Not for Publication

Sun Java System Federated Access Manager 8.0 Developer's Guide

Beta

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–3748–05
April 2008

080214@19028

List of Remarks

Contents

CHAPTER 1

Understanding the Client Software Development Kit

The Sun Java™ System Federated Access Manager Client Software Development Kit (Client SDK) provides Java libraries for integrating access management functionality within stand-alone applications and web applications. The Client SDK can be used in remote applications to take advantage of Federated Access Manager services such as authentication, single sign-on (SSO), authorization, auditing and logging, and Security Assertion Markup Language (SAML). This chapter contains the following sections:

About the Client SDK

Remark 1–1 Reviewer **This chapter talks only about Java SDK. What about .NET and C? Is there stuff I should be writing about an SDK for those languages?**

The Federated Access Manager Client SDK contains Java packages and class files that can be used by developers to implement remote applications with Federated Access Manager services such as authentication, authorization, SSO, and SAML. The Client SDK is a streamlined version of the complete SDK installed with Federated Access Manager. The Client SDK includes only the client-side classes and configuration properties needed by remote applications to communicate with Federated Access Manager services. It is aimed at applications that use identity APIs at run time for authentication, SSO, policy evaluation and enforcement, and obtaining and setting user attributes. It is not for use by applications that perform policy management or identity management (creation and deletion of entries). From a deployment point of view, the Client SDK offers the following:

- The Client SDK communicates directly with Federated Access Manager using XML (SOAP) over HTTP or HTTPS. In turn, Federated Access Manager communicates directly with the data store.
- The Client SDK does not require administrator credentials.
- Applications using the Client SDK can be deployed in demilitarized zones (DMZs), and a firewall can be placed between them and Federated Access Manager.
- The Client SDK includes samples to show how it can be used.

The packages that comprise the Client SDK include:

- `com.iplanet.am.sdk`
- `com.iplanet.am.util`
- `com.iplanet.sso`
- `com.sun.identity.authentication`
- `com.sun.identity.federation`
- `com.sun.identity.idm`
- `com.sun.identity.liberty.ws`
- `com.sun.identity.log`
- `com.sun.identity.policy`
- `com.sun.identity.policy.client`
- `com.sun.identity.saml`
- `com.sun.identity.saml2`
- `com.sun.identity.smt`
- `com.sun.identity.xacml`
- `com.sun.identity.wss`

Descriptions of these packages can be found in *Federated Access Manager 8.0 Java API Reference*. A complete listing of the classes that comprise the Client SDK can be found in the `ClientSDKClasses` file available on the OpenSSO web site.

Caution – It is recommended that developers don't call `com.iplanet.am.sdk`, `com.iplanet.am.util`, `com.sun.identity.policy`, and `com.sun.identity.sm` directly.

Using the Client SDK

[Remark 1–2 Reviewer: More real world type examples for this section would be great!!!] There are many ways to use the Client SDK. Following is a list of some of them.

- Build a proprietary application framework in which the Client SDK is a part. The Client SDK features can allow independence from policy agents.

- Access profile data to perform authentication and authorization beyond what is offered out-of-the-box.
- Allow authenticated and non-authenticated users access to a login process with a registration option that, if accepted, would create a user account.

Running the Client SDK Samples

Federated Access Manager comes with samples and source code that can help developers understand how the Client SDK classes can be implemented. The samples, acting as standalone applications, can be run on the command-line and in a web browser to see the function being performed. By looking at the provided sample source code you can understand how the Client SDK classes were used to perform the sample function.

`fam-client.zip` is the Client SDK sample ZIP and located in the `samples` directory of the inflated Federated Access Manager ZIP. After inflating `fam-client.zip` to its core `fam-client` directory, you will find two subdirectories:

- `sdk` contains the command line samples and source code. You must compile this before using the command line samples.
- `war` contains deployable WAR files comprised of the Client SDK and web-based samples.

The following sections further explain the two directories.

