Deployment Planning Guide

OpenAM 13.5

Mark Craig
David Goldsmith
Gene Hirayama

ForgeRock AS
201 Mission St, Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Guide to planning OpenAM deployments. OpenAM provides open source Authentication, Authorization, Entitlement and Federation software.

Table of Contents

Preface

This guide helps you plan an OpenAM deployment, including performing the following high-level tasks:

- Choose OpenAM software features for your deployment. OpenAM offers a wide-variety of features to meet the needs of any deployment.
- Plan any customizations or extensions that are absolutely required.
- Gather information about the services that your OpenAM deployment uses, and define what those services must provide for your deployment.
- Gather information about the resources that OpenAM protects as well as the consumers of your OpenAM deployment.
- Reach agreement with partners concerning federation and other deployments where multiple actors share control.
- Define your objectives in terms of service levels including availability and security for different services.
- Determine performance requirements and testing procedures to assure performance requirements are met.
- Specify logical and physical deployment topologies, including site and multiple data center deployments.
- Plan the deployment process, including implementing training teams and partners, customization and hardening, and development of a proof-of-concept implementation. Other tasks include performing integration with client applications, integration with auditing tools, automation and continuous integration when necessary, running functional and non-functional testing, documenting procedures and tracking changes, acceptance in production, maintaining and supporting the solution in production, and planning for expansion and upgrade.

1. Who Should Use This Guide

This guide is written for access management architects and project managers who plan OpenAM software deployments within their organizations.

This guide focuses on the tasks that need to be completed for successful OpenAM deployment. It does not show how to accomplish each individual technical task, such as installation, configuration, and maintenance.

Before using this guide, you should have a firm grounding in software deployment, in access management, and in OpenAM software itself. OpenAM deployment training from ForgeRock can help you get started.

2. Getting Deployment Training, Help, and Support

Formal training can help you and your colleagues come up to speed quickly on OpenAM deployment, development, administration, maintenance, and tuning. ForgeRock University regularly offers training courses for OpenAM topics, including OpenAM deployment. For a current list of available courses, see http://forgerock.com/services/university/.

ForgeRock Professional Services offers the support plans for every deployment. For more information about ForgeRock Support offerings, see http://forgerock.com/services/professional-services/.

When your success depends on OpenAM access management services, you want to know that product experts are standing by ready to help should anything go wrong. ForgeRock offers the right support plans for every deployment. For more information about ForgeRock Support offerings, see http://forgerock.com/services/support/.

For more information, email us at info@forgerock.com.

Chapter 1

Introduction to Deployment Planning

The OpenAM Deployment Planning Guide presents the key features and possible solutions to protect your Identity and Access Management (IAM) deployments. The guide discusses the benefits and advantages of implementing OpenAM. It also provides examples of deployment to help you determine which features you might want to include in your deployments.

1.1. Understanding Identity Access Management

The proliferation of cloud-based technologies, mobile devices, social networks, Big Data, enterprise applications, and business-to-business (B2B) services has spurred the exponential growth of identity information, which is often stored in varied and widely-distributed identity environments.

The challenges of securing such identity data and the environments that depend on the identity data are daunting. Organizations that expand their services in-house or globally through internal development or through acquisitions must manage identities across wide spectrum of identity infrastructures. This expansion requires a careful integration of acquisitions must manage identities across a wide spectrum of identity infrastructures. This expansion requires a careful integration of disparate access management systems, platform-dependent architectures with limited scalability, and ad-hoc security components.

ForgeRock, a leader in the IAM market, provides proven solutions to securing your identity data with its Identity Relationship Management (IRM) platform.

Identity Management is the automated provisioning, updating, and de-provisioning of identities over their lifecycles. Access Management is the authentication and authorization of identities who desire privileged access to an organization's resources. Access management encompasses the central auditing of operations performed on the system by customers, employees, and partners. Access management also provides the means to share identity data across different access management systems, legacy implementations, and networks.

1.2. OpenAM is More than Just Single Sign-On

OpenAM is an all-in-one open source, centralized access management solution, securing protected resources across the network and providing authentication, authorization, Web security, and Federation Services in a single, integrated solution. OpenAM is deployed as a simple `.war` file and provides production-proven platform independence, flexible and extensible components, as well

as a high availability and a highly scalable infrastructure. Using open standards, OpenAM is fully extensible, and can expand its capabilities through its SDKs and numerous REST endpoints.

OpenAM is part of ForgeRock's Identity Relationship Management (IRM) *platform*, built upon ForgeRock's Open Identity Stack, an open source identity and access management provider of mobile-ready, cloud, enterprise, social, and partner services. The IRM platform provides global consumer services across any platform for any connected device or any Internet-connected entity.

ForgeRock IRM platform features the following products:

- **OpenAM Context-Based Access Management System**. OpenAM is an all-in-one industry-leading access management solution, providing authentication, authorization, federation, Web services security, adaptive risk, and entitlements services among many other features. OpenAM is deployed as a simple `.war` file, featuring an architecture that is platform independent, flexible, and extensible, and highly available and scalable.

- **OpenIDM**. Cloud-Focused Identity Administration. OpenIDM is a lightweight provisioning system, built on resource-oriented principles. OpenIDM is a self-contained system, providing workflow, compliance, synchronization, password management, and connectors. OpenIDM features a next-generation modular architecture that is self-contained and highly extensible.

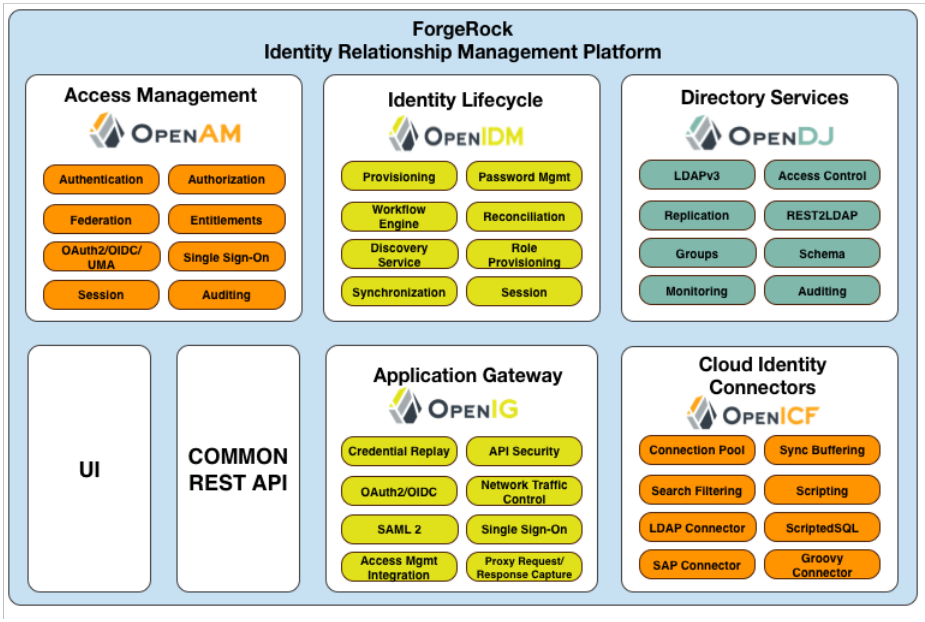
- **OpenDJ**. Internet Scale Directory Server. OpenDJ provides full LDAP protocol support, multi-protocol access, cross-domain replication, common REST framework, SCIM support, and many other features.

- **OpenIG**. No Touch Single Sign-On (SSO) to enterprise, legacy, and custom applications. OpenIG is a reverse proxy server with specialized session management and credential replay functionality. OpenIG works together with OpenAM to integrate Web applications without needing to modify the target application or the container that it runs in.

- **OpenICF**. Enterprise and Cloud Identity Infrastructure Connectors. OpenICF provides identity provisioning connections offering a consistent layer between target resources and applications and exposing a set of programming functions for the full lifecycle of an identity. OpenICF connectors are compatible with OpenIDM, Sun Identity Manager, Oracle® Waveset, Brinqa® GRC Platform, and so forth.

Figure 1.1, "ForgeRock IRM Platform" illustrates the ForgeRock IRM platform.

Figure 1.1. ForgeRock IRM Platform

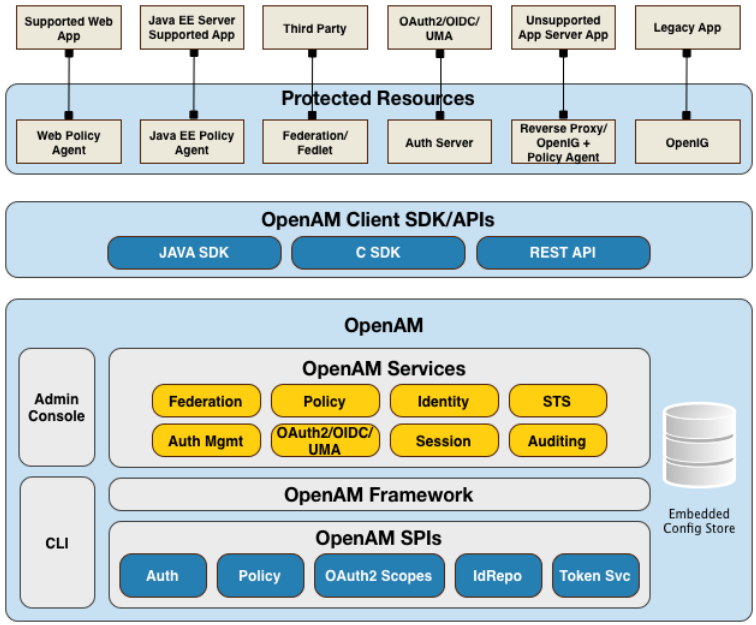


1.3. OpenAM Server Overview

OpenAM is an open source centralized access management server, securing protected resources across the network and providing authentication, authorization, Web security, and federation services in a single, integrated solution. OpenAM manages access to the protected resources by controlling who has access, when, how long, and under what conditions by centralizing disparate hardware and software services for cloud, enterprise, mobile, and business-to-business (B2B) systems.

Figure 1.2, "OpenAM Architecture" illustrates the OpenAM architecture.

Figure 1.2. OpenAM Architecture



OpenAM features a highly modular and flexible architecture with multiple plugin points to meet any customer deployment. It leverages industry standard protocols, such as HTTP, XML, SOAP, REST, SAML 2.0, OAuth 2.0, OpenID Connect 1.0, and so forth to deliver a high performance, highly scalable, and highly available access management solution over the network. OpenAM services are 100% Java-based, proven across multiple platforms and containers in many production deployments.

OpenAM core server can be deployed and integrated within existing network infrastructures. OpenAM provides the following distribution files:

Table 1.1. OpenAM Distribution Files

Distribution	Description
ClientSDK-13.5.1-5.jar	OpenAM's default distribution `.war` file includes the core server code with an embedded OpenDJ directory server, which stores configuration data and simplifies deployments. The distribution includes an administrative graphical user interface (GUI) Web console. During installation, the `.war` file accesses a bootstrap file to obtain the fully qualified domain name, port, context path, and the location of the configuration folder. OpenAM provides a client SDK

Distribution	Description
	for developers to code extensions for their web applications to access OpenAM's services. The client SDK contains the Java packages, classes, property files, and sample code to help you develop your code.
ExampleClientSDK-CLI-13.5.1-5.zip	OpenAM provides client SDK examples to help you run them on OpenAM. The `zip` distribution file contains setup scripts and samples that you can run to learn how to use the client SDK.
ExampleClientSDK-WAR-13.5.1-5.war	The example client SDK also comes in a `.war` file, which installs on your container.
Fedlet-13.5.1-5.zip	OpenAM provides a core server-only `.war` file without the OpenAM Console. This setup is often used in multi-server deployments wherein the deployments is managed using a full server instance using the **ssoadm** command-line tool. The OpenAM server installs with an embedded OpenDJ directory server. OpenAM provides an OpenAM Fedlet, a light-weight SAML v2.0 service provider. The Fedlet lets you set up a federated deployment without the need of a fully-featured service provider.
IDPDiscovery-13.5.1-5.war	OpenAM provides an IdP Discovery Profile (SAMLv2 binding profile) for its IdP Discovery service. The profile keeps track of the identity providers for each user.
OpenAM-13.5.1-5.war	OpenAM provides a smaller subset of the OpenAM SDK, providing client-side API to OpenAM services. The Client SDK allows you to write remote standalone or Web applications to access OpenAM and run its core services. OpenAM's distribution `.war` file includes the core server code with an embedded OpenDJ directory server, which stores configuration data and simplifies deployments. The distribution includes an administrative graphical user interface (GUI) Web console. During installation, the `.war` file accesses properties to obtain the fully qualified domain name, port, context path, and the location of the configuration folder. These properties can be obtained from the `boot.properties` file in the OpenAM installation directory, from environment variables, or from a combination of the two.
openam-soap-sts-server-13.5.1-5.war	OpenAM provides a SOAP-based security token service (STS) server that issues tokens based on the WS-Security protocol[a].
SSOAdminTools-13.5.1-5.zip	OpenAM provides an `ssoadm` command-line tool that allows administrators to configure and maintain OpenAM as well as create their own configuration scripts. The `zip` distribution file contains binaries,

Distribution	Description
	properties file, script templates, and setup scripts for UNIX and windows servers.
SSOConfiguratorTools-13.5.1-5.zip	OpenAM provides configuration and upgrade tools for installing and maintaining your server. The `zip` distribution file contains libraries, legal notices, and supported binaries for these configuration tools. Also, you can view example configuration and upgrade properties files that can be used as a template for your deployments.

[a]OpenAM also provides REST-based STS service endpoints, which you can directly utilize on the OpenAM server.

ForgeRock's OpenAM product is built on open-source code. ForgeRock maintains the OpenAM product, providing the community an open-source code repository, issue tracking, mailing lists, and web sites. ForgeRock also provides commercial releases of fully tested builds. ForgeRock offers the services you need to deploy OpenAM commercial builds into production, including training, consulting, and support.

1.4. OpenAM Key Benefits

The goal of OpenAM is to provide secure, low friction access to valued resources while presenting the user with a consistent experience. OpenAM provides excellent security, which is totally transparent to the user.

OpenAM provides the following key benefits to your organization:

- **Enables Solutions for Additional Revenue Streams**. OpenAM provides the tools and components to quickly deploy services to meet customer demand. For example, OpenAM's Federation Services supports quick and easy deployment with existing SAMLv2, OAuth2, and OpenID Connect systems. For systems that do not support a full SAMLv2 deployment, OpenAM provides a *Fedlet*, a small SAML 2.0 application, which lets service providers quickly add SAML 2.0 support to their Java applications. These solutions open up new possibilities for additional revenue streams.
- **Reduces Operational Cost and Complexity**. OpenAM can function as a hub, leveraging existing identity infrastructures and providing multiple integration paths using its authentication, SSO, and policies to your applications without the complexity of sharing Web access tools and passwords for data exchange. OpenAM decreases the total cost of ownership (TCO) through its operational efficiencies, rapid time-to-market, and high scalability to meet the demands of our market.
- **Improves User Experience**. OpenAM enables users to experience more services using SSO without the need of multiple passwords.
- **Easier Configuration and Management**. OpenAM centralizes the configuration and management of your access management system, allowing easier administration through its console and command-line tools. OpenAM also features a flexible deployment architecture that unifies services through its modular and embeddable components. OpenAM provides a common REST framework and common user interface (UI) model, providing scalable solutions as your customer base increases

to the hundreds of millions. OpenAM also allows enterprises to outsource IAM services to system integrators and partners.

- **Increased Compliance**. OpenAM provides an extensive entitlements service, featuring attribute-based access control (ABAC) policies as its main policy framework with features like import/export support to XACML, a policy editor, and REST endpoints for policy management. OpenAM also includes an extensive auditing service to monitor access according to regulatory compliance standards.

