs/aaa_key.der -outform DER;
   ```

7. Import the aaa_key.der and aaa_cert.der into clientKeyStore.jks; and the aaa_chain.pem into clientTrustStore.jks with the below steps

   ```
   -cd $IDM_HOME/oam/server/tools/importcert/;

   -unzip importcert.zip;

   -java -cp importcert.jar
    oracle.security.am.common.tools.importcerts.CertificateImport -keystore
   ```

```
 /tmp/clientCertArtifatcs/clientKeyStore.jks -privatekeyfile
 /tmp/clientCertArtifatcs/aaa_key.der -signedcertfile
 /tmp/clientCertArtifatcs/aaa_cert.der -storetype jks -genkeystore yes

-keytool -importcert -file /tmp/clientCertArtifatcs/aaa_chain.pem -trustcacerts
 -keystore /tmp/clientCertArtifatcs/clientTrustStore.jks -storetype JKS
```

Enter the keystore passwords when prompted. The password needs to be defined in the input properties file for the addPartnerForMultiDataCentre WLST command as well.

8. If not done when creating the certificates for the WebGate, import the aaa_key.der and aaa_cert.der formatted certificates into the .oamkeystore using the same Oracle provided importcert.jar used inthe previous step.

```
-java -cp importcert.jar
 oracle.security.am.common.tools.importcerts.CertificateImport
 -keystore /scratch/Oracle/Middleware/domains/
 base_domain/config/fmwconfig/.oamkeystore -privatekeyfile
 /tmp/clientCertArtifacts/aaa_key.der -signedcertfile
 /tmp/clientCertArtifacts/aaa_cert.der -alias mycertmode1 -storetype JCEKS
```

alias is the alias name defined when setting CERT mode in Access Manager.

The following is a sample input properties file for use with the addPartnerForMultiDataCentre WLST command.

```
remoteDataCentreClusterId=
 <CLUSTER ID OF REMOTE DC FOR WHICH THE AGENT IS BEING ADDED>
oamMdcAgentId=<AGENT ID OF THE REGISTERED PARTNER IN datacenter ABOVE>
PrimaryHostPort=<fully-qualified-host-name:OAM-port>
 for example:PrimaryHostPort=adc.example.com:5575
SecondaryHostPort=<fully-qualified-host-name:OAM-port>
 for example:SecondaryHostPort=adc.example.com:5577
AccessClientPasswd=<ACCESS CLIENT PASSWORD OF oamMdcAgentId IN datacenter>
oamMdcSecurityMode=CERT
agentVersion=<WEBGATE AGENT VERSION 10g or 11g>

trustStorePath=</tmp/clientCertArtifatcs/clientTrustStore.jks >
keyStorePath=</tmp/clientCertArtifatcs/clientKeyStore.jks >

globalPassPhrase=NA

#use keystore password used for generating keystore in the previous step
keystorePassword=<keystore password given while generating keystore>
```

2.6 WLST Commands for Multi-Data Centers

The following WebLogic Scripting Tool (WLST) commands are specific to Multi-Data Center deployment. More information is in the following sections.

- enableMultiDataCentreMode
- disableMultiDataCentreMode
- addPartnerForMultiDataCentre
- removePartnerForMultiDataCentre
- setMultiDataCenterType
- setMultiDataCenterWrite

- setMultiDataCentreClusterName
- validateMDCConfig
- exportAccessStore
- importAccessStore

See the *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference* for information on the WebLogic Scripting Tool.

2.6.1 enableMultiDataCentreMode

Online command used to enable Multi-Data Center mode.

2.6.1.1 Description

This command enables Multi-Data Center mode. It takes a value equal to the full path to, and name of, the MDC.properties file.

Note: Setting the SSO Token version to 5 is not supported from the administration console. To do this, modify the Access Manager Settings page and run the `enableMultiDataCentreMode` WLST command to set.

2.6.1.2 Syntax

```
enableMultiDataCentreMode(propfile="../MDC_properties/oamMDCProperty.properties")
```

Argument	Definition
propfile	Mandatory. Takes a value equal to the full path to, and name of, the `oamMDCProperty.properties` file. Table 2–2 documents the properties that comprise `the file`. Example 2–3 (following the table) is a sample oamMDCProperty.properties file.

Table 2–2 oamMDC.properties Properties

Property	Definition
SessionMustBeAnchoredToDataCenterServicing User	Takes a value of True (Invalidate) or False (No Invalidation).
SessionDataRetrievalOnDemand	Takes a value of True (Cross DC retrieval) or False (No). Data retrieval can be turned off without disabling MDC. If False, session data is not transferred but SSO is still performed as the user moves across DCs. NOTE: SessionDataRetrievalOnDemand must be set to False when deploying in Co-existence mode. See *Oracle Fusion Middleware Migration Guide for Oracle Identity and Access Management* for information on co-existence scenarios.
Reauthenticate	Takes a value of True (force reauthentication) or False (No forced reauthentication).
SessionDataRetrievalOnDemandMax_retry_ attempts	Takes a value equal to a binary that represents the number of times to retry data retrieval when it fails. Default is 2.

Table 2–2 (Cont.) oamMDC.properties Properties

Property	Definition
SessionDataRetrievalOnDemandMax_conn_wait_time	Takes a value equal to a binary that represents the total amount of time in seconds to wait for a connection. Default is 1000.
SessionContinuationOnSyncFailure	Decides the session adoption action on fail over. When set to 'true', the session will continue on the DC servicing the current request even though the parent DC is down/not reachable. The session will be created in the DC servicing the current request from the mandatory minimal information available in the incoming token. When set to 'false', the user will be challenged on fail-over scenarios.
MDCGitoCookieDomain	Specifies the domain with which the OAM_GITO cookie should be set. In MDC deployments where a common cookie domain hierarchy cannot be derived, this setting should be commented or removed as described in Inactivity time outs scenario.

Example 2–3 Sample oamMDCProperty.properties File

```
SessionMustBeAnchoredToDataCenterServicingUser=false
SessionDataRetrievalOnDemand=true
Reauthenticate=true
SessionDataRetrievalOnDemandMax_retry_attempts=3
SessionDataRetrievalOnDemandMax_conn_wait_time=80
SessionContinuationOnSyncFailure=true

#MDCGitoCookieDomain=.example.com <This setting should be provided only if there
is a common cookie subdomain across the WGs and DCs>
```

2.6.1.3 Example

The following command enables this data center.

```
enableMultiDataCentreMode(propfile="../MDC_properties/oamMDCProperty.properties")
```

2.6.2 disableMultiDataCentreMode

Online command used to disable Multi-Data Center mode.

2.6.2.1 Description

This command disables Multi-Data Center mode.

2.6.2.2 Syntax

```
disableMultiDataCentreMode()
```

There are no arguments for this command.

2.6.2.3 Example

The following command disables Multi-Data Center mode.

```
disableMultiDataCentreMode()
```

2.6.3 addPartnerForMultiDataCentre

In an MDC deployment with *n* number of Data Centers, each Data Center has a registered partner to communicate with each of the other (n-1) Data Centers. This makes the total number of partner registrations (n) x (n-1). This online command is used to add a partner for inter Data Center OAP communication.

Note: An MDC partner profile is exposed by each data center and used by other data centers to communicate with it. Registering an MDC partner is a two step process. Consider an MDC with three data centers. In DC1, expose an MDC partner profile by creating a 10g or 11g WebGate (DC1_MDC_Partner). Then, register DC1_MDC_Partner in DC2 and DC3 using addPartnerForMultiDataCentre. See Section 2.6.3, "addPartnerForMultiDataCentre" for details.

2.6.3.1 Description

This command adds a partner to the Data Center. It takes a value equal to the full path to, and name of, the `partnerInfo.properties` file.

2.6.3.2 Syntax

```
addPartnerForMultiDataCentre(propfile="../MDC_properties/partnerInfo.properties")
```

Argument	Definition
propfile	Mandatory. Takes a value equal to the path to, and name of, the partnerInfo.properties file.
RESTEndpoint	Optional. Takes as a value the HTTP/HTTPS URL from which the Access Manager REST services can be accessed.

Table 2–3 documents the properties that comprise `partnerInfo.properties`. See Understanding Multi-Data Center Security Modes for properties file samples.

Table 2–3 partnerInfo.properties Properties

Property	Definition
remoteDataCentreClusterId	Cluster id of the remote Data Center with which the OAP communication needs to be established.
oamMdcAgentId	Partner ID of the registered partner profile in the remote Data Center. The "allow management operations" flag for this partner should be set in the remote Data Center.
PrimaryHostPort	Takes a *fully-qualified-host-name*:*OAM-port* for the primary Access Manager server corresponding to the remote DC identified by remoteDataCentreClusterId; for example: PrimaryHostPort=abc.example.com:5575

Table 2–3 (Cont.) partnerInfo.properties Properties

Property	Definition
SecondaryHostPort	Takes a *fully-qualified-host-name*:*OAM-port* for the secondary Access Manager server corresponding to the remote DC identified by remoteDataCentreClusterId; for example: SecondaryHostPort=abc.example.com:5577 Consider an OAM MDC member Data Center with two managed servers at abc.example.com with ports as follows: oam_server1 (5575) and oam_server2 (5577). High availability/failover of the OAP SDK partner can be achieved by setting the PrimaryHostPort and SecondaryHostPort as below. PrimaryHostPort=abc.example.com:5575 SecondaryHostPort=abc.example.com:5577
AccessClientPasswd	The access client password of the MDC partner registered in the remote Data Center.
oamMdcSecurityMode	Defines the MDC security mode. Takes a value of OPEN/SIMPLE/CERT. (CERT Mode is preferred, SIMPLE is fine but OPEN is discouraged.) For SIMPLE and CERT modes, the following values should be supplied appropriately. For OPEN mode, these values are not applicable. See Understanding Multi-Data Center Security Modes.
agentVersion	Valid agent version 11g/10g.
trustStorePath	Absolute path to the truststore file [SIMPE/CERT].
keyStorePath	Absolute path to the keyStore file [SIMPLE/CERT].
globalPassPhrase	Global passphrase set during the partner registration [SIMPLE/CERT].
keystorePassword	Key store password set during partner configuration [SIMPLE/CERT].

2.6.3.3 Example

The following command defines this data center as a Master.

