



**Developer Guide: SAP Mobile Server
Runtime**

SAP Mobile Platform 2.3

DOCUMENT ID: DC01934-01-0230-01

LAST REVISED: February 2013

Copyright © 2013 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Documentation Roadmap for SAP Mobile Platform	1
Security API	3
Introducing Security API	3
Quick Start Task Flow	3
Getting Started with CSI Provider Development	4
Developing a Custom Provider	5
Packaging a Custom Provider into a JAR File	6
Installing a Custom Provider JAR File on SAP Mobile Server	6
Validating a Custom Provider Using the HTTP Channel Debug Application	6
SAP Mobile Platform Security Framework	7
Authentication	7
Authorization	11
Attribution	14
Multiple Provider Interaction	16
Shared Data Structures for Providers	18
Capability Query	18
CSI Audit Generation and Configuration	20
Configuration Validation	22
Internationalization in CSI	23
CSI and Provider Localization	23
CSI Core Java Logging	24
SAP Mobile Platform Logging Configuration	24
Logging Localization	24
Error Handling for Providers	24
Error Localization	25
Reporting Errors and Warnings from Providers ...	25
Security API Reference	26
security package	26

Management API	327
Introducing Management API	327
Management API Features	327
Management API Javadoc	328
Documentation Roadmap for SAP Mobile Platform	328
Management API Changes in Version 2.3	328
SUPCluster API Changes	329
SUPConfiguration API Changes	330
SUPServerConfiguration API Changes	333
SUPRelayServer API Changes	336
SUPDomain API Changes	338
SUPMobileWorkflow API Changes	338
SUPMobileHybridApp API Changes	339
SUPApplication API Changes	340
SUPSecurityConfiguration API Changes	341
Management API	342
Contexts	342
Administration Interfaces	343
SUObjectFactory	345
Metadata	345
Exceptions and Error Codes	345
Best Practices	346
Getting Started with Client Development	346
Prerequisites	346
Required Files	347
Starting Required Services	347
Connecting to an SAP Mobile Server Instance ..	348
Developing Client Contexts, Objects, and Operations	349
Code Samples	350
Controlling SAP Mobile Server (SUPServer Interface)	350
Managing Clusters	352
Managing Relay Servers	386

Managing Domains	402
Managing Packages	421
Managing Mobile Business Objects	441
Managing Operations	444
Managing Applications and Application Connections and Templates	446
Monitoring SAP Mobile Platform Components .	492
Managing SAP Mobile Server Logs	516
Managing Domain Logs	522
Configuring SAP Mobile Platforms	530
Configuring Security Configurations	554
Managing Mobile Workflows	566
Managing Hybrid Apps	578
Management Client Application Shutdown	590
Client Metadata	590
Security Configuration	590
Cluster Configuration	646
Server Configuration	667
Server Log Configuration	671
Property Reference	673
Application Connection Properties	673
EIS Data Source Connection Properties Reference	679
Error Code Reference	702
Backward Compatibility	717
Notification API	721
Notification Mode	721
Notification URL	722
Notification Headers	722
Apple Header Fields	722
Google Header Fields	723
BlackBerry Header Fields	723
Generic Header Fields	723
Index	725

Contents

Documentation Roadmap for SAP Mobile Platform

SAP® Mobile Platform documents are available for administrative and mobile development user roles. Some administrative documents are also used in the development and test environment; some documents are used by all users.

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Product Documentation Web site regularly for updates: <http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories>, then navigate to the most current version.

Security API

Use the Security API to develop a custom authentication or authorization provider. The audience is advanced developers who are experienced in working with APIs, but who may be new to SAP® Mobile Platform.

This guide describes the Common Security Infrastructure (CSI) component, the security framework in SAP Mobile Platform, and how to develop a custom provider for extending the authentication and authorization functionality to use a back-end repository that is not supported by the default providers included in SAP Mobile Platform.

Introducing Security API

Use the Security API to develop a custom provider. The audience is advanced developers who are experienced in working with APIs, but who may be new to SAP Mobile Platform.

SAP Mobile Platform delegates the functions of storing and maintaining users and access control rules to the enterprise's existing security solutions. It uses a plug-in model to delegate the security checks to the configured providers using the CSI component.

CSI has a service provider plug-in model that integrates with the customer's existing security infrastructure. If none of the default providers shipped with SAP Mobile Platform meet the security needs, you can use the Security API to implement a custom login module, authorizer, or attributer that interfaces with a security back-end of your choice and plug it into SAP Mobile Platform.

Companion documentation for Security API includes:

- *Fundamentals*
- *Security*
- *SAP Control Center for SAP Mobile Platform*
- *Troubleshooting*

Quick Start Task Flow

This section provides a quick reference to information and task flows relevant to Security API.

1. *Getting Started with CSI Provider Development*

This section describes the task flow for developing a custom CSI provider.

2. *Developing a Custom Provider*

Develop your custom provider by writing a provider metadata file specifying valid configuration options.

3. *Packaging a Custom Provider into a JAR File*

After you have developed a custom provider, you must package the provider for installation.

4. *Installing a Custom Provider JAR File on SAP Mobile Server*

After you have developed your custom provider and packaged the JAR file, you must install the JAR file.

5. *Validating a Custom Provider Using the HTTP Channel Debug Application*

After installing your custom provider, validate your configuration using the HTTP Channel debug application.

Getting Started with CSI Provider Development

This section describes the task flow for developing a custom CSI provider.

Before you get started with CSI provider development, be aware of provider project dependencies including:

- `csi-core.jar` - This JAR encapsulates the CSI security framework packages, containing classes and interfaces that are required for developing CSI providers. More details on the packages contained in this JAR file can be found in *Developing a Custom Provider* and in the *CSI SDK API Reference*.

The file can be found at `SMP_HOME\Servers\UnwiredServer\ServerSDK\securityAPI\lib`.

- `providermetadata44.xsd` - This file specifies the XML schema that dictates the structure of a provider metadata file describing the provider configuration options, type, and default values. The provider metadata file is used to display the configuration options for the provider in SAP Control Center and should be present in every CSI provider JAR. For more details on the provider metadata file, see *Developing a Custom Provider* and *Packaging a Custom Provider into a JAR File*. Also refer to *Validating a Custom Provider Using the HTTP Channel Debug Application*.

This file can be accessed from one of the following locations:

- `SMP_HOME\Servers\UnwiredServer\ServerSDK\securityAPI\metadata\`
- From within the `csi-core-admin.jar` at `com\sybase\security\admin\` (the `csi-core-admin.jar` file can be found at `SMP_HOME\Servers\UnwiredServer\lib\ext`)

A CSI sample provider project, complete with working code and supporting documentation, can be found at: `SMP_HOME\Servers\UnwiredServer\ServerSDK\securityAPI\samples`. This sample project provides a hands-on approach to exploring the CSI SDK.

The `quickStartGuide.html` file in the CSI SDK describes the steps to set up the sample provider project. This file can be found at: `SMP_HOME\Servers\UnwiredServer\ServerSDK\securityAPI\`.

Developing a Custom Provider

Develop your custom provider by writing a provider metadata file specifying valid configuration options.

Packages in `csi-core.jar` library

Before starting development, be sure to understand the packages in the CSI core library. Packages include:

- The `com.sybase.security` package contains the main classes and interfaces that define the SAP CSI framework.
- The `com.sybase.security.authorization` package provides various types of security authorization request types with which a CSI consumer could construct more complex authorization requests.
- The `com.sybase.security.callback` package contains the callback and callback handler implementations used and supported by the default authentication providers.
- The `com.sybase.security.core` package contains the default provider implementations packaged as part of `csi-core.jar` file, as well as utility classes that are useful in implementing new providers.
- The `com.sybase.security.provider` package contains the interfaces implemented by various provider types and helper abstract classes. These helper classes provide an abstract implementation of the corresponding provider interfaces, complete with placeholders for all of the implemented interface's methods, so that subclasses need only to override relevant methods. The `AbstractLoginModule`, `AbstractAuthorizer`, and `AbstractAttributer` present in this package can be extended to overwrite the necessary methods to develop a custom login module, authorizer, and attributer respectively.

For more information, see *Security API Reference*.

Writing a provider metadata file

The CSI provider metadata XML file specifies the valid configuration options, including names, types, default values, as well as required or optional properties, for a particular CSI provider. The custom provider's details and configuration options in SAP Control Center is derived from the contents of this file. This metadata file should be named `sybcsi-provider.xml` and should conform to the latest `providermetadata.xsd` file included in the Security API.

A sample `sybcsi-provider.xml` file is available in the sample project located at `SMP_HOME\Servers\UnwiredServer\ServerSDK\securityAPI\samples\src\main\resources`.

Packaging a Custom Provider into a JAR File

After you have developed a custom provider, you must package the provider for installation.

The custom CSI provider classes are packaged into a JAR file that adheres to the standard JAR file specification at <http://docs.oracle.com/javase/6/docs/technotes/guides/jar/jar.html>.

In addition to the standard layout, include a provider metadata file named `sybcsi-provider.xml` at the root directory level of the provider JAR. The details about the custom provider and its configuration options displayed in SAP Control Center come from this file.

Installing a Custom Provider JAR File on SAP Mobile Server

After you have developed your custom provider and packaged the JAR file, you must install the JAR file.

You must make your new CSI provider available to SAP Mobile Server so it can be used in your security configuration. The JAR file needs to be installed onto the server instance by following the steps below:

1. Shutdown the SAP Mobile Server.
2. Copy the custom provider JAR file into the `SMP_HOME\Servers\UnwiredServer\lib\ext` folder.
3. Create a file named `sup.coff` in the `SMP_HOME\Servers\UnwiredServer\bin\private` directory, if it does not already exist.
4. Start the server using the batch command `start mode` in order to regenerate the class path.

To confirm that the installation succeeded, create a security configuration using SAP Control Center, and confirm the provider classes packaged in the installed JAR are listed as available providers. For more information, see *Security Configurations* in *SAP Control Center for SAP Mobile Platform*.

Validating a Custom Provider Using the HTTP Channel Debug Application

After installing your custom provider, validate your configuration using the HTTP Channel debug application.

To validate that a security configuration in SAP Mobile Server is setup correctly, use the HTTP Channel debug Web application pre-installed in the SAP Mobile Server.

For more information, see *Debug Security Configurations Using HTTP Channel* in the *Troubleshooting*.

SAP Mobile Platform Security Framework

SAP Mobile Platform delegates the functions of storing and maintaining users and access control rules to the enterprise's existing security solutions. It uses a plug-in model to delegate the security checks to the configured providers that use the CSI component.

CSI has a service provider plug-in model that integrates with the customer's existing security infrastructure. If none of the default providers shipped with SAP Mobile Platform meet your security needs, you can implement a custom login module, authorizer, and attributer that interfaces with the security back-end of choice, and plug it into SAP Mobile Platform as long as it implements the interfaces described in this section.

Administrative access to the SAP Mobile Platform server and the domains is controlled by the providers in the "admin" security configuration. Access to the packages deployed in the various SAP Mobile Platform domains is controlled by the security configuration associated with the packages. Different message-based synchronization (MBS) subscriptions and the mobile business object (MBO) operations in a package may require different roles to access them. The user authentication and role requirement at runtime is enforced by the security configuration associated with the package. The security configuration includes authentication, authorization, attribution, and audit providers.

A CSI SecContextFactory instance is associated with a single security configuration in SAP Mobile Platform and is cached to avoid the overhead in creating and initializing the factory for each authentication. A separate SecContext instance is created for each client authentication request, and is saved for the duration of the client connection. For example, a replication-based synchronization (RBS) session involves authenticating the user, the begin sync event, and the end sync event. For MBS, each message sent by the client results in a separate session in SAP Mobile Platform that triggers user authentication, and an authorization check to verify the user's authority to execute the operation and method invocation requested in the message.

Authentication

Authentication within CSI is performed internally using the Java Authentication and Authorization Services (JAAS) model by supplying the client credentials in an object instance that implements the `javax.security.auth.callback.CallbackHandler` interface.

SAP Mobile Platform caches a successful authentication result for a configurable time interval. If another authentication request using the same credentials is received within the authentication cache time interval after a successful authentication, the request is not delegated to CSI.

The authentication provider interfaces in CSI are primarily based on JAAS. The goal of the design is to allow any implementation of the JAAS pluggable authentication module system to alternatively plug into the CSI framework. Specifically, the

`javax.security.auth.spi.LoginModule` interface must be implemented by all authentication providers. CSI provides a flexible mechanism for defining active authentication providers and their control flags. All control flags defined in JAAS are fully implemented in the SAP Mobile Platform architecture. See the JAAS documentation for a complete discussion of the configuration options available with JAAS login modules.

CSI instantiates new authentication provider instances, upon demand, for each authentication request. This is distinctly different from authorization and attribution providers, for which instances are reused and must be thread-safe. Normally, the first method called on a security context after creation results in authentication. The framework creates the authentication providers using the default public, no-argument constructor. If a provider class does not have a constructor meeting these requirements, a `com.sybase.security.SecException` object is created and thrown with the appropriate detail information.

After initialization, the provider methods are called in the following order; however, not all methods are necessarily called:

1. `initialize()`
2. `login()`
3. `commit()`

At any time after the `login()` method is called on a provider, be prepared for the abort method to be called to abort the authentication attempt.

The `initialize()` method call includes four arguments. The callback handler is how an authentication provider should attempt to retrieve credentials from the client. Also included is a `javax.security.auth.Subject` object, into which an authentication provider typically adds custom principals and credentials. The next argument is a `java.util.Map` instance that provides a way for all configured providers to share information. There are no standard way for providers to communicate with each other, but many of the providers that ship with the JDK support certain well-defined communication capabilities. The last argument is also a `java.util.Map` instance, which represents all of the provider-specific configuration options. Included in this argument are any URLs, user names, passwords, and anything else a provider needs to know about its environment to provide authentication services.

Principals play a crucial role in the JAAS and Java security models. It is through principals that a direct JAAS client can discover an authenticated user. A typical authentication provider might add one or more principals to the subject's principal set (see the `Subject.getPrincipals()` method) after verifying the user's identity. Each principal instance must include at least one method, `getName()`, which is defined in the `java.security.Principal` interface. The return value of this method is a unique string that represents this user in the underlying security system.

For an example, the `LDAPLoginModule` adds these principal objects to the `Subject` after authentication

- `com.sybase.security.ldap.LDAPDNPrincipal`, which represents the user's LDAP DN
- `com.sybase.security.ldap.LDAPUsernamePrincipal`, which represents the name used to authenticate the user

Either of these objects can be used individually to determine exactly who the authenticated user is, which makes them good candidates for principals. Examples of other potentially good choices for principals include e-mail address, phone number, or user name. Poor choices for principals include first name, last name, or birthday.

Principals themselves are not useful without any context about where they came from and what format they are in. After authentication is complete and the user's principals are collected, CSI makes a pass through the set of principals and attempts to classify them according to some rudimentary rules. In particular, two categories are defined: name and ID. A principal that is the name of a user is thought to be a principal whose string representation presents itself in a relatively human-readable form. Something like the user name or e-mail address might be considered a name principal. A principal that is an ID principal is typically not very human-friendly—it might be a large number or a string of seemingly random characters that form a machine-readable unique identifier.

Both of these principals are useful in their own places, so CSI provides a way to categorize a principal as a name and/or ID, or neither. You can use the empty interfaces `com.sybase.security.providers.SecIDPrincipal` and `com.sybase.security.providers.SecNamePrincipal` to tag custom authentication providers' principals as name or ID, or both. If a principal instance implements the `SecIDPrincipal` interface, then the value returned from the `getName()` method is added to the security context as an ID attribute. Logically, principals that implement the `SecNamePrincipal` interface are added as NAME attributes in the security context. The actual principal instances themselves remain available to CSI clients and providers within the JAAS subject object, which you can retrieve using the `SecSubject.getJAASSubject()` method.