1.5. OpenAM History

OpenAM's timeline is summarized as follows:

- In 2001, Sun Microsystems releases iPlanet Directory Server, Access Management Edition.
- In 2003, Sun renames iPlanet Directory Server, Access Management Edition to Sun ONE Identity Server.
- Later in 2003, Sun acquires Waveset.
- In 2004, Sun releases Sun Java Enterprise System. Waveset Lighthouse is renamed to Sun Java System Identity Manager and Sun ONE Identity Server is renamed to Sun Java System Access Manager. Both products are included as components of Sun Java Enterprise System.
- In 2005, Sun announces an open-source project, OpenSSO, based on Sun Java System Access Manager.
- In 2008, Sun releases OpenSSO build 6, a community open-source version, and OpenSSO Enterprise 8.0, a commercial enterprise version.
- In 2009, Sun releases OpenSSO build 7 and 8.
- In January 2010, Sun was acquired by Oracle and development for the OpenSSO products were suspended as Oracle no longer planned to support the product.

In February 2010, a small group of former Sun employees founded ForgeRock to continue OpenSSO support, which was renamed to OpenAM. ForgeRock continued OpenAM's development with the following releases:

- 2010: OpenAM 9.0
- 2011: OpenAM 9.5
- 2012: OpenAM 10 and 10.1
- 2013: OpenAM 11.0
- 2014: OpenAM 11.1 and 12.0

ForgeRock continues to develop, enhance, and support the industry-leading OpenAM product to meet the changing and growing demands of the market.

ForgeRock also took over responsibility for support and development of the OpenDS directory server, which was renamed as OpenDJ. ForgeRock plans to continue to maintain, enhance, and support OpenDJ.

Chapter 2

Planning the Deployment Architecture

This chapter presents the elements involved in planning an OpenAM deployment architecture.

2.1. Importance of Deployment Planning

Deployment planning is critical to ensuring your OpenAM system is properly implemented within the time frame determined by your requirements. The more thoroughly you plan your deployment, the more solid your configuration will be, and you will meet timelines and milestones while staying within budget.

A deployment plan defines the goals, scope, roles, and responsibilities of key stakeholders, architecture, implementation, and testing of your OpenAM deployment. A good plan ensures that a smooth transition to a new product or service is configured and all possible contingencies are addressed to quickly troubleshoot and solve any issue that may occur during the deployment process. The deployment plan also defines a training schedule for your employees, procedural maintenance plans, and a service plan to support your OpenAM system.

2.2. Deployment Planning Considerations

When planning a deployment, you must consider some important questions regarding your system.

- **What are you protecting?** You must determine which applications, resources, and levels of access to protect? Are there plans for additional services, either developed in-house or through future acquisitions that also require protected access?

- **How many users are supported?** It is important to determine the number of users supported in your deployment based on system usage. Once you have determined the number of users, it is important to project future growth.

- **What are your product service-level agreements?** In addition to planning for the growth of your user base, it is important to determine the production service-level agreements (SLAs) that help determine the current load requirements on your system and for future loads. The SLAs help define your scaling and high-availability requirements.

  For example, suppose you have 100,000 active users today, and each user has an average of two devices (laptop, phone) that get a session each day. Suppose that you also have 20 protected applications, with each device hitting an average of seven protected resources an average of 1.4 times daily. Let's say that works out to about 200,000 sessions per day with 7 x 1.4 = ~10 updates to each session object. This can result in 200K session creations, 200K session deletions, and 2M session updates.

  Now, imagine next year you still have the same number of active users, 100K, but each has an average of three devices (laptop, phone, tablet), and you have added another 20 protected applications. Assume the same average usage per application per device, or even a little less per device. You can see that although the number of users is unchanged, the whole system needs to scale up considerably.

  You can scale your deployment using vertical or horizontal scaling. Vertical scaling involves increasing components to a single host server, such as increasing the number of CPUs or increasing heap memory to accommodate the number of users in your system. Horizontal scaling involves adding additional host servers, possibly behind a load balancer, so that the servers can function as a single unit. Configuring OpenAM to issue stateless sessions greatly simplifies deployment when using horizontal scaling.

- **What are your high availability requirements?** High availability refers to your system's ability to operate continuously for a specified length of time. It is important to design your system to prevent single points of failure and for continuous availability. Based on the size of your deployment, you can create an architecture using a single-site configuration. For larger deployments, consider implementing a multi-site configuration with replication over WAN in combination with one of the following two alternatives:

  - Stateless sessions

  - Sticky load-balancing and session failover

Which type of clients will be supported? The type of client determines the components required for the deployment. For example, applications supported in a Web server require the use of a Web policy agent. Applications deployed in Java EE containers require the use of a Java EE policy agent. An AJAX application can use the OpenAM's RESTful API. Legacy or custom applications can use the OpenIG Identity Gateway. Applications in an unsupported application server can use a reverse proxy with a policy agent. Third party applications can use federation or a fedlet, or an OpenID Connect or an OAuth 2.0 component.

- **What are your SSL/TLS requirements?** There are two common approaches to handling SSL. First, using SSL through to the application servers themselves, for example, using SSL on the containers. Or second, using SSL offloading via a network device and running HTTP clear internally. You must determine the appropriate approach as each method requires different configurations. Determining SSL use early in the planning process is vitally *important*, as adding SSL later in the process is more complicated and could result in delays in your deployment.
- **What are your other security requirements?** The use of firewalls provides an additional layer of security for your deployment. If you are planning to deploy the OpenAM server behind a firewall, you can deploy a reverse proxy, such as OpenIG. For another level of security, consider using multiple DNS infrastructures using zones; one zone for internal clients, another zone for external clients. To provide additional performance, you can deploy the DNS zones behind a load balancer.
- **Ensure all stakeholders are engaged during the planning phase**. This effort includes but is not limited to delivery resources, such as project managers, architects, designers, implementers, testers, and service resources, such as service managers, production transition managers, security, support, and sustaining personnel. Input from all stakeholders ensures all viewpoints are considered at project inception, rather than downstream, when it may be too late.

2.3. Deployment Planning Steps

The general deployment planning steps can be summarized as follows:

- **Project Initiation**. The Project Initiation phase begins by defining the overall scope and requirements of the deployment. The following items can be planned:
  - Determine the scope, roles and responsibilities of key stakeholders and resources required for the deployment.
  - Determine critical path planning including any dependencies and their assigned expectations.
  - Run a pilot to test the functionality and features of OpenAM and uncover any possible issues early in the process.
  - Determine training for administrators of the environment and training for developers, if needed.
- **Architecting**. The Architecting phase involves designing the deployment. The following items can be planned:

- Determine the use of products, map requirements to features, and ensure the architecture meets the functional requirements.
- Ensure that the architecture is designed for ease of management and scale. TCO is directly proportional to the complexity of the deployment.
- Determine how the Identity, Configuration, and Core Token Service (CTS) data stores are to be configured.
- Determine the Sites and Session Failover configuration.
- Determine where SSL is used in the configuration and how to maintain and update the certificate keystore and truststore for OpenAM's components, such as the agent installer, **ssoadm** tool, agent server, and other OpenAM servers. Planning for SSL at this point can avoid more difficulty later in the process.
- Determine if OpenAM will be deployed behind a load balancer with SSL offloading. If this is the case, you must ensure that the load balancer rewrites the protocol during redirection. If you have a policy agent behind a load balancer with SSL offloading, ensure that you set the policy agent's override request URL properties.
- For multiple OpenAM deployments, there is a requirement to deploy a layer 7 cookie-based load balancer and intelligent keep-alives (for example, `/openam/isAlive.jsp`). The network teams should design the appropriate solution in the architecting phase.
- Determine requirements for vertical scaling, which involves increasing the Java heap based on anticipated session, policy cache, federation session, and restricted token usage. Note that vertical scaling could come with performance cost, so this must be planned accordingly.
- Determine requirements for horizontal scaling, which involves adding additional OpenAM servers and load balancers for scalability and availability purposes.
- Determine whether to configure OpenAM to issue stateful or stateless sessions. Stateless sessions allow for easier horizontal scaling but do not provide equivalent functionality to stateful sessions.
- Determine if any coding is required including extensions and plugins. Unless it is absolutely necessary, leverage the product features instead of implementing custom code. OpenAM provides numerous plugin points and REST endpoints.

- **Implementation**. The Implementation phase involves deploying your OpenAM system. The following items should be considered:
  - Install and configure the OpenAM server, datastores, and components. For information on installing OpenAM, see the Installation Guide.
  - Maintain a record and history of the deployment to maintain consistency across the project.

  - Tune OpenAM's JVM, caches, LDAP connection pools, container thread pools, and other items. For information on tuning OpenAM, see Chapter 25, "*Tuning OpenAM*" in the *Administration Guide*.

  - Tune the OpenDJ directory server. Consider tuning the database back end, replication purge delays, garbage collection, JVM memory, and disk space considerations. For more information, see the OpenDJ directory server documentation.

  - Consider implementing separate file systems for both OpenAM and OpenDJ, so that you can keep log files on a different disk, separate from data or operational files, to prevent device contention should the log files fill up the file system.

- **Automation and Continuous Integration**. The Automation and Continuous Integration phase involves using tools for testing:

  - Set up a continuous integration server, such as Jenkins, to ensure that builds are consistent by running unit tests and publishing Maven artifacts. Perform continuous integration unless your deployment includes no customization.

  - Ensure your custom code has unit tests to ensure nothing is broken.

- **Functional Testing**. The Functional Testing phase should test all functionality to deliver the solution without any failures. You must ensure that your customizations and configurations are covered in the test plan.

- **Non-Functional Testing**. The Non-Functional Testing phase tests failover and disaster recovery procedures. Run load testing to determine the demand of the system and measure its responses. You can anticipate peak load conditions during the phase.

- **Supportability**. The Supportability Phase involves creating the runbook for system administrators including procedures for backup and restores, debugging, change control, and other processes. If you have a ForgeRock Support contract, it ensures everything is in place prior to your deployment.

2.4. Preparing Deployment Plans

When you create a good concrete deployment plan, it ensures that a change request process is in place and utilized, which is essential for a successful deployment. This section looks at planning the full deployment process. When you have addressed everything in this section, then you should have a concrete plan for deployment.

2.4.1. Planning Training

Training provides common understanding, vocabulary, and basic skills for those working together on the project. Depending on previous experience with access management and with OpenAM, both internal teams and project partners might need training.

The type of training team members need depends on their involvement in the project:

- All team members should take at least some training that provides an overview of OpenAM. This helps to ensure a common understanding and vocabulary for those working on the project.
- Team members planning the deployment should take an OpenAM deployment training before finalizing your plans, and ideally before starting to plan your deployment.

  OpenAM not only offers a broad set of features with many choices, but the access management it provides tends to be business critical. OpenAM deployment training pays for itself as it helps you to make the right initial choices to deploy more quickly and successfully.
- Team members involved in designing and developing OpenAM client applications or custom extensions should take training in OpenAM development in order to help them make the right choices. This includes developers customizing the OpenAM UI for your organization.
- Team members who have already had been trained in the past might need to refresh their knowledge if your project deploys newer or significantly changed features, or if they have not worked with OpenAM for some time.

ForgeRock University regularly offers training courses for OpenAM topics, including OpenAM development and deployment. For a current list of available courses, see http://forgerock.com/services/university/.

When you have determined who needs training and the timing of the training during the project, prepare a training schedule based on team member and course availability. Include the scheduled training plans in your deployment project plan.

ForgeRock also offers an accreditation program for partners, offering an in-depth assessment of business and technical skills for each ForgeRock product. This program is open to the partner community and ensures that best practices are followed during the design and deployment phases.

2.4.2. Planning Customization

When you customize OpenAM, you can improve how the software fits your organization. OpenAM customizations can also add complexity to your system as you increase your test load and potentially change components that could affect future upgrades. Therefore, a best practice is to deploy OpenAM with a minimum of customizations.

Most deployments require at least some customization, like skinning end user interfaces for your organization, rather than using the OpenAM defaults. If your deployment is expected to include additional client applications, or custom extensions (authentication modules, policy conditions, and so forth), then have a team member involved in the development help you plan the work. The Developer's Guide can be useful when scoping a development project.

Although some customizations involve little development work, it can require additional scheduling and coordination with others in your organization. An example is adding support for profile attributes in the identity repository.

The more you customize, the more important it is to test your deployment thoroughly before going into production. Consider each customization as sub-project with its own acceptance criteria, and consider plans for unit testing, automation, and continuous integration. See Section 2.4.8, "Planning Tests" for details.

When you have prepared plans for each customization sub-project, you must account for those plans in your overall deployment project plan. Functional customizations, such as custom authentication modules or policy conditions might need to reach the pilot stage before you can finish an overall pilot implementation.

2.4.3. Planning a Pilot Implementation

Unless you are planning a maintenance upgrade, consider starting with a pilot implementation, which is a long term project that is aligned with customer-specific requirements.

A pilot shows that you can achieve your goals with OpenAM plus whatever customizations and companion software you expect to use. The idea is to demonstrate feasibility by focusing on solving key use cases with minimal expense, but without ignoring real-world constraints. The aim is to fail fast before you have too much invested so that you can resolve any issues that threaten the deployment.

Do not expect the pilot to become the first version of your deployment. Instead, build the pilot as something you can afford to change easily, and to throw away and start over if necessary.

The cost of a pilot should remain low compared to overall project cost. Unless your concern is primarily the scalability of your deployment, you run the pilot on a much smaller scale than the full deployment. Scale back on anything not necessary to validating a key use case.

Smaller scale does not necessarily mean a single-server deployment, though. If you expect your deployment to be highly available, for example, one of your key use cases should be continued smooth operation when part of your deployment becomes unavailable.

The pilot is a chance to try and test features and services before finalizing your plans for deployment. The pilot should come early in your deployment plan, leaving appropriate time to adapt your plans based on the pilot results. Before you can schedule the pilot, team members might need training and you might require prototype versions of functional customizations.

Plan the pilot around the key use cases that you must validate. Make sure to plan the pilot review with stakeholders. You might need to iteratively review pilot results as some stakeholders refine their key use cases based on observations.

2.4.4. Planning Security Hardening

When you first configure OpenAM, there are many options to evaluate, plus a number of ways to further increase levels of security. You can change the following default configuration properties:

- The main OpenAM administrative account has a default user name, `amadmin`.
- You can set up OpenAM using HTTPS rather than HTTP.
- An OpenAM policy agent administrative account exists and has a default user name, `UrlAccessAgent`.
- The primary session cookie has a default name, `iPlanetDirectoryPro`.
- Initially, only the top-level realm exists. Other realms and fully qualified domain name realm/DNS aliases must be configured separately.
- The top-level realm includes a demo user, `demo`, with the default password `changeit`.
- Default keystores exist in the *`config-dir`*`/openam/` path, with several self-signed keys and identical passwords. For more information about the default keystores in OpenAM and their demo key aliases, see Chapter 23, "*Managing Certificates and Keystores*" in the *Administration Guide*.
- By default, OpenAM connects to directory servers as the directory root user, `cn=Directory Manager`.
- By default, the list of `goto` and `gotoOnFail` URLs is not restricted.
- On a server that includes OpenAM Console, all the endpoints defined in the Web application descriptor, `WEB-INF/web.xml`, are available for use.
- To prevent cross-site scripting attacks, you can configure session cookies as HTTP Only by setting the property `com.sun.identity.cookie.httponly=true`. This property prevents third-party scripts from accessing the session cookie.
- You can deploy a reverse proxy within delimitarized zone (DMZ) firewalls to limit exposure of service URLs to the end user as well as block access to back end configuration and user data stores to unauthorized users.

You must therefore plan to secure the deployment as described in Chapter 27, "*Securing OpenAM*" in the *Administration Guide*.

At minimum, make sure you include the following tasks in the overall plan:

- Change default settings and administrative user credentials.
- Protect service ports (using firewalls, install a reverse proxy).
- Disable unused endpoints.
- Separate administrative access from client access.
- Secure communications so that OpenAM clients access services over HTTPS.

- Use secure cookies with cookie hijacking protection for CDSSO, so messages for federated configurations perform signing and encryption as necessary, and OpenAM accesses providers securely (for example using LDAP + StartTLS, or LDAPS to access directory services).
- Secure processes and files (for example with SELinux, using a dedicated non-privileged user and port forwarding, and so forth).

