Oracle 9iAS Security - FAQ

Oracle9iAS Single Sign-On

What is Oracle9iAS Single Sign-On (SSO)?
Oracle9iAS SSO supports single sign-on for web (browser) clients. Oracle9iAS SSO allows web users who access Oracle9iAS to sign in once, and be authenticated to multiple web applications including Oracle Portal, Oracle E-Business Suite 11i, and non-Oracle applications. These web applications are classified as either a "partner" or "external" application.

Partner applications are those that work within the Oracle9iAS SSO framework. They are designed (or have been modified) to delegate responsibility for user authentication to Oracle9*i*AS SSO. They accept the user identity presented to them by Oracle9*i*AS SSO. Since partner applications rely on the authentication services of Oracle9*i*AS SSO, they do not need to implement their own authentication modules. User administration is simplified for partner applications, since there is no need to manage passwords for these applications. Deploying an application as a partner application thus can reduce both development and ongoing administrative expenses.

External applications are those that retain their own usernames and passwords, and do not delegate responsibility for authenticating users to Oracle9*i*AS SSO. These applications have not been developed or modified to work within the SSO framework. A typical external application might be one developed or deployed by a third party, such as a portal website which requires username and password for access to custom services like email. Although partner applications are preferable, external applications allow existing or legacy applications to work with Oracle9iAS SSO without any retrofitting.

What are the key strengths of Oracle9iAS?
Oracle9iAS SSO's main strengths are its flexibility, standards-based approach, and scalable design. These are described below:

- Flexibility – Oracle9iAS SSO is designed to work standalone or with your existing infrastructure including 3rd party SSO servers or LDAP directories.
- Standard-based approach - Oracle9iAS SSO uses today's Internet standards including HTTP(S) for communication, and cookies and X.509 certificates for user tokens.
- Scalable design – Oracle9iAS SSO is designed to support terabytes of user information with Oracle Internet Directory (OID) running on an Oracle database.

What are the new SSO features in 9iAS R2?
The new features in 9iAS R2 include:

- Mod_osso to give servlets access to authenticated usernames in HTTP headers
- Client certificate support
- User provisioning through OID's Delegated Administrative Services (DAS)
- Paranoid application support to force reauthentication for highly sensitive applications
- Single Sign-Off

- Global inactivity detection

How does mod_osso work?
Mod_osso is a new feature introduced in Oracle9iAS R2. It is an extension to the Oracle HTTP Server that enables the HTTP Server to be an SSO partner application. Applications running underneath the HTTP Server, such as servlets, can then obtain a user's authenticated identity from mod_osso in the form of an Apache header. Mod_osso therefore allows applications to participate in the Oracle9iAS SSO framework without requiring that the applications embed specific partner application logic. It is the recommended way for applications running on Oracle9iAS to participate in the Oracle9iAS SSO framework.

What is the difference between Login Server and Oracle9iAS SSO?
Login Server is bundled with Portal Server 3.0 (9iAS 1.0) and is the predecessor to Oracle9iAS Single Sign-On (SSO). With 9iAS R2, Login Server has been renamed Single Sign-On (Server) and is part of the 9iAS infrastructure layer.

How does SSO Server work with Oracle Internet Directory?
In Oracle9iAS 1.0.2.2, SSO (formerly known as Login Server) allowed but did not require usernames and passwords to be managed in Oracle's LDAP directory - Oracle Internet Directory (OID). In Oracle9iAS R2, usernames and passwords are always managed in OID. Along with Oracle9iAS SSO, OID is part of the 9iAS R2 infrastructure layer as it supports other 9iAS features.

How does Oracle9iAS SSO improve overall performance versus a non-SSO environment?
For the end user, SSO saves time by requiring only a single sign-on. After the first login to an application, the user no longer sees login screens for all SSO enabled applications, unless there are timeouts or deliberate security settings forcing additional checks. The end result for the user is less keystrokes and reduced steps to access applications.