To facilitate role-based authentication, login modules may also retrieve the roles assigned to a user and add them as Principals to the authenticated subject. If these principals implement the interface `com.sybase.security.core.RoleCheck`, you can use the default authorizer `com.sybase.security.core.RoleCheckAuthorizer`, which is part of the SAP Mobile Platform configurations, to perform authorization checks without the need for an authorization provider for the specific back end.

Table 1. Control Flag Settings for Authentication Providers

Control Flag Value	Description
Required	The authentication provider is always called, and the user must always pass its authentication test. Regardless of whether authentication succeeds or fails, authentication proceeds down the list of providers.

Control Flag Value	Description
Requisite	The user is required to pass the authentication test of the authentication provider. If the user passes the authentication test of the authentication provider, subsequent providers are executed but can fail (except for authentication providers with the JAAS control flag set to required).
Sufficient	The user is not required to pass the authentication test of the authentication provider. If authentication succeeds, no subsequent authentication providers are executed. If authentication fails, authentication proceeds down the list of providers.
Optional	The user is allowed to pass or fail the authentication test of the authentication provider. However, if all authentication providers configured in a security realm have the JAAS control flag set to optional, the user must pass the authentication test of one of the configured providers.

Note: When an authentication provider is added to an existing security configuration, the control flag is set to required by default.

Credentials in Single Sign-on

Created as a direct result of authentication, credentials are usually time-limited objects that you can use to perform tasks such as authorization checks, attribution requests, and single sign-on (SSO).

In addition to principals, credentials help identify users to other remote services. For example, in SAP Mobile Platform where CSI is integrated as the primary security system, credentials perform single sign-on into other systems that are accessed to fulfill a client's request. CSI provides a wrapper on the credential concept to associate a name with a credential. This is to aid SAP Mobile Server in selecting the correct credential to propagate to the remote service when multiple providers (or multiple instances of a provider) add credentials of the same type (class) to the JAAS subject.

You can configure the endpoint connection definition with a property to identify the `NamedCredential` that should be propagated to the remote service to perform SSO. Credentials can also be used by the providers to communicate session information. An authentication provider can store the session information necessary for the attribution and authorization providers to perform their functions.

Add credentials using the `getPublicCredentials()` and `getPrivateCredentials()` methods from the `Subject` class. CSI does not treat public and private credentials differently. Providers should follow standard JAAS best practices when using credentials. When the security context is destroyed using the `destroy()` method, each of the public and private credentials that implements the `javax.security.auth.Destroyable` interface has its `destroy()` method called. This can be used to close any security doors left open by the credential's existence. For example, destroying the credentials may terminate a user's session that was kept open and

stored in the credential for use by the authorization or attribution provider, or it may have the eventual effect of triggering a logout audit record in the underlying security system.

The interface `com.sybase.security.SSOTokenCredential` should be implemented by credentials that contain tokens to be used for single sign-on into an SAP® system. It is used to identify the credential to be forwarded to an SAP endpoint (both JCo and DOEC endpoints).

The interface `com.sybase.security.CertificateCredential` should be implemented by credentials that contain the certificate to be used for single sign on into an SAP system. It is used to identify the credential to be forwarded to an SAP endpoint (both JCo and DOEC endpoints).

The interface `com.sybase.security.NamedCredential` associates a name with a credential object so the Web service endpoints can be configured with the name of the credential to forward to the Web service for single sign-on.

The interface `com.sybase.security.core.ExpiringCredential` allows a credential to be marked with an expiration time. This is used by SAP Mobile Platform server to expire the user session in the authentication cache based on the credential expiration time instead of the configured authentication cache timeout interval.

Authorization

The authorization provider interface is defined by the `com.sybase.security.provider.Authorizer` interface.

There are two primary worker methods:

- `checkRole()`
- `checkAccess()`

There are also two security context life-cycle methods, `initContext()` and `destroyContext()`, and one provider lifecycle method, `init()`, all of which are inherited from `com.sybase.security.provider.SecContextProvider`.

Provider Initialization

Authorization providers are normally initialized once per static class context. The provider instance itself is shared among any security context instances subsequently created.

The provider must expect to be used concurrently by multiple security clients. The security provider is initialized by the CSI infrastructure with a call to the `init()` method, which takes as an argument a `java.util.Map` of configuration data specific to the provider.

The provider should take this opportunity to validate connections to external resources, if possible, so that configuration errors are manifested as early as possible in the business flow. After initialization, the provider list is provided to security context instances as they are being created. During creation, a security context calls each provider's `initContext()` method, passing in an internal context `java.util.Map` (not to be confused with the security

context). This structure should be used to store any working information that the provider needs to maintain state. Because the provider instances are used concurrently by multiple clients, each provider method's first argument is this context object that is unique to the client's security context.

After context initialization has completed, the following context map data is passed into the security provider methods:

Table 2. Context Map Data

Constant	Expected Data Type	Description
ProviderConst.SEC_CONTEXT	SecContext	The client security context associated with the context map.
ProviderConst.CURRENT_SUBJECT	SecSubject	If authenticated, the subject associated with this security context. A provider can use this object to retrieve principals and credentials from the underlying JAAS subject.
ProviderConst.WARNING_MANAGER	WarningManager	Providers may use this object to add warnings for the current operation. Note: Deprecated, use <code>ProviderServices</code> .
ProviderConst.PROVIDER_SERVICES	ProviderServices	Any service exposed internally to all providers is available through this interface, for example <code>WarningManager</code> , access to profiles, certificate validation, and so on.

Authorization Checks

The authorization capabilities in SAP Mobile Platform center around two primary authorization methods: role checks and resource access checks.

Role checks are performed using the context's `checkRole()` method, passing the role ID as the parameter. A list of potential roles that may be used for access checks can be retrieved using the `listRoles()` method. Some provider sets have the ability to enumerate the roles available. The return value of this method does not necessarily comprise the complete list of roles, depending on the providers.

Resource access checks are more complicated than role checks because three factors are applied to resource access checks:

- the resource, represented by the `SecResource` interface
- the action, which is a named object
- the subject

Before attempting an access check, a reference to a resource object must be retrieved through a call to `getResource()`, possibly preceded by a call to `listResources()` to obtain a resource list. Next, the client can either immediately call `checkAccess()` supplying the resource reference and an action identifier, or the client can first use the `listActions()` method to retrieve a list of potential actions that can be expected to be performed on a given resource instance. As with all enumeration methods, providers are not required or guaranteed to implement these and in this case additional valid actions may exist for the given resource.

The calls to the variety of `SecContext.checkAccess()` and `SecContext.checkRole()` methods by the client result in calls to the corresponding methods in the authorization provider. The security context combines the results of the role and access check from all of the configured authorization providers when delegating these calls. Therefore, because one provider identifies a user as having a particular role, it does not mean that the `checkRole()` call succeeds.

Access Check

The access checks performed by the `checkAccess()` method are similar to role checks.

There are two differences regarding access checks:

- Instead of a role name, a `SecResource` instance and an action string are supplied. The `SecResource` object is retrieved with the help of the attribution provider, so it is not possible to perform access checks without an attribution provider installed.
- A `com.sybase.security.SecEnvironment` instance may be present, depending on which variant of the method was called.

This argument is additional information the client must pass to the back-end security provider. This could be information about where the security check occurs or extra rules to evaluate in addition to the standard security check. The contents of this structure are not defined by the CSI except to declare it as attributed. Clients and providers need to coordinate what attributes are supported and meaningful. It is important for authorization providers to know when to return NO and when to return ABSTAIN. The same decision flow should be used to determine the return value for the `checkAccess()` method.

Provider Implementation of Role Check

The `checkRole()` method accepts three arguments: the context map, the role name, and a `SecSubject` object.

The first argument is the context map. This is the same argument that was passed into the `initContext()` method, so any information saved in that method call is now available using this object instance.

The second argument provided to the `checkRole()` method is the role name. The specifics of how the implementation interprets this string argument are determined during

implementation. However, SAP strongly recommends that authorization and attribution providers coordinate their efforts. This ensures the results from an attribution provider's `listRoles()` method return named objects whose identifiers are supported as role name arguments in the authorization provider's `checkRole()` method. It is possible for the provider's `checkRole()` implementation to accept multiple role-name formats beyond those returned by the attribution provider.

The third argument provided to the `checkRole()` method is a `SecSubject` object. Note that this is not necessarily the subject object that is authenticated into the active security context, which can be retrieved from the first argument (context map). In CSI, it is possible for a security client to perform a role check on another security subject. Not all providers need support this capability – if the provider does not, then it should simply return `ProviderConst.ABSTAIN`.

There are three valid return values from an authorization provider authorization check (both role and access check), all of which are contained in the `com.sybase.security.ProviderConst` interface. `YES`, `NO`, and `ABSTAIN` are the three constants. In general, if a provider can verify the existence of a role and can verify that the specified `SecSubject` does not have the role, the provider should return `NO`. Otherwise, the provider should return `ABSTAIN`.

For example, the `RoleCheckAuthorizer` provider provides role checking capabilities when the login module adds principals and credentials that implement the `RoleCheck` interface. This provider checks if a principal or credential exists in the JAAS subject with the specified role name and that it implements the `RoleCheck` interface. If the provider finds one, it returns `YES`. Note that the absence of a principal or credential does not guarantee that the role is not granted to the authenticated user, because the user might be granted the role through an alternate provider in a mixed security model. In this case, the provider must return `ABSTAIN` or lose interoperability with other authorization providers.

Attribution

The attribution capabilities provide attribute information for attributed objects within the security system and provide lists of objects.

The attribution capabilities are classified as those that perform one or both of the following tasks:

- Provide complete attribute information for attributed objects, for example resources and subjects, within the security system
- Provide enumerated lists of objects, for example resources, resource types, actions, and roles, possibly filtering using qualifiers

Attributed objects are represented by the `com.sybase.provider.Attributed` interface, which is sub-classed by several core interfaces such as `SecSubject` and `SecResource`.

In general, attributed objects may have zero or more attributes assigned to them represented by string keys. Each of these keys may have zero or more string values associated with it. Example attributes for a subject may be first name and e-mail address. Each attribution provider defines their own list of attributes, keys, and values. Some standard attributes that all attribution providers should support, if relevant, are:

- `Const.ATT_NAME`
- `Const.ATT_DESCRIPTION`

These map cleanly to extra fields in named objects.

Not all provider sets include an attribution provider. For example `HttpAuthenticationLoginModule` and `NTProxyAuthLoginModule` do not have a corresponding attributer. In SAP Mobile Platform, the attributer is the key interface for making changes to the back-end security system. The attributer is typically used for listing resources and roles in the back-end. For example `LDAPAttributer` is used to list the roles defined in the LDAP server. Using the attributer, you can implement self-registration to create a user account in the back-end security configuration or to change a password.

The attribution provider interface is defined by the `com.sybase.security.provider.Attributer` interface. The attribution providers have the same life-cycle methods as the authorization providers. The same rules should be followed with respect to thread safety. All of the attribution provider methods include a context map which can be used to store state information associated with a security context.

Enumeration Methods

Enumeration of objects is a function of the attribution layer.

Methods that meet this qualification typically return a list of named objects and have a name in the format `listXYZ` where `XYZ` is an object type, such as `Roles` or `Actions`. The enumeration APIs are those which are prefixed by `list` and return a `java.util.List`.

Order is important and relevant within the enumeration methods. CSI calls the providers in the order in which they are configured. Values in lists are concatenated in the order in which providers are called. In some cases, duplicates are removed.

Attribution Methods

There are three attribution methods that an attribution provider should implement: `attributeAuthenticatedSubject()`, `attributeSubject()`, and `attributeResource()`.

After authentication, the method `attributeAuthenticatedSubject()` is called to further elaborate on a subject's attributes. Because authentication is done by combining JAAS `LoginModules` and the results of this authentication process are simply a set of principals and credentials, there is more required after authentication to provide additional information about a user. Immediately following authentication, the CSI architecture reviews the authenticated

subject's principals and initializes the ID and NAME attributes of the `SecSubject` object as described in *Authentication*.

Any attributes other than these must be provided by the `attributeAuthenticatedSubject()` method, which takes two arguments. The first argument is the context map. The second argument is the `SecSubject` object that needs to have attributes defined on it. The attribution provider should use information contained in this object to add more attributes to the subject object.

For example, an LDAP authentication module adds an `LDAPDNPrincipal` principal to the subject's principal set. A companion LDAP attribution provider may contain the `attributeAuthenticatedSubject()` implemented to look for the custom `LDAPDNPrincipal` object in the principal set. It then uses the DN stored in the principal to connect to the LDAP server and retrieve extended attributes about the user represented by the principal. The specifics of the attributes are undefined in the context of CSI – each attribution provider defines its own sets of attributes. The only subject attributes which are used in SAP Mobile Platform are ID and NAME attributes. Others are ignored.

While the most common way to retrieve a `SecSubject` instance is through an authenticated security context's `getSubject()` method, there is also a way to retrieve an unauthenticated subject using the context's `getSubject(String id)` method. It is through this method that the attribution provider's `attributeSubject()` method is called. This method is similar to the `attributeAuthenticatedSubject()` method, except the method is supplied with a single identifier in a third argument that is to be used as the subject identifier. If an attribution provider can recognize and attribute a subject with the given identifier, it should return "true" when complete. This indicates to CSI that the attribution provider recognized a user with the specified identity, and that it should pass an unauthenticated `SecSubject` instance back to the CSI client.

The third attribution method, `attributeResource()` is used in a manner similar to `attributeSubject()`. The security context's `getResource()` method delegates to each of the configured attribution providers, and if one or more signals that it recognized and attributed the provider, a `SecResource` implementation is returned to the client. One difference is that in addition to attributes, this method should populate the resource type lists for the `SecResource` object.

Multiple Provider Interaction

When multiple active providers are configured, CSI reconciles conflicting decisions across the providers.

Security configurations can include a variety of active providers, all interacting and voicing their own perspective on an incoming request. CSI reconciles the conflicting decisions.

Authentication is unique in that it is partly the responsibility of the individual providers and partly the responsibility of the CSI configuration. CSI uses the JAAS authentication model for authentication and allows each provider to be associated with a control flag (required,

requisite, sufficient, and optional). See <http://www.oracle.com/technetwork/java/javase/jaas/index.html> for an introduction to the JAAS model. When reviewing the JAAS model, consider references to `LoginContext` the same as `SecContext` because CSI provides its own entry point.

An additional feature the framework adds on top of the JAAS model is the ability to detect impersonation. This is meaningful when both a user name and other data (certificate or an opaque token) are provided in an authentication request. When the authentication succeeds, the authenticated subject's attributes contain the `Const.ATT_NAME` attribute with the value set to the set of names (derived from the `Principal` objects of type `com.sybase.security.provider.SecNamePrincipal`) aggregated from the configured providers.

If the configured providers only consider the extra data (for example, certificate or token) and add `Principal` objects based on that, the supplied user name may not match any of the values for the `Const.ATT_NAME` attribute of the authenticated subject. When it is critical for the supplied user name to match at least one of the `Principal` names, you can set the configuration property `checkImpersonation` to `true`. This results in authentication failure even when all the configured login modules succeed but none of the `Named` principal names match the supplied user name.

in CSI, the result of authorization operations is either `YES`, `NO`, or `ABSTAIN`. However, there may be multiple authorization providers all vying to respond to a given authorization request. Providers have three possible answers to the authorization question – `true`, `false`, or `abstain`. `Abstain` should be used when a provider is not capable of answering yes or no. CSI denies the authorization request if either of the following occurs:

- Any authorization provider returns `NO`
- No authorization provider returns `YES`

CSI permits the authorization request in any other situation. Another way of describing a successful authorization request is when at least one authorization provider returns `YES` and none return `NO`.