2.4.5. Planning With Providers

OpenAM delegates authentication and profile storage to other services. OpenAM can store configuration, policies, session, and other tokens in an external directory service. OpenAM can also participate in a circle of trust with other SAML entities. In each of these cases, a successful deployment depends on coordination with service providers, potentially outside of your organization.

The infrastructure you need to run OpenAM services might be managed outside your own organization. Hardware, operating systems, network, and software installation might be the responsibility of providers with which you must coordinate.

When working with providers, take the following points into consideration.

- Shared authentication and profile services might have been sized prior to or independently from your access management deployment.

  An overall outcome of your access management deployment might be to decrease the load on shared authentication services (and replace some authentication load with single-sign on that is managed by OpenAM), or it might be to increase the load (if, for example, your deployment enables many new applications or devices, or enables controlled access to resources that were previously unavailable).

  Identity repositories are typically backed by shared directory services. Directory services might need to provision additional attributes for OpenAM. This could affect not only directory schema and access for OpenAM, but also sizing for the directory services that your deployment uses.

- If your deployment uses an external directory service for OpenAM configuration data and OpenAM policies, then the directory administrator must include attributes in the schema and provide access rights to OpenAM. The number of policies depends on the deployment. For deployments with thousands or millions of policies to store, OpenAM's use of the directory could affect sizing.
- If your deployment uses an external directory service as a backing store for the OpenAM Core Token Service (CTS), then the directory administrator must include attributes in the schema and provide access rights to OpenAM.

  CTS load tends to involve more write operations than configuration and policy load, as CTS data tend to be more volatile, especially if most tokens concern short-lived sessions. This can affect directory service sizing.

  CTS can enable cross-site session failover by allowing a remote OpenAM server to retrieve a user session from the directory service backing the CTS. For this feature to work quickly in the event of

a failure or network partition, CTS data must be replicated rapidly including across WAN links. This can affect network sizing for the directory service.

When configured to issue stateless sessions, OpenAM does *not* write the stateless sessions to CTS. Instead, OpenAM uses CTS for session blacklists. Session blacklisting is an optional OpenAM feature that provides logout integrity.

- SAML federation circles of trust require organizational and legal coordination before you can determine what the configuration looks like. Organizations must agree on which security data they share and how, and you must be involved to ensure that their expectations map to the security data that is actually available.

  There also needs to be coordination between all SAML parties, (that is, agreed-upon SLAs, patch windows, points of contact and escalation paths). Often, the technical implementation is considered, but not the *business requirements*. For example, a common scenario occurs when a service provider takes down their service for patching without informing the identity provider or vice-versa.

- When working with infrastructure providers, realize that you are likely to have better sizing estimates after you have tried a test deployment under load. Even though you can expect to revise your estimates, take into account the lead time necessary to provide infrastructure services.

  Estimate your infrastructure needs not only for the final deployment, but also for the development, pilot, and testing stages.

For each provider you work with, add the necessary coordinated activities to your overall plan, as well as periodic checks to make sure that parallel work is proceeding according to plan.

2.4.6. Planning Integration With Client Applications

When planning integration with OpenAM client applications, the applications that are most relevant are those that register with OpenAM; therefore, you should make note of the following types of client applications registering with OpenAM:

- OpenAM policy agents reside with the applications they protect.

  By default, OpenAM policy agents store their configuration profiles in OpenAM. OpenAM then sends policy agents notifications when their configurations change.

  Policy agents authenticate to OpenAM with a user name and password.

  To delegate administration of multiple policy agents, OpenAM lets you create a group profile for each realm to register the policy agent profiles.

  While the OpenAM administration manages policy agent configuration, application administrators are often the ones who install policy agents. You must coordinate installation and upgrades with them.

- OAuth 2.0 clients and OpenID Connect 1.0 relying parties also register profiles with OpenAM.

OpenAM optionally allows registration of such applications without prior authentication. By default, however, registration requires an access token granted to an OAuth 2.0 client with access to register profiles.

If you expect to allow dynamic registration, or if you have many clients registering with your deployment, then consider clearly documenting how to register the clients, and building a client to register clients.

- You must configure Circles of Trust for SAML 2.0 federations, so registration happens at configuration time, rather than at runtime.

  Address the necessary configuration as described in Section 2.4.5, "Planning With Providers".

  If your deployment functions as a SAML 2.0 Identity Provider (IDP) and shares Fedlets with Service Providers (SP), the SP administrators must install the Fedlets, and must update their Fedlets for changes in your IDP configuration. Consider at least clearly documenting how to do so, and if necessary, build installation and upgrade capabilities.

- If you have custom client applications, consider how they are configured and how they must register with OpenAM.

- REST API client applications authenticate based on a user profile.

  REST client applications can therefore authenticate using whatever authentication mechanisms you configure in OpenAM, and therefore do not require additional registration.

For each client application whose integration with OpenAM requires coordination, add the relevant tasks to your overall plan.

2.4.7. Planning Integration With Audit Tools

OpenAM and policy agents can log audit information to flat files or alternatively, to a relational database. Log volumes depend on usage and on logging levels. By default, OpenAM generates both access and error messages for each service, providing the raw material for auditing the deployment. The Reference covers what you can expect to find in log messages.

In order to analyze the raw material, however, you must use other software, such as Splunk, which indexes machine-generated data for analysis.

If you require integration with an audit tool, plan the tasks of setting up logging to work with the tool, and analyzing and monitoring the data once it has been indexed. Consider how you must retain and rotate log data once it has been consumed, as a high volume service can produce large volumes of log data.

Include these plans in the overall plan.

2.4.8. Planning Tests

In addition to planning tests for each customized component, test the functionality of each service you deploy, such as authentication, policy decisions, and federation. You should also perform non-functional testing to validate that the services hold up under load in realistic conditions. Perform penetration testing to check for security issues. Include acceptance tests for the actual deployment. The data from the acceptance tests help you to make an informed decision about whether to go ahead with the deployment or to roll back.

2.4.8.1. Planning Functional Testing

Functional testing validates that specified test cases work with the software considered as a black box.

As ForgeRock already tests OpenAM and policy agents functionally, focus your functional testing on customizations and service-level functions. For each key service, devise automated functional tests. Automated tests make it easier to integrate new deliveries to take advantage of recent bug fixes and to check that fixes and new features do not cause regressions.

Tools for running functional testing include Apache JMeter and Selenium. Apache JMeter is a load testing tool for Web applications. Selenium is a test framework for Web applications, particularly for UIs.

As part of the overall plan, include not only tasks to develop and maintain your functional tests, but also to provision and to maintain a test environment in which you run the functional tests before you significantly change anything in your deployment. For example, run functional tests whenever you upgrade OpenAM, OpenAM policy agents, or any custom components, and analyze the output to understand the effect on your deployment.

2.4.8.2. Planning Service Performance Testing

For written service-level agreements and objectives, even if your first version consists of guesses, you turn performance plans from an open-ended project to a clear set of measurable goals for a manageable project with a definite outcome. Therefore, start your testing with clear definitions of success.

Also, start your testing with a system for load generation that can reproduce the traffic you expect in production, and provider services that behave as you expect in production. To run your tests, you must therefore generate representative load data and test clients based on what you expect in production. You can then use the load generation system to perform iterative performance testing.

Iterative performance testing consists in identifying underperformance and the bottlenecks that cause it, and discovering ways to eliminate or work around those bottlenecks. Underperformance means that the system under load does not meet service level objectives. Sometimes re-sizing and/or tuning the system or provider services can help remove bottlenecks that cause underperformance.

Based on service level objectives and availability requirements, define acceptance criteria for performance testing, and iterate until you have eliminated underperformance.

Tools for running performance testing include Apache JMeter, for which your loads should mimic what you expect in production, and Gatling, which records load using a domain specific language for load testing. To mimic the production load, examine both the access patterns and also the data that OpenAM stores. The representative load should reflect the expected random distribution of client access, so that sessions are affected as in production. Consider authentication, authorization, logout, and session timeout events, and the lifecycle you expect to see in production.

Although you cannot use actual production data for testing, you can generate similar test data using tools, such as the OpenDJ **makeldif** command, which generates user profile data for directory services. OpenAM REST APIs can help with test provisioning for policies, users, and groups.

As part of the overall plan, include not only tasks to develop and maintain performance tests, but also to provision and to maintain a pre-production test environment that mimics your production environment. Security measures in your test environment must also mimic your production environment, as changes to secure OpenAM as described in Section 2.4.4, "Planning Security Hardening", such as using HTTPS rather than HTTP, can impact performance.

Once you are satisfied that the baseline performance is acceptable, run performance tests again when something in your deployment changes significantly with respect to performance. For example, if the load or number of clients changes significantly, it could cause the system to underperform. Also, consider the thresholds that you can monitor in the production system to estimate when your system might start to underperform.

2.4.8.3. Planning Penetration Testing

Penetration testing involves attacking a system to expose security issues before they show up in production.

When planning penetration testing, consider both white box and black box scenarios. Attackers can know something about how OpenAM works internally, and not only how it works from the outside. Also, consider both internal attacks from within your organization, and external attacks from outside your organization.

As for other testing, take time to define acceptance criteria. Know that ForgeRock has performed penetration testing on the software for each enterprise release. Any customization, however, could be the source of security weaknesses, as could configuration to secure OpenAM.

You can also plan to perform penetration tests against the same hardened, pre-production test environment also used for performance testing.

2.4.8.4. Planning Deployment Testing

Deployment testing is used as a description, and not a term in the context of this guide. It refers to the testing implemented within the deployment window after the system is deployed to the production environment, but before client applications and users access the system.

Plan for minimal changes between the pre-production test environment and the actual production environment. Then test that those changes have not cause any issues, and that the system generally behaves as expected.

Take the time to agree upfront with stakeholders regarding the acceptance criteria for deployment tests. When the production deployment window is small, and you have only a short time to deploy and test the deployment, you must trade off thorough testing for adequate testing. Make sure to plan enough time in the deployment window for performing the necessary tests and checks.

Include preparation for this exercise in your overall plan, as well as time to check the plans close to the deployment date.

2.4.9. Planning Documentation and Tracking Changes

The OpenAM product documentation is written for readers like you, who are architects and solution developers, as well as for OpenAM developers and for administrators who have had OpenAM training. The people operating your production environment need concrete documentation specific to your deployed solution, with an emphasis on operational policies and procedures.

Procedural documentation can take the form of a runbook with procedures that emphasize maintenance operations, such as backup, restore, monitoring and log maintenance, collecting data pertaining to an issue in production, replacing a broken server or policy agent, responding to a monitoring alert, and so forth. Make sure in particular that you document procedures for taking remedial action in the event of a production issue.

Furthermore, to ensure that everyone understands your deployment and to speed problem resolution in the event of an issue, changes in production must be documented and tracked as a matter of course. When you make changes, always prepare to roll back to the previous state if the change does not perform as expected.

Include documentation tasks in your overall plan. Also, include the tasks necessary to put in place and to maintain change control for updates to the configuration.

2.4.10. Planning Maintenance and Support in Production

If you own the architecture and planning, but others own the service in production, or even in the labs, then you must plan coordination with those who own the service.

Start by considering the service owners' acceptance criteria. If they have defined support readiness acceptance criteria, you can start with their acceptance criteria. You can also ask yourself the following questions:

- What do they require in terms of training in OpenAM?
- What additional training do they require to support your solution?
- Do your plans for documentation and change control, as described in Section 2.4.9, "Planning Documentation and Tracking Changes", match their requirements?

- Do they have any additional acceptance criteria for deployment tests, as described in Section 2.4.8.4, "Planning Deployment Testing"?

Also, plan back line support with ForgeRock or a qualified partner. The aim is to define clearly who handles production issues, and how production issues are escalated to a product specialist if necessary.

Include a task in the overall plan to define the hand off to production, making sure there is clarity on who handles monitoring and issues.

2.4.11. Planning Rollout Into Production

In addition to planning for the hand off of the production system, also prepare plans to roll-out the system into production. Rollout into production calls for a well-choreographed operation, so these are likely the most detailed plans.

Take at least the following items into account when planning the rollout:

- Availability of all infrastructure that OpenAM depends upon the following elements:
  - Server hosts and operating systems
  - Web application containers
  - Network links and configurations
  - Load balancers
  - Reverse proxy services to protect OpenAM
  - Data stores, such as directory services
  - Authentication providers
- Installation for all OpenAM services.
- Installation of OpenAM client applications:
  - Policy agents
  - Fedlets
  - SDK applications
  - OAuth 2.0 applications
  - OpenID Connect 1.0 applications
- Final tests and checks.

- Availability of the personnel involved in the rollout.

In your overall plan, leave time and resources to finalize rollout plans toward the end of the project.

2.4.12. Planning for Growth

Before rolling out into production, plan how to monitor the system to know when you must grow, and plan the actions to take when you must add capacity.

Unless your system is embedded or otherwise very constrained, after your successful rollout of access management services, you can expect to add capacity at some point in the future. Therefore, you should plan to monitor system growth.

You can grow many parts of the system by adding servers or adding clients. The parts of the system that you cannot expand so simply are those parts that depend on writing to the directory service, and those that can result in crosstalk between OpenAM servers.

The directory service eventually replicates each write to all other servers. Therefore, adding servers simply adds the number of writes to perform. One simple way of getting around this limitation to working with the hierarchical nature of directory data to split a monolithic directory service into several. That said, directory services often are not a bottleneck for growth.

Crosstalk between OpenAM servers can result when one OpenAM server authenticates a user, and a subsequent request regarding that user is sent to a second OpenAM server. In that case, the second server can communicate with the first server to handle the request, resulting in crosstalk from one server to another. A load balancing solution that offers server affinity or stickiness reduces crosstalk and contributes to a system that grows more smoothly.

When should you expand the deployed system? The time to expand the deployed system is when growth in usage causes the system to approach performance threshold levels that cause the service to underperform. For that reason, devise thresholds that can be monitored in production, and plan to monitor the deployment with respect to the thresholds. In addition to programming appropriate alerts to react to thresholds, also plan periodic reviews of system performance to uncover anything missing from regular monitoring results.

2.4.13. Planning for Upgrades

In this section, "upgrade" means moving to a more recent release, whether it is a patch, maintenance release, minor release, or major release. For definitions of the types of release, see Appendix A, "*Release Levels and Interface Stability*" in the *Administration Guide*.

Upgrades generally bring fixes, or new features, or both. For each upgrade, you must build a new plan. Depending on the scope of the upgrade, that plan might include almost all of the original overall plan, or it might be abbreviated, for example, for a patch that fixes a single issue. In any case, adapt deployment plans, as each upgrade is a new deployment.

When planning an upgrade, pay particular attention to testing and to any changes necessary in your customizations. For testing, consider compatibility issues when not all agents and services are

upgraded simultaneously. Choreography is particularly important, as upgrades are likely to happen in constrained low usage windows, and as users already have expectations about how the service should behave.

When preparing your overall plan, include a regular review task to determine whether to upgrade, not only for patches or regular maintenance releases, but also to consider whether to upgrade to new minor and major releases.

2.4.14. Planning for Disaster Recovery

Disaster recovery planning and a robust backup strategy is essential when server hardware fails, network connections go down, a site fails, and so on. Your team must determine the disaster recovery procedures to recover from such events.

Chapter 3

Example Deployment Topology

You can configure OpenAM in a wide variety of deployments depending on your security requirements and network infrastructure. This chapter presents an example enterprise deployment, featuring a highly available and scalable architecture across multiple data centers.