How does SSO work with Java/J2EE applications?
From Oracle9iAS SSO's perspective, Java applications are no different than other applications. Java developers are encouraged, however, to take advantage of the Oracle9iAS JAAS Provider (available in Oracle9iAS R2). JAAS is a specification from the Java community that provides simple APIs for authentication and authorization. Oracle's JAAS Provider is integrated with the Oracle9iAS SSO, giving Java developers a seamless way to support single sign-on.

Is SSO available directly to the Oracle database (without 9iAS SSO)?
Yes. Oracle has supported single sign-on from database (Net8) clients to the database since Oracle 7 through the Oracle Advanced Security Option (ASO). ASO supports mechanisms, such as Kerberos and SSL authentication, which allow a Net8 client to sign in once and access multiple Oracle databases in an enterprise, without having to login to each database independently.

Will 9iAS R2 SSO work with Portal 3.0.8 or 3.0.9 applications?
Oracle9iAS R2 SSO supports backward compatibility with older SSO (Login) Server partner applications. These partner applications will not have access to all the new R2 SSO features, such as timeout, but can get authenticated user identity from Oracle9iAS SSO. The Portal itself has dependencies on the SSO Server, which in 3.0.8 and 3.0.9 was actually part of the Portal build (e.g., many functions involving Portal user management point to the SSO Server). These dependencies will cause problems if a 3.0.8 or 3.0.9 Portal attempts to use a R2 SSO Server instead of the 3.0.8 or 3.0.9 SSO Server it expects.

How do I set the default page after a user logs out?

If you want everyone to go to the same page when they log out, you can set a default page for the public user by editing the portal user profile for the PUBLIC user. You set the default page for this user and upon logout, that is the page that is navigated to.

Does SSO support environments across multiple domains since it's using cookies?

Yes. The following cookies are used in an SSO environment:

1. SSO has its own cookie that only it uses. (no domain issue)
2. Partner apps have their own cookie that only they use. (no domain issue)
3. SSO passes user credentials to partner apps using a token (sometimes called a "URL Cookie" or URLC). Note that this token is not a browser cookie but just part of the HTTP header, so there are no issues with domain restrictions.

Is the SSO 3rd party API different between 9iAS 1.0 and 9iAS R2?

It is the same except for the some functions for external authentication (which are not supported in 9.0.2) as described in the documentation.

Has the SSO SDK API for Java changed between V1 and V2?

No. Although in V2 there is a more standard way to access the functionality through our JAAS provider. JAAS is part of the J2EE 1.3 specification and is an API that provides authentication and authorization to Java applications. Oracle has integrated our JAAS implementation with SSO.

What is the difference between (Database) Server-Based Single Sign-On and Middle Tier Single Sign-On?

You would use one or the other, and it depends on your architecture. Database SSO (with Advanced Security - ASO) is for clients that login directly to the database, whereas 9iAS SSO is for web-based applications using 9iAS. You can use ASO with proxy authentication to get 3-tier SSO for enterprise users, those users stored in OID. When using the database/ASO solution, you'd be using SSL for single sign-on or passwords for single login.

Oracle9iAS JAAS Provider

What is the Oracle9iAS JAAS Provider?

JAAS stands for Java Authentication and Authorization Services. Oracle's JAAS implementation, known officially as the Oracle9iAS JAAS Provider and known internally as "JAZN," provides core security services for developing Java-based applications for Oracle9*i*AS. Oracle's JAAS Provider is part of a larger set of security services in Oracle9*i*AS that includes single sign-on, network encryption, and other features.

How does Oracle's JAAS Provider fit into the Java security model?

JAAS as a specification is not yet integrated with the Java 2 Platform, Enterprise Edition (J2EE) security model. However, Oracle's JAAS Provider *does* provide security for Oracle Containers for Java (OC4J), Oracle's J2EE implementation, to enforce security constraints for Web Servlets, Java Server Pages (JSPs) and EJB components.

What are the security services provided by Oracle's JAAS Provider?