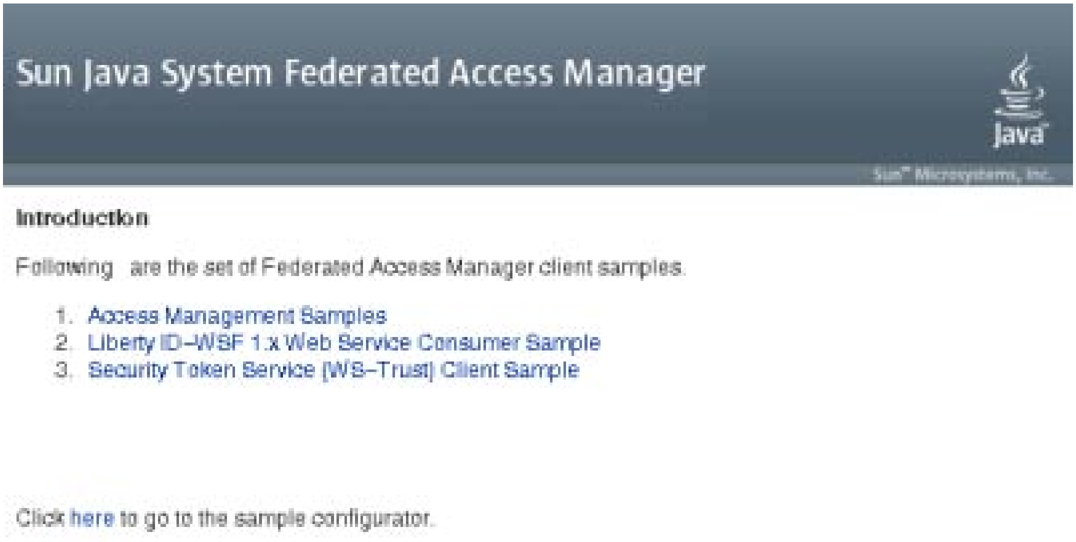
- "Web-based Samples" on page 9
- "Command Line Samples" on page 11

Web-based Samples

The web-based Client SDK samples are run by deploying a WAR file. The Client SDK WAR files are located in the `samples/fam-client/war` directory of the inflated Federated Access Manager download. They are:

- `fam-client-jdk15.war` requires Java Platform, Enterprise Edition 1.5.
- `fam-client-jdk14.war` requires Java 2, Standard Edition 1.4.2.

These WAR files contain the web-based samples and the Client SDK for use with them. Deploy either `fam-client-jdk14.war` or `fam-client-jdk15.war` to your web container, depending on the version of Java installed on the machine. After deploying, launching, and configuring the appropriate WAR, click the resulting link to proceed to the web-based samples Introduction page. This page contains links to the web-based samples.



Introduction

Following are the set of Federated Access Manager client samples.

1. Access Management Samples
2. Liberty ID-WSF 1.x Web Service Consumer Sample
3. Security Token Service (WS-Trust) Client Sample

Click here to go to the sample configurator.

Note – For more information on configuring the Client SDK, see "Using `AMConfig.properties` with Client SDK" on page 12.

The following table documents the web-based sample applications and their corresponding source file. Look in the `samples` directory for additional source code files not specifically called out below. The source files and directories noted in this table are linked to the version on the OpenSSO web site.

TABLE 1–1 Web-based Client SDK Samples

Sample	Function	Source
Service Configuration Sample Servlet	Retrieves and displays attributes of the entered service name	ServiceConfigServlet.java
User Profile (Attribute) Sample Servlet	Retrieves and displays the attributes that correspond to the entered user ID	UserProfileServlet.java
Policy Evaluator Client Sample Servlet	Retrieves from the Policy Service a policy decision that would be passed to a web agent for enforcement	PolicyClientServlet.java

TABLE 1–1 Web-based Client SDK Samples *(Continued)*

Sample	Function	Source
Single Sign-on Token Verification Servlet	Validates a session token and then displays the user profile associated with it	SSOTokenSampleServlet.java
Liberty ID-WSF 1.x Web Service Consumer Sample	Query and modify the Discovery Service and the Liberty Personal Profile Service	wsc Directory
Security Token Service (WS-Trust) Client Sample	Obtain security tokens from the Security Token Service	sts Directory

Command Line Samples

The command line samples are located in the `samples/fam-client/sdk` directory of the inflated Federated Access Manager download. These samples must be compiled before they can be used by running `scripts/compile-samples.sh`.

Caution – Be sure to run all the scripts discussed in this section from outside the `scripts` directory: `scripts/setup.sh`

The `README` in the `sdk` directory contains instructions on how to run the command line samples. The table documents the command line sample applications and their corresponding source file. Look in the `samples` directory for additional source code files not specifically called out below.

Note – The source files in this table are linked to the version on the OpenSSO web site.