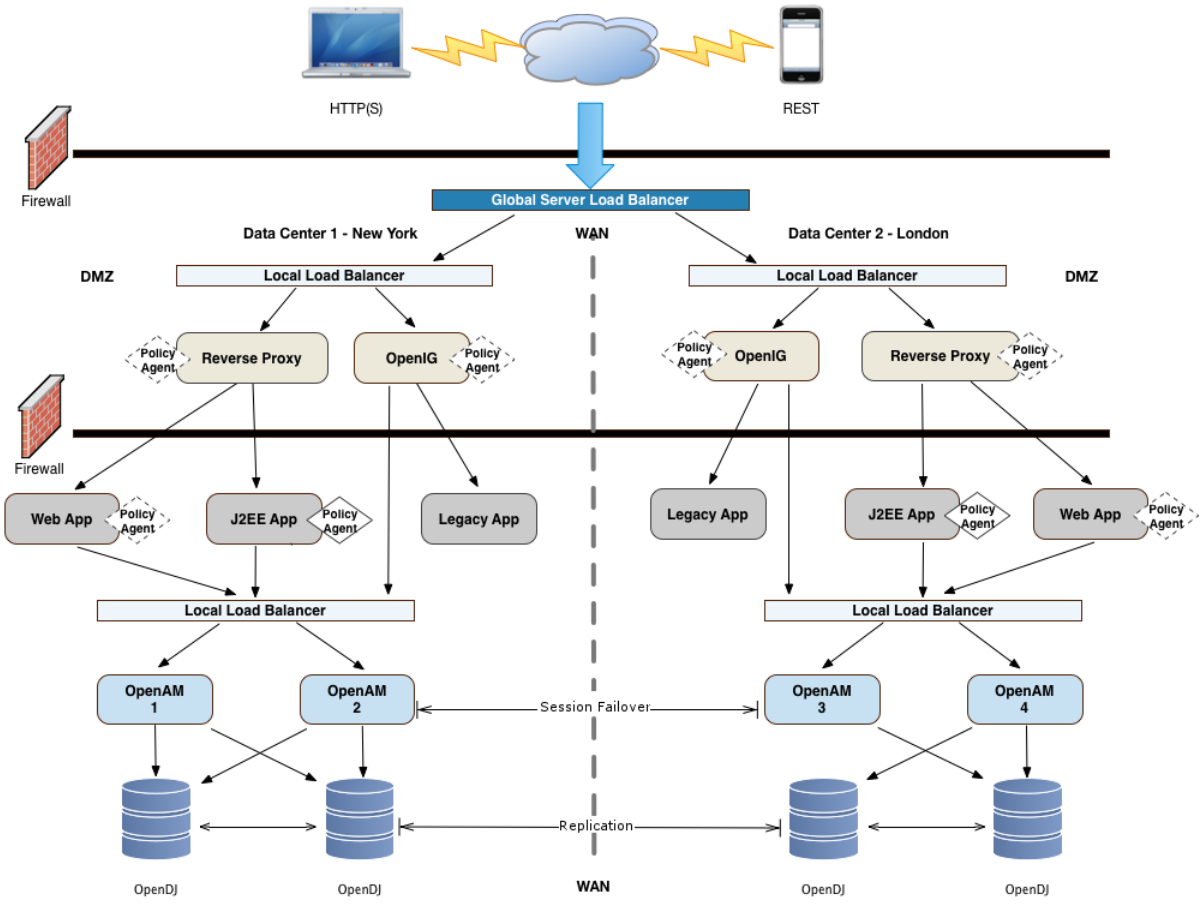
3.1. About the Example Topology

Figure 3.1, "OpenAM Deployment Example" presents an example topology of a multi-city multi-data-center deployment across a wide area network (WAN). The example deployment is partitioned into a two-tier architecture. The top tier is a DMZ with the initial firewall securing public traffic into the network. The second firewall limits traffic from the DMZ into the application tier where the protected resources are housed.

Note

The example components in this chapter are presented for illustrative purposes. ForgeRock does not recommend specific products, such as reverse proxies, load balancers, switches, firewalls, and so forth, as OpenAM can be deployed within your existing networking infrastructure.

Figure 3.1. OpenAM Deployment Example



3.2. The Public Tier

The public tier provides an extra layer of security with a DMZ consisting of load balancers and reverse proxies. This section presents the DMZ elements.

3.2.1. The Global Load Balancer

The example deployment uses a global load balancer (GLB) to route DNS requests efficiently to multiple data centers. The GLB reduces application latency by spreading the traffic workload among data centers and maintains high availability during planned or unplanned down time, during which it quickly re-routes requests to another data center to ensure online business activity continues successfully.

You can install a cloud-based or a hardware-based version of the GLB. The leading GLB vendors offer solutions with extensive health-checking, site affinity capabilities, and other features for most systems. Detailed deployment discussions about global load balancers are beyond the scope of this guide.

3.2.2. Front End Local Load Balancers

Each data center has local front end load balancers to route incoming traffic to multiple reverse proxy servers, thereby distributing the load based on a scheduling algorithm. Many load balancer solutions provide server affinity or stickiness to efficiently route a client's inbound requests to the same server. Other features include health checking to determine the state of its connected servers, and SSL offloading to secure communication with the client.

You can cluster the load balancers themselves or configure load balancing in a clustered server environment, which provides data and session failover and high availability across multiple nodes. Clustering also allows horizontal scaling for future growth. Many vendors offer hardware and software solutions for this requirement. In most cases, you must determine how you want to configure your load balancers, for example, in an active-passive configuration that supports high availability, or in an active-active configuration that supports session failover and redundancy.

There are many load balancer solutions available in the market. You can set up an external network hardware load balancer, or a software solution like HAProxy (L4 or L7 load balancing) or Linux Virtual Server (LVS) (L4 load balancing), and many others.

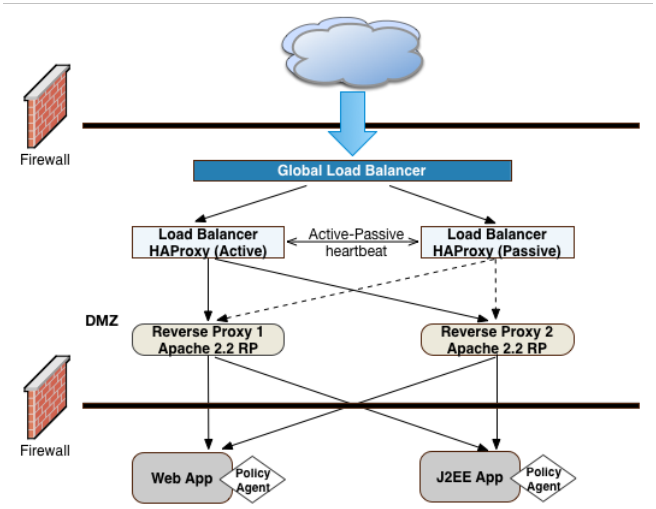
3.2.3. Reverse Proxies

The reverse proxies work in concert with the load balancers to route the client requests to the back end Web or application servers, providing an extra level of security for your network. The reverse proxies also provide additional features, like caching to reduce the load on the Web servers, HTTP compression for faster transmission, URL filtering to deny access to certain sites, SSL acceleration to offload public key encryption in SSL handshakes to a hardware accelerator, or SSL termination to reduce the SSL encryption overhead on the load-balanced servers.

The use of reverse proxies has several key advantages. First, the reverse proxies serve as an highly scalable SSL layer that can be deployed inexpensively using freely available products, like Apache HTTP server or nginx. This layer provides SSL termination and offloads SSL processing to the reverse proxies instead of the load balancer, which could otherwise become a bottleneck if the load balancer is required to handle increasing SSL traffic.

Figure 3.2, "OpenAM Frontend Load Balancer Reverse Proxy Layer" illustrates one possible deployment using HAProxy in an active-passive configuration for high availability. The HAProxy load balancers forward the requests to Apache 2.2 reverse proxy servers. For this example, we assume SSL is configured everywhere within the network.

Figure 3.2. OpenAM Frontend Load Balancer Reverse Proxy Layer

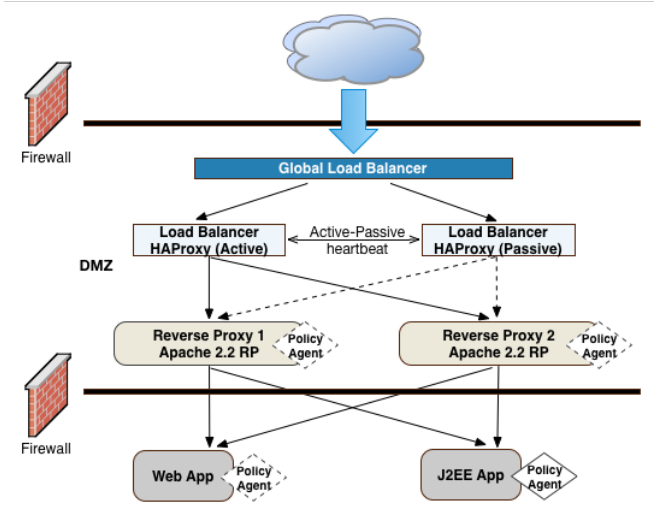


Another advantage to reverse proxies is that they allow access to only those endpoints required for your applications. If you need the authentication user interface and OAuth2/OpenID Connect endpoints, then you can expose only those endpoints and no others. A good rule of thumb is to check which functionality is required for your public interface and then use the reverse proxy to expose only those endpoints.

A third advantage to reverse proxies is when you have applications that sit on non-standard containers for which ForgeRock does not provide a native agent. In this case, you can implement a reverse proxy in your Web tier, and deploy a policy agent on the reverse proxy to filter any requests.

Figure 3.3, "OpenAM Frontend Load Balancers and Reverse Proxies with Agent" shows a simple topology diagram of your Web tier with agents deployed on your reverse proxies. The dotted policy agents indicate that they can be optionally deployed in your network depending on your configuration, container type, and application.

Figure 3.3. OpenAM Frontend Load Balancers and Reverse Proxies with Agent



3.2.4. OpenIG

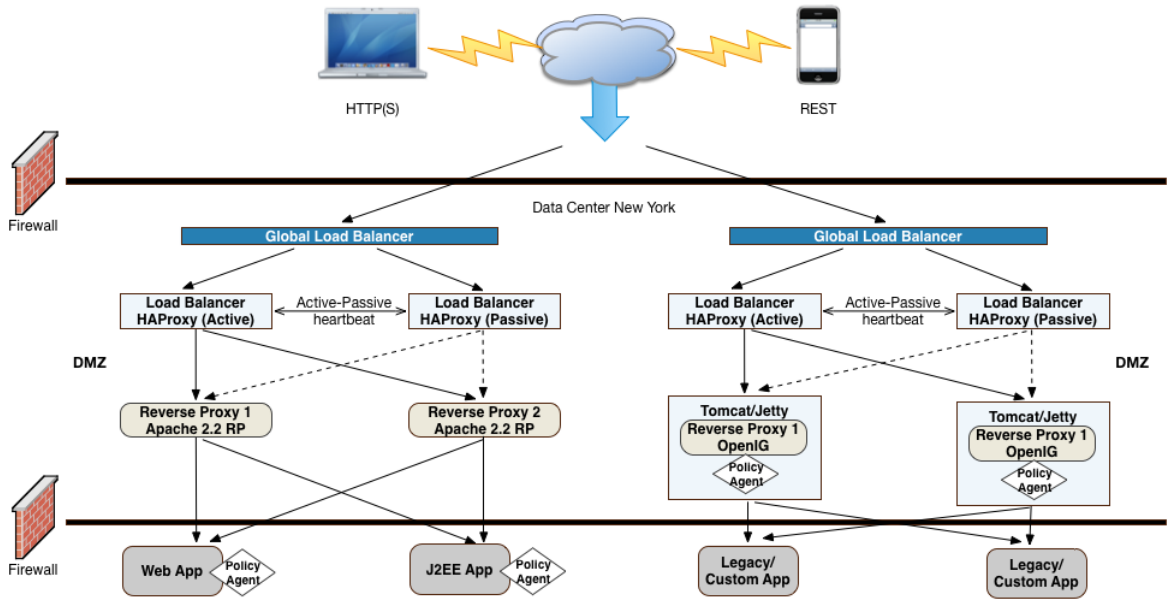
The ForgeRock IRM platform includes an open-source application gateway product, OpenIG, a versatile, Java EE servlet-based reverse proxy server. OpenIG allows you to integrate your applications into existing networks using current standards, such as SAML 2.0 federation, OAuth 2.0 and OpenID Connect 1.0, custom or legacy applications, or other vendor solutions without altering native applications. With OpenIG, you can extend OpenAM's authentication and authorization (policy) services to provide SSO across your mobile, social, partner, and Web applications.

OpenIG provides a set of servlet filters that you can use as-is or chained together with other filters to provide complex operations processing on HTTP requests and responses. You can also write your own custom filters for legacy or custom applications. For more information, see the OpenIG documentation.

You can deploy OpenIG on Tomcat or Jetty servers, allowing it to intercept the HTTP requests and carry out filtering operations on each request, and then log the user directly into the application. In such cases, you can deploy a policy agent for authorization purposes on the request.

However, in the example deployment, you may not need to deploy a policy agent as OpenIG functions strictly as a reverse proxy in the DMZ. The inclusion of the policy agent in the illustration only indicates that you can deploy a policy agent with OpenIG when deployed on a Web container or app server.

Figure 3.4. OpenAM Front End Load Balancers



Note

Some OpenAM authentication modules may require additional user information to authenticate, such as the IP address where the request originated. When OpenAM is accessed through a load balancer or proxy layer, you can configure OpenAM to consume and forward this information with the request headers.

3.2.5. SSL Termination

One important security decision ia whether to terminate SSL or offload your SSL connections at the load balancer. Offloading SSL effectively decrypts your SSL traffic before passing it on as HTTP or at the reverse proxy. Another option is to run SSL pass-through where the load balancer does not decrypt the traffic but passes it on to the reverse proxy servers, which are responsible for the decryption. The other option is to deploy a more secure environment using SSL everywhere within your deployment.

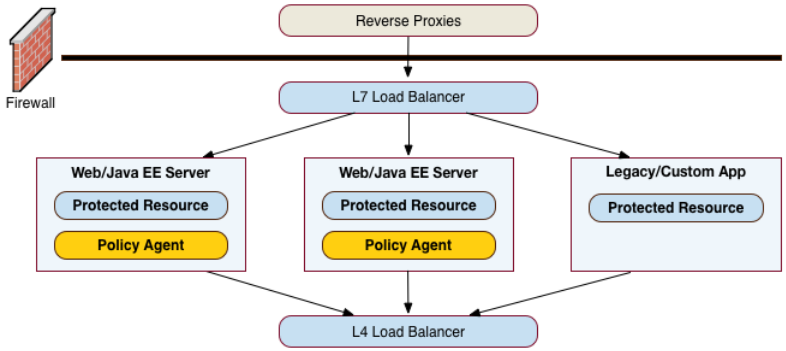
3.3. About the Application Tier

The application tier is where the protected resources reside on Web containers, application servers, or legacy servers. The policy agents intercept all access requests to the protected resources

on the Web or app server and grants access to the user based on policy decisions made on the OpenAM servers. For a list of supported Web application containers, see Section 2.3, "OpenAM Web Application Container Requirements" in the *Release Notes*.

Because OpenAM is Java-based, you can install the server on a variety of platforms, such as Linux, Solaris, and Windows. For a list of platforms that OpenAM has been tested on, see Section 2.1, "OpenAM Operating System Requirements" in the *Release Notes*.

Figure 3.5. OpenAM App Server Deployment



OpenAM provides a cookie (default: `amlbcoookie`) for *sticky load balancing* to ensure that the load balancer properly routes requests to the OpenAM servers. When the client sends an access request to a resource, the policy agent redirects the client to an authentication login page. Upon successful authentication, the policy agent forwards the request via the load balancer to one of the OpenAM servers.

The OpenAM server that authenticated the user becomes the authoritative server during that user's session with OpenAM. Each authentication and authorization request related to the user's session is then evaluated by the authoritative server as long as that server is available. It is therefore important when load balancing, to send requests concerning the user to the authoritative server directly to reduce additional crosstalk from other servers trying contact the authoritative server.

Directing OpenAM requests to the authoritative server is necessary only for OpenAM deployments that use stateful sessions. Because stateless sessions reside in the session token cookie (default: `iPlanetDirectoryPro`) rather than on the OpenAM server, any OpenAM server in a cluster can handle a request with a stateless session without crosstalk.