The result of attribution operations is also a combined exercise. For methods such as enumeration methods, the union of the corresponding values for each provider is created and returned to the client. For attributed objects, the union of the attributes that each provider states are created.

For example, Provider A knows that Resource A has a `NAME` attribute of "Resource A" and an `OWNER` attribute of "Bob". Provider B knows that resource A has a `NAME` attribute of "Resource A-B" and a `GROUP` attribute of "Sales". The net attribute names and values for Resource A would be `NAME = {"Resource A", "Resource A-B"}, OWNER="Bob", GROUP="Sales"`.

Shared Data Structures for Providers

Provider instances use a shared map to store information and share it with other providers.

All the provider instances are passed a shared map when context-specific methods are invoked. This map can be used by the provider instances to store context-specific information and share it with other providers that need to access it. If providers need to store and share data at the factory level so they can access the data across multiple contexts, they can do so by storing the data in the map indexed with the key

`ProviderConst.FACTORY_LEVEL_SHARED_STATE`.

When using the core provider `CertificateValidationLoginModule` or `CertificateAuthenticationLoginModule`, the validated certificate chain is stored in the shared map and available to all the authentication providers using a pre-defined key. Providers can set and retrieve the certificate chain using the key

`ProviderConst.CERTIFICATE_SHARED_KEY` to support certificate authentication principals.

In SAP Mobile Platform, all the client parameters (including personalization parameters as well as the HTTP headers and cookies from the client session) are populated in the shared map passed into the `initialize()` method. A map is stored in the shared state with the key `ProviderConst.HTTP_PROPERTIESCOOKIES_SHARED_KEY`. It contains the client parameter names as the keys.

Capability Query

CSI provides the capabilities API to allow a CSI client to interrogate the capabilities of the underlying providers within CSI.

A capability query result has a Boolean value, either true or false. Capabilities are dynamic and may change over time even within the same security context. Standard capabilities are defined in the `com.sybase.security.Const` class. Some of these are listed in the table below:

Table 3. CSI Capabilities APIs

Capability Name	Description
<code>com.sybase.security.capabilities.provider.SelfRegistration</code>	If the security context has the ability to perform self registration, this capability is available. Note: Not used in SAP Mobile Platform.
<code>com.sybase.security.capabilities.provider.X509Authentication</code>	If the security context is able to authenticate X.509 certificates, this capability is available.

Capability Name	Description
<code>com.sybase.security.capabilities.provider.PasswordChange</code>	Not used in SAP Mobile Platform.
<code>com.sybase.security.capabilities.provider.FineGrainAccessControl</code>	The provider implements the <code>checkAccess</code> methods, not just the <code>checkRole</code> methods.

The capability API supports provider-specific capabilities. This allows a provider to define its own set of capabilities that may be checked by SAP Mobile Platform. Providers can implement the `com.sybase.security.provider.SecProviderCapabilities` interface to respond to a capabilities query.

Capabilities API that may be implemented by providers is as follows:

```
package com.sybase.security.provider;

/**
 * Optional provider interface that allows providers to respond to
 * capability requests.
 */
public interface SecProviderCapabilities
{
    /**
     * Called when when building the capability set of a provider. The
     * result of this call will not be cached by the SecContext
     * implementation.
     * @param context the context map
     * @param capability the capability to check
     * @throws SecException if some sort of error occurs that should abort
     * the entire capability query.
     */
    boolean hasCapability(Map<String, Object> context, String
        capability) throws SecException;
}
```

For example, a method implementation in a provider implements `SecProviderCapabilities` interface and supports certificate authentication capability:

```
public boolean hasCapability(Map<String, Object> context, String
    capability) throws SecException
{
    if (capability.equals(CAPABILITY_X509_AUTHENTICATION))
    {
        return true;
    }

    return false;
}
```

CSI Audit Generation and Configuration

CSI provides audit trail generation and configuration.

An audit destination may write an audit record to a file. All audit destinations have the ability to use an audit formatter, although they may be configured to ignore this if they do their own formatting. Audit destinations may be configured to initialize a default formatter. Audit destinations must be thread safe, unlike other CSI providers. CSI guarantees that each audit destination is created only once per factory instance.

However, it may be desirable for an audit destination to be aware of other instances of itself instantiated in the same VM and ensure that these instances appropriately share access to common resources, such as audit files. Audit destinations may also add additional attributes to the audit record, such as a sequential record ID, to guard against log tampering or signature attributes that can be later verified to check for audit record tampering.

The primary sources for generating audit records are:

- CSI core classes perform a variety of operations that generate audit records. Examples of events that are audited include authentication and authorization decisions, self registration attempts, and configuration information such as active provider sets.
- Providers may audit additional information about the operations to which they contribute. An example of extra information might be an extended response from the authentication server that indicates why an operation failed in more detail.

Providers that need to issue audit records can take advantage of auditing APIs defined in the `com.sybase.security.providers.ProviderServices` interface. Those audit APIs require the provider pass in an audit token. This token identifies which provider is issuing the audit request. The provider can retrieve and save this token from the provider configuration using the constant `ProviderConst.AUDIT_TOKEN_CONFIG_PROPERTY` in the provider's `init()` method or a `LoginModule`'s `initialize` method.

An audit record includes five common parts:

- timestamp - the time at which the event occurred
- resource class - the scope used when processing the resource ID
- resource ID - the object on which an operation is being performed
- action - the operation that is being performed
- decision - the result of the security check

The decision has four potential values: `PERMIT`, `DENY`, `ABSTAIN` or `NOTAPPLICABLE`. The `NOTAPPLICABLE` decision is used for those cases where the audit is an event rather than a security decision.

In addition to the core of the audit record, there are additional attributes. These attributes vary with the type of audit record being generated but may include context ID, provider identifier, and failure reason.

The auditing providers consist of the following configurable components:

- The `com.sybase.security.provider.AuditDestination` interface defines APIs for the audit destination component, which decides where to write audit record. The framework includes a simple file-based audit destination component, `com.sybase.security.core.FileAuditDestination`, that uses an audit formatter to retrieve a string representation of the audit record, and writes it to a file.
- The `com.sybase.security.provider.AuditFilter` interface defines APIs for audit filtering components, which decide whether an audit record should be audited or not. The default filter component, `com.sybase.security.core.DefaultAuditFilter`, adopts a filter string to perform audit record filtering. The syntax for the filter consists of zero or more filter expressions, delimited by parenthesis (brackets denote optional values). For example, `(expr1)[(expr2)...]`. Each of these expressions has syntax like the following: `[key1=value [,key2=value...]]`. The allowed keys are: `ResourceClass`, `Action`, and `Decision`. For example, filter string `(ResourceClass=core.*,Decision=Deny)` enables auditing all of the CSI core resource classes involved in a deny decision.

The default formatter, `com.sybase.security.core.XmlAuditFormatter`, stores the audit record in one type of XML format. Each audit destination component is associated with one audit filter and one audit formatter. The framework supports an audit destination control flag scheme similar to Java's JAAS LoginModule control flags of required, requisite, sufficient, and optional.

Table 4. Control Flag Settings for AuditDestination

Control Flag Value	Description
Required	If the audit filter associated with the destination enables logging a particular event, the audit record is always logged to the audit destination, and the logging must succeed. Regardless of whether or not the audit record is successfully logged to the destination, the other configured audit providers are invoked.
Requisite	If the audit filter associated with the destination enables logging a particular event, the audit record is required to be successfully logged to the audit destination. If the audit record is successfully logged to the audit destination provider, subsequent providers are executed but can fail (except for providers with the JAAS control flag set to required).
Sufficient	If the audit filter associated with the destination enables logging a particular event, the audit record is not required to be successfully logged to the audit destination. However, if the audit record is successfully logged by the provider, no subsequent providers are executed. If audit record logging fails, the audit filter proceeds down the list of configured providers.

Control Flag Value	Description
Optional	If the audit filter associated with the destination enables logging a particular event, the logging of the audit record by the provider is allowed to succeed or fail. However, if all the audit destination providers are configured with control flag set to optional, the audit record must be successfully logged by at least one provider.

Note: The audit results are combined in the same way authentication results of login modules are combined to decide if the authentication succeeds or not overall. If the combined audit result fails, then the security operation being audited fails. For example, if an authorization check is audited and the audit filter is set up to log that event, the authorization check fails if the audit record cannot be successfully logged even if the authorization provider returns `VOTE_YES`.

Configuration Validation

The provider architecture allows each configured provider to have its own namespace of configuration options.

When a provider's `init()` method (or a login module's `initialize()` method) is called, a configuration map is provided. The keys and values in the map always appear as string types.

To detect any configuration errors before they are saved, CSI validates that the supplied internal configuration in the properties format adheres to the stored provider metadata, and the providers validate the supplied configuration by performing runtime checks where possible. The provider implementations should include a `sybasecsi-provider.xml` file that contains the metadata about the valid configuration options (including names, types, default values, and whether they are required or optional properties, and so on) to simplify the configuration administration and validation. The provider metadata should conform to the latest configuration XSD included in the SDK.

A provider can participate in configuration validation by implementing the optional interface `com.sybase.security.provider.SecConfigurationValidatingProvider` to perform the same validation checks on the specified configuration options that it performs at runtime when instantiated with the configuration using the `init()` method. It should return the validation errors as a list of `com.sybase.security.provider.ConfigurationProblem`.

A configuration problem report includes an error description, its severity, and the configuration property with which it is associated. In the case of a missing required property or an invalid property combination, the problem is associated with the provider itself.

Login modules can implement `com.sybase.security.provider.NamedCredentialProvider` as an

optional interface to aid in validating that the `NamedCredential` added by the provider does not conflict with the credentials added by other configured login modules.

Internationalization in CSI

Internationalization ensures that data integrity is maintained while passing through the CSI core.

The core CSI classes do not provide any character set conversions. All string type values are expected to be encoded in Java standard UCS-16. Any providers that receive or send character-based information in any other character set are responsible for character set conversions.

The following examples show how the API handles character sets:

XML Configuration

The XML configuration provider retrieves configuration data from the XML file. It uses standard Java APIs to read and process the XML data. Any non-ASCII data embedded in the XML file is properly handled and converted to strings using an active Java XML parser (JAXP).

LDAP Attributer

The LDAP attributer uses the JNDI LDAP provider to retrieve attribute data about specific subjects. Exact string references are returned from the JNDI LDAP provider, and the CSI core does not inspect the string.

Login Module

The login module requests any credentials from the supplied callback. The transmission of credentials to the login module is done through the JAAS callback mechanism, requiring the login module to request any credentials from the supplied callback.

CSI and Provider Localization

Message localization within the CSI core and the default providers is performed using Java resource bundles along with some internal-only helper classes.

Messages are localized to the system locale before being written to a file or shown to a user.

Note: No translations are maintained, except the English default.

CSI core does not provide localization services to the providers. Provider authors are encouraged to use the Java resource bundle technique to provide localization services, but may choose to use other methods.

CSI Core Java Logging

CSI core and its default providers use Java logging APIs.

SAP Mobile Platform using the CSI core enables you to configure and localize logging.

CSI does not provide any logging services to providers. It is the responsibility of each provider author to log messages using their framework of choice. Providers are encouraged to use Java logging and resource bundles, which ensure that custom provider logging can be configured from SAP Control Center for SAP Mobile Platform.

SAP Mobile Platform Logging Configuration

The SAP Mobile Platform logging configuration sets the logging level for the core CSI framework classes and the default providers shipped with SAP Mobile Platform when the **Security** component logging level is set in SAP Control Center for SAP Mobile Platform.

If you develop and deploy a custom provider with a different namespace, the log level for it can be set using the **Other** component in the SAP Control Center for SAP Mobile Platform log configuration screen. The **Other** component includes all third-party libraries deployed and used in SAP Mobile Platform, and results in a large number of log messages.

SAP recommends that a custom security provider be developed in the namespace `sup.custom.security.*`.

Logging Localization

All log messages generated at the INFO level and higher within the framework and the default providers can be localized.

Localization of messages within CSI core and the default providers is performed using Java resource bundles along with some internal-only helper classes. In contrast, DEBUG messages cannot be localized because they are intended to be interpreted by technical support and development personnel only. Implement a custom provider implementation using the same guideline.

For more information, see *Localization*.

Error Handling for Providers

Errors in the providers are typically reported by throwing exceptions.

Localized error messages are made available from the `java.lang.Throwable.getMessage()` method that all exception classes implement. However, the framework defines a few exception types and interfaces that are

intended to be used for reporting errors. The exception `com.sybase.security.SecException` can be thrown from most methods.

The framework aggregates results from multiple providers where each can throw their own errors for a given operation. Also, the providers can be stacked where only a subset of providers are required to fulfill a request (authentication or audit) controlled by the control flag. For example, if the first authentication provider fails with an exception, and the second one succeeds, the exception thrown by the first provider is not propagated to the client if the control flag dictates that the error is irrelevant in the authentication process. Control flag is a JAAS concept. For more detailed information, refer to the `javax.security.auth.login.Configuration` javadoc at <http://docs.oracle.com/javase/6/docs/api/javax/security/auth/login/Configuration.html>.

The framework allows all warnings to be tracked and retrieved so they can be propagated to SAP Mobile Platform clients. Warnings are represented by the `com.sybase.security.SecWarning` interface. Providers can introduce new warnings using special provider-side APIs. There are several predefined warning sub-interfaces for standard security messages such as `password is expiring in the future at this time/date`.

See the javadoc for more details. SAP Mobile Platform inspects the warnings after an authentication attempt and although not all warnings are returned to the client, SAP Mobile Platform does look for a few pre-defined warnings that are propagated to clients.

Error Localization

Exception messages generated by the security framework and the default providers can be localized.

All messages are propagated in the system locale only. The security context does not provide a way to specify an alternative location for message translation

CSI does not provide error or message localization services to the providers. Each provider author must choose a localization facility. Providers are encouraged to use Java resource bundles for error or message localization.

Reporting Errors and Warnings from Providers

Providers indicate errors and warnings in a variety of ways, including framework and stack-trace logging to the logging system. Provider authors can customize exception handling and reporting.

Nearly all provider methods include `SecException` to indicate a failure. At a minimum, the framework logs the exception's message and stack trace to the logging system. The framework may propagate the message to the client, depending on the provider type and the exception type. The framework may add the exceptions to the context warning list or log them for troubleshooting purposes.

Security API

In the provider implementation, use

```
com.sybase.security.provider.ProviderServices to add a warning to the  
SecContext. The ProviderServices can be accessed from the shared map passed into the  
initialize method in the provider  
(ProviderServices)sharedState.get(ProviderConst.PROVIDER_SERVICES).
```

Authentication providers have an alternative mechanism to propagate warnings. When a `LoginException` is thrown from the login method of an authentication provider, the framework automatically adds a warning to the `SecContext`. If the `LoginException` instance already implements the `SecWarning` interface, the exception itself is added as a warning. Otherwise, the exception is wrapped in a lightweight wrapper

```
(com.sybase.security.provider.SecLoginExceptionWarningImpl).
```

The authentication provider can use the

```
com.sybase.security.provider.SecLoginExceptionAuthentication  
FailureWarningImpl class to simultaneously signify login failure and supply a more  
specific failure reason mapped from  
com.sybase.security.core.AuthenticationFailureReasons.
```

For more details, see the javadoc for these classes, and refer to the sample provider implementation in the server SDK.