TABLE 1–2 Command Line Client SDK Samples

Sample	Function	Source
`setup.sh`	Create `AMConfig.properties` and populate it with values based on your deployment.	Main.java

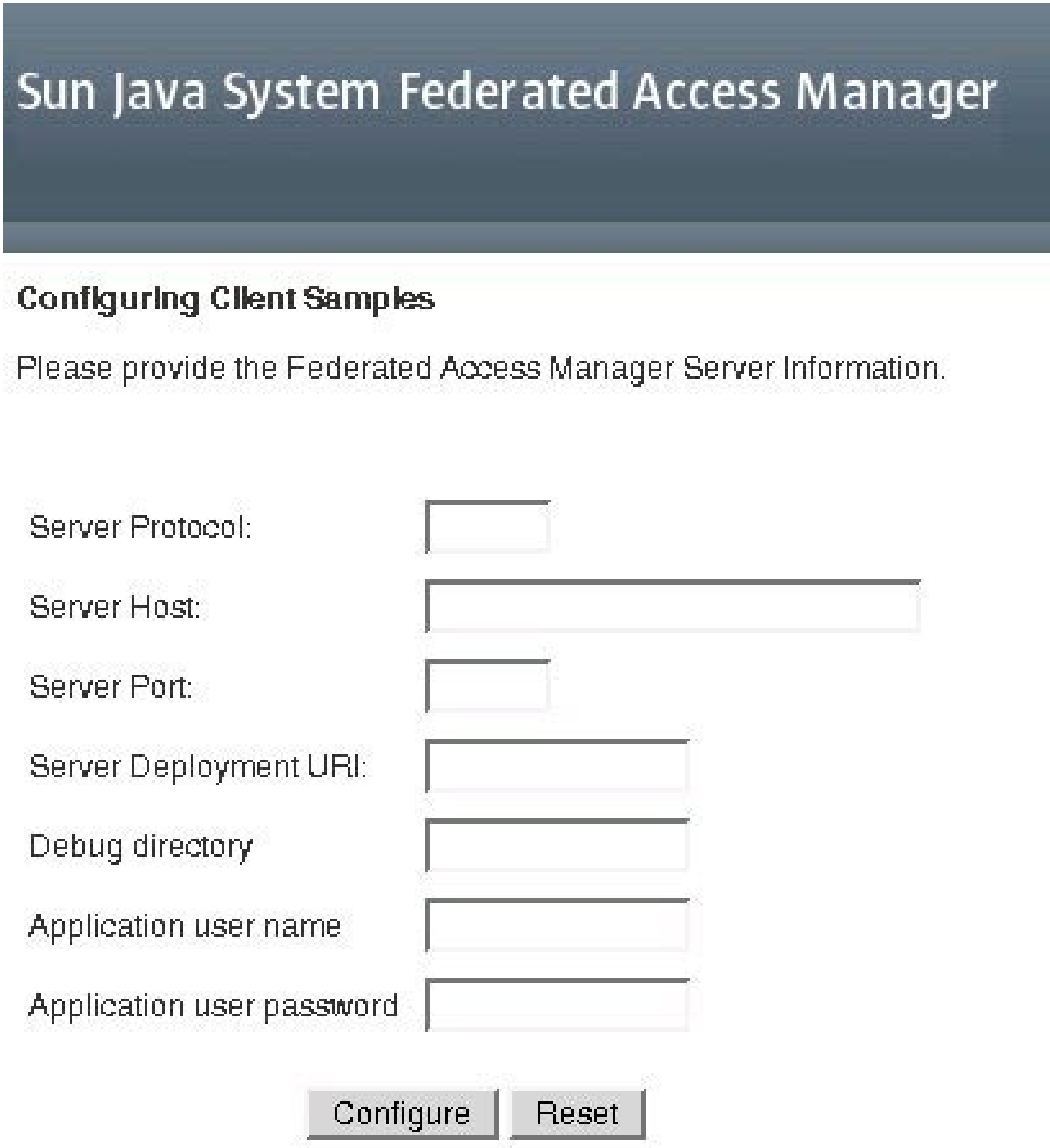
TABLE 1–2 Command Line Client SDK Samples *(Continued)*

Sample	Function	Source
`Login.sh`	Logs in and then logs out the user	Login.java
`CommandLineSSO.sh`	Demonstrates how to retrieve a user profile	CommandLineSSO.java
`CommandLineIdrepo.sh`	Perform operations on the Identity Repository; for example, create an identity, delete an identity and search or select an identity	idrepo Directory
`SSOTokenSample.sh`	Verifies a session token from a SSOTokenID input	SSOTokenSample.java
`run-policy-evaluation-sample.sh`	Returns a policy decision based on console created user and configured policy	policy Directory
`run-xacml-client-sample.sh`	Constructs a XACML request, makes an authorization query, receives the decision, and prints out the response	XACMLClientSample.java

Using `AMConfig.properties` with Client SDK

[Remark 1–3 Reviewer:] Although `AMConfig.properties` has been deprecated as the configuration data store for the Federated Access Manager application, the file is still used to store configuration data for the Client SDK. This `AMConfig.properties` points to the instance of Federated Access Manager that will be used by the Client SDK samples. After deploying and launching one of the sample WAR files (as discussed in "Web-based Samples" on page 9), a Client SDK configuration page is displayed.