To direct requests directly to the authoritative OpenAM server, the load balancer should use the value specified in the OpenAM load balancer cookie, `amlbcookie`, which you can configure to uniquely identify a server within a site.

The load balancer inspects the sticky cookie to determine which OpenAM server should receive the request. This ensures that all subsequent requests involving the session are routed to the correct server.

3.4. OpenAM Policy Agents

Policy agents are OpenAM components that are installed on Web containers or application servers to protect the resources deployed there. Policy agents function as a type of gatekeeper to ensure clients are authenticated and authorized to access the resource as well as enforce SSO with registered devices.

OpenAM provides two main policy agents: Web Policy Agent (WPA) and J2EE Policy Agent. The Web Policy Agent is a native plugin to a Web server and is distributed as a zip file. Web policy agents filter requests for Web server resources without any changes to the resources. The J2EE Policy Agent is set up as a servlet filter within the application server. Protected Java EE application configurations must be changed to filter requests through the Java EE policy agent.

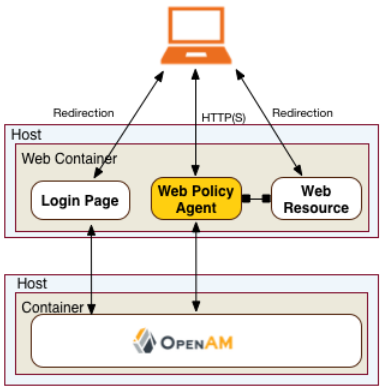
Both policy agents have the following features:

- **Cookie Reset**. Policy agents can be configured to reset any number of cookies in the session before the client is redirected for authentication. This feature is typically used when the policy agent is deployed with a parallel authentication mechanism and cookies need to be reset. Make sure that the `name`, `domain`, and `path` properties are defined.
- **Disable Policy Evaluation**. Policy agents act as a policy enforcement point (PEP) during the authorization phase for a client application. This feature is typically used when the policy agent is only used for SSO and does not require a policy evaluation request to OpenAM.
- **Not-Enforced URLs/URIs List**. Policy agents protect all resources on the Web server or in a Web application that it serves and grants access only if the client has been authenticated and authorized to access the resources. However, there may be some resources, such as public HTML pages, graphics, or stylesheet files that do not require policy evaluation. To account for such files, the policy agent maintains a Not-Enforced URL list, specifying the URLs or resources that are available to any user. J2EE agents use a Not-Enforced URI list.
- **URL Correction**. OpenAM is aware of the access management network and its registered clients, implementing a fully qualified domain name (FQDN) mapper that can be configured to correct invalid URLs. It also holds a list of invalid URLs and compares them to the URL the policy agent is attempting to access.
- **Attribute Injection Into Requests**. Policy agents can be configured to inject user profile attributes into cookies, requests, and HTTP headers.
- **Notifications**. Both agents have the ability to receive configuration notifications from OpenAM. In deployments with stateful sessions, both agents can receive session notifications from OpenAM.
- **Cross-Domain Single Sign-On**. In deployments with stateful sessions, both agents can be configured for cross-domain single sign-on (CDSSO).

3.4.1. Web Policy Agents

A Web policy agent is an installable component on a Web server that is configured to be called by the Web server when a client requests access to a protected resource on a Web site. The Web policy agent runs authentication and authorization services to allow the user access to a protected resource.

Figure 3.6. OpenAM Web Policy Agent

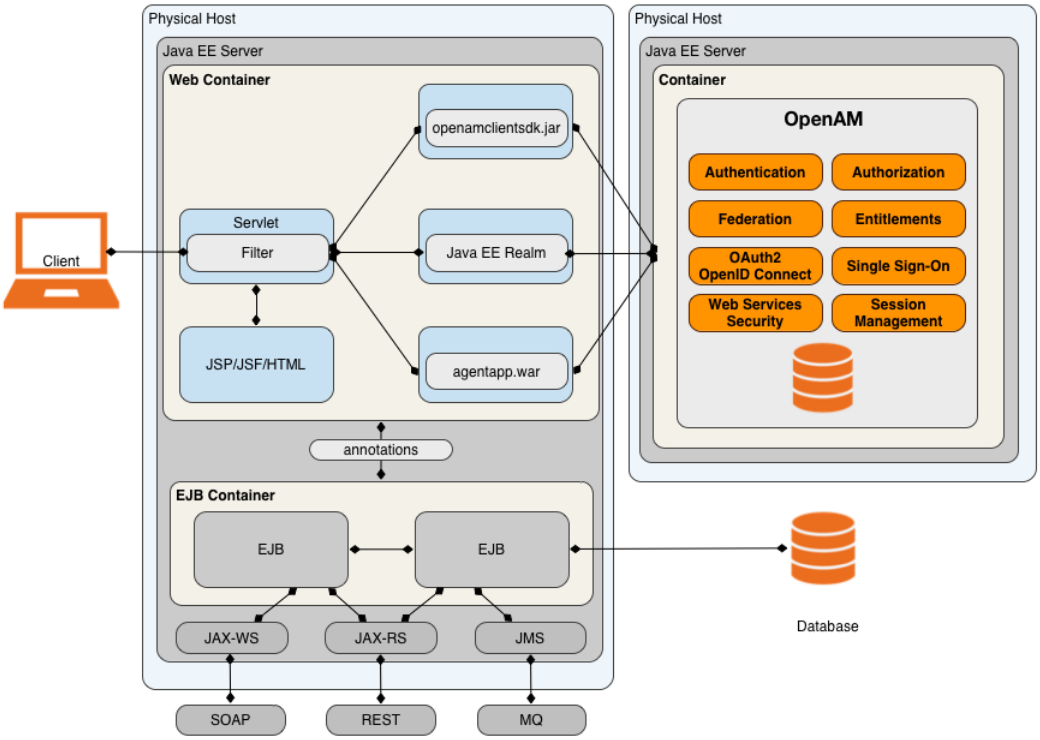


Web Policy Agents are supported on different architectures, although not all Web server types and architecture combinations are supported. You can view the list of supported Web policy agents in the OpenAM Web Policy Agent documentation.

3.4.2. Java EE Policy Agents

The J2EE policy agent is made up of a servlet filter and a J2EE realm. The servlet filter manages authentication and URL-based authorization to the protected application and implements SSO. The filter must be integrated into the application using the application's Web deployment descriptor. The J2EE realm is configured into the security settings of the application server and maps J2EE roles to OpenAM users and groups.

Figure 3.7. Java EE Policy Agent



OpenAM provides a variety of J2EE policy agents for application servers. You can view the list of supported Java EE policy agents in the OpenAM Java EE Policy Agent documentation.

3.5. Sites

OpenAM provides the capability to logically group two or more redundant OpenAM servers into a *site*, allowing the servers to function as a single unit identified by a site ID across a LAN or WAN. When you set up a single site, you place the OpenAM servers behind a load balancer to spread the

load and provide system failover should one of the servers go down for any reason. You can use round-robin or load average for your load balancing algorithms.

Note

Round-robin load balancing should only be used for a first time access of OpenAM or if the `amlbcookie` is not set; otherwise, cookie-based load balancing should be used.

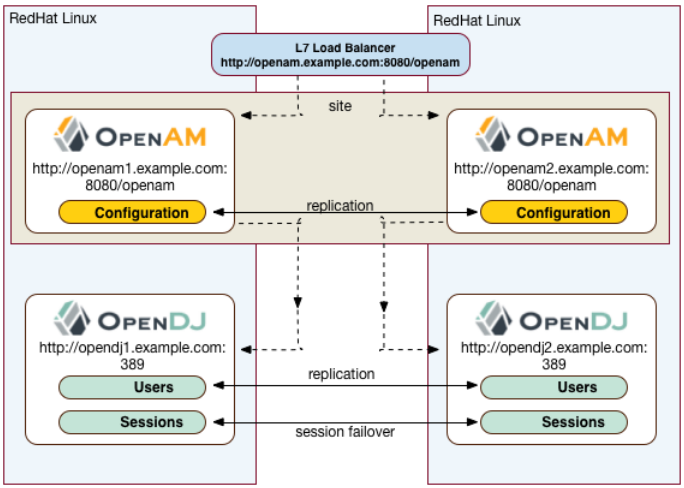
In OpenAM deployments with stateful sessions, you configure each server in a site for session failover, in which the user's authenticated session continues uninterrupted in the event one of the servers go down. Session failover uses OpenAM's CTS to store and share user session data between servers in the site. When an OpenAM server goes does down, the other server(s) in the site reads user session data in the CTS store, allowing the user to run new transactions or requests without re-authenticating to the system. When the server becomes available, it reads the session data in the CTS store and services transactions for active users.

Session failover requires that all servers in a site use the same Core Token Service, which is replicated across all servers. For more information, see Chapter 7, "*Setting Up OpenAM Session Failover*" in the *Installation Guide*.

OpenAM deployments with stateless sessions do not use the CTS for session storage and retrieval to achieve session failover. Instead, the session is stored in a browser cookie.

Figure 3.8, "OpenAM Application Tier Deployment" shows a possible implementation using RedHat Linux servers with OpenAM installed on each server. You can implement routing software, like Keepalived in such a deployment. If you require L7 load balancing, you can consider many other software and hardware solutions. OpenAM relies on OpenDJ's SDK for load balancing, failover, and heartbeat capabilities to spread the load across the directory servers or to throttle performance.

Figure 3.8. OpenAM Application Tier Deployment



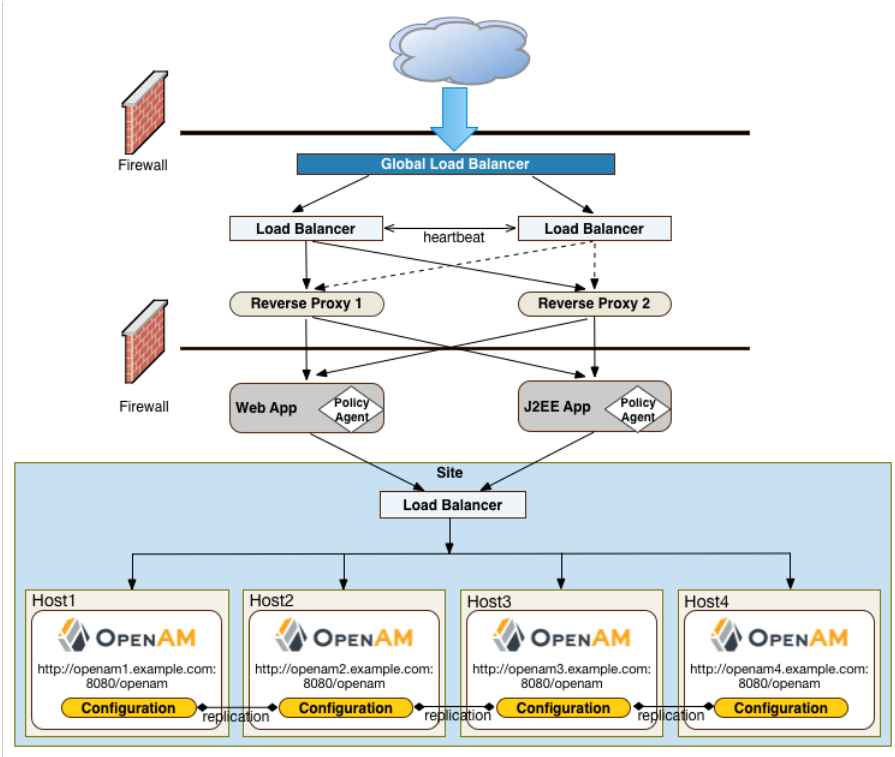
Note

When protecting OpenAM with a load balancer or proxy service, configure your container so that OpenAM can trust the load balancer or proxy service.

One possible configuration (seen in Figure 3.9, "OpenAM Site Deployment With a Single Load Balancer") is to set up a load balancer with multiple OpenAM servers. You configure the load balancer to be sticky using the value of the OpenAM cookie, `amlbcookie`, which routes client requests to that primary server. If the primary OpenAM server goes down for any reason, it fails over to another OpenAM server. Session data also continues uninterrupted if a server goes down as it is shared between OpenAM servers. You must also ensure that the container trusts the load balancer.

You must determine if SSL should be terminated on the load balancer or communication be encrypted from the load balancer to the OpenAM servers.

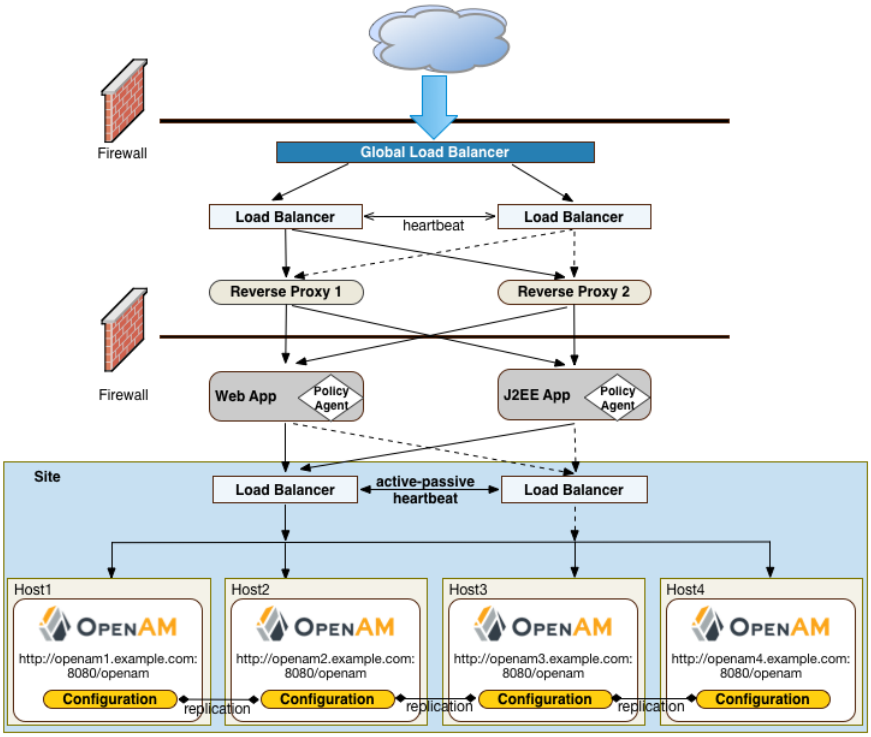
Figure 3.9. OpenAM Site Deployment With a Single Load Balancer



One problem with Figure 3.9, "OpenAM Site Deployment With a Single Load Balancer" is that the load balancer is a single point of failure. If the load balancer goes down, then the system becomes inoperable.

To make the deployment highly available, you can set up another variation of the deployment by fronting more than one load balancer to the set of OpenAM servers in an active/passive configuration that provides high availability should one load balancer go down for an outage.