Security API Reference

Use the Quick Start Guide and the javadoc as a Security API reference.

Refer to the sample project in the *Quick Start Guide* located in the SAP Mobile Platform installation directory: `SMP_HOME\Servers\UnwiredServer\ServerSDK\securityAPI`

Refer to the Security API javadoc for applicable documentation for each available package.

security package

The `com.sybase.security` package contains the main classes and interfaces that define the Sybase Common Security Infrastructure (CSI) framework.

Members

All public members of the security package.

- **authorization package** – The authorization package provides various types of security authorization request types with which a CSI consumer could construct more complex authorization requests.
- **callback package** – The `com.sybase.security.callback` package contains the callback and callback handler implementations used/supported by the default authentication providers.

- **core package** – The com.sybase.security.core package contains the default provider implementations that are packaged as part of csi-core.jar file as well as some utility classes that are useful in implementing new providers.
- **provider package** – Custom providers allow for the customization and extension of security enforcement as needed, by implementing Provider-side interfaces to author custom providers.
- **Attributed interface** – All objects that have Attributes should implement this interface.
- **CertificateCredential class** – This class exposes the method to retrieve the certificate after a user has been authenticated so that other components can retrieve it for performing SSO.
- **Const interface** – This interface defines constants for use with CSI.
- **Decision class** – Represents decisions within the audit subsystems.
- **Named interface** – This interface defines a base for objects that have names and unique identifiers.
- **NamedCredential interface** – This interface associates a name with a credential object added to the JAAS subject.
- **Operation class** – Represents operations that can be performed on cryptographic objects.
- **SecAdminContext interface** – This interface provides access to administrative CSI services.
- **SecConfiguration interface** – The interface that describes the configuration interface for CSI.
- **SecConfiguration2 interface** – The interface that describes the configuration interface for CSI.
- **SecConfiguration3 interface** – The interface that describes the configuration interface for CSI.
- **SecContext interface** – This interface defines methods available for performing security checks, and factories for Resource, Subject, and Environment classes.
- **SecContextFactory class** – This class exposes factory methods that other Sybase products can use for programmatic security.
- **SecEnvironment interface** – This interface wraps classes that can convey arbitrary attributes of the execution environment to the PDP.
- **SecException class** – Various security exceptions that occur during the use of methods in this package will throw this exception.
- **SecProfile interface** – This interface defines methods available for obtaining information on profiles.
- **SecResource interface** – This interface defines classes that can represent a protected resource in a security system.
- **SecSubject interface** – This interface defines methods available for obtaining information on subjects.
- **SecWarning interface** – An interface to propagate error information from the framework or the provider to the client.

- **SSOTokenCredential interface** – This represents a credential that contains Single Sign On (SSO) data that can be retrieved from an authenticated Subject.

Remarks

CSI has a service provider plug-in model defined by the classes and interfaces in the `com.sybase.security` package that allows providers to be plugged into the framework to integrate with the existing security infrastructure. If none of the default providers meet the security needs, one can implement a new provider that interfaces with a security back-end of choice and plug it into CSI as long as it implements the relevant interfaces.

authorization package

The authorization package provides various types of security authorization request types with which a CSI consumer could construct more complex authorization requests.

Members

All public members of the authorization package.

- **AbstractAuthzChecker class** – The abstract base class that implements the setters/getters in the AuthorizationChecker interface.
- **AbstractAuthzRequest class** – Base class that implements the setters/getters in the AuthzRequest interface.
- **AuthorizationChecker interface** – Interface that should be implemented by all authorization checkers.
- **AuthorizationCheckerFactory class** – Factory to obtain an AuthorizationChecker that knows how to evaluate a given AuthzRequest type.
- **AuthzRequest interface** – The Marker interface to define authorization requests.
- **AuthzResponse interface** – The Marker interface to define the authorization response returned.
- **AuthzResponseImpl class** – Basic implementation of AuthzResponse interface.
- **CompositeAuthzRequest interface** – An authorization request interface that groups multiple requests together in some fashion.
- **FineGrainedAuthzChecker class** – Authorization checker that checks if the subject is allowed to perform specified action on the specified resource using the `SecContext.checkAccess()` method.
- **FineGrainedAuthzRequest class** – Fine grained authorization request to check if the subject is allowed to perform specified action on the specified resource.
- **LogicalAndAuthzChecker class** – Authorization checker that returns a response with PERMIT decision only if all of the authorization requests are successful.
- **LogicalAndAuthzRequest class** – Composite authorization request for authorization check to succeed only if all of the authorization requests are successful.
- **LogicalOrAuthzChecker class** – Authorization checker that returns a response with PERMIT decision if at least one of the authorization requests is successful.

- **LogicalOrAuthzRequest class** – Composite authorization request for authorization check to succeed if at least one of the authorization requests is successful.
- **RoleAuthzChecker class** – Authorization checker that performs role check using the `SecContext.checkRole()` method.
- **RoleAuthzRequest class** – An implementation class for Role based authorization requests.

Remarks

In cases where the access control policy is not straight forward role based access control, these request objects may be combined to pose a more complex authorization request, and use the `SecContext.checkAuthorization` method. One benefit of this is to avoid false negative audit records.

For example, if the security policy says "grant access if the user is in any one of this set of roles {A, B, C}", the Policy Enforcement Point (PEP), without the support for complex authorization requests, could have checked each role separately and granted access if any of the checks succeeded, but we might have ended up with DENY audit records for failed checks without any indication that all the role checks are tied to one single access check.

Using this package, a composite request can be constructed as demonstrated in the following sample code:

```
CompositeAuthzRequest request =
    RoleAuthzRequest.getCompositeAuthzRequest(
Arrays.asList(new String[]
    { "A", "B", "C" }), LogicalOrAuthzRequest.class);
request.setPackage("Application A");
AuthzResponse resp = context.checkAuthorization(request);
if (resp.getDecision() == Decision.PERMIT)
{
    // access granted
```

The audit records generated for a composite authorization check are all tied with the same request ID to indicate that they are all part of one composite authorization check.

At the moment there are 4 types of `AuthzRequest`: `RoleAuthzRequest` uses `checkRole` to check a role. `LogicalOrAuthzRequest` returns `PERMIT` if any of the requests contained within the composite request returns `PERMIT`. `LogicalAndAuthzRequest` returns `PERMIT` only if all of the requests contained within the composite return `PERMIT`. `FineGrainedAuthzRequest` uses `checkAccess` to perform a fine grained access check.

For each `AuthzRequest` there is a corresponding `AuthorizationChecker`. The `AuthorizationCheckerFactory` is used to correlate a `AuthzRequest` with its corresponding checker. At some future point we may allow extensions beyond these 4 request types, but for the moment this is closed.

AbstractAuthzChecker class

The abstract base class that implements the setters/getters in the AuthorizationChecker interface.

Syntax

```
public class AbstractAuthzChecker
```

Derived classes

- *com.sybase.security.authorization.FineGrainedAuthzChecker* on page 40
- *com.sybase.security.authorization.LogicalAndAuthzChecker* on page 42
- *com.sybase.security.authorization.LogicalOrAuthzChecker* on page 44
- *com.sybase.security.authorization.RoleAuthzChecker* on page 45

Remarks

Sub-classes should implement the performAuthorization() method.

getAuthorizationCheckerFactory() method

Method to retrieve an alternate AuthorizationCheckerFactory when the current AuthorizationChecker is required to process a nested AuthzRequest of a different type.

Syntax

```
AuthorizationCheckerFactory getAuthorizationCheckerFactory ()
```

Returns

The factory for producing Authorization Checkers

Usage

AuthzRequests can nest other AuthzRequests of different types internally. When an AuthorizationChecker finds a nested request of a different type from which it was coded to process, it uses the AuthorizationCheckerFactory to retrieve an alternate AuthorizationChecker it can chain to process the nested request.

inheritAttributes(AuthzRequest , AuthzRequest) method

Transfer the request attributes (SecSubject, Environment, Package) from a parent request to a child request.

Syntax

```
void inheritAttributes ( AuthzRequest parent , AuthzRequest child )
```

Parameters

- **parent** – The parent request
- **child** – The child request

performAuthorization(AuthzRequest , Map< String, Object >, Map< String, Object >)
method

This method implements the authorization logic behind a specific AuthzRequest type.

Syntax

```
abstract AuthzResponse performAuthorization ( AuthzRequest
request , Map< String, Object > context , Map< String, Object >
auditDetails ) throws SecException
```

Parameters

- **request** – the AuthzRequest being processed.
- **context** – the security context in which the request is being processed.
- **auditDetails** – map containing the information to be added into the audit record.

Returns

An AuthzResponse describing if the request was Denied or Permitted.

Exceptions

- **SecException class** – security exceptions that occurs when an error is encountered while performing the authorization check.

setAuthorizationCheckerFactory(AuthorizationCheckerFactory) method

Sets the AuthorizationCheckerFactory to map an AuthzRequest to the corresponding AuthorizationChecker implementation.

Syntax

```
void setAuthorizationCheckerFactory
( AuthorizationCheckerFactory acf)
```

Parameters

- **acf** – The AuthorizationCheckerFactory to be used to obtain an AuthorizationChecker.

_acf variable

The AuthorizationCheckerFactory used to get the AuthorizationChecker.

Syntax

```
AuthorizationCheckerFactory _acf
```

MESSAGES variable

Member used to retrieve appropriate messages from resource bundles.

Syntax

```
final Messages MESSAGES
```

AbstractAuthzRequest class

Base class that implements the setters/getters in the AuthzRequest interface.

Syntax

```
public class AbstractAuthzRequest
```

Derived classes

- *com.sybase.security.authorization.FineGrainedAuthzRequest* on page 40
- *com.sybase.security.authorization.LogicalAndAuthzRequest* on page 42
- *com.sybase.security.authorization.LogicalOrAuthzRequest* on page 44
- *com.sybase.security.authorization.RoleAuthzRequest* on page 46

getEnvironment() method

Retrieves the environment object set in the request.

Syntax

```
SecEnvironment getEnvironment ()
```

Returns

Returns the SecEnvironment object set in the request

getPackage() method

Retrieves the scope to be used for the authorization request.

Syntax

```
String getPackage ()
```

Returns

A string representing the package scope to be used for the authorization request.

getSubject() method

Retrieves the subject to be used for the authorization request.

Syntax

```
SecSubject getSubject ()
```

Returns

Returns the SecSubject to be used for the authorization request

setEnvironment(SecEnvironment) method

Sets the environment object to be used for evaluating the authorization request.

Syntax

```
void setEnvironment ( SecEnvironment environment )
```

Parameters

- **environment** – The SecEnvironment object to be used for evaluating the authorization

setPackage(String) method

Sets the scope to be used for the authorization request.

Syntax

```
void setPackage ( String pkg )
```

Parameters

- **pkg** – The package scope to be used for the authorization request.

setSubject(SecSubject) method

Sets the subject to be used for the authorization check.

Syntax

```
void setSubject ( SecSubject subject )
```

Parameters

- **subject** – The SecSubject to be used for the authorization check.

_environment variable

The Security environment of the PDP.

Syntax

```
SecEnvironment _environment
```

_pkg variable

The scope to be used for authorization check.

Syntax

```
String _pkg
```

_subject variable

The subject to be used for the authorization check.

Syntax

```
SecSubject _subject
```

AuthorizationChecker interface

Interface that should be implemented by all authorization checkers.

Syntax

```
public interface AuthorizationChecker
```

Derived classes

- *com.sybase.security.authorization.AbstractAuthzChecker* on page 30

getAuthorizationCheckerFactory() method

Method to retrieve an alternate AuthorizationCheckerFactory when the current AuthorizationChecker is required to process a nested AuthzRequest of a different type.

Syntax

```
AuthorizationCheckerFactory getAuthorizationCheckerFactory ()
```

Returns

The factory for producing Authorization Checkers

Usage

AuthzRequests can nest other AuthzRequests of different types internally. When an AuthorizationChecker finds a nested request of a different type from which it was coded to process, it uses the AuthorizationCheckerFactory to retrieve an alternate AuthorizationChecker it can chain to process the nested request.

performAuthorization(AuthzRequest, Map< String, Object >, Map< String, Object >)
method

This method implements the authorization logic behind a specific AuthzRequest type.

Syntax

```
AuthzResponse performAuthorization ( AuthzRequest request , Map<
String, Object > context , Map< String, Object > auditDetails ) throws
SecException
```

Parameters

- **request** – the AuthzRequest being processed.
- **context** – the security context in which the request is being processed.
- **auditDetails** – map containing the information to be added into the audit record.

Returns

An AuthzResponse describing if the request was Denied or Permitted.

Exceptions

- **SecException class** – security exceptions that occurs when an error is encountered while performing the authorization check.

setAuthorizationCheckerFactory(AuthorizationCheckerFactory) method

Sets the AuthorizationCheckerFactory to map an AuthzRequest to the corresponding AuthorizationChecker implementation.

Syntax

```
void setAuthorizationCheckerFactory
( AuthorizationCheckerFactory acf)
```

Parameters

- **acf** – The AuthorizationCheckerFactory to be used to obtain an AuthorizationChecker.

AuthorizationCheckerFactory class

Factory to obtain an AuthorizationChecker that knows how to evaluate a given AuthzRequest type.

Syntax

```
public class AuthorizationCheckerFactory
```

getChecker(AuthzRequest) method

Returns an AuthorizationChecker that knows how to evaluate a given AuthzRequest type.

Syntax

```
AuthorizationChecker getChecker ( AuthzRequest request ) throws  
SecException
```

Parameters

- **request** – authorization request type

Returns

An AuthorizationChecker that knows how to evaluate a given AuthzRequest type

Exceptions

- **SecException class** –

AuthzRequest interface

The Marker interface to define authorization requests.

Syntax

```
public interface AuthzRequest
```

Derived classes

- *com.sybase.security.authorization.AbstractAuthzRequest* on page 32
- *com.sybase.security.authorization.CompositeAuthzRequest* on page 39

getEnvironment() method

Retrieves the environment object set in the request.

Syntax

```
SecEnvironment getEnvironment ()
```

Returns

Returns the SecEnvironment object set in the request

getPackage() method

Retrieves the scope to be used for the authorization request.

Syntax

```
String getPackage ()
```

Returns

A string representing the package scope to be used for the authorization request.

getSubject() method

Retrieves the subject to be used for the authorization request.

Syntax

```
SecSubject getSubject ()
```

Returns

Returns the SecSubject to be used for the authorization request

setEnvironment(SecEnvironment) method

Sets the environment object to be used for evaluating the authorization request.

Syntax

```
void setEnvironment ( SecEnvironment environment )
```

Parameters

- **environment** – The SecEnvironment object to be used for evaluating the authorization

setPackage(String) method

Sets the scope to be used for the authorization request.

Syntax

```
void setPackage ( String pkg )
```

Parameters

- **pkg** – The package scope to be used for the authorization request.

setSubject(SecSubject) method

Sets the subject to be used for the authorization check.

Syntax

```
void setSubject ( SecSubject subject )
```

Parameters

- **subject** – The SecSubject to be used for the authorization check.

AuthzResponse interface

The Marker interface to define the authorization response returned.

Syntax

```
public interface AuthzResponse
```

Derived classes

- *com.sybase.security.authorization.AuthzResponseImpl* on page 38

getDecision() method

Returns the decision of the authorization check.

Syntax

```
Decision getDecision ()
```

Returns

the Decision object representing the authorization decision

setDecision(Decision) method

Sets the decision of the authorization check.