Remark 1–4 Writer **Need new graphic header now says 'configure client SDK'**



Sun Java System Federated Access Manager

Configuring Client Samples

Please provide the Federated Access Manager Server Information.

Server Protocol:

Server Host:

Server Port:

Server Deployment URI:

Debug directory

Application user name

Application user password

Entering the appropriate values and clicking Configure creates an `AMConfig.properties` file under the home directory of the user running the web container. This value is indicated by the JDK system property `user.home`. When running the command line interface samples (as discussed in “Command Line Samples” on page 11) `AMConfig.properties` is created in the `samples/sdk/resources` directory of the inflated Federated Access Manager ZIP.

Note – Both famclientsdk.jar and servlet.jar are required in the CLASSPATH of the machine on which the Client SDK is installed.

An AMConfig.properties file with the information needed to point to the remote Federated Access Manager server must be accessible to the Client SDK from the machine on which the client application is hosted. The AMConfig.properties created by the sample WAR can be modified for this purpose. The following sections explain how to do this.

- “Federated Access Manager Properties for AMConfig.properties” on page 14
- “Initializing the AMConfig.properties Properties” on page 22

Note – An AMConfig.properties file is also created and populated with values when the setup.sh script is run as discussed in “Command Line Samples” on page 11.

Federated Access Manager Properties for AMConfig.properties

Federated Access Manager properties used by the Client SDK are contained in the AMConfig.properties file generated by the Client SDK configured during installation. (See “Using AMConfig.properties with Client SDK” on page 12.) Additional properties can be added to this file as the client application can register for notification of changes to session and user attributes, and policy decisions. The following sections describe these properties.

Remark 1–5 Reviewer **Please review the properties in this section carefully. Let me know if any are missing or if any need to be removed. Also check the values: I modifed them from the opensso values I saw - mostly replacing opensso and amserver with fam. Finally, should the properties below match the properties in the AMConfig.properties file generated by the Client SDK configuration page?**

- “Naming Properties” on page 15
- “Debug Properties” on page 15
- “Notification URL Property” on page 15
- “Security Credentials Properties” on page 16
- “Encryption Properties” on page 16
- “Cache Update Properties” on page 16
- “Client Services Properties” on page 17
- “Cookie Property” on page 17
- “Session Service Properties” on page 17
- “Certificate Database Properties” on page 18
- “Policy Client Properties” on page 18
- “Monitoring Framework Property” on page 19
- “Remote Client SDK Property” on page 19

- "Federation Properties" on page 19

Naming Properties

`com.iplanet.am.naming.url`

This is a required property. The value of this property is the URI of the Naming Service from which the Client SDK would retrieve the URLs of Federated Access Manager internal services. Example:

```
com.iplanet.am.naming.url=http://FAM_Host_Machine.domain_name:port
/fam/namingservice
```

`com.iplanet.am.naming.failover.url`

This property can be used by any remote application developed with the Client SDK that wants failover in, for example, session validation or getting the service URLs. Example:

```
com.iplanet.am.naming.failover.url=http://FAM_Host_Machine.domain_name:port
/fam/failover
```

Debug Properties

`com.iplanet.services.debug.level`

Specifies the debug level. Values are:

- **`Off`** specifies that no debug information is recorded.
- **`error`** specifies that there should be no errors in the debug files. This level is recommended for production environments.
- **`warning`** is not a recommended value at this time.
- **`message`** alerts to possible issues using code tracing. Most Federated Access Manager modules use this level to send debug messages.

Caution – **`warning`** and **`message`** should not be used in production. They cause severe performance degradation and an abundance of debug messages.