Oracle's JAAS Provider provides key security services for:

- authentication (identifying users)
- authorization (limiting what they can do)
- delegation (enabling code to run securely, with privileges of other users).

Specific feature descriptions follow:

Authentication

- Oracle's JAAS Provider Web PlugIn integrates Oracle9*i*AS Single Sign-On (SSO) with OC4J
- LoginModules enable customers to develop strong authentication for Java-based applications

The ability to integrate Java-based applications with single sign-on provides greater security unification across Oracle9*i*AS, since any application can take advantage of Oracle9*i*AS Single Sign-on. Also, the ability to add strong authentication gives developers extensible security for Java-based applications. A Java-based banking application might require stronger authentication (e.g. using a challenge-response mechanism) than a web site offering static content, for example.

Authorization

- centrally manage access control policies in Oracle Internet Directory (through JAZN-LDAP) or in the file system (JAZN-XML)
- ability to partition security policy by subscriber
- support for hierarchical, role-based access control
- support for principal (that is, user) and code-based policies

Each Realm (an association of users, such as a subscriber base) can have its own private, partitioned access control policy, administered by its own realm-specific administrator. This allows Java-based applications to leverage the centralized user management capabilities of Oracle Internet Directory, yet allow delegated administration by organization. Java-based applications can retrieve roles for users, and roles can be hierarchical (that is, role can be assigned to other roles). Principal and code-based policies can require that a user only assume particular privileges when accessing particular Java applications from particular locations, allowing Java developers to fine-tune their security policies.

Delegation

- support for impersonation of a specified user (both RunAsClient and RunAsID)

RunAsClient means an enterprise bean, servlet or Java Server Page (JSP) can be configured to run with the permissions associated with the current client. RunAsID means an enterprise bean, servlet or JSP can be configured to run with the permissions associated with a specified user (e.g. run as "DBAdmin"). This allows developers to enforce "least privilege" in their applications, allowing users only those privileges needed to perform a function, because users can only exercise privileges in the context of a well-formed business rule (e.g. an enterprise bean).

What are the key benefits of Oracle's JAAS Provider?

Oracle's JAAS Provider provides benefits to any customer developing Java-based applications. One of the biggest benefits of Oracle's JAAS Provider is integration with Oracle9*i*AS Single Sign-On (SSO). This integration enables any Java-based application to participate in web single sign-on.

What administration tools exist for Oracle's JAAS Provider?

Oracle's JAAS Provider has both a command line tool and a graphical management tool accessible through the Oracle Enterprise Manager console. The command line tool has additional functionality not yet found in the GUI tool, such as migration capabilities from principals.xml to JAZN-XML.

What extensions has Oracle made to JAAS?

Oracle has made several extensions to JAAS to enhance the basic security mechanisms:

- *Authentication* — Oracle's JAAS Provider integrates with Oracle9*i*AS Single Sign-on (formerly known as Login Server) for authentication and single sign-on in J2EE application environments. Any Java-based application can thus be integrated with Oracle9*i*AS Single Sign-on. Oracle's JAAS Provider also includes extensible authentication (through a RealmLoginModule class), which enables developers to add their own authentication mechanisms to J2EE applications. For example, a banking application may wish to use a custom challenge-response mechanism to authenticate users accessing a JSP.
- *Authorization* — Oracle's JAAS Provider provides full support for role-based access control (RBAC), including the ability to grant roles to other roles. Authorizations can be stored in (and retrieved from) either Oracle Internet Directory (an LDAP-based directory server) or from the file system.
- *Secure Configuration and Management* — Oracle provides an Oracle-proprietary Realm API package to support realms (user communities), enabling more granular associations of users and roles. Through JAZN-XML, Oracle's JAAS Provider provides secure configuration through password obfuscation (so that passwords are not stored unencrypted).

What is JAZN-LDAP?

JAZN-LDAP is an implementation of the JAAS API that retrieves user and authorization information securely from Oracle Internet Directory (OID). JAZN-LDAP is particularly useful for applications that have a large user community, for which scalability is a strong requirement.

What is JAZN-XML?