Syntax

```
void setDecision ( Decision decision )
```

Parameters

- **decision** – The Decision of the authorization check

AuthzResponseImpl class

Basic implementation of AuthzResponse interface.

Syntax

```
public class AuthzResponseImpl
```

getDecision() method

Returns the decision of the authorization check.

Syntax

```
Decision getDecision ()
```

Returns

The decision of the authorization check.

setDecision(Decision) method

Sets the decision of the authorization check.

Syntax

```
void setDecision ( Decision decision )
```

Parameters

- **decision** – The decision of the authorization check.

CompositeAuthzRequest interface

An authorization request interface that groups multiple requests together in some fashion.

Syntax

```
public interface CompositeAuthzRequest
```

Derived classes

- *com.sybase.security.authorization.LogicalAndAuthzRequest* on page 42
- *com.sybase.security.authorization.LogicalOrAuthzRequest* on page 44

getRequestList() method

Returns the list of authorization requests.

Syntax

```
List< AuthzRequest > getRequestList ()
```

Returns

the list of authorization requests.

setRequestList(List< AuthzRequest >) method

Sets the list of authorization requests.

Syntax

```
void setRequestList ( List< AuthzRequest > requestList )
```

Parameters

- **requestList** – list of authorization requests.

FineGrainedAuthzChecker class

Authorization checker that checks if the subject is allowed to perform specified action on the specified resource using the `SecContext.checkAccess()` method.

Syntax

```
public class FineGrainedAuthzChecker
```

```
performAuthorization( AuthzRequest , Map< String, Object >, Map< String, Object > )  
method
```

This method implements the authorization logic for a `FineGrainedAuthzRequest`.

Syntax

```
AuthzResponse performAuthorization ( AuthzRequest request , Map<  
String, Object > context , Map< String, Object > auditDetails ) throws  
SecException
```

Parameters

- **request** – the `AuthzRequest` being processed.
- **context** – the security context in which the request is being processed
- **auditDetails** – map containing the information to be added into the audit record.

Returns

an `AuthzResponse` describing if the request was Denied or Permitted. The decision is set to PERMIT if the current subject is allowed to perform the action on the resource specified in the request.

Exceptions

- **SecException class** –

FineGrainedAuthzRequest class

Fine grained authorization request to check if the subject is allowed to perform specified action on the specified resource.

Syntax

```
public class FineGrainedAuthzRequest
```

FineGrainedAuthzRequest(String, SecResource) constructor

Constructs a fine grained authorization request to check if the subject is allowed to perform specified action on the specified resource.

Syntax

```
FineGrainedAuthzRequest ( String action ,    SecResource resource )
```

getAction() method

Returns the action set in the request.

Syntax

```
String getAction ()
```

Returns

the action set in the request.

getResource() method

Returns the resource set in the request.

Syntax

```
SecResource getResource ()
```

Returns

the resource set in the request.

setAction(String) method

Sets the action to be used in the authorization request.

Syntax

```
void setAction ( String action )
```

Parameters

- **action** – Action to be used in the authorization request.

setResource(SecResource) method

Sets the resource to be used in the authorization request.

Syntax

```
void setResource ( SecResource resource )
```

Parameters

- **resource** – resource to be used in the authorization request.

LogicalAndAuthzChecker class

Authorization checker that returns a response with PERMIT decision only if all of the authorization requests are successful.

Syntax

```
public class LogicalAndAuthzChecker
```

performAuthorization(AuthzRequest, Map<String, Object>, Map<String, Object>)
method

This method implements the authorization logic for a LogicalAndAuthzRequest.

Syntax

```
AuthzResponse performAuthorization ( AuthzRequest request , Map<  
String, Object > context , Map< String, Object > auditDetails ) throws  
SecException
```

Parameters

- **request** – the AuthzRequest being processed.
- **context** – the security context in which the request is being processed
- **auditDetails** – map containing the information to be added into the audit record.

Returns

an AuthzResponse describing if the request was Denied or Permitted. The decision is set to PERMIT only if all the authorization requests in the list are permitted.

Exceptions

- **SecException class** –

LogicalAndAuthzRequest class

Composite authorization request for authorization check to succeed only if all of the authorization requests are successful.

Syntax

```
public class LogicalAndAuthzRequest
```


LogicalAndAuthzRequest() constructor

Constructs a composite LogicalAndAuthzRequest for authorization check to succeed only if all of the authorization requests are successful.

Syntax

```
LogicalAndAuthzRequest ()
```

LogicalAndAuthzRequest(List< AuthzRequest >) constructor

Constructs a composite LogicalAndAuthzRequest for authorization check to succeed only if all of the authorization requests are successful.

Syntax

```
LogicalAndAuthzRequest ( List< AuthzRequest > requestList )
```

Parameters

- **requestList** – list of authorization requests

getRequestList() method

Returns the list of authorization requests set in the composite request.

Syntax

```
List< AuthzRequest > getRequestList ()
```

Returns

list of authorization requests

setRequestList(List< AuthzRequest >) method

Sets the list of authorization requests in the composite request.

Syntax

```
void setRequestList ( List< AuthzRequest > requestList )
```

Parameters

- **requestList** – list of authorization requests

_requests variable

The list of authorization requests used in the LogicalAndAuthzRequest.

Syntax

```
List< AuthzRequest > _requests
```

LogicalOrAuthzChecker class

Authorization checker that returns a response with PERMIT decision if at least one of the authorization requests is successful.

Syntax

```
public class LogicalOrAuthzChecker
```

```
performAuthorization( AuthzRequest , Map< String, Object >, Map< String, Object >)  
method
```

This method implements the authorization logic for a LogicalOrAuthzRequest.

Syntax

```
AuthzResponse performAuthorization( AuthzRequest request , Map<  
String, Object > context , Map< String, Object > auditDetails ) throws  
SecException
```

Parameters

- **request** – the AuthzRequest being processed.
- **context** – the security context in which the request is being processed
- **auditDetails** – map containing the information to be added into the audit record.

Returns

an AuthzResponse describing if the request was Denied or Permitted. The decision is set to PERMIT if least one of the authorization requests is permitted.

Exceptions

- **SecException class** –

LogicalOrAuthzRequest class

Composite authorization request for authorization check to succeed if at least one of the authorization requests is successful.

Syntax

```
public class LogicalOrAuthzRequest
```

```
LogicalOrAuthzRequest() constructor
```

Constructs a composite LogicalOrAuthzRequest for authorization check to succeed if at least one of the authorization requests in the list is successful.

Syntax

```
LogicalOrAuthzRequest ()
```

LogicalOrAuthzRequest(List< AuthzRequest >) constructor

Constructs a composite LogicalOrAuthzRequest for authorization check to succeed if at least one of the authorization requests in the list is successful.

Syntax

```
LogicalOrAuthzRequest ( List< AuthzRequest > requestList )
```

Parameters

- **requestList** – list of authorization requests

getRequestList() method

Returns the list of authorization requests set in the composite request.

Syntax

```
List< AuthzRequest > getRequestList ( )
```

Returns

list of authorization requests

setRequestList(List< AuthzRequest >) method

Sets the list of authorization requests in the composite request.

Syntax

```
void setRequestList ( List< AuthzRequest > requestList )
```

Parameters

- **requestList** – list of authorization requests

_requests variable

The list of authorization requests used in the LogicalOrAuthzRequest.

Syntax

```
List< AuthzRequest > _requests
```

RoleAuthzChecker class

Authorization checker that performs role check using the SecContext.checkRole() method.

Syntax

```
public class RoleAuthzChecker
```

performAuthorization(AuthzRequest, Map<String, Object>, Map<String, Object>)
method

This method implements the authorization logic for a RoleAuthzRequest.

Syntax

AuthzResponse performAuthorization (AuthzRequest *request* , Map<String, Object > *context* , Map<String, Object > *auditDetails*) throws
SecException

Parameters

- **request** – the AuthzRequest being processed.
- **context** – the security context in which the request is being processed
- **auditDetails** – map containing the information to be added into the audit record.

Returns

an AuthzResponse describing if the request was Denied or Permitted. The decision is set to PERMIT if the checkRole() method on the context returns true for the role specified in the RoleAuthzRequest.

Exceptions

- **SecException class** –

RoleAuthzRequest class

An implementation class for Role based authorization requests.

Syntax

```
public class RoleAuthzRequest
```

RoleAuthzRequest(String) constructor

Authorization request that involves role based check.

Syntax

```
RoleAuthzRequest ( String role )
```

Parameters

- **role** – role to check for

getCompositeAuthzRequest(List< String >, Class<?extends CompositeAuthzRequest >) method

Constructs a composite authorization request containing a list of roles.

Syntax

```
CompositeAuthzRequest getCompositeAuthzRequest ( List< String
> roles , Class<?extends CompositeAuthzRequest > container )
```

Parameters

- **roles** – list of roles to be checked.
- **container** – CompositeAuthzRequest Class into which the list of RoleAuthzRequest wrapping the specified roles is set.

Returns

composite authorization request containing a list of RoleAuthzRequest.

Usage

This can be used to enforce complex authorization requirements (require a subject to be granted to at least one of the roles or all of the roles etc).

getRole() method

returns the role to check for.

Syntax

```
String getRole ()
```

Returns

the role to check for.

setRole(String) method

Sets the role to base the authorization check.

Syntax

```
void setRole ( String role )
```

Parameters

- **role** – role for the authorization check.

callback package

The com.sybase.security.callback package contains the callback and callback handler implementations used/supported by the default authentication providers.

Members

All public members of the callback package.

- **CertificateCallback class** – Callback implementation for use by LoginModule implementations that use Certificates for authentication.
- **CertificateCallbackHandler class** – A rudimentary CallbackHandler that supports passing a certificate to the LoginModule by handling CertificateCallback.
- **ServletRequestCallback class** – Callback implementation for use by JAAS LoginModule implementations that can use a HttpServletRequest for authentication.
- **ServletRequestCallbackHandler class** – A rudimentary CallbackHandler that supports passing an HttpServletRequest implementation to the LoginModule.
- **UsernamePasswordCallbackHandler class** – A simple implementation of callback handler that supports simple username/password authentications.

CertificateCallback class

Callback implementation for use by LoginModule implementations that use Certificates for authentication.

Syntax

```
public class CertificateCallback
```

getCertificateChain() method

Gets the certificate chain to be authenticated.

Syntax

```
Certificate[] getCertificateChain ()
```

Returns

Certificate chain to be authenticated

Usage

The chain, or path, begins with the certificate of the subject to be authenticated, and each certificate in the chain is signed by the entity identified by the next certificate in the chain.

setCertificateChain(Certificate[]) method

Sets the certificate chain to be authenticated in the callback.

Syntax

```
void setCertificateChain ( Certificate[] certChain )
```

Parameters

- **certChain** – certificate chain to be authenticated

CertificateCallbackHandler class

A rudimentary CallbackHandler that supports passing a certificate to the LoginModule by handling CertificateCallback.

Syntax

```
public class CertificateCallbackHandler
```

CertificateCallbackHandler(Certificate[]) constructor

Constructor that accepts the supplied certificate chain to return when handling the CertificateCallback.

Syntax

```
CertificateCallbackHandler ( Certificate[] certs )
```

Parameters

- **certs** – certificate chain to be returned when handling the CertificateCallback.

CertificateCallbackHandler(javax.net.ssl.SSLSocket) constructor

Constructor that extracts the certificate chain from the specified socket.

Syntax

```
CertificateCallbackHandler ( javax.net.ssl.SSLSocket socket )  
throws SSLException
```

Parameters

- **socket** – SSLSocket to extract the peer certificate chain

CertificateCallbackHandler(javax.servlet.http.HttpServletRequest) constructor
Certificates will be extracted from the servlet request object.

Syntax

```
CertificateCallbackHandler  
( javax.servlet.http.HttpServletRequest req )
```

Parameters

- **req** – servlet request object to extract the certificate chain from the javax.servlet.request.X509Certificate attribute

Usage

Constructor that extracts the certificate chain from the specified servlet request. The extracted certificate chain is used to handle the CertificateCallback.

handle(Callback[]) method

Retrieves the information requested in the specified Callbacks.

Syntax

```
void handle ( Callback[] cbarray ) throws IOException,  
UnsupportedCallbackException
```

Parameters

- **cbarray** – An array of Callback objects that contain the information requested to be retrieved.

Exceptions

- **IOException** – if an input or output error occurs.
- **UnsupportedCallbackException** – if UnsupportedCallbackException if any Callback other than CertificateCallback is supplied in the Callback array.

ServletRequestCallback class

Callback implementation for use by JAAS LoginModule implementations that can use a HttpServletRequest for authentication.

Syntax

```
public class ServletRequestCallback
```


ServletRequestCallback() constructor

Default constructor.

Syntax

```
ServletRequestCallback ()
```

Usage

The setRequest method should be invoked after an instance of the Callback is constructed.

getRequest() method

Gets the the value of the HttpServletRequest as set by the CallbackHandler.

Syntax

```
HttpServletRequest getRequest ()
```

Returns

the HttpServletRequest object or null if it was not set

setRequest(HttpServletRequest) method

The CallbackHandler can use this method to set the value of the HttpServletRequest.

Syntax

```
void setRequest ( HttpServletRequest request )
```

Parameters

- **request** – the HttpServletRequest value

ServletRequestCallbackHandler class

A rudimentary CallbackHandler that supports passing an HttpServletRequest implementation to the LoginModule.

Syntax

```
public class ServletRequestCallbackHandler
```

ServletRequestCallbackHandler(HttpServletRequest) constructor

Constructor to initialize the CallbackHandler with the specified HttpServletRequest.

Syntax

```
ServletRequestCallbackHandler ( HttpServletRequest request )
```

Parameters

- **request** – The HttpServletRequest value to supply to the LoginModule

handle(Callback[]) method

This callback implementation will pass the constructor-initialized HttpServletRequest to an ServletRequestCallback implementation supplied in the Callback array.

Syntax

```
void handle ( Callback[] callbacks ) throws  
UnsupportedCallbackException
```

Exceptions

- **UnsupportedCallbackException** – if any Callback other than ServletRequestCallback is supplied in the Callback array.

UsernamePasswordCallbackHandler class

A simple implementation of callback handler that supports simple username/password authentications.

Syntax

```
public class UsernamePasswordCallbackHandler
```

Remarks

For security reasons, this implementation will automatically clear the password by default after the first authentication attempt. This may be bypassed by modifying the allowMultipleAuthentications flag in the complete constructor.

UsernamePasswordCallbackHandler(String, char[]) constructor

Constructor that accepts only the username and password credentials.

Syntax

```
UsernamePasswordCallbackHandler ( String username , char[]  
password )
```

Parameters

- **username** – Username to use when authenticating
- **password** – Password to use when authenticating

Usage

Same as the other constructor, assumes the value "false" for allowMultipleAuthentications flag.

UsernamePasswordCallbackHandler(String, char[], boolean) constructor

Constructor that accepts the username and password credentials along with the boolean flag that dictates if multiple retrieval of the password are allowed.

Syntax

```
UsernamePasswordCallbackHandler ( String username , char[]  
password , boolean allowMultipleAuthentications )
```

Parameters

- **username** – Username to use when authenticating
- **password** – Password to use when authenticating
- **allowMultipleAuthentications** – Whether or not multiple authentication attempts are allowed. If this value is false then additional attempts to retrieve the password (after the first one) using the handle method will result in an `UnsupportedCallbackException`.

handle(Callback[]) method

This callback implementation will use the constructor-initialized username and password values to handle the `javax.security.auth.callback.NameCallback` and `javax.security.auth.callback.PasswordCallback`.