`com.iplanet.services.debug.directory`

The value of this property is the output directory for the debug information. The directory should be writable by the server process. Example:

```
com.iplanet.services.debug.directory=/fam/debug
```

Notification URL Property

`com.iplanet.am.notification.url`

The value of this property is the URI of the Notification Service running on the machine where the Client SDK is installed. Example:

```
com.iplanet.am.notification.url=http://SDK_Host_Machine.domain_name:port
/fam/notificationservice
```

Security Credentials Properties

`com.sun.identity.agents.app.username`
User with permission to read Federated Access Manager configuration data. Default:

```
com.sun.identity.agents.app.username=UrlAccessAgent
```

`com.iplanet.am.service.password`
Password of user with permission to read Federated Access Manager configuration data.

Note – Before running the Client SDK sample applications, you need to add `changeit` as a value for this property.

`com.iplanet.am.service.secret`
The encryption key used to encrypt the password. Example:

```
com.iplanet.am.service.secret=AQIC24u86rq9RRZGr/HN25OcIuO6w+ne+0lG
```

Encryption Properties

`am.encryption.pwd`
The encryption key used to decrypt service configuration passwords. Example:

```
am.encryption.pwd=ENCRYPTION_KEY
```

`com.sun.identity.client.encryptionKey`
Encryption key used to encrypt and decrypt data used locally within the client application. Example:

```
com.sun.identity.client.encryptionKey=ENCRYPTION_KEY_LOCAL
```

`com.iplanet.security.encryptor`
Property to set the default encrypting class. Values are:

- `com.iplanet.services.util.JCEEncryption`
- `com.iplanet.services.util.JSSEncryption`

Cache Update Properties

`com.sun.identity.sm.cacheTime`
Cache update time (in minutes) for service configuration data if notification URL is not provided. Example:

```
com.sun.identity.sm.cacheTime=1
```

`com.iplanet.am.sdk.remote.pollingTime`

Cache update time (in minutes) for user management data if notification URL is not provided. Example:

```
com.iplanet.am.sdk.remote.pollingTime=1
```

Client Services Properties

These properties are defined by the Client SDK configuration page.

`com.iplanet.am.server.protocol`

Protocol of machine on which Federated Access Manager is deployed. Example:

```
com.iplanet.am.server.protocol=http
```

`com.iplanet.am.server.host`

Name and domain of machine on which Federated Access Manager is deployed. Example:

```
com.iplanet.am.server.host=machine2.sun.com
```

`com.iplanet.am.server.port`

Port of machine on which Federated Access Manager is deployed. Example:

```
com.iplanet.am.server.port=8080
```

`com.iplanet.am.services.deploymentDescriptor`

URI of the deployed instance of Federated Access Manager. Example:

```
com.iplanet.am.server.protocol=fam
```

Cookie Property

`com.iplanet.am.cookie.name`

The name of the Federated Access Manager cookie. Example:

```
com.iplanet.am.cookie.name=iPlanetDirectoryPro
```

Session Service Properties

`com.iplanet.am.session.client.polling.enable`

A value of `true` or `false` enables or disables, respectively, client-side session polling.

`com.iplanet.am.session.client.polling.period`

Specifies the number of seconds in the polling period. Example

```
com.iplanet.am.session.client.polling.period=180
```

Certificate Database Properties

Remark 1–6 Writer

This only covers JSS. Most recently we've added some new properties for JCE/JSSE based provider to support SSL with client auth. We need to doc those properties too. This section should be tied to first two use cases in new use case section

`com.iplanet.am.admin.cli.certdb.dir`
Identifies the directory path to the certificate database for initializing the JSS Socket Factory when the Federated Access Manager web container is configured for SSL.

`com.iplanet.am.admin.cli.certdb.passfile`
Identifies the certificate database password file for initializing the JSS Socket Factory when the Federated Access Manager web container is configured for SSL. Example:

```
com.iplanet.am.admin.cli.certdb.passfile=/config/.wtpass
```

`com.iplanet.am.admin.cli.certdb.prefix`
Identifies the certificate database prefix for initializing the JSS Socket Factory when the Federated Access Manager web container is configured for SSL.

Policy Client Properties

`com.sun.identity.agents.server.log.file.name`
Specifies name of the client's policy log file. Example:

```
com.sun.identity.agents.server.log.file.name=amRemotePolicyLog
```

`com.sun.identity.agents.logging.level`
Specifies the granularity of logging to the client's policy log file.

- **NONE** is the default value. Nothing is logged.
- **ALLOW** logs allowed access decisions.
- **DENY** logs denied access decisions.
- **BOTH** logs allowed and denied access decisions.
- **DECISION**

`com.sun.identity.agents.notification.enabled`
A value of `true` or `false` enables or disables, respectively, notifications from Federated Access Manager for updating the client cache.