Figure 3.10. OpenAM Site Deployment With Multiple Load Balancers



3.5.1. Multiple Sites

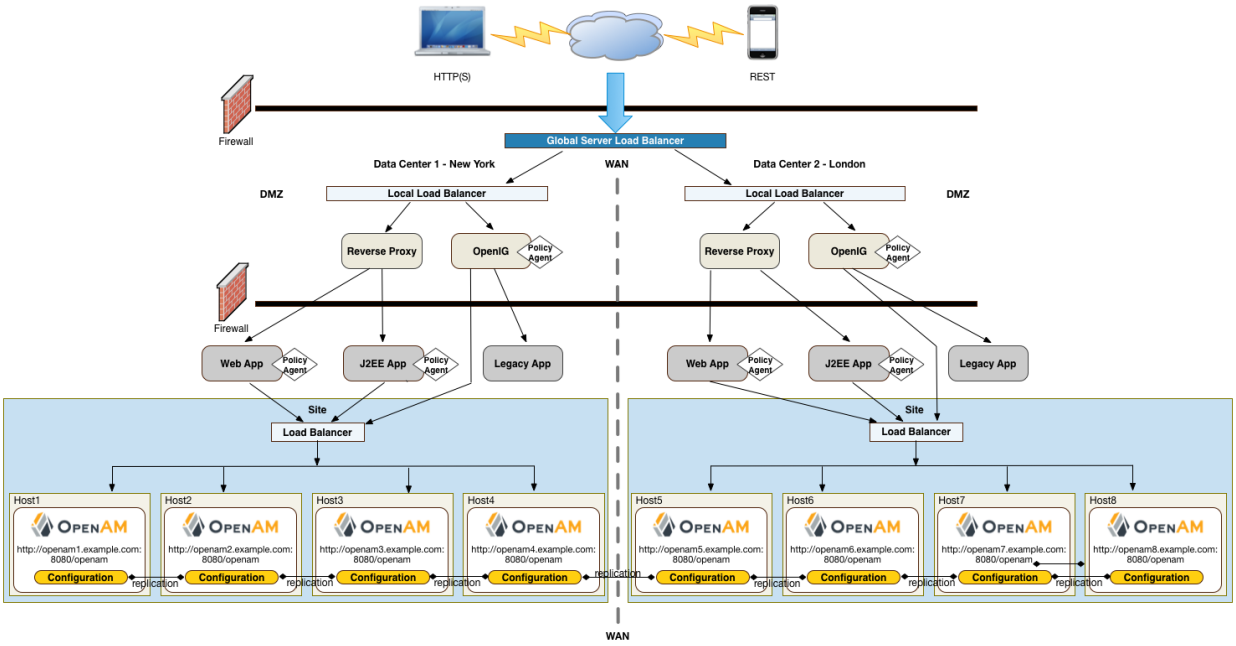
Another deployment variation is to set up multiple redundant sites, typically across a WAN network, which provides high availability for stateful sessions through system and session failover. This setup can be seen in Figure 3.11, "OpenAM Site Deployment With Multiple Sites" If the load balancer in one site goes down, the other site can resume processing requests with the authenticated user session running without interruption. If an OpenAM server goes down, it fails over to another OpenAM server while also keeping the authenticated user session active with uninterrupted service.

Policy agent configuration and other configuration data can be shared among multiple, redundant sites, so that if one site fails, the other site can continue without requiring re-logging.

For optimum performance, you want to keep sites local to your geographical location with session failover taking place only within a data center. The possible loss of a data center means clients must reestablish sessions, which may be an acceptable trade-off given the performance cost of highly-

replicated systems across multiple sites over WAN. You must determine the optimum topology based on your performance and availability objectives.

Figure 3.11. OpenAM Site Deployment With Multiple Sites



For more information, see Chapter 4, "*Installation Considerations for Multiple Servers*" in the *Installation Guide*.

3.6. Back End Directory Servers

Each OpenAM server comes out-of-the-box with an embedded OpenDJ directory server that you can configure to store policies, configuration data, identity data, and CTS tokens. The embedded directory server is best suited for small systems or for evaluation purposes. It is not generally recommended for large-scale production systems.

- **Identity Data Stores**. For identity repositories, OpenAM provides built-in support for LDAP and JDBC storage systems. You can implement a number of different directory server vendors for storing your identity data, allowing you to configure your directory servers in a number of deployment typologies. For a list of supported data stores, see Section 2.4, "Data Store Requirements" in the *Release Notes*.

When configuring external LDAP Identity data stores, you must manually carry out additional installation tasks that could require a bit more time for the configuration process. For example, you must manually add schema definitions, access control instructions (ACIs), privileges for reading and updating the schema, and resetting user passwords. For more information, see Section 1.4, "Preparing an External Identity Repository" in the *Installation Guide*.

If OpenAM does not support your particular identity data store type, you can develop your own customized plugin to allow OpenAM to run method calls to fetch, read, create, delete, edit, or authenticate to your identity store data. For more information, see Section 3.6, "Customizing Identity Data Storage" in the *Developer's Guide*.

You can configure the Data Store authentication module to require the user to authenticate against a particular identity data store for a specific realm. OpenAM associates a realm with at least one identity repository and authentication process. When you initially configure OpenAM, you define the identity repository for authenticating at the top level realm (/), which is used to administer OpenAM. From there, you can define additional realms with different authentication and authorization services as well as different identity repositories if you have enough identity data. For more information, see Chapter 4, "*Configuring Realms*" in the *Administration Guide*.

- **Configuration Data Stores**. OpenAM stores configuration data in the embedded OpenDJ directory server. Configuration data includes authentication information that defines how users and groups authenticate, identity data store information, service information, policy information for evaluation, and partner server information that can send trusted SAML assertions. For a list of supported data stores, see Section 2.4, "Data Store Requirements" in the *Release Notes*.

  The embedded OpenDJ directory server may be sufficient for your system, but you may want to deploy an external configuration store if required for large-scale systems with many policies or many realms. Like external identity stores, you must manually add schema definitions, ACIs, privileges for reading and updating the schema, and indexes for attributes used to access the configuration data.

  SAML keys are stored in the configuration store and are thus replicated. Also, OpenAM's signing keys are shipped with a test certificate. If you upgrade the keystore, you need to redistribute the certificates to all nodes so that they can continue to communicate with each other. For more information, see Chapter 23, "*Managing Certificates and Keystores*" in the *Administration Guide*.

- **CTS Data Stores**. The CTS provides persistent and highly available token storage for OpenAM session, OAuth 2.0, and SAML 2.0 tokens. If configured, CTS supports session token persistence for stateful session failover.

  CTS traffic is volatile compared to configuration data, so deploying CTS as a dedicated external data store is advantageous for systems with many users and many sessions. For more information, see Chapter 6, "*Configuring the Core Token Service*" in the *Installation Guide*.

When configuring multiple external directory servers, make sure to deploy them with an active/passive load balancing algorithm. This setup eliminates the possibility of directory read-write errors if replication is not quick enough. For example, if an attribute is updated on OpenDJ-1 but read

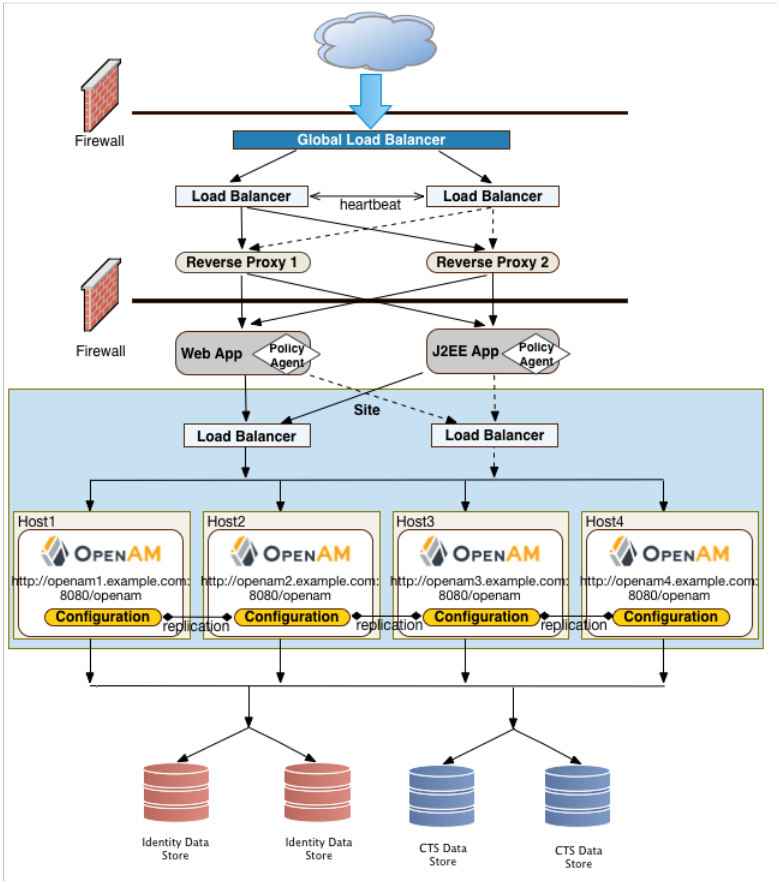
from OpenDJ-2, and if replication is not quick enough and the attribute is not written or updated in OpenDJ-2, an error could result.

Figure 3.12, "OpenAM Site Deployment With External Datastores" shows a basic back end deployment with separate external identity, configuration, and CTS data stores. You can use load balancers to spread the load or throttle performance for the external data stores. Although not shown in the diagram, you can also set up a directory tier, separating the application tier from the repositories with another firewall. This tier provides added security for your identity repository or policy data.

Note

ForgeRock recommends that you use the OpenAM's embedded OpenDJ directory server as the configuration data store and only set up an external configuration store if necessary.

Figure 3.12. OpenAM Site Deployment With External Datastores



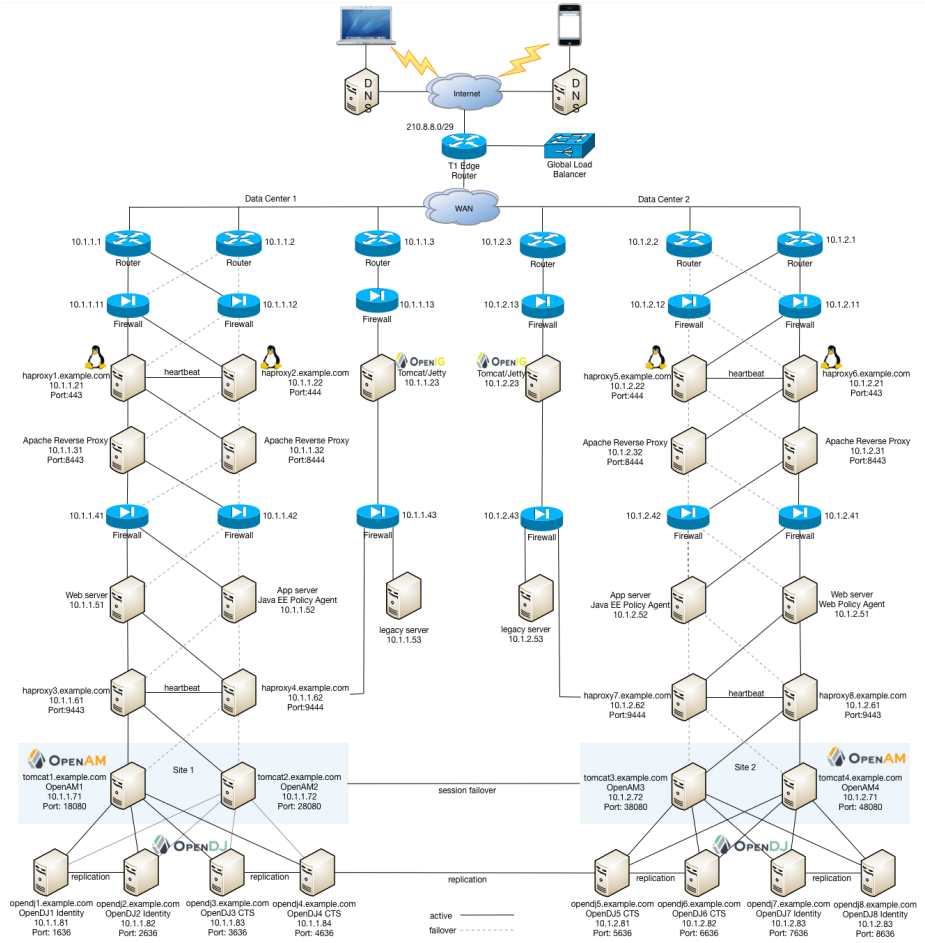
3.7. Example Topology Configuration Diagram

Figure 3.13, "OpenAM Example Deployment Configuration Diagram" shows a simplified configuration diagram of the example deployment presented in this chapter (shown in Figure 3.1, "OpenAM Deployment Example"). The example deploys the various servers on Linux hosts.

The firewalls can be a hardware or software solution or a combination firewall-router can be implemented in the deployment. The local load balancers are implemented using HAProxy servers in an active-passive configuration. You can also use Linux Keepalived for software load balancing or one of the many other solutions available. The Web and application servers have the Web policy agent and Java EE policy agent installed on each server respectively. OpenAM is deployed on Tomcat hosted on a Linux server. Within each datacenter, the OpenAM servers are configured as sites for failover and stateful session failover capabilities.

The directory servers are OpenDJ servers that store identity and CTS data. For presentation purposes only, the configuration data is assumed to be stored within the embedded directory store on each OpenAM server. The OpenIG example does not show redundancy for high availability also due to presentation purposes.

Figure 3.13. OpenAM Example Deployment Configuration Diagram



3.8. Realms

The previous sections in this chapter present the logical and physical topologies of an example highly available OpenAM deployment, including the clustering of servers using *sites*. One important configuration feature of OpenAM is its ability to run multiple client entities to secure and manage applications through a single OpenAM instance.

OpenAM supports its multiple clients through its use of *realms*. You configure realms within OpenAM to handle different sets of users to whom you can set up different configuration options, storage requirements, delegated administrators, and customization options per realm.

Typically, you can configure realms for customers, partners, or employees within your OpenAM instance, for different departments, or for subsidiaries. In such cases, you create a global administrator who can delegate privileges to realm administrators, each specifically responsible for managing their respective realms.

Chapter 4

Sizing Hardware and Services For Deployment

This chapter covers sizing servers, network, storage, and service levels required by your OpenAM deployment.

4.1. Sizing For Availability

Any part of a system that can fail eventually will fail. Keeping your service available means tolerating failure in any part of the system without interrupting the service. You make OpenAM services highly available through good maintenance practices and by removing single points of failure from your architectures.

Removing single points of failure involves replicating each system component, so that when one component fails, another can take its place. Replicating components comes with costs not only for the deployment and maintenance of more individual components, but also for the synchronization of anything those components share. Due to necessary synchronization between components, what you spend on availability is never fully recovered as gains in capacity. (Two servers cannot do quite twice the work of one server.) Instead you must determine the right trade offs for the deployment.

To start thinking about the trade offs, answer the following questions.

- What is the impact of the OpenAM service becoming unavailable?

  In an online system this could be a severe problem, interrupting all access to protected resources. Most deployments fall into this category.

  In an embedded system protecting local resources, it might be acceptable to restart the service.

  Deployments that require always-on service availability require some sort of load balancing solution at minimum between OpenAM and OpenAM client applications. The load balancer itself must be redundant, too, so that it does not become a single point of failure. Illustrations in Chapter 3, "*Example Deployment Topology*", show multiple levels of load balancing for availability and firewalls for security.

- Is the service critical enough to warrant deployment across multiple sites?

  OpenAM allows you to deploy replicated configurations in different physical locations, so that if the service experiences complete failure at one site, you can redirect client traffic to another site and continue operation. The question is whether the benefit in reducing the likelihood of failure outweighs the costs of maintaining multiple sites.

When you need failover across sites, one of the costs is (redundant) WAN links scaled for inter-site traffic. OpenAM synchronizes configuration and policy data across sites, and by default also synchronizes session data as well. OpenAM also expects profiles in identity data stores to remain in sync. As shown in Chapter 3, "*Example Deployment Topology*", the deployment involves directory service replication between sites.

- What happens if individual session information is lost?

  In OpenAM session failover is different from service failover. Session failover consists of maintaining redundant information for stateful sessions, so that if a server fails, another server recovers the session information, preventing the user from having to authenticate again. Service failover alone consists of maintaining redundant servers, so that if one server fails, another server can take the load. With service failover alone, users who authenticated with a failed server must authenticate again to start a new session.

  In deployments where an interruption in access to a protected resource could cause users to lose valuable information, session failover can prevent the loss. To provide for session failover, OpenAM replicates the session information held by the CTS, relying on the underlying directory service to perform the replication. Session information can be quite volatile, certainly more volatile than configuration and policy data. Session failover across sites can therefore call for more WAN bandwidth, as more information is shared across sites.

Once you have the answers to these questions for the deployment, you can draw a diagram of the deployment, checking for single points of failure to avoid. In the end, you should have a count of the number of load balancers, network links, and servers that you need, with the types of clients and an estimated numbers of clients that access the OpenAM service.