```
addPartnerForMultiDataCentre(propfile="../MDC_properties/partnerInfo.properties")
```

2.6.4 removePartnerForMultiDataCentre

Online command used to remove a registered remote partner from the Data Center configuration.

2.6.4.1 Description

This command removes a registered remote partner from a configured Data Center. It takes a value equal to a valid remoteDataCentreClusterId.

2.6.4.2 Syntax

```
removePartnerForMultiDataCentre=("<cluster_ID>")
```

Argument	Definition
cluster_ID	Mandatory. Takes a string value equal to the cluster ID.

2.6.4.3 Example

The following command defines the partner to be removed.

```
removePartnerForMultiDataCentre("99bf9-adc2120609")
```

2.6.5 setMultiDataCenterType

Online command used to set the type of data center - either Master or Clone.

2.6.5.1 Description

In an MDC deployment one Data Center is designated as the Master and the others as a Clone. Essentially all MDC wide global configurations and policy updates should be applied to the Master and propagated to the Clones using the supported T2P commands. This command sets the type of the data center. Values are Master or Clone.

2.6.5.2 Syntax

```
setMultiDataCenterType(DataCenterType="<Master|Clone>")
```

Argument	Definition
DataCenterType	Mandatory. Takes a string value of Master or Clone.

2.6.5.3 Example

The following command defines this data center as a Master.

```
setMultiDataCenterType(DataCenterType="Master")
```

2.6.6 setMultiDataCenterWrite

Online command used to set write protection for modifications to system and policy configurations on the Clone Data Center.

2.6.6.1 Description

A Clone Data Center can be write protected by setting `WriteEnabledFlag` to false. In this case, the Clone Data Center will not allow updates through the Oracle Access Management Console or WLST commands. Data synchronization will still continue to update as the command is used to write protect the Clone Data Center against accidental updates after the initial set up is complete.

2.6.6.2 Syntax

```
setMultiDataCenterWrite(WriteEnabledFlag="<true|false>")
```

Argument	Definition
WriteEnabledFlag	Mandatory. Takes a string value of true or false.

2.6.6.3 Example

The following example protects the Clone Data Center from accidental overwrites.

```
setMultiDataCenterWrite(WriteEnabledFlag = "false")
```

2.6.7 setMultiDataCentreClusterName

Online command to set the cluster name of the Data Center to the supplied string.

2.6.7.1 Description

This command sets the Multi-Data Center cluster name. Value is a string.

2.6.7.2 Syntax

```
setMultiDataCentreClusterName(clusterName="<string_value>")
```

Argument	Definition
clusterName	Mandatory. Takes a string equal to the cluster name.

2.6.7.3 Example

The following command enables this data center.

```
setMultiDataCentreClusterName(clusterName="MyCluster")
```

2.6.8 validateMDCConfig

Online command used to insure the Multi-Data Center configuration is correct.

2.6.8.1 Description

This command validates that the required entries in the Multi-Data Center configuration are present in `oam-config.xml`. For the MDC solution, a new Access Manager event named `mdc_session_update` is required to create or update MDC sessions during authorization. The Access Manager event model requires a set of configurations to be present in the `oam-config.xml` configuration file. The required configurations cannot be added statically so `validateMDCConfig` validates the required entries for `mdc_session_update` and seeds any configurations not already present.

2.6.8.2 Syntax

```
validateMDCConfig()
```

There are no arguments for this command.

2.6.8.3 Example

The following command validates the MDC configuration.

```
validateMDCConfig()
```

2.6.9 exportAccessStore

Online command to create a ZIP file of the Master Data Center UDM metadata.

2.6.9.1 Description

This command will create a ZIP file of the Master Data Center UDM metadata.

2.6.9.2 Syntax

```
exportAccessStore(toFile="<name and location of ZIP", namePath="/")
```

2.6.9.3 Example

```
exportAccessStore(toFile="/master/location/dc1metadata.zip", namePath="/")
```

2.6.10 importAccessStore

Online command to import a ZIP file of the Master Data Center UDM metadata to a Clone Data Center.

2.6.10.1 Description

This command will import a ZIP file of the Master Data Center UDM metadata to the Clone Data Center.

2.6.10.2 Syntax

```
importAccessStore(fromFile="<name and location of ZIP", namePath="/")
```

2.6.10.3 Example

```
importAccessStore(fromFile="/master/location/dc1metadata.zip", namePath="/")
```

3

Synchronizing Data In A Multi-Data Center

The Multi-Data Center infrastructure can be configured to keep Access Manager data synchronized across multiple data centers. This can be done using the Automated Policy Synchronization Replication Service or data can be replicated manually.

The following sections contain details on the replication procedures.

3.1 Understanding the Multi-Data Center Sync

The Multi-Data Center (MDC) infrastructure can be configured to keep Access Manager data synchronized across multiple data centers. This release introduces Automated Policy Synchronization (APS), a replication mechanism that removes administrator and manual intervention from the data synchronization process. Policy, system configuration and partner metadata are all synchronized with APS.

> **Note:** T2P tooling is required for creating a Clone (also referred to as a Consumer) from a Master (also referred to as a Supplier). Once the MDC infrastructure is deployed (as documented in Chapter 2, "Configuring Multi-Data Centers"), APS can be enabled to automatically sync any changes from the Master to the Clones.

APS (also referred to as the Replication Service) is a set of REST API. The binaries are installed as part of the Access Manager application and deployed in the AdminServer. It is disabled by default but can be enabled by setting the `oracle.oam.EnableMDCReplication` property to true. After enabling the service, create a pull model Replication Agreement between the Clone data center and the Master. The Clone polls for changes from the Master as long as the Replication Agreement is valid for it. Conversely, the Master will respond to the Clone's request as long as it finds a valid replication agreement. The Clone applies the changes locally

> **Note:** APS is optional; administrator-initiated import and export based replication is still available using the T2P tooling and WLST command procedure used previous to this release.

When setting up the Replication Service, the following may occur:

- Establishment of a Replication Agreement with the registration of one data center as a replication Clone and another in a separate geographical location as its Master; the changes are pulled from the Master and applied to the Clone.
- Definition of data center specific configurations which may not be replicated across data centers.
- Tracking of Access Manager configuration changes in each data center and querying the current replication state in any of the data centers.
- Generation of a changelog which can be applied in the context of a similar setup running in another data center.
- Trigger of a pull from the Master data center if there is a need; for example, if automated replication fails.
- Replication of Access Manager configuration artifacts in the Master-Clone model.

> **Note:** APS does not sync IDS Profiles, OAM Keystores and Oracle Platform Security Services artifacts (jps-config.xml changes, credential store configuration and the like).

The following sections contain additional details.

- How Replication Works
- Understanding the Replication Agreement
- Manually Syncing Data in a Multi-Data Center

3.1.1 How Replication Works

Replication works in a Master-Clone topology. In this topology, multiple Clones pull changes from a single Master. One Data Center is defined by the administrator as the Master and one or more other Data Centers are Clones. The administrator makes changes to the Master that are replicated to the Clones. Only Master to Clone replication is supported; changes to Clones are not replicated back to the Master.

> **Note:** Multi-master replication is not supported.

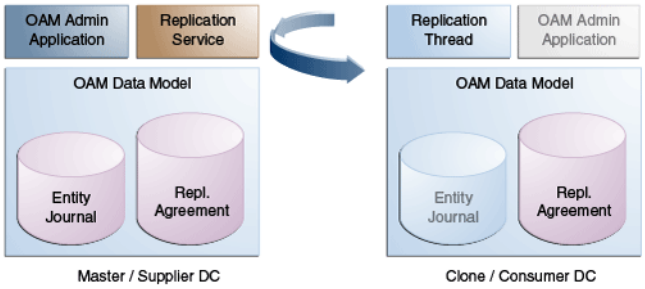
To partake in replication, the Master data center (initiator of the replication) and the Clone data center (receiver of the changes) must have a Replication Agreement stored in the Access Manager data store. Table 3–1 documents the states in which replication can be deployed.

Table 3–1 Replication States

State	Definition
Active	An Access Manager domain (including Admin and managed servers) is setup to serve access requests. In an active state, the Access Manager server provides the web access management functionality without additional MDC features.
Bootstrapping	This state is optional for some Data Centers; for example, the first one in an MDC topology. A Data Center goes through this intermediate state when added to an existing MDC topology. The new DC contacts the master and bootstraps itself to the same state. The bootstrap includes synchronizing the server keys, policy artifacts, partners, and system configuration. After completion of bootstrapping, the DC will be Replication Ready.
Replication Ready	In this state, MDC is enabled, the DC is made part of the topology, and the replication service is enabled. Once enabled, a clone can be registered with the master via a Replication Agreement. Once established, clone DCs can query and start pulling changelogs from the master.

Figure 3–1 illustrates the replication flow.

Figure 3–1 Replication Flow



Flow diagram illustrating Replication

****

Each Clone pulls changes from the Master. A replication thread runs on the Clone after the pre-configured interval of time and fetches changes from the Replication Service (REST endpoint) running on the Master environment.

For every cloned environment, the Master keeps track of a change sequence number indicating when it was last synced. The Master also keeps track of the list of changes (Create/Update/Delete) that have been pulled by the Clones in a changelog using specific change sequence numbers. When updating configuration metadata, the Clone can also change environment specific parameter values depending on transformation rules. See Using and Customizing Transformation Rules for details.

3.1.2 Understanding the Replication Agreement

Configuration changes (defined as *journals*) are replicated from a Master node to Clone nodes. On receiving the journals, each node updates it's configuration to match the journal and remain in a synchronized state. The nodes, though, need to enter a Replication Agreement to receive change journals.

When a new data center is added to an existing MDC topology, it has to bootstrap itself to be in sync with the existing data centers. This bootstrap operation will get the current Access Manager policies, system configuration, partner metadata and server keys for the existing MDC topology. After the bootstrap operation, the new data center captures the last change sequence number from the topology's Master so that during replication it can be used to determine the current state.

Note: Automated bootstrap is the ideal scenario but you can execute T2P tooling first to ensure the Master and Clones are in the same state.

To establish a Replication Agreement, the Clone data center must know the Master's changelog sequence number. If the data center is added to the topology on 'day 0' and the Replication Agreement was created on 'day 1', there is a need to bootstrap again. To avoid this and to keep the flow simple, creating a Replication Agreement should take care of the bootstrap and actual replication agreement creation. The steps to create the Replication Agreement are documented in Section 2.3, "Setting Up a Multi-Data Center."

3.1.3 Manually Syncing Data in a Multi-Data Center

Data in an MDC topology can also be synced manually. The manual option uses WLST export and import calls to migrate the data from one data center to another. Partner profiles and policies should both be exported and imported using WLST. The WLST commands are documented in *Oracle Fusion Middleware WebLogic Scripting Tool Command Reference for Identity and Access Management*.

3.2 Enabling Data Replication

The following procedures will enable data replication on the Master and Clone Data Centers.

1. Enable Replication for the Master Data Center using the following procedure.
   a. Set the `oracle.oam.EnableMDCReplication` VM property to `true` in the `setDomainEnv.sh/cmd` file on the Master Data Center machine.

      ```
      -Doracle.oam.EnableMDCReplication=true
      ```

   b. Stop and start the Access Manager AdminServer.
   c. Validate that the REST endpoints are enabled by running the following command.

      ```
      curl -u <user> 'https://dc1-admin.example.com:7002/
         oam/services/rest/_replication/hello
      ```

2. Enable Replication for the Clone Data Center(s) using the following procedure.
   a. Set the `oracle.oam.EnableMDCReplication` VM property to `true` in the `setDomainEnv.sh/cmd` file on the Clone Data Center machine.

      ```
      -Doracle.oam.EnableMDCReplication=true
      ```

Figure 3–2 EnableMDCReplication Java Property