`com.sun.identity.client.notification.url`
Specifies the URL to which policy, session, and agent configuration notifications from Federated Access Manager are sent.

`com.sun.identity.agents.polling.interval`
Specifies the number of minutes after which an entry is dropped from the Client SDK cache. Example:

```
com.sun.identity.agents.polling.interval=3
```

`com.sun.identity.policy.client.cacheMode`
Specifies the cache mode for the client policy evaluator. Values are:

- **`subtree`** specifies that the policy evaluator obtains policy decisions from the server for all the resources from the root of resource actually requested.
- **`self`** specifies that the policy evaluator obtains policy decisions from the server only for the resource actually requested.

`com.sun.identity.policy.client.usePre22BooleanValues`
Define and set this property to `false` if you do not want to use Boolean values. The default value is `true` if the property is not defined.

Monitoring Framework Property

`com.sun.identity.monitoring=off` **[Remark 1–7 Reviewer: How do I reword this? No JES, right?]** Explicitly disables Java Enterprise System (JES) monitoring services in the sample client applications.

Remote Client SDK Property

`com.iplanet.am.sdk.package` If you want to use a remote instance of the Client SDK, set the value of this property to **`remote`**.

The default value is `ldap` if not explicitly defined.

Federation Properties

You must manually add these federation properties to `AMConfig.properties` as needed. They are not automatically placed in the file when generated.

com.sun.identity.wss.provider.plugins.AgentProvider

`com.sun.identity.liberty.ws.soap.supportedActor`
Supported SOAP actors. Each actor must be separated by a pipe (|). Example:

```
com.sun.identity.liberty.ws.soap.supportedActors=
http://schemas.xmlsoap.org/soap/actor/next
```

`com.sun.identity.liberty.interaction.wspRedirectHandler`
Indicates the URL for `WSPRedirectHandlerServlet` to handle Liberty the WSF web service provider-resource owner. Interactions are based on user agent redirects. The servlet should be running in the same JVM where the Liberty service provider is running.

`com.sun.identity.liberty.interaction.wscSpecifiedInteractionChoice`
Indicates whether the web service client should participate in an interaction. Valid values are `interactIfNeeded | doNotInteract | doNotInteractForData`. Default value is `interactIfNeeded`. Default value is used if an invalid value is specified.

`com.sun.identity.liberty.interaction.wscWillInlcudeUserInteractionHeader`
Indicates whether the web service client should include `userInteractionHeader`. Valid values are `yes` and `no` (case ignored). Default value is `yes`. Default value is used if no value is specified.

`com.sun.identity.liberty.interaction.wscWillRedirect`
Indicates whether the web service client will redirect user for an interaction. Valid values are `yes` and `no`. Default value is `yes`. Default value is used if no value is specified.

`com.sun.identity.liberty.interaction.wscSpecifiedMaxInteractionTime`
Indicates the web service client preference for acceptable duration (in seconds) for an interaction. If the value is not specified or if a non-integer value is specified, the default value is `60`.

`com.sun.identity.liberty.interaction.wscWillEnforceHttpsCheck`
Indicates whether the web service client enforces that redirected to URL is HTTPS. Valid values are `yes` and `no` (case ignored). The Liberty specification requires the value to be `yes`. Default value is `yes`. Default value is used if no value is specified.

`com.sun.identity.liberty.interaction.wspWillRedirect`
Indicates whether the web service provider redirects the user for an interaction. Valid values are `yes` and `no` (case ignored). Default value is `yes`. Default value is if no value is specified.

`com.sun.identity.liberty.interaction.wspWillRedirectForData`
Indicates whether the web service provider redirects the user for an interaction for data. Valid values are `yes` and `no`. Default value is `yes`. If no value is specified, the value is `yes`.

`com.sun.identity.liberty.interaction.wspRedirectTime`
Web service provider expected duration (in seconds) for an interaction. Default value if the value is not specified or is a non-integer value is `30`.

`com.sun.identity.liberty.interaction.wspWillEnforceHttpsCheck`
Indicates whether the web service client enforces that `returnToURL` is HTTP. Valid values are `yes` and `no` (case ignored). Liberty specification requires the value to be `yes`. Default value is `yes`. If no value is specified, then the value used is `yes`.