JAZN-XML is a fast, lightweight implementation of the JAAS API that is based on XML as an encoding mechanism. JAZN-XML allows Java developers to retrieve user and role information securely from operating system files rather than retrieving information from Oracle Internet Directory (as is the case with JAZN-LDAP). JAZN_XML supports lightweight deployments of Oracle9*i*AS and provides a more secure alternative to principals.xml.

What integration does Oracle's JAAS Provider have with Secure Sockets Layer (SSL)?

Oracle's JAAS Provider supports the use of SSL in multiple ways. Users are able to authenticate to Apache (or to the SSO Server) using SSL, and that authentication is accepted by JAAS. This supports client-side authentication via SSL for Java-based applications.

Additionally, Oracle's JAAS Provider is able to assign permissions based on the distinguished name (DN) from an X.509 certificate. For example, a developer could enforce the access control condition that users having an Organizational Unit (OU) component of "XYZ Corporation" are able to run an Enterprise Java Bean, but other users cannot access the EJB. This allows developers to utilize the information within an X.509 certificate for access control, thereby leveraging an organization's deployment of a public key infrastructure (PKI).

Is information that Oracle's JAAS Provider retrieves from Oracle Internet Directory cached?

Yes, Oracle has implemented time-based caching.

What forms of authentication are supported with Oracle's JAAS Provider?

The JAAS specification supports the Pluggable Authentication Module (PAM) framework. So any login module that complies with the JAAS API will plug-in to Oracle's JAAS Provider. The Oracle JAAS Provider ships with the ability to support the following forms of authentication: Basic, Form-based, SSL client certificate, and Oracle9iAS Single Sign-On Server.

How is Oracle's JAAS Provider packaged?

Oracle's JAAS Provider is a standard feature of all versions of Oracle9*i*AS, and installs by default. JAZN-XML is configured by default. JAZN-LDAP is not configured as the default since Oracle Internet Directory is not installed on all versions of Oracle9iAS.

Is JAAS just for web applications with a web front end?

No, JAAS can be used with EJBs directly without any web interface.

How can I debug the JAAS Provider if it is not working properly with OID?

Verify that you are providing valid credentials and using the correct realm. You can use the "ldapbind" command to verify. For example, user "scott" with password "tiger" in the "jazn.com" realm can be verified as follows:

> ldapbind -h oidhost -p port -Dcn=scott,cn=allusers,cn=jazn.com,cn=subscribers -w tiger

This should return a "bind successful" command, otherwise there is something wrong with your credentials and realm setup.

Is there a difference between principals.xml and the JAAS Provider for protecting EJB and Web (JSP/Servlet) components?

No. Using JAZN-XML or JAZN-LDAP instead of principals.xml is completely transparent. One can still define who can access what down to the method level using the JAAS Provider.

How do J2EE applications running on OC4J take advantage of SSO/OID?

In 9iAS R2, J2EE apps can take advantage of SSO/OID using the JAAS Provider. The JAAS Provider supports any JAAS-compliant LoginModule and it also has the option to get authentication information from Oracle9iAS SSO directly, which is unique to Oracle. This provides a unified authentication framework across not only J2EE apps but also all applications covered by Oracle9iAS SSO. The JAAS Provider also has the ability to use OID as its repository (known as "JAZN-LDAP") for optimum scalability and manageability.

How do I associate a permission to a block of code that only users from a certain group can run?

If your "block of code" is a servlet, you can use J2EE static declarative constraints - e.g. specify logical security roles in your web.xml, specify which roles can access what URLs, and then map those roles to JAZN users and roles in orion-application.xml. These are all documented in the 9iAS R2 documentation set.

If your "block of code" is more akin to a library method call or if your code is not in a servlet, then you should use Java2 permissions (unless you want to roll your own). You can grant the permissions to a user/role via JAZN (either JAZN-LDAP or JAZN-XML is fine) and then in your code do a Subject.doAs() - with your "block of code" as the action argument. This is further documented in the JAAS documentation.