```
#Added -Doracle.oam.EnableMDCReplication=true below to enable APS configuration by Kiran

EXTRA_JAVA_PROPERTIES=" -Doracle.oam.EnableMDCReplication=true -DCONFIG_DS=jdbc/oamds -DCONFIG_HISTORY=true -Doam.oes.new=true -DOAM_POLICY_FILE=${DOMA
IN_HOME}/config/fmwconfig/oam-policy.xml -DOAM_CONFIG_FILE=${DOMAIN_HOME}/config/fmwconfig/oam-config.xml -DOAM_ORACLE_HOME=${OAM_ORACLE_HOME} -Doracle
.security.am.SERVER_INSTNCE_NAME=${SERVER_NAME} -Does.jars.home=${OAM_ORACLE_HOME}/server/lib/oes-d8 -Does.integration.path=${OAM_ORACLE_HOME}/server/l
ib/oeslib/oes-integration.jar -Does.enabled=true  -Djavax.xml.soap.SOAPConnectionFactory=weblogic.wsee.saaj.SOAPConnectionFactoryImpl -Djavax.xml.soap.
MessageFactory=oracle.j2ee.ws.saaj.soap.MessageFactoryImpl -Djavax.xml.soap.SOAPFactory=oracle.j2ee.ws.saaj.soap.SOAPFactoryImpl ${EXTRA_JAVA_PROPERTIE
S}"
export EXTRA_JAVA_PROPERTIES
```

Add EnableMDCReplication Java property

****

b. Stop and start the Access Manager AdminServer.

c. Validate that the REST endpoints are enabled by running the following command.

```
curl -u <user> 'https://dc1-admin.example.com:7002/
   oam/services/rest/_replication/hello
```

d. Repeat this process on all Clone Data Centers.

3.3 Syncing Master and Clone Metadata

The process for syncing metadata across an MDC involves first syncing Access Manager UDM metadata and then creating a replication agreement (as discussed in Understanding the Replication Agreement). The procedures are documented in the following sections.

- Syncing the UDM Metadata
- Creating the Replication Agreement
- Modifying the Replication Agreement

3.3.1 Syncing the UDM Metadata

It is required to sync Access Manager UDM metadata stored in the Master to all Clones and this step must be executed before creating the replication agreement.

1. Execute the `exportAccessStore` WLST command on the Master Data Center to create a ZIP file containing the UDM metadata.

```
exportAccessStore(toFile="/master/location/dc1metadata.zip",
   namePath="/")
```

2. Copy `dc1metadata.zip` to the Clone DC location.
3. Execute the `importAccessStore` WLST command on the Clone Data Center to import the UDM metadata.

```
importAccessStore(fromFile="/clone/location/dc1metadata.zip",
   namePath="/")
```

4. Repeat on all Clone DCs.

3.3.2 Creating the Replication Agreement

This is a one time operation which will enable the Clone DCs to pull changes from the master DC. The replication agreement can be created using any REST client. In this procedure, we use the standard curl utility. This command will:

- Insert an entry in the Master's Replication Agreement store containing details regarding the Clone that wants to pull changes.
- Insert an entry in the Clone's Replication Agreement store containing details regarding the Master from which it will pull changes. Replication configuration values like the poll interval will also be set.

1. Ensure the Master and Clone DC REST endpoints are up and running.
2. Execute the following command on the Master DC.

   This command will use the `repluser` specified for replication queries from the Master to the Clone. repluser is expected to be available in the default identity stores for all involved DCs.

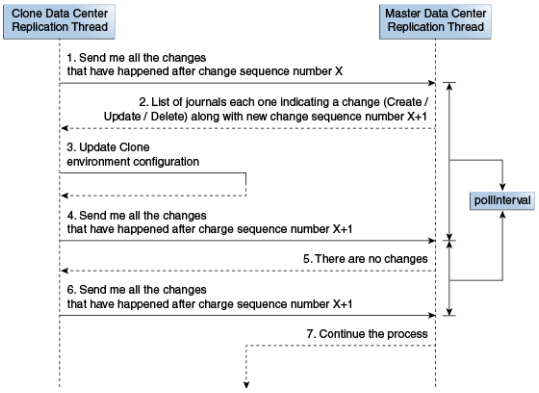
   ```
   curl -u <repluser> -H 'Content-Type: application/json' -X POST
    'https://supplier.example.com:7002/oam/services/rest/
     _replication/setup' -d '{"name":"DC12DC2",
    "source":"DC1","target":"DC2","documentType":"ENTITY"}'
   ```

   The following is an example of output for the command.

   ```
   {"enabled":"true","identifier":"201409231329353668","ok":"true",
     "pollInterval":"900","startingSequenceNumber":"110","state"
     :"READY"}
   ```

   Be sure to note the values of the replication identifier, pollInterval and startingSequence Number. The identifier is a reference specific to this Replication Agreement and is used for replication related queries. The pollInterval is a value (in seconds) after which the Clone will poll for changes against the Master. (Typically policy and configuration are not changed often so this number can be as high as the default value of 900 seconds.) The startingSequenceNumber is the value before which all records will be unavailable. In the example, all records before the value of 110 are unavailable. It is implicit that bootstrapping happened before creating the Replication Agreement thus the Clone can start pulling changes from sequence number 110. The Clone also has an entry created in its local replication table which keeps track of the last sequence number. The starting sequence process is illustrated in Figure 3–3.

Figure 3–3 Starting Sequence Illustrated



Starting sequence process illustrated

****

The create replication agreement command will return details of an already existing replication agreement if applicable. In this case, the value of ok will be false.

```
{"enabled":"true","identifier":"201409231329353668","ok":"false",
  "pollInterval":"900","startingSequenceNumber":"110",
  "state":"READY"}
```

Note: If a specific user needs to be used for replication, the user's credentials can be provided in the command in the format "BASIC base64(user:password)".

For example, "BASIC base64(weblogic:welcome1)" is specified as "BASIC d2VibG9naWM6d2VsY29tZTE=" in the following command.

```
curl -u <repluser> -H 'Content-Type: application/json' -X POST
  'https://supplier.example.com:7002/oam/services/rest/
  _replication/setup' -d
  '{"source":"DC1","target":"DC2","documentType":"ENTITY","config":
  {"entry":{"key":"authorization","value":"BASIC
  d2VibG9naWM6d2VsY29tZTE="}}}''
```

Basic Authorization is supported for replication REST API.

3. Restart the Master and Clone AdminServers.

   Once the replication agreement is created and the AdminServers restarted, the Clone will start polling for changes. The default poll interval is '900' seconds or 15 minutes. The poll interval can be changed by executing an edit replication agreement command. For example, the following command will change the polling interval to 60 seconds. Restart the Clone AdminServer after running the command.