`com.sun.identity.liberty.interaction.wspWillEnforceReturnToHostEqualsRequestHost`
Indicates whether the web services client enforces that `returnToHost` and `requestHost` are the same. Valid values are `yes` and `no`. Liberty specification requires the value to be `yes`.

`com.sun.identity.liberty.interaction.htmlStyleSheetLocation`
Indicates the path to the style sheet used to render the interaction page in HTML.

`com.sun.identity.liberty.interaction.wmlStyleSheetLocation`
Indicates the path to the style sheet used to render the interaction page in WML.

Example: `com.sun.identity.liberty.interaction.wmlStyleSheetLocation=/opt/SUNWam/lib/is-wml.`

`com.sun.identity.liberty.ws.interaction.enable`
Default value is false.

`com.sun.identity.wss.provider.config.plugin=`
`com.sun.identity.wss.provider.plugins.AgentProvider`
Used by the web services provider to determine the plug-in that will be used to store the configuration.

Example: `com.sun.identity.wss.provider.config.plugin=`
`com.sun.identity.wss.provider.plugins.AgentProvider`

`com.sun.identity.loginurl`
Used by the web services clients in Client SDK mode. Example:

`com.sun.identity.loginurl=https://`*hostName*`:`*portNumber*`/amserver/UI/Login`

`com.sun.identity.liberty.authnsvc.url`
Indicates the Liberty authentication service URL.

`com.sun.identity.liberty.wsf.version`
Used to determine which version of the Liberty identity web services framework is to be used when the framework can not determine from the inbound message or from the resource offering. This property is used when Access Manager is acting as the web service client. The default version is 1.1. The possible values are 1.0 or 1.1.

`com.sun.identity.liberty.ws.soap.certalias`
Value is set during installation. Client certificate alias that will be used in SSL connection for Liberty SOAP Binding.

`com.sun.identity.liberty.ws.soap.messageIDCacheCleanupInterval`
Default value is 60000. Specifies the number of milliseconds to elapse before cache cleanup events begin. Each message is stored in a cache with its `ownmessageID` to avoid duplicate messages. When a message's current time less the received time exceeds `thestaleTimeLimit` value, the message is removed from the cache.

`com.sun.identity.liberty.ws.soap.staleTimeLimit`
Default value is 300000. Determines if a message is stale and thus no longer trustworthy. If the message timestamp is earlier than the current timestamp by the specified number of milliseconds, the message the considered to be stale.

`com.sun.identity.liberty.ws.wsc.certalias`
Value is set during installation. Specifies default certificate alias for issuing web service security token for this web service client.

`com.sun.identity.liberty.ws.trustedca.certaliases`

Value is set during installation. Specifies certificate aliases for trusted CA. SAML or SAML BEARER token of incoming request. Message must be signed by a trusted CA in this list. The syntax is:

```
cert alias 1[:issuer 1]|cert alias 2[:issuer 2]|.....
```

Example: `myalias1:myissuer1|myalias2|myalias3:myissuer3`. The value issuer is used when the token doesn't have a KeyInfo inside the signature. The issuer of the token must be in this list, and the corresponding certificate alias will be used to verify the signature. If KeyInfo exists, the keystore must contain a certificate alias that matches the KeyInfo and the certificate alias must be in this list.

Initializing the `AMConfig.properties` Properties

[Remark 1–8 Reviewer: Please check these three sections and make sure they still work as documented.] When you configure the Client SDK (as documented in "Using `AMConfig.properties` with Client SDK" on page 12) you are minimally configuring it to communicate with a remote instance of Federated Access Manager. The properties listed in "Federated Access Manager Properties for `AMConfig.properties`" on page 14 can also be initialized. The following sections describe different ways in which these properties can be initialized.

- "Using the `AMConfig.properties` Properties File" on page 22
- "Using the Java API" on page 22
- "Setting Individual Properties" on page 23

Using the `AMConfig.properties` Properties File

You can set properties in the `AMConfig.properties` properties file created during installation. The properties are formatted as follows:

property_name=property_value

Note – The properties files must be in the CLASSPATH. If necessary, declare the Java Virtual Machine (JVM) option as follows:

`-Damconfig=`*properties_file_name*

Using the Java API

[Remark 1–9 Reviewer: Code sample still valid?] The Client SDK properties can be set programmatically using the class `com.iplanet.am.util.SystemProperties`. The following code sample illustrates how this can be accomplished.