Also, you must consider the requirements for non-functional testing, covered in Section 2.4.8, "Planning Tests". While you might be able to perform functional testing by using a single OpenAM server with the embedded OpenDJ server for user data store, other tests require a more complete environment with multiple servers, secure connections, and so forth. Performance testing should reveal any scalability issues. Performance testing should also run through scenarios where components fail, and check that critical functionality remains available and continues to provide acceptable levels of service.

4.2. Sizing For Service Levels

Beyond service availability, your aim is to provide some level of service. You can express service levels in terms of throughput and response times. For example, the service level goal could be to handle an average of 10,000 authentications per hour with peaks of 25,000 authentications per hour, and no more than 1 second wait for 95% of users authenticating, with an average of 100,000 live SSO sessions at any given time. Another service level goal could be to handle an average of 500 policy requests per minute per policy agent with an average response time of 0.5 seconds.

When you have established your service level goals, you can create load tests for each key service as described in Section 2.4.8.2, "Planning Service Performance Testing". Use the load tests to check your sizing assumptions.

To estimate sizing based on service levels, take some initial measurements and extrapolate from those measurements.

- For a service that handles policy decision (authorization) requests, what is the average policy size? What is the total size of all expected policies?

  To answer these questions, you can measure the current disk space and memory occupied by the configuration directory data. Next, create a representative sample of the policies that you expect to see in the deployment, and measure the difference. Then, derive the average policy size, and use it to estimate total size.

  To measure rates of policy evaluations, you can monitor policy evaluation counts over SNMP. For details, see Section 24.2.2, "SNMP Monitoring for Policy Evaluation" in the *Administration Guide*.

- For a service that stores stateful sessions, access, and refresh tokens, or other authentication-related data in the CTS backing store, what is the average session or access/refresh token size? What is the average total size of CTS data?

  As for policy data, you can estimate the size of CTS data by measuring the CTS directory data.

  The average total size depends on the number of live CTS entries, which in turn depends on session and token lifetimes. The lifetimes are configurable and depend also on user actions like logout, that are specific to the deployment.

  For example, suppose that the deployment only handles stateful SSO sessions, session entries occupy on average 20 KB of memory, and that you anticipate on average 100,000 active sessions. In that case, you would estimate the need for 2 GB (100,000 x 20,000) RAM on average to hold the session data in memory. If you expect that sometimes the number of active sessions could rise to 200,000, then you would plan for 4 GB RAM for session data. Keep in mind that this is the extra memory needed in addition to memory needed for everything else that the system does including running OpenAM server.

  Session data is relatively volatile, as the CTS creates sessions entries when sessions are created. CTS deletes session entries when sessions are destroyed due to logout or timeout. Sessions are also modified regularly to update the idle timeout. Suppose the rate of session creation is about 5 per second, and the rate of session destruction is also about 5 per second. Then you know that the underlying directory service must handle on average 5 adds and 5 deletes per second. The added entries generate on the order of 100 KB replication traffic per second (5/s x 20 KB plus some overhead). The deleted entries generate less replication traffic, as the directory service only needs to know the distinguished name (DN) of the entry to delete, not its content.

  You can also gather statistics about CTS operations over SNMP. For details, see Section 24.2, "Monitoring CTS Tokens" in the *Administration Guide*.

- What level of network traffic do you expect for notifications and crosstalk?

When sizing the network, you must account both for inter-site replication traffic and also for notifications and crosstalk in high-throughput deployments.

In OpenAM deployments using stateful sessions, much of the network traffic between OpenAM servers consists of notifications and crosstalk. When the session state changes on session creation and destruction, the OpenAM server performing the operations can notify other servers.

Crosstalk between OpenAM servers arises either when configured (i.e., in OpenAM 12.0 or later), where you enable crosstalk to resolve non-local sessions instead of having OpenAM rely on the session persistence mechanism, or when a client is not routed to the OpenAM server that originally authenticated the client. In an OpenAM site, the server that originally authenticates a client deals with the session, unless that server becomes unavailable. If the client is routed to another server, then the other server communicates with the first, resulting in local crosstalk network traffic. Sticky load balancing can limit crosstalk by routing clients to the same server with which they started their session.

When the OpenAM servers are all on the same LAN, and even on the same rack, notifications and crosstalk are less likely to adversely affect performance. If the servers are on different networks or communicating across the WAN, the network could become a bottleneck.

- What increase in user and group profile size should you expect?

  OpenAM stores data in user profile attributes. OpenAM can use or provision many profile attributes, as described in Procedure 4.5, "To Configure a Data Store" in the *Administration Guide*.

  When you know which attributes are used, you can estimate the average increase in size by measuring the identity data store as you did for configuration and CTS-related data. If you do not manage the identity data store as part of the deployment, you can communicate this information with the maintainers. For a large deployment, the increase in profile size can affect sizing for the underlying directory service.

- How does the number of realms affect the configuration data size?

  In a centrally managed deployment with only a few realms, the size of realm configuration data might not be consequential. Also, you might have already estimated the size of policy data. For example, each new realm might add about 1 MB of configuration data to the configuration directory, not counting the policies added to the realm.

  In a multi-tenant deployment or any deployment where you expect to set up many new realms, the realm configuration data and the additional policies for the realm can add significantly to the size of the configuration data overall. You can measure the configuration directory data as you did previously, but specifically for realm creation and policy configuration, so that you can estimate an average for a new realm with policies and the overall size of realm configuration data for the deployment.

4.3. Sizing Systems

Given availability requirements and estimates on sizing for services, estimate the required capacity for individual systems, networks, and storage. This section considers the OpenAM server systems, not the load balancers, firewalls, independent directory services, and client applications.

Although you can start with a rule of thumb, you see from the previous sections that the memory and storage footprints for the deployment depend in large part on the services you plan to provide. With that in mind, to performance test a basic deployment providing SSO, you can start with OpenAM systems having at least 4 GB free RAM, 4 CPU cores (not throughput computing cores, but normal modern cores), plenty of local storage for configuration, policy, and CTS data, and LAN connections to other OpenAM servers. This rule of thumb assumes the identity data stores are sized separately, and that the service is housed on a single local site. Notice that this rule of thumb does not take into account anything particular to the service levels you expect to provide. Consider it a starting point when you lack more specific information.

4.3.1. Sizing System CPU and Memory

OpenAM services use CPU resources to process requests and responses, and essentially to make policy decisions. Encryption, decryption, signing, and checking signatures can absorb CPU resources when processing requests and responses. Policy decision evaluation depends both on the number of policies configured and on their complexity.

Memory depends on space for OpenAM code, on the number of live connections OpenAM maintains, on caching of configuration data, user profile data, and stateful session data, and importantly, on holding embedded directory server data in memory. The OpenAM code in memory probably never changes while the server is running, as JSPs deployed are unlikely ever to change in production.

The number of connections and data caching depending on server tuning, as described in Chapter 25, "*Tuning OpenAM*" in the *Administration Guide*.

If OpenAM uses the embedded OpenDJ directory server, then the memory needed depends on what you store in the embedded directory and what you calculated as described in Section 4.2, "Sizing For Service Levels". The embedded OpenDJ directory server shares memory with the OpenAM server process. By default, the directory server takes half of the available heap as database cache for directory data. That setting is configurable as described in the OpenDJ directory server documentation.

4.3.2. Sizing Network Connections

When sizing network connections, you must account for the requests and notifications from other servers and clients, as well as the responses. This depends on the service levels that the deployment provides, as described in Section 4.2, "Sizing For Service Levels". Responses for browser-based authentication can be quite large if each time a new user visits the authentication UI pages, OpenAM must respond with the UI page, plus all images and JavaScript logic and libraries included to complete the authentication process. Inter-server synchronization and replication can also require significant bandwidth.

For deployments with sites in multiple locations, be sure to account for configuration, CTS, and identity directory data over WAN links, as this is much more likely to be an issue than replication traffic over LAN links.

Make sure to size enough bandwidth for peak throughput, and do not forget redundancy for availability.

4.3.3. Sizing Disk I/O and Storage

As described in Section 5.1.1, "Disk Storage Requirements", the largest disk I/O loads for OpenAM servers arise from logging and from the embedded OpenDJ directory server writing to disk. You can estimate your storage requirements as described in that section.

I/O rates depend on the service levels that the deployment provides, as described in Section 4.2, "Sizing For Service Levels". When you size disk I/O and disk space, you must account for peak rates and leave a safety margin when you must briefly enable debug logging to troubleshoot any issues that arise.

Also, keep in mind the possible sudden I/O increases that can arise in a highly available service when one server fails and other servers must take over for the failed server temporarily.

Another option is to consider placing log, configuration, and database files on a different file system to maximize throughput and minimize service disruption due to a file system full or failure scenarios.

Chapter 5

OpenAM Hardware and Software Requirements

You can configure OpenAM in a wide variety of deployments depending on your security requirements and network infrastructure.

5.1. Hardware Requirements

This section covers hardware requirements for OpenAM.

5.1.1. Disk Storage Requirements

This section considers disk storage requirements for OpenAM server, OpenAM policy agents, and OpenIG gateway.

5.1.1.1. Server Disk Storage Requirements

Disk storage requirements for OpenAM servers depend partly on OpenAM itself and partly on your deployment. Disk storage requirements also depend on the space needed for binaries and configuration data, space for log files and rate of writes for logs, space for directory data and file system requirements when using an embedded OpenDJ directory server.

For initial installation, a few hundred MB is sufficient, not including the downloaded files.

- The OpenAM `.war` file size varies from release to release, but if your container holds one `.war` file and one directory with the contents of the `.war` file, the disk space required is on the order of 300 MB.

  This space requirement remains stable as you use OpenAM.

- The OpenAM configuration directory initially fits in approximately 50 MB of disk space including the embedded OpenDJ directory server.

  This space requirement grows as you use OpenAM.

By default, OpenAM servers write audit logs to flat files under *`config-dir`*`/openam/logs/`. Alternatively, OpenAM servers can write audit logs to `syslog`, or to a relational database.

When using flat-file audit logging, OpenAM lets you configure rotation and purging for logs under`openam/logs/`, so you can effectively cap the maximum disk space used for logs. Make sure, however, that you retain the information you need before logs are purged. Also make sure that your disk can keep pace with the volume of logging, which can be significant in high volume deployments, as OpenAM logs not only errors, but also access messages.

For details about audit logging configuration, see Chapter 6, "*Configuring Audit Logging*" in the *Administration Guide*.

By default, OpenAM servers write debug logs to flat files under `config-dir/openam/debug/`. OpenAM lets you configure rotation for debug logs. As you can change debug log levels at runtime when investigating issues, debug log volume is not as predictable as for regular logs. Leave a margin in production environments, so that you can turn up debug log levels to diagnose problems.

For details about debug logging configuration, see Section 24.4, "Debug Logging" in the *Administration Guide*.

When using the embedded OpenDJ directory server, take the following into account:

- OpenDJ is designed to work with local storage for the database, but not for network file system (NFS) nor network-attached storage (NAS) due to some file system locking functions that OpenDJ needs. High performance storage, like solid state drives (SSD), is essential if you need to handle high write throughput.

  By default, OpenAM's configuration directory resides under the `$HOME` directory of the user running the container. `$HOME` directories can be mounted over the network.

  This is not an issue if you are using OpenDJ mainly for configuration data. It can however be a serious problem when you use OpenDJ to back the CTS in a high-volume deployment.

- Embedded OpenDJ directory server log files are stored under `config-dir/opends/logs/`. As for OpenAM, you can configure OpenDJ directory server log rotation and purging. The default cap for access logs is 2 GB.

- OpenAM stores policy information in the configuration directory. The space this takes up depends on the policies you have.

- By default, OpenAM stores CTS information in the configuration directory. The space this takes up depends on the volume of traffic to the server and whether OpenAM is configured for stateless sessions.

- With the default database implementation, OpenDJ database files handling sustained writes can grow to about double their initial size on disk.

- For OpenDJ on Linux systems, enable file system write barriers and ensure the file system journaling mode is ordered to avoid directory database file corruption after crashes or power failures. For details on enabling write barriers and setting the journaling mode for data, see the options for your file system in the **mount** command manual page.

- OpenDJ directory server uses file descriptors when handling connections.

Defaults can be limited to 1024 file descriptors per user on Linux systems. Consider increasing this limit to at least 64K. For details, see Section 1.3, "Setting Maximum File Descriptors" in the *Installation Guide*.

5.1.1.2. Policy Agent Disk Storage Requirements

Policy agents are implemented as libraries or Web applications, and so tend to be small on disk, not more than a few MB.

You can configure policy agents to perform local logging to files, or to send log messages to OpenAM for remote logging. For details, see the *Configuration Reference* for your policy agent.

Debug messages are logged to local files, however, not remotely. Debug logging volume depends on log level. As for OpenAM, leave a margin in production environments so that you can turn up debug log levels to diagnose problems.

5.1.1.3. OpenIG Disk Storage Requirements

The OpenIG Web application can vary in size from release to release. On disk, the `.war` file is under 50 MB. For containers that keep both the `.war` file and an unpacked version, the total size is under 100 MB.

By default, OpenIG configuration resides under the `$HOME` directory of the user who runs the container.

If you use the default log sink, messages are sent to the container logs. Manage those as you would any container logs.

Capture logging and any logging you perform from scriptable filters and handlers can potentially generate significant write traffic. Furthermore, OpenIG does not run rotation or purging for such logs. You must manage any logs OpenIG generates using a `CaptureFilter` or log messages from scriptable filters and handlers.

Both normal log messages and debug messages go to the log sink. As for other components, debug logging volume depends on log level. Leave a margin in production environments so that you can turn up debug log levels to diagnose problems.

5.1.1.4. Disk Storage Recommendations

The following are based on the preceding information in this section. When deciding on disk storage, keep the following recommendations in mind:

- Plan enough space and enough disk I/O to comfortably absorb the load for logs.

  Check your assumptions in testing. For example, make sure that logs are cleaned up so that they do not exceed your space threshold even in long-duration testing.

- For deployments where an embedded OpenDJ directory service handles high throughput, make sure you use a local file system and that the user running the container has enough file descriptors.
- When using local policy agent logs, make sure you have a mechanism in place to clean them up.
- For OpenIG, make sure you turn off `CaptureFilter` logging, scriptable filter, and handler debug logging before moving to production.

5.1.2. Random Access Memory Requirements

OpenAM core services require a minimum JVM heap size of 1 GB and, when running on JDK 7, a minimum permanent generation size of 256 MB. If you are including the embedded OpenDJ directory, OpenAM requires at least a 2 GB heap, as 50% of that space is allocated to OpenDJ.

Note

Ensure that the `Xms` and `Xmx` JVM parameters are set to the same value to prevent a large garbage collection as the memory profile increases from the default up to the `Xms` value. Also, setting `Xms` and `Xmx` to the same value ensures that small controlled garbage collection events minimize application unresponsiveness.

5.2. Software Requirements

The following sections list software requirements for deploying OpenAM server and policy agent software.

5.2.1. OpenAM Operating System Requirements

ForgeRock supports customers using OpenAM server software on the following operating system versions:

Table 5.1. Supported Operating Systems

Operating System	Version
Red Hat Enterprise Linux, Centos	6, 7
SuSE	11
Ubuntu	12.04 LTS, 14.04 LTS
Solaris x64	10, 11
Solaris Sparc	10, 11
Windows Server	2008, 2008 R2, 2012, 2012 R2

5.2.2. Java Requirements

Table 5.2. JDK Requirements

Vendor	Version
Oracle JDK	7, 8
IBM SDK, Java Technology Edition (Websphere only)	7

5.2.3. OpenAM Web Application Container Requirements

Table 5.3. Web Containers

Web Container	Version
Apache Tomcat	7, 8[a]
Oracle WebLogic Server	12c
JBoss Enterprise Application Platform	6.1+
JBoss Application Server	7.2+
WildFly AS	9
IBM WebSphere	8.0, 8.5.5.8+

[a]OpenAM supports Tomcat 8.0.x, but not 8.5.x. Tomcat 8.5.x is supported in Access Management 5.