```
curl -u <repluser> -H 'Content-Type: application/json' -X PUT
  'https://supplier.example.com:7002/oam/services/rest/
  _replication/201409231
  329353668' -d '{"pollInterval":"60","replicaType":"CONSUMER"}''
```

To query the details of a Clone's replication agreement (including the polling interval), use the following command.

```
curl -u <repluser>
  'https://supplier.example.com:7002/oam/services/rest/_replication/201409231
  329353668?type=consumer'
```

The output would be similar to the following.

```
{"enabled":"true","identifier":"201409231329353668","ok":"true",
  "pollInterval":"60","startingSequenceNumber":"110","state":"READY"}
```

To query the details of a Master's replication agreement (including the polling interval), use the following command.

```
curl -u <repluser>
  'https://supplier.example.com:7001/oam/services/rest/_replication/201409231
  329353668'
```

The output would be similar to the following. (The poll Interval of the Master's replication agreement does not affect the actual replication.)

```
{"enabled":"true","identifier":"201409231329353668","ok":"true",
  "pollInterval":"3600","startingSequenceNumber":"110","state":"ACTIVE"}
```

In R2PS3, the following command can also be used on the Master or the Clone to get details of any replication agreements. In cases where the replication agreement identifier is unknown, this command can be used to list all the replication agreement identifiers for input in the previous commands.

```
curl -k -u weblogic
'https://oamadmin.example.com:7002/oam/services/rest/_replication/agreements'

Sample output 1:
{"featureEnabled":"true","identifiers":"201411211137273612","ok":"true"}

Sample output 2:
{"featureEnabled":"true","identifiers":["201411211137273612","201411211137273900"]
,"ok":"true"}
```

To remove a replication agreement, first disable it on the Clone side, then disable it on the Master side and then delete it on both sides. The following commands illustrate this process.

```
curl -u <repluser> -H 'Content-Type: application/json' -X PUT
  'https://supplier.example.com:7002/oam/services/rest/_replication/201409231
  329353668' -d '{"enabled":"false","replicaType":"CONSUMER"}''

curl -u <repluser> -H 'Content-Type: application/json' -X PUT
  'https://supplier.example.com:7002/oam/services/rest/_replication/201409231
  329353668' -d '{"enabled":"false","replicaType":"SUPPLIER"}''

curl -u weblogic:welcome1 -H 'Content-Type: application/json' -X DELETE
  'https://supplier.example.com:7001/oam/services/rest/_replication/201409231
  329353668'
```

3.3.3 Modifying the Replication Agreement

Using the Replication Agreement identifier, changes can be made to the Replication Agreement configuration. In this example, the value of pollInterval will be changed to 60 seconds.

Service responds back with JSON object that is the status of replication agreement before making the change. You need to fetch replication agreement status again to see updated configuration.

1. Execute the following command to get the current status of the Replication Agreement.

   ```
   curl -u weblogic:****** -H 'Content-Type: application/json'
     'http://oam1-nyc.example.com:7001/oam/services/rest/
     _replication/201409040157218184'
   ```

   The JSON response would be:

   ```
   {"enabled":"true","identifier":"201409040157218184?,"ok":"true",
     "pollInterval":"3600?,"startingSequenceNumber":"101?,"state":"ACTIVE"}
   ```

2. Execute the following command to modify the value of pollInterval.

   ```
   curl -u weblogic:****** -H 'Content-Type: application/json'
     -X PUT 'http://oam1-nyc.example.com:7001/oam/services/rest/
     _replication/201409040157218184'
   ```

```
  -d '{"pollInterval":"60?,"replicaType":"consumer"}'
```

The JSON response would be:

```
{"enabled":"true","identifier":"201409040157218184?,"ok":"true","pollInterval":
"3600?,"startingSequenceNumber":"101?,"state":"ACTIVE"}
```

3. Restart the AdminServer on both Master and Clone machines.
4. Execute the following command to get the current status of the Replication Agreement.

   This will validate that the change has been made. Note the value of pollInterval in the JSON Response is different from the value returned in the first step of this procedure.

```
curl -u weblogic:****** -H 'Content-Type: application/json'
  'http://oam1-nyc.example.com:7001/oam/services/rest/
  _replication/201409040157218184'
```

   The JSON response would be:

```
{"enabled":"true","identifier":"201409040157218184?,"ok":"true",
  "pollInterval":"60?,"startingSequenceNumber":"101?,"state":"ACTIVE"}
```

3.4 Using and Customizing Transformation Rules

Transformation rules are used by APS. The transformation rules illustrated in Example 3–1 are the default rules provided by Access Manager. A Clone can be configured to override these OOTB rules. This section documents how some of these rules can be modified and how to configure Access Manager to recognize these custom rules.

Example 3–1 Default Transformation Rules

```
<?xml version="1.0" encoding="UTF-8"?>
<mdc-transform-rule>
 <changes-to-include entity-path="/policy"/>
 <changes-to-include

entity-path="/config/NGAMConfiguration/DeployedComponent/Agent/WebGate/Instance">
  <replace attribute-match="/*/PrimaryServerList/*/host" value-match="(.*)">
   <replace-with
n="1">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverhost}</replace-with>
  </replace>
  <replace attribute-match="/*/UserDefinedParameters/logoutRedirectUrl"
   value-match="(.*)://(.*):(.*)/oam/server/logout">
  <replace-with
n="1">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverprotocol}</replace-with>
  <replace-with
n="2">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverhost}</replace-with>
  <replace-with
n="3">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverport}</replace-with>
  </replace>
  <replace attribute-match="/*/logoutRedirectUrl"
   value-match="(.*)://(.*):(.*)/oam/server/logout">
```

```
  <replace-with
n="1">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverprotocol}</replace-with>
  <replace-with
n="2">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverhost}</replace-with>
  <replace-with
n="3">${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMS
erverProfile/OAMSERVER/serverport}</replace-with>
 </replace>
</changes-to-include>
<changes-to-include entity-path=
"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/Authenticat
ionModules"/>
<changes-to-include entity-path=
"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/oamproxy"/>
<changes-to-include entity-path=
"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/Sme/Session
Configurations"/>
<changes-to-include entity-path=
"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/OAMServerPr
ofile/OAMSERVER">
<ignore attribute-match="/serverprotocol"/>
<ignore attribute-match="/serverhost"/>
<ignore attribute-match="/serverport"/>
</changes-to-include>
<changes-to-include
entity-path="/config/NGAMConfiguration/DataCenterConfiguration/Cluster">
<ignore attribute-match="/DataCenterType"/>
<ignore attribute-match="/ClusterId"/>
<ignore attribute-match="/WriteEnabledFlag"/>
</changes-to-include>
</mdc-transform-rule>
```

These transformation rules make changes to WebGate agent definitions. The following information details how you can modify these changes for the PrimaryServerList and logoutRedirectUrl attributes.

- PrimaryServerList updates the primary server list for all WebGate agents and replaces them with the Access Manager server host from the Clone environment. This change can be viewed in the oam-config.xml file; it replaces the value of the PrimaryServerList attribute with the value equal to ${DeployedComponent/Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/serverhost}; for example, oam1-lon.example.com. The limitation of this rule is that it updates all servers in the primary list. You can use the transformation rule in Example 3–2 to update servers in PrimaryServerList with the different Clone servers.

Example 3–2 Modified PrimaryServerList Transformation Rule

```
<changes-to-include entity-path=
 "/config/NGAMConfiguration/DeployedComponent/Agent/WebGate/Instance">
    <replace attribute-match="/*/PrimaryServerList/0/host" value-match="(.*)">
        <replace-with
n="1?>${"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/
 Instance/oam_server1/host"}
        </replace-with>
    </replace>
    <replace attribute-match="/*/PrimaryServerList/1/host" value-match="(.*)">
        <replace-with
```

```
n="1?>${"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/
 Instance/oam_server2/host"}
        </replace-with>
    </replace>
</changes-to-include>
```

A load balancer is recommended between the WebGate and Access Manager server. In this case, you do not have to update the PrimaryServerList across data centers and can remove this transformation rule from the XML. However, you do need to update the PrimaryServerList parameter for IAMSuiteAgent and accessgate-oic unless you have configured these agents to communicate with the load balancer as well. Example 3–3 illustrates how to change the transformation rule to update the PrimaryServerList only for IAMSuiteAgent and accessgate-oic agents and not WebGate agents.

Example 3–3 Modified Transformation Rule for Different Agents

```
<changes-to-include entity-path="/config/NGAMConfiguration/DeployedComponent/
 Agent/WebGate/Instance">
    <replace attribute-match="/IAMSuiteAgent/PrimaryServerList/
     */host" value-match="(.*)">
        <replace-with
n="1?>${"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/
 Profile/OAMServerProfile/OAMSERVER/serverhost"}
        </replace-with>
    </replace>
    <replace attribute-match="/accessgate-oic/PrimaryServerList/
     */host" value-match="(.*)">
        <replace-with
n="1?>${"/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/
 Profile/OAMServerProfile/OAMSERVER/serverhost"}
        </replace-with>
    </replace>
</changes-to-include>
```

- The logoutRedirectUrl attribute updates the logout URL protocol, host and port for all WebGate agents with respective values from the Clone. If a load balancer is used globally to define the logout URL for all WebGate agents in the Master environment, you don't need to replace the logout URL in the Clone environment and can remove the transformation rule. If you are using a DCC authentication scheme and a global load balancer host name to define the DCC login and logout URL, then again you don't need to replace the login and logout URL in the Clone environment and can remove the transformation rule.

To configure Access Manager to use custom transformation rules, update the `setDomainEnv.xml` file on the Clone machine. Each Clone can use different transformation rules. Be sure to restart the Clone's AdminServer after changing a transformation rule. Figure 3–4 illustrates how to apply these custom rules.

Figure 3–4 Applying Custom Transformation Rules