Syntax

```
void handle ( Callback[] callbacks ) throws IOException,  
UnsupportedCallbackException
```

Exceptions

- **IOException** – if an input or output error occurs.
- **UnsupportedCallbackException** – if any `Callback` other than `javax.security.auth.callback.NameCallback` and `javax.security.auth.callback.PasswordCallback` is supplied in the `Callback` array.

core package

The `com.sybase.security.core` package contains the default provider implementations that are packaged as part of `csi-core.jar` file as well as some utility classes that are useful in implementing new providers.

Members

All public members of the core package.

- **AlreadyExistsException class** – This exception is thrown when an attempt is made to create or rename an item where an item already exists by that name.

- **AuthenticationFailureWarning interface** – Defines the pre-defined set of authentication failure codes.
- **CertificateAuthenticationLoginModule class** – Certificate authentication provider authenticates the user by verifying that the password field contains the certificate as well as the certificate digest encrypted with the corresponding private key.
- **CertificateBlob class** – This class is an implementation of an ASN.1 SEQUENCE type.
- **CertificateIDPrincipal class** – Identity Principal that is derived from the information in the certificate used to authenticate a user.
- **CertificatePrincipal class** – Principal that associates the given name with the authenticated certificate chain.
- **CertificateTools class** – Utility functions to convert Java Certificate class instances to string form and back again, for use with attributes in CSI which can currently only be in string format.
- **CertificateValidationLoginModule class** – Authenticates the user based on the supplied certificate chain and adds the Certificate DN or the Certificate chain itself to the Principal set based on the configuration options.
- **ClientValuePropagatingLoginModule class** – Login module whose only purpose is to add the specified client http values from the map accessible with the key `ProviderConst.HTTP_PROPERTIESCOOKIES_SHARED_KEY` from shared context as `NamedCredentials`.
- **DefaultAuditFilter class** – The default audit filter implementation.
- **ExpiringCredential interface** – Interface to mark credentials that expire.
- **FileAuditDestination class** – A file-based audit destination.
- **HierarchicalItem< C, P, S > class** – Abstract class for maintaining role mapping information and scopes in sorted maps.
- **JCESecureDataServices class** – This class creates and initializes a `javax.crypto.Cipher` object for encrypt/decrypt operations based on the properties specified in the specified profile Map.
- **LogicalRole class** – `LogicalRole` maintains a list of physical role mappings to a single logical role within a `RolePackage`.
- **NamedConfiguration class** – Provides the named configuration facility for CSI.
- **NamedCredentialImpl class** – Simple implementation of `NamedCredential` interface.
- **NoSecAttributer class** – A simple implementation of `Attributer` interface.
- **NoSecAuthorizer class** – This authorizer ALWAYS grants access.
- **NoSecLoginModule class** – This authenticator implementation always authenticates the user.
- **PasswordExpirationWarning interface** – Warning to propagate the password expiration date.
- **PasswordUtils class** – Utility class that houses methods to compute encoded password hash and method to compare a specified password against such encoded password hash.

- **PhysicalRole class** – PhysicalRole represents a physical role assigned to a LogicalRole within a RolePackage.
- **PreConfiguredUserLoginModule class** – This login module only authenticates pre-configured users.
- **ProfilerImpl class** –
- **PropertiesConfiguration class** – Provides the standard property file-based configuration facility for CSI.
- **RoleCheck interface** – An interface that principals and credentials can implement to signal that they convey role membership.
- **RoleCheckAuthorizer class** – This authorizer provides only role-based authorization checks.
- **RoleMapperAdmin class** – Allows role mappings to be managed programmatically.
- **RoleMappings class** – Maintains a list of role package objects.
- **RolePackage class** – RolePackage maintains a list of logical role mappings within a package.
- **XmlAuditFormatter class** – An audit formatter that formats audit data into an XML record.
- **XmlConfiguration class** – Provides the standard XML-based configuration facility for CSI.
- **XMLFileRoleMapper class** – The RoleMapper provider maintains role mapping data in an XML File format.

AlreadyExistsException class

This exception is thrown when an attempt is made to create or rename an item where an item already exists by that name.

Syntax

```
public class AlreadyExistsException
```

AlreadyExistsException(String) constructor

Constructor that accepts a message to be included in the exception.

Syntax

```
AlreadyExistsException ( String message )
```

Parameters

- **message** –

AuthenticationFailureWarning interface

Defines the pre-defined set of authentication failure codes.

Syntax

```
public interface AuthenticationFailureWarning
```

Derived classes

- *com.sybase.security.provider.AuthenticationFailureWarningImpl* on page 198
- *com.sybase.security.provider.SecLoginExceptionAuthenticationFailureWarningImpl* on page 257

getFailureCode() method

Returns the failure code for the failed authentication attempt.

Syntax

```
int getFailureCode ()
```

Returns

Authentication failure code

FAILURE_CODE_ACCOUNT_DISABLED variable

Failure code to indicate authentication failed as the account is disabled.

Syntax

```
final int FAILURE_CODE_ACCOUNT_DISABLED
```

FAILURE_CODE_ACCOUNT_EXPIRED variable

Failure code to indicate authentication failed as the account has expired.

Syntax

```
final int FAILURE_CODE_ACCOUNT_EXPIRED
```

FAILURE_CODE_ACCOUNT_LOCKED variable

Failure code to indicate authentication failed as the account is locked.

Syntax

```
final int FAILURE_CODE_ACCOUNT_LOCKED
```

FAILURE_CODE_IMPERSONATION_ERROR variable

Failure code to indicate authentication failed because the specified username does not match the validated token/certificate etc.

Syntax

```
final int FAILURE_CODE_IMPERSONATION_ERROR
```

FAILURE_CODE_INVALID_CREDENTIALS variable

Failure code to indicate authentication failed because invalid credentials were supplied.

Syntax

```
final int FAILURE_CODE_INVALID_CREDENTIALS
```

FAILURE_CODE_PASSWORD_EXPIRED variable

Failure code to indicate authentication failed because the password has expired.

Syntax

```
final int FAILURE_CODE_PASSWORD_EXPIRED
```

Remarks

The expired password cannot be changed.

FAILURE_CODE_PASSWORD_EXPIRED_CAN_CHANGE variable

Failure code to indicate authentication failed because the password has expired and the expired password can be changed.

Syntax

```
final int FAILURE_CODE_PASSWORD_EXPIRED_CAN_CHANGE
```

FAILURE_CODE_TOKEN_VALIDATION_ERROR variable

Failure code to indicate authentication failed because the token validation failed.

Syntax

```
final int FAILURE_CODE_TOKEN_VALIDATION_ERROR
```

FAILURE_CODE_UNSPECIFIED variable

Failure code to indicate authentication failed due to unknown reason.

Syntax

```
final int FAILURE_CODE_UNSPECIFIED
```

CertificateAuthenticationLoginModule class

Certificate authentication provider authenticates the user by verifying that the password field contains the certificate as well as the certificate digest encrypted with the corresponding private key.

Syntax

```
public class CertificateAuthenticationLoginModule
```

Remarks

This provider authenticates the user by verifying that the password field contains the certificate as well as the certificate digest encrypted with the corresponding private key. Further, it compares the specified username with the subjectDN extracted from the certificate using the configured regular expression. After successfully validating the certificate, the provider stores the certificate in the shared context state indexed with the key `ProviderConst.CERTIFICATE_SHARED_KEY` so other providers in the security configuration can access it.

Upon successful authentication, the following a `CertificateIDPrincipal` with the specified username and a public credential `CertificateCredential` containing the certificate used for authentication with the configured credential name are added to the authenticated JAAS subject.

The provider expects the common name (from the certificate) as the username and the base64 encoded ASN.1 structure

```
CertBlob ::= SEQUENCE {
    x509Cert OCTET STRING,
    signature OCTET STRING,
    algorithm INTEGER
}
```

in the password field.

The digest of the certificate is expected to be encrypted using the private key. The only supported value for the algorithm identifier in the ASN.1 sequence is "1". It corresponds to the algorithms, "SHA1" for computing the digest and "RSA/ECB/PKCS1Padding" for decrypting the encrypted digest.

AlgorithmSet class

Syntax

```
package class AlgorithmSet
```


AlgorithmSet(String, String) constructor

Syntax

AlgorithmSet (String *digestAlg* , String *cipherAlg*)

cipherTransformation variable

Syntax

String cipherTransformation

messageDigestAlgorithm variable

Syntax

String messageDigestAlgorithm

getCredentialNames(Map< String, String >) method

Returns the set of names associated with the NamedCredentials that the provider may add to an authenticated subject.

Syntax

Set< String > getCredentialNames (Map< String, String > *configuration*)

Parameters

- **configuration** – provider configuration properties

Returns

the set of names associated with the NamedCredentials that the provider may add to an authenticated subject

Usage

This is used to detect conflicts among the configured providers when validating the configuration so as to avoid duplicate NamedCredentials added by multiple providers.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

String getProviderDescription ()

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

hasCapability(Map< String, Object >, String) method

Returns true for the capability query Const.CAPABILITY_X509_AUTHENTICATION.

Syntax

```
boolean hasCapability ( Map< String, Object > context , String  
capability ) throws SecException
```

Parameters

- **context** – the context map
- **capability** – the capability to check

Returns

true for the capability query Const.CAPABILITY_X509_AUTHENTICATION; false for other capability queries

Exceptions

- **SecException class** – if some sort of error occurs that should abort the entire capability query.

initialize(Subject, CallbackHandler, Map, Map) method

Syntax

```
void initialize ( Subject subject , CallbackHandler callback , Map  
sharedState , Map options )
```

login() method

Authenticates the user by verifying that the password field contains the certificate as well as the certificate digest encrypted with the corresponding private key.

Syntax

```
boolean login () throws LoginException
```

Returns

true if authentication is successful

Exceptions

- **LoginException** – if authentication fails

Usage

Further, it compares the specified username with the subjectDN extracted from the certificate using the configured regular expression.

CERTIFICATE_AUTH_REGEX_MATCH variable

Configuration property to specify the regular expression to use for matching the supplied username with the subjectDN in the certificate.

Syntax

```
final String CERTIFICATE_AUTH_REGEX_MATCH
```

Remarks

The string "{username}" in the regex will be replaced with the specified username before using it to match with the subjectDN from the certificate.

DEF_CERTIFICATE_AUTH_REGEX_MATCH variable

Default value for CERTIFICATE_AUTH_REGEX_MATCH that matches the specified username to the common name (CN) attribute in the certificate.

Syntax

```
final String DEF_CERTIFICATE_AUTH_REGEX_MATCH
```

Remarks

The common name may appear not just in the beginning of the distinguished name (DN) but anywhere.

MESSAGE_DIGEST_ALGORITHM_1 variable

Message digest algorithm associated with algorithm identifier "1".

Syntax

```
final String MESSAGE_DIGEST_ALGORITHM_1
```

CertificateBlob class

This class is an implementation of an ASN.1 SEQUENCE type.

Syntax

```
public class CertificateBlob
```

Remarks

The certificate blob consists of the following ASN.1 structure

```
CertBlob ::= SEQUENCE {  
  x509Cert OCTET STRING,  
  signature OCTET STRING,  
  algorithm INTEGER  
}
```

This class relies heavily on the Codec library to decode the encoded ASN.1 structure.

CertificateBlob() constructor

Constructor to create an empty blob which can then be used to decode the encoded ASN.1 sequence.

Syntax

```
CertificateBlob ()
```

CertificateBlob(byte[], byte[], int) constructor

Constructs the Certificate Blob with the specified certificate, the encrypted digest and the algorithm used to compute the encrypted digest so that it can be encoded.

Syntax

```
CertificateBlob (byte[] cert , byte[] digest , int alg)
```

Parameters

- **cert** – certificate
- **digest** – encrypted certificate digest
- **alg** – algorithm identifier used to compute the encrypted digest

decode(byte[]) method

Decodes the byte array (encoded ASN.1 structure) passed as argument.

Syntax

```
void decode ( byte[] encodedData ) throws ASN1Exception, IOException
```

Parameters

- **encodedData** – the encoded ASN.1 structure that should be decoded into the member variables.

Exceptions

- **ASN1Exception** – error decoding the ASN.1 structure
- **IOException** – if an input or output error occurs while dealing with the byte array streams

Usage

The decoded values are stored in the member variables of this class that represent the components of the corresponding ASN.1 type.

getAlgorithmID() method

Returns the algorithm identifier for the algorithm used to encrypt the certificate digest.

Syntax

```
int getAlgorithmID ()
```

Returns

algorithm identifier

getCertificate() method

Returns the stored/decoded certificate.

Syntax

```
byte[] getCertificate ()
```

Returns

certificate

getEncoded() method

Returns the encoded certificate blob i.e., ASN.1 structure.

Syntax

```
byte[] getEncoded () throws ASN1Exception, IOException
```

Returns

encoded certificate blob.

Exceptions

- **ASN1Exception** – if any error is encountered while encoding the provided certificate data into the ASN.1 structure.
- **IOException** – if an input or output error occurs while dealing with the byte array streams

getEncryptedDigest() method

Returns the encrypted certificate digest.

Syntax

```
byte[] getEncryptedDigest ()
```

Returns

encrypted certificate digest

CertificateIDPrincipal class

Identity Principal that is derived from the information in the certificate used to authenticate a user.

Syntax

```
public class CertificateIDPrincipal
```

CertificateIDPrincipal(String) constructor

Constructs the Principal that identifies the certificate used for authentication.

Syntax

```
CertificateIDPrincipal (String id)
```

Parameters

- **id** – certificate identifier

Usage

This id may be the entire certificate itself, but may also be the subjectDN retrieved from the certificate or the username matched with an attribute in the certificate.

CertificatePrincipal class

Principal that associates the given name with the authenticated certificate chain.

Syntax

```
public class CertificatePrincipal
```

Remarks

The name is derived from the certificate used to authenticate the user. This id may be the entire certificate itself, but may also be the subjectDN retrieved from the certificate or the username matched with an attribute in the certificate.

CertificatePrincipal(String, Certificate[]) constructor

Constructor that accepts the name to be set in the principal and certificate chain used to authenticate the user.

Syntax

```
CertificatePrincipal ( String name , Certificate[] certs )
```

Parameters

- **name** – Name to be associated with the certificate chain
- **certs** – certificate chain used to authenticate the user

equals(Object) method

Syntax

```
boolean equals ( Object o )
```

getCertificateChain() method

Retrieves the certificate chain used to authenticate the user.

Syntax

```
Certificate[] getCertificateChain ()
```

Returns

certificate chain used to authenticate the user

getName() method

Retrieves the name associated with the certificate chain.

Syntax

```
String getName ()
```

Returns

name set in the Principal.

hashCode() method

Syntax

```
int hashCode ()
```

_certs variable

Certificate chain used to authenticate the user.

Syntax

```
final Certificate[] _certs
```

_name variable

Name set in the principal.

Syntax

```
final String _name
```

CertificateTools class

Utility functions to convert Java Certificate class instances to string form and back again, for use with attributes in CSI which can currently only be in string format.

Syntax

```
public class CertificateTools
```

fromAttributeString(String) method

Standard method to be used to convert an attribute value to a certificate instance.

Syntax

```
Certificate fromAttributeString ( String attributeString )
```

Parameters

- **attributeString** – certificate encoded in special attribute format

Returns

certificate instance reconstructed from the string format

fromPEMString(String) method

Parse PEM format certificate string to retrieve certificate.

Syntax

Certificate fromPEMString (String *pemString*)

Parameters

- **pemString** – The PEM formatted string data

Returns

Certificate instance reconstructed from the PEM string format

getAttributeStringFromEncodedCert(byte[]) method

Given a binary certificate representation, return the encoded attribute string.