The web application container must be able to write to its own home directory, where OpenAM stores configuration files.

5.2.4. Data Store Requirements

Table 5.4. Supported Data Stores

Data Store	Version	CTS Datastore	Config Datastore	User Datastore	UMA Datastore
Embedded OpenDJ	3.5	✓	✓	✓	✓
External OpenDJ	2.6, 2.6.4, 3.0, 3.5	✓	✓	✓	✓
Oracle Unified Directory	11g			✓	
Oracle Directory Server Enterprise Edition	11g			✓	
Microsoft Active Directory	2008, 2008 R2, 2012, 2012 R2			✓	

Data Store	Version	CTS Datastore	Config Datastore	User Datastore	UMA Datastore
IBM Tivoli Directory Server	6.3			✓	

5.2.5. Supported Clients

The following table summarizes supported clients:

Table 5.5. Supported Clients

Client Platform	Native Apps[a]	Chrome 16+[b]	IE 9+, Microsoft Edge	Firefox 3.6+	Safari 5+
Windows 7 or later	✓	✓	✓	✓	✓
Mac OS X 10.8 or later	✓	✓		✓	
Ubuntu 12.04 LTS or later	✓	✓		✓	✓
iOS 7 or later	✓	✓			✓
Android 4.3 or later	✓	✓			

[a] *Native Apps* is a placeholder to indicate OpenAM is not just a browser-based technology product. An example of a native app would be something written to use our REST APIs, such as the sample OAuth 2.0 Token Demo app.
[b] Chrome, Firefox, and Safari are configured to update automatically, so customers will typically running the latest versions. The versions listed in the table are the minimum required versions.

5.2.6. Java EE Agents Platform Requirements

The following table summarizes platform support.

Table 5.6. Supported Operating Systems & Web Application Containers

Operating Systems (OS)	OS Versions	Web Application Containers & Versions
CentOS Red Hat Enterprise Linux Oracle Linux	5, 6, 7	Apache Tomcat 6, 7, 8 IBM Web Sphere Application Server 8, 8.5 JBoss Enterprise Application Platform 6 JBoss Application Server 7 Jetty 8 (at least 8.1.13) Oracle WebLogic Server 11g, 12c
Microsoft Windows Server	2008, 2008 R2, 2012, 2012 R2	Apache Tomcat 6, 7, 8
Oracle Solaris x64 Oracle Solaris SPARC	10, 11	Apache Tomcat 6, 7, 8 Oracle WebLogic Server 11g, 12c

Operating Systems (OS)	OS Versions	Web Application Containers & Versions
Ubuntu Linux	12.04 LTS, 14.04 LTS	Apache Tomcat 6, 7, 8 IBM Web Sphere Application Server 8, 8.5 JBoss Enterprise Application Platform 6 JBoss Application Server 7 Jetty 8 (at least 8.1.13) Oracle WebLogic Server 11g, 12c

5.2.7. Web Policy Agents Platform Requirements

The following table summarizes platform support.

Table 5.7. Supported Operating Systems & Web Servers

Operating Systems (OS)	OS Versions	Web Servers & Versions
CentOS Red Hat Enterprise Linux Oracle Linux	5, 6, 7	Apache HTTP Server 2.2 Apache HTTP Server 2.4
Microsoft Windows Server	2008 R2	Microsoft IIS 7
	2008 R2	Microsoft IIS 7.5
	2012, 2012 R2	Microsoft IIS 8
Oracle Solaris x64 Oracle Solaris SPARC	10, 11	Apache HTTP Server 2.2 Apache HTTP Server 2.4
Ubuntu Linux	12.04 LTS, 14.04 LTS	Apache HTTP Server 2.2 Apache HTTP Server 2.4

Before installing web policy agents on your platform, also make sure that the system provides the required components.

All Systems

If agents use secure connections (SSL, TLS), then also make sure that OpenSSL is installed.

Linux Systems

Before installing web policy agents on Linux, make sure the system can run **gcc** 4.4.7. `libc.so.6` must be available and it must support the GLIBC_2.3 ABI. You can check this by running the following command: **strings libc.so.6 | grep GLIBC_2**.

Microsoft Windows Systems

Before installing the IIS 7 web policy agent on Microsoft IIS 7 or IIS 8, make sure that the optional Application Development component of Web Server (IIS) is installed. In the Windows Server 2012 Server Manager for example, Application Development is a component of Web Server (IIS) | Web Server.

Oracle Solaris Systems

Before installing web policy agents on Solaris 10, make sure you have applied the latest shared library patch for C++, at least 119963-16 on SPARC or 119964-12 on x64. The library is bundled on Solaris 10 update 5 and later.

Chapter 6

Getting Started for Architects and Deployers

The following section provides general instructions to get started with an OpenAM deployment.

6.1. Plan the Deployment

The initial process is the planning phase of your project.

- **Learn about OpenAM**. You can access online information, meet with your ForgeRock Sales representative, go to a seminar, or call ForgeRock about OpenAM's capabilities.

  The following are some general questions that you may want to have answered:

Table 6.1. OpenAM Initial Questions

OpenAM Initial Tasks	Done ?	
Understand the access management problems that OpenAM helps to solve	Y	N
Learn how to protect a Web site with OpenAM	Y	N
Get to know the OpenAM software deliverables	Y	N
Get to know the tools for administering OpenAM	Y	N
Get to know the APIs for OpenAM client applications	Y	N
Find out how to get help and support from ForgeRock and partners	Y	N
Find out how to get training from ForgeRock and partners	Y	N
Find out how to keep up to date on new development and new releases	Y	N
Find out how to report problems	Y	N

- **Set up a Demo or Pilot**. View an OpenAM demo or set up a pilot to determine how you want to use OpenAM to protect your site(s). ForgeRock Sales representatives can assist you with a demo or pilot.

- **Attend a Training Class**. ForgeRock presents effective training classes to deploy OpenAM in your environment. See ForgeRock University for more information.

- **Complete the Accreditation Program**. Complete the product-specific ForgeRock Accreditation Program to gain in-depth design and deployment expertise or seek partners who are ForgeRock Accredited Partners.

- **Determine Your Service Level Agreements**. ForgeRock provides a set of standard service level agreements that you can sign up for. ForgeRock also provides custom service level agreements if the standard set does not meet your needs.

Table 6.2. OpenAM Standard SLAs

Priority	Gold	Silver	Bronze
Urgent (P1)	2 Hour	4 Hour	Next Business Day
High (P2)	4 Hour	8 Hour	2 Business Days
Normal (P3)	6 Hour	Next Business Day	3 Business Days
Low (P4)	Next Business Day	2 Business Days	4 Business Days

- **Determine Your Services**. ForgeRock provides a full, proven-production Identity Management stack to meet your requirements.

Table 6.3. OpenAM Services

Services Task	Done ?	
Understand the services OpenAM software provides	Y	N
Determine which services to deploy	Y	N
Determine which services the deployment consumes (load balancing, application container, authentication services, configuration storage, profile storage, token/session storage, policy storage, log storage)	Y	N
Determine which services the deployment provides (SSO, CDSSO, SAML Federation IDP/SP, XACML PDP, REST STS, OAuth 2.0/OpenID Connect 1.0, and so forth)	Y	N
Determine which resources OpenAM protects (who consumes OpenAM services)	Y	N

- **Determine Your Deployment Objectives**. OpenAM provides proven performance and security in many production deployments. You should determine your overall deployment objectives.

Table 6.4. OpenAM Deployment Objectives

Deployment Objectives	Done ?	
Define deployment objectives in terms of service levels (expectations for authentication rates, active sessions maintained, session life cycles, policies managed, authorization decision rates, response times, throughput, and so forth)	Y	N
Define deployment objectives in terms of service availability (OpenAM service availability, authentication availability, authorization decision availability, session availability, elasticity)	Y	N
Understand how OpenAM services scale for high availability	Y	N
Understand the restrictions in an OpenAM deployment that uses stateless sessions	Y	N
Plan for availability (number of sites and servers, load balancing and OpenAM software configuration)	Y	N

Deployment Objectives	Done ?	
Define the domains managed and domains involved in the deployment	Y	N
Define deployment objectives for delegated administration	Y	N
Agree with partners for federated deployments on circles of trust and terms	Y	N

- **Plan Sizing**. At this stage, you should determine the sizing estimates for your deployment. ForgeRock Sales Engineers can assist you in this task.

Table 6.5. OpenAM Sizing

Sizing	Done ?	
Derive sizing estimates from service levels and availability	Y	N
Understand how to test sizing estimates (load generation tools?)	Y	N
Size servers for OpenAM deployment: CPU	Y	N
Size servers for OpenAM deployment: Memory	Y	N
Size servers for OpenAM deployment: Network	Y	N
Size servers for OpenAM deployment: I/O	Y	N
Size servers for OpenAM deployment: Storage	Y	N
Quantify impact on external services consumed (LDAP, other auth services, load balancing, and so forth)	Y	N
Plan testing and acceptance criteria for sizing	Y	N

- **Plan the Topology**. Plan your logical and physical deployment.

Table 6.6. OpenAM Topology Planning

Topology	Done ?	
Specify the logical and physical deployment topology (show examples of each)	Y	N
Determine whether to use the embedded or external directory service for configuration, CTS, and user data	Y	N
Plan installation of OpenAM services (including external dependencies)	Y	N
Plan installation of OpenAM policy agents, Fedlets, and OpenIG (might be done by partner service providers)	Y	N
Plan integration with client applications	Y	N
Plan customization of OpenAM (classic UI or XUI, user profile attributes, authentication modules, identity repositories, OAuth 2.0 scope handling, OAuth 2.0 response types, post-authentication actions, policy evaluation, session quota exhaustion actions, policy evaluation, identity data storage, OpenAM service, custom logger, custom Web policy agents).	Y	N

- **Plan Security**. At this stage, you must plan how to secure your deployment.

Table 6.7. OpenAM Security

Security	Done ?	
Understand security guidelines, including legal requirements	Y	N
Change default settings and administrative user credentials	Y	N
Protect service ports (Firewall, Dist Auth UI, reverse proxy)	Y	N
Turn off unused service endpoints	Y	N
Separate administrative access from client access	Y	N
Secure communications (HTTPS, LDAPS, secure cookies, cookie hijacking protection, key management for signing and encryption)	Y	N
Determine if components handle SSL acceleration or termination	Y	N
Securing processes and files (e.g. with SELinux, dedicated non-privileged user and port forwarding, and so forth)	Y	N

- **Post-Deployment Tasks**. At this stage, you should plan your post-deployment tasks to sustain and monitor your system.

Table 6.8. OpenAM Post-Deployment Tasks

Post Deployment Tasks	Done ?	
Plan administration following OpenAM deployment (services, agents/OpenIG, delegated administration)	Y	N
Plan monitoring following deployment	Y	N
Plan how to expand the deployment	Y	N
Plan how to upgrade the deployment	Y	N

6.2. Install the Components

The installation process requires that you implement your deployment plan.

- **Plan the Overall Deployment**. The initial planning step involves establishing the overall deployment. You should determine who is responsible for each task and any external dependencies.
- **Determine What To Install**. Based on your deployment plan, determine what you need to install.
- **Determine Your System Requirements**. Based on your deployment plan, determine your system requirements.
- **Prepare the Operating System**. Prepare your operating system, depending on the OS: Linux, Solaris, Windows, Cloud (Amazon EC2, OpenStack, and so forth), Virtual Machines (VMWare, Xen, Hyper-V, and so forth)

- **Prepare the Java Environment**. Prepare your Java environment, depending on your vendor type: Oracle, IBM, OpenJDK.
- **Prepare the App Server**. Prepare your application server, depending on type: Apache Tomcat, JBoss 4/5, WildFly, Jetty, Oracle WebLogic, IBM WebSphere. Also, prepare each app server for HTTPS.
- **Prepare the Directory Servers**. Prepare the configuration directory server, OpenDJ for the core token service (CTS), and the LDAP identity repository. For information on installing data repositories, see Chapter 1, "*Preparing For Installation*" in the *Installation Guide*.
- **Obtain the OpenAM Software**. You should obtain a supported ForgeRock release of OpenAM or an archive build. For the latest stable version of OpenAM, click *Enterprise Downloads*.
- **Configure OpenAM**. Install and configure OpenAM with or without the console, the setup tools (configurator), configuration tools (`ssoadm`, `ampassword`, `amverifyarchive`), or set up your scripted install and configuration of OpenAM. For information on installing OpenAM, see the Installation Guide.
- **Set up your Realms**. Within OpenAM, set up your realms and realm administrators if any. For more information on realms, see Chapter 4, "*Configuring Realms*" in the *Administration Guide*.
- **Configure Session State**. Configure sessions as stateful or stateless. For more information on session state, see Chapter 9, "*Configuring Session State*" in the *Administration Guide*.
- **Install Another OpenAM Instance**. Set up an identical instance of your first OpenAM instance. For information on installing multiple OpenAM servers, see Chapter 4, "*Installation Considerations for Multiple Servers*" in the *Installation Guide*.
- **Secure OpenAM**. Configure OpenAM to access external resources over HTTPS and LDAPS. Set up secure cookies and certificates. For more information, see Chapter 27, "*Securing OpenAM*" in the *Administration Guide*.
- **Configure High Availability**. Configure the load balancers, reverse proxies, and site(s). Configure OpenAM for session failover and server failover. For simple instructions to deploy OpenAM behind a load balancer, see *Deploying OpenAM behind a load balancer*. For an example of a reverse proxy with OpenAM, see *Simple Apache Reverse Proxy for OpenAM with Certificate-Based Authentication*. For information on configuring sites, see Chapter 4, "*Installation Considerations for Multiple Servers*" in the *Installation Guide*.
- **Prepare the Policy Agent Profiles**. Prepare the policy agent profile, agent authenticator, policy agent configuration, bootstrap configuration for a Java EE or Web policy agent. For more information, see Chapter 5, "*Configuring Policy Agent Profiles*" in the *Administration Guide*.
- **Install the Policy Agents**. Install the policy agents depending on the app server or Web server type. For app servers, Apache Tomcat, JBoss, Jetty, Oracle WebLogic, IBM WebSphere. For Web servers, Apache , Microsoft IIS. Set up any script installations of the policy agents. For more information, see the OpenAM Web Policy Agent documentation.
- **Customizing OpenAM**. Customize OpenAM for your organization. For information on customizing the OpenAM end-user pages, see Chapter 5, "*Customizing the OpenAM End User Pages*" in the *Installation Guide*.

- **Install OpenIG**. Determine which OpenIG deliverable to install (whether federation is involved). Prepare the Apache Tomcat, JBoss, Jetty, Oracle WebLogic app servers for installation. Install OpenIG. See the OpenIG documentation for details.
- **Plan Application and Host Backup**. Determine your backup strategy including LDIF exports, file system backups, tar files, and so forth. Also, consider log rotation and retention policies. For more information on backups, see Chapter 21, "*Backing Up and Restoring OpenAM Configurations*" in the *Administration Guide*.
- **Plan an OpenAM Upgrade**. You should know what is new or fixed in an upgrade version as well as the differences and compatibility between the current version and an upgrade. Know the limitations of an upgrade version. Plan a live upgrade without service interruption. Plan an offline upgrade with service interruption. Plan the test of the upgrade and revert a failed upgrade. For more information on upgrades, see the Upgrade Guide.
- **Upgrade OpenAM**. Upgrade OpenAM and other instances with or without the console. Upgrade the setup tools (configurator), configuration tools (`ssoadm`, `ampassword`, `amverifyarchive`), and the Java EE and/or Web policy agents. Upgrade OpenIG. For more information on upgrades, see the Upgrade Guide.
- **Remove OpenAM**. If required, remove OpenAM with or without the console. Remove setup and configuration tools. Remove the Java EE and/or Web policy agents. Remove OpenIG. For more information on removing OpenAM, see Chapter 8, "*Removing OpenAM Software*" in the *Installation Guide*.

Index

W