```
#Added -Doracle.oam.EnableMDCReplication=true -Doracle.oam.MDCRuleFile=/u01/bits/APS/TransformRules.xml below to enable APS configuration

EXTRA_JAVA_PROPERTIES="  -Doracle.oam.EnableMDCReplication=true -Doracle.oam.MDCRuleFile=/u01/bits/APS/TransformRules.xml -DCONFIG_DS=jdbc/oamds  -DCONFI
G_HISTORY=true -Doam.oes.new=true -DOAM_POLICY_FILE=${DOMAIN_HOME}/config/fmwconfig/oam-policy.xml -DOAM_CONFIG_FILE=${DOMAIN_HOME}/config/fmwconfig/oa
m-config.xml  -DOAM_ORACLE_HOME=${OAM_ORACLE_HOME} -Doracle.security.am.SERVER_INSTNCE_NAME=${SERVER_NAME} -Does.jars.home=${OAM_ORACLE_HOME}/server/lib
/oes-d8 -Does.integration.path=${OAM_ORACLE_HOME}/server/lib/oeslib/oes-integration.jar -Does.enabled=true  -Djavax.xml.soap.SOAPConnectionFactory=webl
ogic.wsee.saaj.SOAPConnectionFactoryImpl -Djavax.xml.soap.MessageFactory=oracle.j2ee.ws.saaj.soap.MessageFactoryImpl -Djavax.xml.soap.SOAPFactory=oracl
e.j2ee.ws.saaj.soap.SOAPFactoryImpl ${EXTRA_JAVA_PROPERTIES}"
export EXTRA_JAVA_PROPERTIES
```

Applying custom transformation rules on command line

****

3.5 Modifying a Rule Document

A rule document is created for the purpose of replicating. The allowed transformation rules are represented by the XML document in Example 3–4.

Example 3–4 Replication Rules XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<mdc-transform-rule>
 <changes-to-include entity-path="/policy"/>
 <changes-to-include entity-path=
  "/config/NGAMConfiguration/DeployedComponent/Agent/WebGate/Instance">
   <replace attribute-match="/*/PrimaryServerList/*/host" value-match="(.*)">
    <replace-with n="1">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
     /OAMServerProfile/OAMSERVER/serverhost}</replace-with>
    </replace>
   <replace attribute-match="/*/UserDefinedParameters/logoutRedirectUrl"
    value-match="(.*)://(.*):(.*)/oam/server/logout">
    <replace-with n="1">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
      /OAMServerProfile/OAMSERVER/serverprotocol}</replace-with>
    <replace-with n="2">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
     /OAMServerProfile/OAMSERVER/serverhost}</replace-with>
    <replace-with n="3">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
     /OAMServerProfile/OAMSERVER/serverport}</replace-with>
   </replace>
   <replace attribute-match="/*/logoutRedirectUrl"
    value-match="(.*)://(.*):(.*)/oam/server/logout">
    <replace-withn="1">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
     /OAMServerProfile/OAMSERVER/serverprotocol}</replace-with>
    <replace-with n="2">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
     /OAMServerProfile/OAMSERVER/serverhost}</replace-with>
    <replace-with n="3">
     ${/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile
     /OAMServerProfile/OAMSERVER/serverport}</replace-with>
   </replace>
  </changes-to-include>
 <changes-to-include entity-
  path="/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/Profile/
  AuthenticationModules"/>
 <changes-to-include entity-
  path="/config/NGAMConfiguration/DeployedComponent/Server/NGA
  MServer/Profile/oamproxy"/>
 <changes-to-include entity-
  path="/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/
  Profile/Sme/SessionConfigurations"/>
 <changes-to-include entity-
  path="/config/NGAMConfiguration/DeployedComponent/Server/NGAMServer/
  Profile/OAMServerProfile/OAMSERVER">
 <ignore attribute-match="/serverprotocol"/>
 <ignore attribute-match="/serverhost"/>
 <ignore attribute-match="/serverport"/>
 </changes-to-include>
```

```
 <changes-to-include entity-
  path="/config/NGAMConfiguration/DataCenterConfiguration/Cluster">
 <ignore attribute-match="/DataCenterType"/>
 <ignore attribute-match="/ClusterId"/>
 <ignore attribute-match="/WriteEnabledFlag"/>
 </changes-to-include>
</mdc-transform-rule>
```

The rule document will mention the XPath of system configuration artifacts to be replicated for a clone. If there is any transformation to be done during replication for the entry in XPath, it can be provided as replacement rule for that clone. To add a new XPath for replication to a clone, create a new transformation XML file, using the above XML document as a template. Add and remove XPaths as required. For example, adding the following XPath as the child of an <mdc-transformation-rule> node and saving the file to the clone's file system will modify Available Services.

```
<changes-to-include entity-path=
  "/config/NGAMConfiguration/DeployedComponent/Descriptors/
   OAMSEntityDescriptor" />
```

Set the following VM property in the Clone's adminserver (setDomainEnv.sh) to use the newly created transformation rule document for replication instead of the OOTB one.

```
-Doracle.oam.MDCRuleFile=/path/to/rule/mdcrule.xml
```

For developer recommendations when using replication, see Section 3.8, "Best Practices for Replication."

3.6 Using REST API for Replication Agreements

The following sections contain details on how to use the REST API provided by Access Manager.

- Querying for Replication Agreement Details
- Modifying an Existing Replication Agreement
- Deleting a Replication Agreement

3.6.1 Querying for Replication Agreement Details

A REST request can be executed at the Master's endpoint to query the details of the Replication Agreement between a Master and a Clone.

```
GET http://oam1.example.com/oam/services/rest/
  _replication/201312040602298762 HTTP/1.1
  Content-Type: application/json
```

To query details of a Clone, use the following:

```
GET http://oam1.example.com/oam/services/rest/
  _replication/201312040602298762?type=CONSUMER HTTP/1.1 HTTP/1.1
  Content-Type: application/json
```

3.6.2 Modifying an Existing Replication Agreement

Replication Agreement properties (enabled status, poll interval and the like) can be updated by executing a REST request at the Master's endpoint. Either the Master or

Clone Replication Agreement will be updated as specified by the value of the replicaType parameter. The clone will poll for changes, apply them and wait the duration specified as the pollInterval.

```
PUT http://supplier.example.com/oam/services/rest/
  _replication/201312040602298762 HTTP/1.1
  Content-Type: application/json
  {"enabled":"false","pollInterval":"60","replicaType":"CONSUMER"}
```

This example will disable the Clone Replication Agreement and change the poll interval to '60' seconds. If a value for replicaType is not defined (or it is defined as SUPPLIER), the Master's Replication Agreement will be updated.

To edit the poll interval using cURL, use the following command. Note that the replicaType value for updating the clone in this case is SUPPLIER or CONSUMER.

```
curl -u <repluser> -H 'Content-Type: application/json' -X PUT
 'https://supplier.example.com/oam/services/rest/_replication/201409231329353668'
 -d '{"pollInterval":"60","replicaType":"CONSUMER"}'
```

Table 3–2 lists properties that can be modified using cURL and REST.

Table 3–2 Modifying Replication Agreement Properties

Property	Modification Command

BatchSize	Number of change records (journals) returned by the master as a result of a getChanges query by clone. Ideally the default batch size of 32 is sufficient as all changes are pulled in multiple batches as part of fetching. However if the setup needs a large batch size, execute the following command:
	`curl -u <repluser> -H 'Content-Type: application/json' -X PUT`
	`'https://master.example.com/oam/services/rest/_`
	`replication/<replid>'`
	`-d '{"batchSize":"100","replicaType":"SUPPLIER"}'`

User Context	In rare instances, the user context for replication poll may need to be modified.
	`curl -u <repluser> -H 'Content-Type: application/json' -X PUT`
	`'https://supplier.example.com:7002/oam/services/rest/_`
	`replication/201409231329353668'`
	`-d '{"replicaType":"CONSUMER",`
	`"config":{"entry":{"key":"authorization","value":"`
	`BASIC cG9sbHVzZXI6c2VjcmV0"}}}'`
	
	'cG9sbHVzZXI6c2VjcmV0' is a base 64 encoded value for polluser credentials. Any user credentials can be used here instead of the repluser which is used to execute the command.

3.6.3 Deleting a Replication Agreement

A Replication Agreement can be deleted by executing the following REST API at the master DC's endpoint. Replication Agreements that are currently active and in use cannot be deleted until the Master and Clone have been disabled.

```
DELETE http://oam1.example.com/oam/services/rest/_replication/
 201312040602298762 HTTP/1.1
```

3.7 Replicating Domains in Identity Manager Deployments

If you have a deployment where Access Manager 11.1.2.1.0 and Oracle Identity Manager (11.1.2.1.0) are integrated in the same domain T2P cannot be used for domain replication because Identity Manager does not support it. In this case, Access Manager and Identity Manager should be installed in different domains using the following procedure.