Syntax

String getAttributeStringFromEncodedCert (byte[] *cert*) throws
IOException

Parameters

- **cert** – the certificate data

Returns

The encoded attribute string

Exceptions

- **IOException** – if encoding error occurs

getEncodedCertFromAttributeString(String) method

Given a certificate attribute string, return the binary certificate information.

Syntax

byte[] getEncodedCertFromAttributeString (String *attributeString*)
throws IOException

Parameters

- **attributeString** – the certificate attribute string

Returns

binary certificate data

Exceptions

- **IOException** – if decoding error occurs

toAttributeString(Certificate) method

Standard method to be used to convert a certificate object to a string which can be supplied to an attributer.

Syntax

```
String toAttributeString ( Certificate cert )
```

Parameters

- **cert** – certificate instance to be converted into string format

Returns

certificate encoded in special attribute format

CertificateValidationLoginModule class

Authenticates the user based on the supplied certificate chain and adds the Certificate DN or the Certificate chain itself to the Principal set based on the configuration options.

Syntax

```
public class CertificateValidationLoginModule
```

Remarks

The authentication is successful if the provider is able to validate the certificate chain retrieved from the CertificateCallback successfully based on the configuration properties.

After successfully validating the certificate, the provider stores the certificate in the shared context state indexed with the key `ProviderConst.CERTIFICATE_SHARED_KEY` so other providers in the security configuration can access it. The subjectDN from the client certificate (or the certificate itself) is added to the authenticated JAAS subject as `CertificateIDPrincipal` if the configuration property `ProviderConst.VALIDATED_CERT_IS_IDENTITY` is set to true. If the configuration property value is false, a `CertificatePrincipal` with the subjectDN from the certificate and validated certificate chain is added to the authenticated JAAS subject.

getProviderDescription() method

Syntax

String getProviderDescription ()

getProviderVersion() method

Syntax

String getProviderVersion ()

hasCapability(Map< String, Object >, String) method

Returns true for the capability query Const.CAPABILITY_X509_AUTHENTICATION.

Syntax

boolean hasCapability (Map< String, Object > *context* , String *capability*) throws SecException

Parameters

- **context** – the context map
- **capability** – the capability to check

Returns

true for the capability query Const.CAPABILITY_X509_AUTHENTICATION; false for other capability queries

Exceptions

- **SecException class** – if some sort of error occurs that should abort the entire capability query.

initialize(Subject, CallbackHandler, Map< String,?>, Map< String,?>) method

Initializes the LoginModule.

Syntax

void initialize (Subject *subject* , CallbackHandler *callback* , Map< String,?> *sharedState* , Map< String,?> *options*)

Parameters

- **subject** – Subject to be authenticated
- **callback** – CallbackHandler to retrieve credentials

- **sharedState** – state shared with other configured providers
- **options** – configuration options for this provider

Usage

This method is invoked after the LoginModule has been instantiated. The purpose of this method is to initialize this LoginModule with the configuration properties. If this LoginModule does not understand any of the data stored in sharedState or options parameters, they can be ignored.

login() method

Validates the certificate chain retrieved from the callback against the configured CRLs.

Syntax

```
boolean login () throws LoginException
```

Returns

true if authentication is successful

Exceptions

- **LoginException** – if authentication fails

ClientValuePropagatingLoginModule class

Login module whose only purpose is to add the specified client http values from the map accessible with the key ProviderConst.HTTP_PROPERTIESCOOKIES_SHARED_KEY from shared context as NamedCredentials.

Syntax

```
public class ClientValuePropagatingLoginModule
```

Remarks

This provider always returns false from the login method and should be configured as "optional" so as to not affect the outcome of the authentication process.

CVPLMRolePrincipal class

Principal that implements RoleCheck interface to add a principal with the specified role name.

Syntax

```
public class CVPLMRolePrincipal
```

Remarks

This facilitates the delegation of authorization check to RoleCheckAuthorizer.

CVPLMRolePrincipal(String) constructor

Constructor that accepts the role name to be set as Principal name.

Syntax

```
CVPLMRolePrincipal ( String name )
```

Parameters

- **name** – role name that should be set as the Principal name.

*checkRole(String) method***Syntax**

```
boolean checkRole ( String role )
```

*equals(Object) method***Syntax**

```
boolean equals ( Object o )
```

*getName() method***Syntax**

```
String getName ( )
```

*hashCode() method***Syntax**

```
int hashCode ( )
```

CVPLMUserPrincipal class

SecNamePrincipal implementation to add a principal with the specified username.

Syntax

```
public class CVPLMUserPrincipal
```

CVPLMUserPrincipal(String) constructor

Constructor that accepts the username to be set as Principal name.

Syntax

```
CVPLMUserPrincipal ( String name )
```

Parameters

- **name** – username that should be set as the Principal name.

commit() method

Override commit method in the super class as it only adds the credentials if login succeeded in this provider.

Syntax

```
boolean commit () throws LoginException
```

Returns

true if this method succeeded, or false if this LoginModule should be ignored.

Usage

This provider needs to add the credentials even though login returns false.

getCredentialNames(Map< String, String >) method

Returns all the NamedCredentials the configuration specifies.

Syntax

```
Set< String > getCredentialNames ( Map< String, String >  
configuration )
```

Parameters

- **configuration** – configuration property map based on which to return the list of NamedCredentials added by this provider

Returns

list of NamedCredentials added by this provider

Usage

No configuration validation is performed to detect invalid/duplicate NamedCredentials. This check will be performed by SecAdminContext.validateExternalConfiguration method that aggregates the named credentials added by the configured providers.

initialize(Subject, CallbackHandler, Map, Map) method

Syntax

```
void initialize ( Subject subject , CallbackHandler callback , Map  
sharedState , Map options )
```

login() method

Always returns false.

Syntax

```
boolean login () throws LoginException
```

Returns

false always

Exceptions

- **LoginException** – if credential mapping is specified in invalid format or duplicate mappings are detected or if no client values are available in the session

Usage

This provider should always be configured with the control flag value of optional so that it does not affect the overall outcome of the authentication process. The method implementation simply adds the client values available in the session as credentials/Principals to the Subject based on the configuration properties.

validateConfiguration(Map< String, String >) method

Validate the supplied configuration and report any errors.

Syntax

```
List< ConfigurationProblem > validateConfiguration ( Map< String, String > configuration ) throws SecException
```

Parameters

- **configuration** – provider configuration to be validated

Returns

the list of configuration problems.

Exceptions

- **SecException class** – if there is an error validating the specified configuration. *The configuration problems should not be reported using the Exception mechanism.*

Usage

If no errors are discovered an empty set should be returned. If possible, the same runtime checks that are done in the init() method should be performed so that the configuration errors

can be fixed prior to deploying the configuration. A `SecException` should not be thrown to report the configuration problems, it should only be thrown in case of an error in validating the specified configuration.

PROP_HTTP_VALUES_AS_NAME_PRINCIPALS variable

Comma separated list of mappings that specify client http values that should be added as name principals after successful authentication.

Syntax

```
final String PROP_HTTP_VALUES_AS_NAME_PRINCIPALS
```

PROP_HTTP_VALUES_AS_ROLE_PRINCIPALS variable

Comma separated list of mappings that specify client http values that should be added as role principals after successful authentication.

Syntax

```
final String PROP_HTTP_VALUES_AS_ROLE_PRINCIPALS
```

PROP_HTTP_VALUES_TO_NAMED_CREDENTIAL_MAPPING variable

Comma separated list of mappings that specify the credential names for the client http values that should be added as credentials after successful authentication and the name to be associated with the credential.

Syntax

```
final String PROP_HTTP_VALUES_TO_NAMED_CREDENTIAL_MAPPING
```

Remarks

Ex: `clientName1:credentialName1`

DefaultAuditFilter class

The default audit filter implementation.

Syntax

```
public class DefaultAuditFilter
```

Remarks

This provides a simple wildcard syntax for selecting the resource classes, actions and decisions to be included in the audit logs.

init(Map< String, ?>) method

Implementations that care about initialization properties should override this method.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```


Parameters

- **configuration** – Provider specific configuration data.

Exceptions

- **SecException class** – if a required property is missing, or a provided property had bad syntax or couldn't be used. If a provider throws this exception it will be printed to the log and the provider will be inaccessible.

isAuditEnabled(String, String, Decision) method

This method will be called to query if an audit record with the specified attributes should be logged to the audit destination.

Syntax

```
boolean isAuditEnabled ( String resourceClass , String action ,
Decision decision ) throws SecException
```

Parameters

- **resourceClass** – the scope used when processing the resource ID.
- **action** – the operation that is being performed.
- **decision** – the result of the security check.

Returns

the decision on whether or not an audit record with the specified attributes should be logged to the audit destination.

Exceptions

- **SecException class** – generated when an error is encountered while performing the audit record query.

isFilteringCaseSensitive() method

Retrieves the case sensitivity of the filter.

Syntax

```
boolean isFilteringCaseSensitive ()
```

Returns

Whether or not the filtering is case sensitive which is based on the configuration property values.

validateConfiguration(Map< String, String >) method

Validate the supplied configuration and report any errors.

Syntax

```
List< ConfigurationProblem > validateConfiguration ( Map<  
String, String > configuration ) throws SecException
```

Parameters

- **configuration** – provider configuration to be validated

Returns

the list of configuration problems.

Exceptions

- **SecException class** – if there is an error validating the specified configuration. *The configuration problems should not be reported using the Exception mechanism.*

Usage

If no errors are discovered an empty set should be returned. If possible, the same runtime checks that are done in the init() method should be performed so that the configuration errors can be fixed prior to deploying the configuration. A SecException should not be thrown to report the configuration problems, it should only be thrown in case of an error in validating the specified configuration.

OPTION_CASE_SENSITIVE_FILTERING variable

The configuration option to specify whether or not to use case sensitivity when matching resource classes and actions.

Syntax

```
final String OPTION_CASE_SENSITIVE_FILTERING
```

Remarks

The default is not to be case sensitive.

OPTION_FILTER variable

The configuration option to specify the actual filter string to use.

Syntax

```
final String OPTION_FILTER
```

Remarks

The default value is specified by the

OPTION_FILTER_DEFAULT variable

The default value for the configuration option filter.

Syntax

```
final String OPTION_FILTER_DEFAULT
```

Remarks

This default enables role and resource authorization, profile access, authentication, logout and all provider and client audit events.

ExpiringCredential interface

Interface to mark credentials that expire.

Syntax

```
public interface ExpiringCredential
```

getExpirationTime() method

Returns the time when the credential expires.

Syntax

```
long getExpirationTime ()
```

Usage

The time is expressed as milliseconds since Jan 1, 1970

FileAuditDestination class

A file-based audit destination.

Syntax

```
public class FileAuditDestination
```

Remarks

This implementation supports concurrent access to the same file from multiple instances of this class. This can happen under normal circumstances if, for example, multiple factories are created with the same audit configuration. The implementation supports audit file rollover based upon size. It has the capability to compress rolled over items using GZIP and remove rolled over items beyond a certain count.

audit(String, String, String, Map< String, Object >, Decision) method

Generates and logs an audit record with the specified audit information.

Syntax

```
void audit ( String resourceClass , String resourceID , String action ,  
Map< String, Object > attributes , Decision decision ) throws  
SecException
```

Parameters

- **resourceClass** – the scope used when processing the resource ID.
- **resourceID** – the object on which an operation is being performed.
- **action** – the operation that is being performed.
- **attributes** – A map of attributes associated with the audit record.
- **decision** – the result of the security check.

Exceptions

- **SecException class** – If an error occurs while issuing the audit request.

init(Map< String,?>) method

Implementations that care about initialization properties should override this method.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```

Parameters

- **configuration** – Provider specific configuration data.

Exceptions

- **SecException class** – if a required property is missing, or a provided property had bad syntax or couldn't be used. If a provider throws this exception it will be printed to the log and the provider will be inaccessible.

rollover(File) method

Subclasses may override this method to modify rollover behavior.

Syntax

```
void rollover ( File log ) throws SecException
```

Parameters

- **log** – The base log file to roll over

Exceptions

- **SecException class** – if an error occurs in rolling the file over.

Usage

This method is used to perform all of the rolling over.

setFormatter(AuditFormatter) method

Sets the formatter to be used to format the audit information before writing it to the destination.

Syntax

```
void setFormatter ( AuditFormatter formatter )
```

Parameters

- **formatter** – Audit formatter to be used to format the audit information before writing it to the destination.

shouldRollover(File, long) method

Subclasses may override this method to modify rollover behavior.

Syntax

```
boolean shouldRollover ( File log , long size ) throws SecException
```

Parameters

- **log** – The base log file that may be rolled over
- **size** – The current size of the log file (this is a generally more accurate value than that returned from the filesystem).

Returns

Whether or not it is time to roll over the log file

Usage

This method is called on every audit record write so it should be fast. The default implementation delegates to the size rollover implementation.

validateConfiguration(Map< String, String >) method

Validate the supplied configuration and report any errors.

Syntax

```
List< ConfigurationProblem > validateConfiguration ( Map<  
String, String > configuration ) throws SecException
```

Parameters

- **configuration** – provider configuration to be validated

Returns

the list of configuration problems.

Exceptions

- **SecException class** – if there is an error validating the specified configuration. *The configuration problems should not be reported using the Exception mechanism.*

Usage

If no errors are discovered an empty set should be returned. If possible, the same runtime checks that are done in the init() method should be performed so that the configuration errors can be fixed prior to deploying the configuration. A SecException should not be thrown to report the configuration problems, it should only be thrown in case of an error in validating the specified configuration.

OPTION_AUDIT_FILE variable

This configuration option **MUST** be supplied to specify the file to which audit data will be written.

Syntax

```
final String OPTION_AUDIT_FILE
```

OPTION_COMPRESSION_THRESHOLD variable

This configuration option may be supplied to specify the number of uncompressed audit log rollover files that are created, before GZIP compression is used to archive the audit data.

Syntax

```
final String OPTION_COMPRESSION_THRESHOLD
```

Remarks

The default value is to never compress the data.

OPTION_DELETION_THRESHOLD variable

This configuration option may be supplied to specify the number of audit log files that will be preserved.

Syntax

```
final String OPTION_DELETION_THRESHOLD
```

Remarks

This value includes the main audit log, so a value of "3" will allow an audit.log, audit.log.0 and audit.log.1 before deleting old logs. The default value is to never delete old audit log data.

OPTION_ENCODING variable

This configuration option may be supplied to specify the encoding to use when writing the audit log.

Syntax

```
final String OPTION_ENCODING
```

Remarks

If this is not specified then UTF-8 will be used.

OPTION_ERROR_THRESHOLD variable

This configuration option may be supplied to specify the maximum number of audit log files that will be allowed -- when this threshold is reached then an error occurs and all auditing will fail.

Syntax

```
final String OPTION_ERROR_THRESHOLD
```

Remarks

For example, with this value set to "3", audit.log, audit.log.0 and audit.log.1 will be created according to the maximum log size value. If another audit log rollover is triggered then all audit operations will fail until one of the rollover files is removed.

This value is mutually exclusive with the deletion threshold -- the smallest value of the two will take effect.

OPTION_MAXIMUM_LOG_SIZE variable

This option may be supplied to specify the maximum audit log file size before a rollover occurs.

Syntax

```
final String OPTION_MAXIMUM_LOG_SIZE
```

Remarks

The default value never rolls over.

HierarchicalItem< C, P, S > class

Abstract class for maintaining role mapping information and scopes in sorted maps.

Syntax

```
public class HierarchicalItem< C, P, S >
```

addChild(String) method

Creates a new named child item.