EXAMPLE 1–1 Setting Client SDK Properties Programmatically

```
import com.iplanet.am.util.SystemProperties;
import java.util.Properties;
public static void main(String[] args) {
        // To initialize a set of properties
        Properties props = new Properties();
        props.setProperty(”com.iplanet.am.naming.url’,
                    ”http://sample.com/amserver/namingservice’);
        props.setProperty(”com.sun.identity.agents.app.username’, ”amAdmin’);
        props.setProperty(”com.iplanet.am.service.password’, ”11111111’);
        SystemProperties.initializeProperties(props) ;

        // To initialize a single property
        SystemProperties.initializeProperties(“com.iplanet.am.naming.url’,
                    ”http://sample.com/amserver/namingservice’);
    // Application specific code ...
}
```

Setting Individual Properties

You can set properties one at a time. For example, you can declare the following JVM option at run time to assign a value to a particular property:

`-D`*propertyName=propertyValue*

Setting Up a Client SDK Identity

[Remark 1–10 Reviewer: Changed this section. Please review carefully. How does the client send the username/PW that is stored in AMConfig?] Some Federated Access Manager components (such as SAML, user management, and policy) require an identity to be authenticated before the client application can read configuration data. The client can provide either a username and password that can be authenticated, or an implementation of the `com.sun.identity.security.AppSSOTokenProvider` interface. Either option will return a session token which the client can then use to access Federated Access Manager configuration data.

- “To Set Username and Password Properties” on page 23
- “To Set an SSO Token Provider” on page 24

To Set Username and Password Properties

The following properties in `AMConfig.properties` can be used to set the username and password. The authenticated username should have permission to read the Federated Access Manager configuration data.

- The property to provide the user name is `com.sun.identity.agents.app.username`.
- The property to provide the plain text password is `com.iplanet.am.service.password`.

Note – If a plain text password is a security concern, an encrypted password can be provided as the value of `com.iplanet.am.service.secret`. If an encrypted password is provided, the encryption key must also be provided as the value of `am.encryption.pwd`.

To Set an SSO Token Provider

[Remark 1–11 Reviewer: I don't see this property in AMConfig. Is it still there? Has this option changed? Shouldn't the implementation be used in the client app? Please explain.] Provide the implementation of the `com.sun.identity.security.AppSSOTokenProvider` interface as the value of the `com.sun.identity.security.AdminToken` property.

Client SDK Use Cases

Remark 1–12 Writer **Need to get these procedures**

This section contains the procedures for the following Client SDK use cases.

- Enabling the Client SDK to run against an SSL enabled instance of Federated Access Manager
- Enabling the Client SDK to run against an SSL enabled instance of Sun Directory Server
- how to enable Client SDK failover. Though this has been described on pager 15 via naming URL. It's good to list this as a separate section

SAE API

See SAE_README in home directory

Building Custom Web Applications

[Remark 1–13 Reviewer: It seems to me that this section should change. I need to speak with the appropriate engineer regard this.] [Remark 1–14 Writer: add details on what are needed in terms of jars , config files , properties files etc.] The Client SDK is contained in a small Java archive (JAR) named `famclientsdk.jar`. If using the Client SDK to write client applications, download (or retrieve from the `libraries/jars` directory of the Federated Access Manager ZIP) `famclientsdk.jar`, and include it in the class path for the application.

The Client SDK package contains `Makefile.clientsdk` that you can use to generate and build samples and web applications. The makefile defines targets to build configuration properties, samples and web applications.

- "Building Stand-Alone Applications" on page 25
- "Targets Defined in clientsdk" on page 25

Building Stand-Alone Applications

Remark 1–15 Writer **add details on what are needed in terms of jars , config files , properties files etc.**

Use this procedure for building identity-enabled web applications.

▼ To Build a Stand-Alone Application

1 **Install the Client SDK.**

See "Running the Client SDK Samples" on page 9.

2 **Copy** `servlet.jar` **to the** `lib` **directory.**

3 **Run the stand-alone application.**

Change directory to respective components within `clientsdk-samples`. Each has a `Readme.html` file explaining the changes and a Makefile to rebuild and run the program.

Targets Defined in clientsdk

For web deployment, `amclientwebapps.war` is ready to be deployed. However, you can make changes in the `clientsdk-webapps` directory and the WAR file can be recreated.

Custom web applications can use the following as a template to build their identity enabled web application.

properties: Generates `AMConfig.properties` in the temp directory that can used as a template for setting AM SDK's properties

samples: Copies standalone samples and corresponding Makefiles to samples directory.

webapp: Generates `amclientwebapps.war` that can be deployed on any Servlet 2.3 compliant web container.

Index