1. Install Access Manager.
2. Run `configureSecurityStore (-create)`.
3. Start Access Manager.

   Remember to enable TRACE logging with instrumented EAR.
4. Install Identity Manager.
5. Run `configureSecurityStore (-join)`.
6. Update the default passwords for the Access Manager and Identity Manager domains in *$DOMAIN_HOME*`/config/fmwconfig/default-keystore.jks` password using the `keytool` command.
7. Set the same password values in the CSF using the EM console.
   a. Navigate to the *domain_name* of the appropriate Weblogic domain.
   b. Right click the *domain_name* and navigate to Security --> Credentials.
   c. Expand the `oracle.wsm.security` Credential map and edit the value of `keystore-csf-key`.
   d. Update password and confirm password fields with the password.

      This password should be same as the new password for `default-keystore.jks` in both Access Manager and Identity Manager domains
8. Map oracle.wsm.security with the Key keystore-csf-key.
9. Start Identity Manager.
10. Restart Access Manager and Identity Manager.

3.8 Best Practices for Replication

The following points and the information in the sections should be taken into account when setting up data replication.

- It is recommended that as many Policy Domain artifacts are created as possible before cloning. This will help the replication manager work efficiently during incremental updates.
- The OAM server instance list will be used as the Well Known Addresses (WKA) to create a Coherence cluster so do not add other data center servers to the server instance list.
- To allow for a WebGate profile to point to a remote data center in the secondary server list, use the Other option to provide OAP with the host and port details of the remote data center.
- Enabling Replication Logs
- Changing the User Identifier

3.8.1 Enabling Replication Logs

To get detailed logs on replication agreement and replication poll related issues, enable the logger 'oracle.oam.replication' by executing the WLST command

```
setLogLevel(logger="oracle.oam.replication", level="TRACE:32", persist="0",
target="AdminServer")
```

This will enable logger only till next shutdown of AdminServer. To keep the logger state across restart, set the persist attribute to "1"

3.8.2 Changing the User Identifier

While creating replication agreement if you have not specified any authorization header of the user to be used for replication if the user's password got changed at later point, you can edit the replication agreement with the latest user identity and password using the following command.

```
curl -u <repluser> -H 'Content-Type: application/json' -X PUT
'https://supplier.example.com:7002/oam/services/rest/_replication/201409231
329353668' -d '{"replicaType":"CONSUMER","config":{"entry":
{"key":"authorization","value":"BASIC d2VibG9naWM6d2VsY29tZTE="}}}'
```

4

Setting Up the Multi-Data Center: A Sequence

The sequence of steps in this chapter will help you to setup a Multi-Data Center with four nodes using Oracle Access Manager 11.1.1.2. The configuration spans two Data Centers with two nodes per Data Center. The nodes are configured in Active/Active Mode.

This chapter contains the following section.

4.1 Before You Begin

Read the following chapters before beginning the steps documented in this sequence for an understanding of Multi-Data Center and its features.

- Chapter 1, "Understanding Multi-Data Centers"
- Chapter 2, "Configuring Multi-Data Centers"
- Chapter 3, "Synchronizing Data In A Multi-Data Center"

Confirm the following before you begin the Multi-Data Center set-up sequence.

- Check that your operating system is up-to-date with all necessary patches applied.
- Mount the binaries you will be using. The applicable Oracle software includes:
  - Oracle Fusion Middleware Identity and Access Management 11g (11.1.2.3.0)
  - Oracle WebLogic Server 10g (10.3.6)
  - Oracle Database 11g (11.2.0.4)
  - Oracle Fusion Middleware Repository Creation Utility 11g (11.1.2.3.0)
- Add `/etc/hosts` entries on all four nodes being configured.
- Verify that the Oracle Database is connected and accessible.
- Verify that each machine has more than 30 GB space available and more than 8GB of memory available.

4.2 Setting Up a Multi-Data Center

Be sure to follow this sequence as documented for a successful set-up of a Multi-Data Center with data replication using T2P. The configuration spans two Data Centers with two nodes per Data Center. The nodes are configured in Active/Active Mode.

1. Install the Java Development Kit (JDK) 1.7.0.60 on all four of the Nodes and set the appropriate environment variables.
2. Run the Repository Creation Utility (RCU) 11.1.2.3.0 on Data Center 1 and Data Center 2.

   This will create and load the appropriate database schemas for Oracle Identity and Access Management products.
3. Install WebLogic Server 10g (10.3.6) on Data Center 1, Node 1.

   This process creates the Middleware Home (<MW_HOME>).
4. Install the Oracle Identity and Access Management 11g (11.1.2.3.0) software on Data Center 1, Node 1.

   Oracle Identity and Access Management contains the Oracle Access Management suite which includes Oracle Access Manager. The default name of this Oracle product home directory after installation is Oracle_IDM1.
5. Run the Oracle Fusion Middleware Configuration Wizard script to configure Oracle Access Management on Data Center 1, Node 1.

   The Wizard script is `Oracle_IDM1/common/bin/config.sh` script (on Linux or UNIX), or `Oracle_IDM1\common\bin\config.cmd` (on Windows). Minimally, you will be configuring:

   - a new WebLogic domain
   - an Oracle Access Management Administration Server
   - an Oracle Access Management Managed Server
   - Oracle Access Manager
6. Run the `configureSecurityStore.py` script on Data Center 1, Node 1 to configure the Database Security Store.

   a. `<MW_HOME>/oracle_common/common/bin/wlst.sh`

   b.
   ```
   <MW_HOME>/Oracle_IDM1/common/tools/configureSecurityStore.py
   -c IAM -d <MW_HOME>/user_projects/domains/OAMDomain
   -p Oracle123 -m create -v
   ```
7. Modify the following WebLogic scripts on Data Center 1, Node 1.

   a. Open `startWeblogic.sh` and `startManagedWeblogic.sh` using vi and enter the appropriate value for WLS_USER.

      Enter the password when asked; do not hard code it here.

   b. Save `startWeblogic.sh` and `startManagedWeblogic.sh`.

   c. Open `setDomainEnv.sh` using vi and add the following line:

      ```
      USER_MEM ARGS="-Xms1024m -Xmx1024m -XX:MaxPermSize=512m"
      ```

   d. Save `setDomainEnv.sh`.
8. Create and run a `cConfig.sh` script in the MDC folder on Data Center 1, Node 1.

The `cConfig.sh` script concatenates the necessary environment variables and `copyConfig.sh` into one script. You will need to create the MDC folder to serve as the T2P_HOME.

a. Add the following contents and save as `cConfig.sh`.

```
export JAVA_HOME=/u01/app/jdk1.7.0_60;
export MW_HOME=/u01/app/Middleware;
export T2P_HOME=/u01/bits/MDC;
export WL_DOMAIN_HOME=$MW_HOME/user_projects/domains/OAMDomain;
```

b. Source `cConfig.sh`.

```
$<>. cConfig.sh
```

9. Execute `copyBinary.sh` on Data Center 1, Node 1.

`copyBinary.sh` and `pasteBinary.sh` will be used to avoid a time-consuming installation process on the remaining nodes. When running `copyBinary.sh`, the Administration and Managed Servers can be running or stopped.

a. Change to the `bin` directory.

```
cd $MW_HOME/oracle_common/bin/;
```

b. Run the script.

```
./copyBinary.sh -javaHome $JAVA_HOME
 -archiveLoc $T2P_HOME/oamt2pbin.jar -sourceMWHomeLoc $MW_HOME
 -idw true -ipl $MW_HOME/oracle_common/oraInst.loc -silent true
 -ldl $T2P_HOME/oam_cln_log
```

10. Copy the following files to the MDC folder on Data Center 1, Node 1.

- $T2P_HOME/cConfig.sh (already in the MDC folder)
- $T2P_HOME/oamt2pbin.jar (already in the MDC folder)
- $MW_HOME/oracle_common/bin/pasteBinary.sh
- $MW_HOME/oracle_common/jlib/cloningclient.jar
- $MW_HOME/oracle_common/oraInst.loc

11. Copy the MDC folder (populated with the five files) to Data Center 1, Node 2, and Data Center 2, Nodes 1 and 2.

12. Execute `pasteBinary.sh` on Data Center 1, Node 2.

a. Source `cConfig.sh`.

```
$<>. cConfig.sh
```

b. Run `pasteBinary.sh` on Data Center 1, Node 2.

```
$T2P_HOME/pasteBinary.sh -javaHome $JAVA_HOME
-al $T2P_HOME/oamt2pbin.jar -tmw $MW_HOME -silent true
-idw true -esp false -ipl $T2P_HOME/oraInst.loc
-ldl $T2P_HOME/oam_cln_log -silent true
```

13. Create a Managed Server JAR on Data Center 1, Node 1 and copy it to Data Center 1, Node 2.

`pack.sh` is used to create the JAR and is located in the <MW_HOME>/oracle_common/common/bin directory. The pack and unpack (used in the next step) scripts must be executed in the same Data Center only whereas copyConfig and

pasteConfig (used later in the procedure) must be executed to the Master node of the other Data Center and then run Pack/UnPack within those data centers.

a. Run `pack.sh`.

```
./pack.sh -domain=$MW_HOME/user_projects/domains/OAMDomain
-template=OAMManagedServer.jar -template_name="OAM Domain" -managed=true
```

b. Copy `OAMManagedServer.jar` to the MW_HOME/oracle_common/common/bin directory on Data Center 1, Node 2.

14. Unpack the Managed Server JAR on Data Center 1, Node 2 using `unpack.sh`.

    The JAR is used as a template to create the OAMDomain Domain Structure on Data Center 1, Node 2.

    a. `mkdir -p $MW_HOME/user_projects/domains/OAMDomain`

    b. `cd <MW_HOME>/oracle_common/common/bin`

    c. `./unpack.sh -domain=$MW_HOME/user_projects/domains/OAMDomain -template=OAMManagedServer.jar`

15. Modify the following WebLogic scripts on Data Center 1, Node 2.

    a. Open `startManagedWeblogic.sh` using vi and enter the appropriate values for WLS_USER and WLS_PW.

    b. Save `startWeblogic.sh` and `startManagedWeblogic.sh`.

    c. Open `setDomainEnv.sh` using vi and add the following line:

    ```
    USER_MEM ARGS="-Xms1024m -Xmx1024m -XX:MaxPermSize=512m"
    ```

    d. Save `setDomainEnv.sh`.

    At this point in the sequence, the Data Center 1 cluster and its two nodes are configured and ready for Multi-Data Center configurations. Start the Administration Server and the oam_server1 and oam_server2 Managed Servers. Disable the SSL port number 14101.

16. Enable Multi-Data Center mode on Data Center 1, Node 1.

    a. `cd $T2P_HOME/MDC`

    b. Create `OAMMDC.properties` on Data Center 1, Node 1 using vi.

    Add the following lines to `OAMMDC.properties` and save.

    ```
    SessionMustBeAnchoredToDataCenterServicingUser=false
    SessionDataRetrievalOnDemand=true
    Reauthenticate=false
    SessionDataRetrievalOnDemandMax_retry_attempts=3
    SessionDataRetrievalOnDemandMax_conn_wait_time=80
    SessionContinuationOnSyncFailure=true
    MDCGitoCookieDomain=.customerpoc.com
    ```

    c. Change to the ../Oracle_IDM1/common/bin directory and run WLST.

    d. `./wlst.sh`

    e. `connect()`

    f. `domainRuntime()`

    g. `enableMultiDataCentreMode(propfile="../OAMMDC.properties")`

    h. `setMultiDataCentreClusterName(clusterName="<string_value>")`

i. `setMultiDataCenterWrite(WriteEnabledFlag="true")`

j. `validateMDCConfig()`

k. `exit()`

17. Create `oamt2pconfig.jar` on Data Center 1, Node 1 and copy it to Data Center 2, Node 1.

    `copyConfig.sh` is located in $MW_HOME/oracle_common/bin/ and must be executed on the Master node. To run `copyConfig.sh`, the Administration and Managed Servers should be up and running.

    a. Source `cConfig.sh`.

    ```
    $<>. cConfig.sh
    ```

    b. Create `$T2P_HOME/t2p_domain_pass.txt` using vi.

    Add a password value for use with copyConfig.sh; for example, Oracle123 (without quotes).

    c.
    ```
    ./copyConfig.sh -javaHome $JAVA_HOME
    -archiveLoc $T2P_HOME/oamt2pConfig.jar
    -sourceDomainLoc $WL_DOMAIN_HOME
    -sourceMWHomeLoc $MW_HOME
    -domainHostName oam1-dc1.customerpoc.com
    -domainPortNum 7001 -domainAdminUserName weblogic
    -domainAdminPassword $T2P_HOME/t2p_domain_pass.txt
    -silent true -ldl $T2P_HOME/oam_cln_log_config
    -opssDataExport true -debug true
    ```

    d. Copy `oamt2pconfig.jar` to the Data Center 2, Node 1.

18. Execute `pasteBinary.sh` on Data Center 2, Node 1.

    a. Source `cConfig.sh`.

    ```
    $<>. cConfig.sh
    ```

    b. Run:

    ```
    $T2P_HOME/pasteBinary.sh -javaHome $JAVA_HOME
    -al $T2__HOME/oamt2pbin.jar -tmw $MW_HOME -silent true
    -idw true -esp false -ipl $T2P_HOME/oraInst.loc
    -ldl $T2P_HOME/oam_cln_log -silent true
    ```

19. Execute `pasteBinary.sh` on Data Center 2, Node 2.

    a. Source `cConfig.sh`.

    ```
    $<>. cConfig.sh
    ```

    b. Run:

    ```
    $T2P_HOME/pasteBinary.sh -javaHome $JAVA_HOME
    -al $T2__HOME/oamt2pbin.jar -tmw $MW_HOME -silent true
    -idw true -esp false -ipl $T2P_HOME/oraInst.loc
    -ldl $T2P_HOME/oam_cln_log -silent true
    ```

20. Execute `extractmovePlan.sh` on Data Center 2, Node 1.

    a. `mkdir $T2P_HOME/moveplan`

    b. `cd $MW_HOME/oracle_common/bin/`

c. Source `cConfig.sh`.

```
$<>. cConfig.sh
```

d. `./extractMovePlan.sh -javaHome $JAVA_HOME -al $T2P_HOME/oamt2pConfig.jar -planDirLoc $T2P_HOME/moveplan/`

e. Backup the moveplan and then make the following modifications using vi.

```
Search and Replace the hostnames

:1,$s/oam1-dc1/oam1-dc2/g
:1,$s/oam2-dc1/oam2-dc2/g

Search and replace datasource names

:1,$s/DC1/DC2/g

Search for the two instances of "Password File" and add the previously
created t2p_domain_pass.txt Password File location as a <value>.

<value>/u01/bits/final/MDC/t2p_domain_pass.txt</value>
```

f. Create $T2P_HOME/t2p_pass.txt with a password value you want.

This file is used to create new components on the target environment with the associated password. The moveplan has a reference to it so that when the components are created the password will be assigned.

21. Run the `psa.sh` PSA script to update the Oracle Platform Security Services (OPSS) schema on Data Center 2, Node 1.

The script is in the Oracle_IDM1/bin/ directory. Use the following procedure to verify that the PSA has updated the OPSS version from 11.1.1.7.0 to 11.1.1.7.2.

a. Connect to the system as sysdba.

You can use SQL Plus or SQL Developer.

b. Enter the following SQL statement.

select * from DC2_OPSS.JPS_ATTRS where JPS_ATTRS.ATTRNAME='orclProductVersion';

22. Execute `pasteConfig.sh` on Data Center 2, Node 1.

The same JDK used on the source must be used on the target.

```
$MW_HOME/oracle_common/bin/pasteConfig.sh
-javaHome $JAVA_HOME -archiveLoc $T2P_HOME/oamt2pConfig.jar
-targetMWHomeLoc $MW_HOME -targetDomainLoc $WL_DOMAIN_HOME
-movePlanLoc $T2P_HOME/moveplan/moveplan.xml -domainAdminPassword
$T2P_HOME/t2p_domain_pass.txt -ldl $T2P_HOME/oam_cln_log
-silent true
```

Note: Comment out all keystore <> tags in the moveplan if there is an issue.

23. Modify the following WebLogic scripts on Data Center 2, Node 1.

a. Open `startWeblogic.sh` and `startManagedWeblogic.sh` using vi and enter the appropriate values for WLS_USER and WLS_PW.

b. Save `startWeblogic.sh` and `startManagedWeblogic.sh`.

c. Open `setDomainEnv.sh` using vi and add the following line:

```
USER_MEM ARGS="-Xms1024m -Xmx1024m -XX:MaxPermSize=512m"
```

d. Save `setDomainEnv.sh`.

24. Create a Managed Server JAR on Data Center 2, Node 1 and copy it to Data Center 2, Node 2.

    `pack.sh` is used to create the JAR and is located in the `<MW_HOME>/oracle_common/common/bin` directory. The pack and unpack (used in the next step) scripts must be executed in the same Data Center only.

    a. Run `pack.sh`.

    ```
    ./pack.sh -domain=$MW_HOME/user_projects/domains/OAMDomain
    -template=OAMManagedServer.jar -template_name="OAM Domain" -managed=true
    ```

    b. Copy `OAMManagedServer.jar` to the `<MW_HOME>/oracle_common/common/bin` directory on Data Center 2, Node 2.

25. Unpack the Managed Server JAR on Data Center 2, Node 2 using `unpack.sh`.

    The JAR will be used as a template to create the OAMDomain Domain Structure on Data Center 2, Node 2.

    a. `mkdir -p $MW_HOME/user_projects/domains/OAMDomain`

    b. `cd <MW_HOME>/oracle_common/common/bin`

    c. `./unpack.sh -domain=$MW_HOME/user_projects/domains/OAMDomain -template=OAMManagedServer.jar`

26. Modify the following WebLogic scripts on Data Center 2, Node 2.

    a. Open `startManagedWeblogic.sh` using vi and enter the appropriate values for WLS_USER and WLS_PW.

    b. Save `startWeblogic.sh` and `startManagedWeblogic.sh`.

    c. Open `setDomainEnv.sh` using vi and add the following line:

    ```
    USER_MEM ARGS="-Xms1024m -Xmx1024m -XX:MaxPermSize=512m"
    ```

    d. Save `setDomainEnv.sh`.

    At this point in the sequence, the Data Center 2 cluster and its two nodes are configured and ready for Multi-Data Center configurations. Start the Administration Server and the oam_server1 and oam_server2 Managed Servers. Disable the SSL port number 14101.

27. Enable Multi-Data Center mode on Data Center 2, Node 1.

    a. Restart the Administration Server on Data Center 2, Node 1.

    b. Change to the ../Oracle_IDM1/common/bin directory and run WLST.

    c. `./wlst.sh`

    d. `connect()`

    e. `domainRuntime()`

    f. `enableMultiDataCentreMode(propfile="//OAMMDC.properties")`

    g. `setMultiDataCentreClusterName(clusterName="<string_value>")`

h. `setMultiDataCenterWrite(WriteEnabledFlag="true")`

i. `exit()`

28. Create two WebGate agents using the Oracle Access Management Console on Data Center 1, Node 1 only.
    - Name the agents MDC-DC1 and MDC-DC2
    - Check AccessClientPassword and AllowManagementOperations
    - Be sure the Primary Server List has Access Server listed as "Other" - ideally it will have global load balancer/LTM entries rather than the local hosts entries.

29. Create the MDCPartner-DC1 and MDCPartner-DC2 property files using vi

    Create these files on both Data Center 1, Node 1 and Data Center 2, Node 1 with the following data.

```
vi MDCPartner-DC1.properties

remoteDataCentreClusterId=FINALDC1
oamMdcAgentId=MDC-DC1
PrimaryHostPort=oam1-dc1.poc.com:5575
SecondaryHostPort
AccessClientPasswd
oamMdcSecurityMode=Open
agentVersion=11g
trustStorePath
keyStorePath
globalPassPhrase
keystorePassword
RESTEndpoint=http://oam1-dc1.poc.com:7001

vi MDCPartner-DC2.properties

remoteDataCentreClusterId=FINALDC2
oamMdcAgentId=MDC-DC2
PrimaryHostPort=oam1-dc2.poc.com:5575
SecondaryHostPort
AccessClientPasswd
oamMdcSecurityMode=Open
agentVersion=11g
trustStorePath
keyStorePath
globalPassPhrase
keystorePassword
RESTEndpoint=http://oam1-dc2.poc.com:7001
```

30. Register the partners on Data Center 1, Node 1 using wlst.sh.

a. Change to the ../Oracle_IDM1/common/bin directory and run WLST.

b. `./wlst.sh`

c. `connect()`

d. `domainRuntime()`

e. `addPartnerForMultiDataCentre(propfile="../MDCPartner-DC1.properties")`

f. `addPartnerForMultiDataCentre(propfile="../MDCPartner-DC2.properties")`

g. `setMultiDataCenterType(DataCenterType="Master")`

h. `exit()`

31. Register the partners on Data Center 2, Node 1 using wlst.sh.
    a. Change to the ../Oracle_IDM1/common/bin directory and run WLST.
    b. `./wlst.sh`
    c. `connect()`
    d. `domainRuntime()`
    e. `addPartnerForMultiDataCentre(propfile="../MDCPartner-DC1.properties")`
    f. `addPartnerForMultiDataCentre(propfile="../MDCPartner-DC2.properties")`
    g. `setMultiDataCenterType(DataCenterType="Clone")`
    h. `exit()`
32. Export the partner and policy information from Data Center 1, Node 1 and then import it to Data Center 2, Node 1.
    a. Change to the ../Oracle_IDM1/common/bin directory and run WLST to export from Data Center 1, Node 1.
    b. `./wlst.sh`
    c. `connect()`
    d. `exportPartners(pathTempOAMPartnerFile="<oampartner.xml>")`
    e. `exportPolicy(pathTempOAMPolicyFile="<oampolicy.xml>")`
    f. `exit()`
    g. Copy `oampolicy.xml` and `oampartner.xml` to Data Center 2, Node 1.
    h. Change to the ../Oracle_IDM1/common/bin directory and run WLST to import on Data Center 2, Node 1.
    i. `./wlst.sh`
    j. `connect()`
    k. `importPolicy(pathTempOAMPolicyFile="<oampolicy.xml>")`
    l. `importPartners(pathTempOAMPartnerFile="<oampartner.xml>")`
    m. `exit()`

4.3 Enabling Automated Policy Synchronization

The sequence in this section will enable the Automated Policy Synchronization (APS) feature for automated data synchronization among the servers. The procedure includes commands for testing the REST services as well as details on adding custom transformation rules to the synchronization. See Chapter 3, "Synchronizing Data In A Multi-Data Center" for details on APS and transformation rules.

1. Stop all the Administration and Managed Servers.
2. Add the following line to `$WL_DOMAIN_HOME/bin/setDomainEnv.sh` on both Data Center 1, Node 1 and Data Center 2, Node 1 and save the file.

```
EXTRA_JAVA_PROPERTIES="
 -Doracle.oam.EnableMDCReplication=true -DCONFIG_DS=jdbc/oamds …
```

3. Start the Administration Servers only.
4. Test the REST services using the following commands:

curl -u weblogic 'http://oam1-dc1.customerpoc.com:7001/oam/services/rest/_replication/hello'

RESPONSE: {"ok":"true”}

curl -u weblogic 'http://oam1-dc2.poc.com:7001/oam/services/rest/_replication/hello'

RESPONSE: {"ok":"true”}

curl -u weblogic:Oracle123 -H 'Content-Type: application/json' -X POST 'http://oam1-dc1.poc.com:7001/oam/services/rest/_replication/setup' -d '{"name”:"DC1toDC2", "source”:"FINALDC1","target”:"FINALDC2","documentType":"ENTITY”}'

RESPONSE: {"enabled":"true","identifier":"201409040157218184","ok":"true","pollInterval":"900","startingSequenceNumber":"101","state":"READY"}

#The random long number will be unique to every replication agreement, so don’t use that number as is, though use the number which comes as an output from the 3rd curl command below

curl -u weblogic:Oracle123 -H 'Content-Type: application/json’ 'http://oam1-dc1.poc.com:7001/oam/services/rest/_replication/201409040157218184'

RESPONSE: {"enabled":"true","identifier":"201409040157218184","ok":"true","pollInterval":"3600","startingSequenceNumber":"101","state":"ACTIVE"}

curl -u weblogic:Oracle123 -H 'Content-Type: application/json’ 'http://oam1-dc1.poc.com:7001/oam/services/rest/_replication/201409040157218184?type=consumer'

RESPONSE: {"enabled":"true","identifier":"201409040157218184","ok":"true","pollInterval":"900","startingSequenceNumber":"101","state":"READY"}

curl -u weblogic:Oracle123 -H 'Content-Type: application/json' -X PUT 'http://oam1-dc1.poc.com:7001/oam/services/rest/_replication/201409040157218184' -d '{"pollInterval":"60","replicaType":"consumer"}'

RESPONSE: {"enabled":"true","identifier":"201409040157218184","ok":"true","pollInterval":"3600","startingSequenceNumber":"101","state":"ACTIVE”}

#Run this command on NODE1DC2 ONLY IF you want to disable Policy Writes to DC2 (or Clones) and just accept policy writes via the Master Policy Server using APS Synch: setMultiDataCenterWrite(WriteEnabledFlag="false”)

5. Create a transformation rules file using vi as `$T2P_HOME/transformationrules.xml`.

   `transformationrules.xml` should contain the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<mdc-transform-rule>
    <changes-to-include
entity-path="/DeployedComponent/Agent/WebGate/Instance">
        <replace attribute-match="/IAMSuiteAgent/PrimaryServerList/*/host”
value-match="(.*)">
            <replace-with
n="1">${DeployedComponent/Server/NGAMServer/Profile/OAMServerProfile/OAMSERVER/
```

```
serverhost}</replace-with>
        </replace>
    </changes-to-include>
    <changes-to-include
entity-path="/DeployedComponent/Server/NGAMServer/Profile/AuthenticationModules
"/>
</mdc-transform-rule>
```

6. Add the following line to `$WL_DOMAIN_HOME/bin/setDomainEnv.sh` on Data Center 2, Node 1 only and save the file.

```
EXTRA_JAVA_PROPERTIES="
-Doracle.oam.MDCRuleFile=/u01/bits/customer/MDC/transformationrules.xml
-Doracle.oam.EnableMDCReplication=true -DCONFIG_DS=jdbc/oamds …
```

7. Start the Administration and Managed Servers.

This completes the Multi-Data Center setup with T2P data replication and APS configuration! You can test the APS function by creating an agent and a policy on data Center 1 and verifying that it auto migrates to Data Center 2.

4.4 Troubleshooting the Multi-Data Center Setup

Follow this procedure if you have identified inconsistencies or errors in your setup and want to revert the APS.

1. Disable replication on the Master (Supplier).

```
curl -u weblogic:welcome1 -H 'Content-Type: application/json'
  -X PUT 'http://supplier.example.com:7001/oam/services/
  rest/_replication/201311271226476658'
  -d '{"enabled":"false","replicaType":"SUPPLIER"}'
```

2. Disable replication on the Clone (Consumer).

```
curl -u weblogic:welcome1 -H 'Content-Type: application/json'
  -X PUT 'http://supplier.example.com:7001/oam/services/
  rest/_replication/201311271226476658'
  -d '{"enabled":"false","replicaType":"CONSUMER"}'
```

3. Delete the replication agreements on the Master and Clone using this command.

```
curl -u weblogic:welcome1 -H 'Content-Type: application/json'
  -X DELETE 'http://supplier.example.com:7001/oam/services/
  rest/_replication/201311271226476658'
```

> **Note:** The AM_REPLICATION_SETTINGS table in the OAM Schema defines the replication agreements so you can also delete the agreements by manual deletion using the following SQL statement.
>
> - Select * from FINALDC1_OAM.AM_REPLICATION_SETTINGS;
> - Select * from FINALDC2_OAM.AM_REPLICATION_SETTINGS;
>
> Alternately:
>
> - Delete from FINALDC1_OAM.AM_REPLICATION_SETTINGS;
> - Delete from FINALDC2_OAM.AM_REPLICATION_SETTINGS;

4. commit;