Syntax

```
abstract C addChild ( String name ) throws AlreadyExistsException
```

Parameters

- **name** – Name of the child item to add

Returns

the newly created item.

Exceptions

- **AlreadyExistsException class** – it a child by that name already exists.

clear() method

Clears all children of the current item.

Syntax

```
void clear ()
```

deepClone() method

Return a deep clone of the object in question.

Syntax

```
Object deepClone ()
```

Returns

the deep clone

get(String) method

Returns a child of the current item with the given name or null if the child does not exist.

Syntax

```
C.get ( String name )
```

Parameters

- **name** – Name of the child to return

Returns

a child with the given name or null if the child does not exist.

getName() method

Returns the name of the item.

Syntax

```
String getName ()
```

Returns

the name of the item

*HierarchicalItem(String, HierarchicalItem< S,?, P >, boolean) method***Syntax**

```
HierarchicalItem ( String name , HierarchicalItem< S,?, P > parent ,  
boolean hasChildren )
```

iterator() method

Returns an iterator to iterate over the child objects.

Syntax

```
Iterator< C > iterator ()
```

Returns

an iterator to iterate over child objects. Objects will be returned in alphabetically increasing order based on their names.

remove() method

Deletes the current item.

Syntax

```
void remove ()
```

rename(String) method

Renames the item to the specified name.

Syntax

```
void rename ( String name ) throws AlreadyExistsException
```

Parameters

- **name** – new name for the item

Exceptions

- **AlreadyExistsException class** – if there is already an item by that name in the parent collection.

size() method

Returns the number of child items.

Syntax

```
int size ()
```

Returns

the number of children

_children variable

sorted child list

Syntax

```
TreeMap< String, C > _children
```

_name variable

Name of the item.

Syntax

```
String _name
```

_parent variable

Parent item.

Syntax

```
HierarchicalItem< S,?, P > _parent
```

*MESSAGES variable**Syntax*

```
final Messages MESSAGES
```

JCESecureDataServices class

This class creates and initializes a javax.crypto.Cipher object for encrypt/decrypt operations based on the properties specified in the specified profile Map.

Syntax

```
public class JCESecureDataServices
```

*JCESecureDataServices() constructor***Syntax**

```
JCESecureDataServices ()
```

getCipher(Map< String, Object >, SecProfile , String) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
Cipher getCipher ( Map< String, Object > context , SecProfile
profile , String operation )
```

*getMessageDigest(Map< String, Object >, SecProfile) method***Syntax**

```
MessageDigest getMessageDigest ( Map< String, Object > context ,
SecProfile profile )
```

Parameters

- **context** – CSI context
- **profile** – SecProfile object containing the properties to create and initialize a MessageDigest instance.

Returns

MessageDigest object created based on the properties specified in the profile Map and initialized for the digest operation. Returns null if unable to handle the profile.

Exceptions

- **SecException class** –

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

String getProviderDescription ()

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

String getProviderVersion ()

Returns

the version

getSignature(Map< String, Object >, SecProfile , String) method

Syntax

Signature getSignature (Map< String, Object > *context* ,
SecProfile *profile* , String *operation*)

Parameters

- **context** – CSI context
- **profile** – SecProfile object containing the properties to create and initialize a Signature instance.
- **operation** – specifies the operation for which to initialize the Signature object. Valid values are Const.OP_SIGN and Const.OP_VERIFY

Returns

Signature object created based on the properties specified in the profile Map and initialized for the specified operation. Returns null if unable to handle the profile.

Exceptions

- **SecException class** –

LogicalRole class

LogicalRole maintains a list of physical role mappings to a single logical role within a RolePackage.

Syntax

```
public class LogicalRole
```

LogicalRole(String, RolePackage) constructor

Constructor that accepts the logical role name and the role package to scope the mapping to.

Syntax

```
LogicalRole ( String name ,    RolePackage rp )
```

Parameters

- **name** – logical role name
- **rp** – role package to scope the mapping to

addChild(String) method

Adds a new physical role mapped to the current logical role for the specific package.

Syntax

```
PhysicalRole addChild ( String name ) throws  
AlreadyExistsException
```

Parameters

- **name** – role name to be mapped to the logical role

Returns

PhysicalRole that represents a physical role assigned to a LogicalRole within a RolePackage

NamedConfiguration class

Provides the named configuration facility for CSI.

Syntax

```
public class NamedConfiguration
```

Remarks

This configuration provider can be used to retrieve configuration data from two different sources in various combinations:

Security API

- A file on the filesystem
- A `java.util.Properties` object specified programmatically

The `Properties` object overrides those values loaded from the file.

System properties may be used to alter the default behavior when the class is instantiated, or the values may be modified programmatically. The system property `"com.sybase.security.core.NamedConfiguration.ConfigurationProvider"` determines the configuration provider that is delegated the task of retrieving the desired configuration file. The specified configuration provider must implement the interface `CompatibleConfiguration`. By default the `XmlConfiguration` provider is used.

System property values may be embedded into any provider option values included in the configuration file. Properties are referenced using the format `${system.property.name}`.

ConfigKey class

Class that represents a unique configuration possibility, based on the supplied selectors.

Syntax

```
protected class ConfigKey
```

ConfigKey(String, URL) constructor

Constructor that accepts configuration name and the base repository URL.

Syntax

```
ConfigKey ( String configName , URL baseURL )
```

Parameters

- **configName** – configuration name
- **baseURL** – repository URL

equals(Object) method

Syntax

```
boolean equals ( Object obj )
```

getBaseURL() method

Gets the repository URL.

Syntax

```
URL getBaseURL ( )
```

Returns

repository URL

getConfigName() method

Gets the configuration name.

Syntax

```
String getConfigName ()
```

Returns

configuration name

*hashCode() method***Syntax**

```
int hashCode ()
```

*toString() method***Syntax**

```
String toString ()
```

CompatibleConfiguration interface

Interface that providers should implement to make themselves compatible with Named Configuration.

Syntax

```
public interface CompatibleConfiguration
```

Derived classes

- *com.sybase.security.core.PropertiesConfiguration* on page 119
- *com.sybase.security.core.XmlConfiguration* on page 135

getSuggestedExtension() method

Gets the configuration file extension.

Syntax

```
String getSuggestedExtension ()
```

Returns

file extension to fully qualify the file name.

setURL(URL) method

Sets the configuration file URL.

Syntax

```
void setURL ( URL url )
```

Parameters

- **url** – URL to retrieve the configuration from.

NamedConfiguration() constructor

Instantiates NamedConfiguration with default values.

Syntax

```
NamedConfiguration ( )
```

NamedConfiguration(boolean) constructor

Subclasses should call this constructor with false as the doBootstrap argument.

Syntax

```
NamedConfiguration ( boolean doBootstrap )
```

Parameters

- **doBootstrap** – Flag to indicate if automatic bootstrapping should be performed by this method.

Usage

This will allow the subclasses constructor to call automaticBootstrap() to perform bootstrapping.

bootstrap(Properties) method

Saves bootstrap properties so they can be used to bootstrap the underlying configuration provider.

Syntax

```
void bootstrap ( Properties props ) throws SecException
```

Parameters

- **props** – properties to bootstrap the underlying configuration provider

Exceptions

- **SecException class** – to propagate any initialization errors

buildConfigurationInstance() method

Subclasses can override this to customize how the configuration provider instance is initialized.

Syntax

```
CompatibleConfiguration buildConfigurationInstance () throws
SecException
```

Returns

An instance of the underlying configuration provider to which the configuration reading/storing tasks are delegated.

Exceptions

- **SecException class** – to propagate any initialization errors

getConfigurationKeys(Map< String, Object >) method

Gets a list of configuration keys that will be searched in order, given the selectors.

Syntax

```
List< ConfigKey > getConfigurationKeys ( Map< String, Object >
selectors ) throws SecException
```

Parameters

- **selectors** – Map of key/value pairs that dynamically alter the configuration selection

Returns

list of ConfigKey that identifies the configuration in the primary and/or alternate repository specified by the selectors

Exceptions

- **SecException class** – if the repository URL identified by the selectors is invalid

getProperty(String) method

Gets the value of the specified property.

Syntax

```
Object getProperty ( String name )
```

Parameters

- **name** – Name of the property to retrieve the value for

Returns

value of the specified property

getProviderDescription() method

Gets the provider description.

Syntax

```
String getProviderDescription ()
```

Returns

Provider description string.

getProviderVersion() method

Gets the provider version.

Syntax

```
String getProviderVersion ()
```

Returns

Provider version string.

getRepository() method

Gets the configuration repository URL.

Syntax

```
URL getRepository ()
```

Returns

repository URL

initializeNewProvider(CompatibleConfiguration) method

Cycle through the configuration properties that have been set, setting them using reflection.

Syntax

```
void initializeNewProvider ( CompatibleConfiguration config )  
throws SecException
```

Parameters

- **config** – Configuration provider instance in which the configuration properties are to be set

Exceptions

- **SecException class** – to propagate any initialization errors

invalidateCache() method

Clears the configuration cache.

Syntax

```
void invalidateCache ()
```

setBootstrapProperty(String, String) method

Overrides bootstrapping to ignore properties destined for underlying provider.

Syntax

```
void setBootstrapProperty ( String name , String value )
```

Parameters

- **name** – bootstrap property name
- **value** – bootstrap property value

setConfigurationProviderName(String) method

Sets the underlying configuration provider to delegate the configuration retrieval.

Syntax

```
void setConfigurationProviderName ( String configProviderName ) throws  
SecException
```

Parameters

- **configProviderName** – Name of the configuration provider to delegate the configuration retrieval to

Exceptions

- **IllegalArgumentException** – if the configuration provider name is null
- **SecException class** – if the specified provider class cannot be instantiated or if the provider does not implement the interface `CompatibleConfiguration`.

setOverrideProperties(Properties) method

Sets the properties/values that will override the ones read from the configuration repository using the specified selectors.

Syntax

```
void setOverrideProperties ( Properties props )
```

Parameters

- **props** – override properties

setProperty(String, Object) method

Sets the specified property value in the provider configuration map.

Syntax

```
void setProperty ( String name , Object value )
```

Parameters

- **name** – Property name to be set
- **value** – Property value to be set

setRepository(File) method

Sets the configuration repository by specifying the directory containing the configuration files.

Syntax

```
void setRepository ( File repository ) throws SecException
```

Parameters

- **repository** – path to the directory containing the configuration files

Exceptions

- **IllegalArgumentException** – if the specified repository path is null
- **SecException class** – if the specified repository is invalid (i.e., not a directory or if it does not exist etc)

setRepositoryURL(URL) method

Sets the configuration repository URL.

Syntax

`void setRepositoryURL (URL repository) throws SecException`

Parameters

- **repository** – repository URL

Exceptions

- **IllegalArgumentException** – if the specified url is null
- **SecException class** – if the specified url is invalid.

writeNewConfiguration(Map< String, Object >, Map< String, String >) method

Stores the specified configuration properties as the Named configuration identified by the selectors.

Syntax

`void writeNewConfiguration (Map< String, Object > selectors , Map< String, String > configuration) throws SecException`

Parameters

- **selectors** – Map of key/value pairs that identify the configuration to be stored to.

Exceptions

- **SecException class** – if the repository identified by the selectors does not exist or if an error occurs trying to store the specified configuration

Usage

For valid selector definitions, please refer to SecConfiguration2 interface.

_bootstrapProps variable

Properties for use in bootstrapping the provider.

Syntax

Properties `_bootstrapProps`

_configCache variable

Configuration cache indexed by the ConfigKey.

Syntax

```
Map< ConfigKey, CompatibleConfiguration > _configCache
```

_configProvider variable

Underlying configuration provider that will be delegated the task to retrieve the desired configuration file.

Syntax

```
String _configProvider
```

_initProviderMap variable

Map containing the provider configuration properties.

Syntax

```
Map< String, Object > _initProviderMap
```

_repository variable

Syntax

```
URL _repository
```

CONFIGURATION_PROVIDER variable

The system property that should be used to select the configuration provider that will be delegated the task to retrieve the desired configuration file from the specified repository.

Syntax

```
final String CONFIGURATION_PROVIDER
```

Remarks

By default, the XmlConfiguration provider is used. The configured provider **MUST** implement the CompatibleConfiguration interface.

CONFIGURATION_PROVIDER_DEFAULT variable

Default configuration provider used to retrieve the desired configuration.

Syntax

```
final String CONFIGURATION_PROVIDER_DEFAULT
```

REPOSITORY_URL_SYSTEM_PROPERTY variable

The system property that should be set to specify the directory where the configuration files should be loaded from.

Syntax

```
final String REPOSITORY_URL_SYSTEM_PROPERTY
```

Remarks

This system property has no default value.

NamedCredentialImpl class

Simple implementation of NamedCredential interface.

Syntax

```
public class NamedCredentialImpl
```

NamedCredentialImpl(String, String) constructor

Constructor that accepts the credential name and value.

Syntax

```
NamedCredentialImpl ( String name , String value )
```

Parameters

- **name** – Name to be associated with the credential
- **value** – Value of the credential

getName() method

Returns Name associated with the credential.

Syntax

```
String getName ()
```

Returns

Name of the credential.

getValue() method

Returns the value stored in the credential.

Syntax

```
String getValue ()
```

Returns

value stored in the credential.

Usage

This should be in the format appropriate to relay to the backend system to perform SSO i.e., base64 encoded certificate, http header/cookie value in the format expected by the backend.

NoSecAttributer class

A simple implementation of Attributer interface.

Syntax

```
public class NoSecAttributer
```

Remarks

It simply returns true from the attribute methods but does not add any attributes to the SecSubject or SecResource.

attributeResource(Map< String, Object >, SecResource , String) method

Always returns true without adding any attributes to the SecResource.

Syntax

```
boolean attributeResource ( Map< String, Object > context ,  
SecResource resource , String id )
```

Parameters

- **resource** – resource to be attributed
- **context** – CSI context
- **id** – the ID of the resource to retrieve the attributes for

Returns

true always

attributeSubject(Map< String, Object >, SecSubject , String) method

Always returns true without adding any attributes to the SecSubject.

Syntax

```
boolean attributeSubject ( Map< String, Object > context ,  
SecSubject subject , String id )
```


Parameters

- **context** – CSI context
- **subject** – subject that should be attributed
- **id** – The ID to look up the subject when retrieving attributes.

Returns

true always

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

NoSecAuthorizer class

This authorizer ALWAYS grants access.

Syntax

```
public class NoSecAuthorizer
```

Remarks

This provider does not enforce any security at all. So, it is not recommended for use in production environment. Because many access checks are dependent on the combined results of all configured authorizers, including this provider in the security configuration will not guarantee that access checks will always be successful when combined with other providers.

checkAccess(Map< String, Object >, String, SecResource , SecSubject , SecEnvironment) method

This always votes YES for access checks.

Syntax

```
int checkAccess ( Map< String, Object > context , String action ,  
SecResource resource , SecSubject subject , SecEnvironment  
env )
```

Parameters

- **action** – the action to check
- **resource** – the resource on which to check the action
- **subject** – the subject to check access on
- **env** – the security environment
- **context** – the CSI context

Returns

Const.VOTE_YES, always

checkRole(Map< String, Object >, String, SecSubject) method

This always votes YES for role membership.

Syntax

```
int checkRole ( Map< String, Object > context , String roleName ,  
SecSubject subject )
```

Parameters

- **context** – the CSI context
- **roleName** – the role name to check
- **subject** – the subject to check

Returns

Const.VOTE_YES, always

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ( )
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

NoSecLoginModule class

This authenticator implementation always authenticates the user.

Syntax

```
public class NoSecLoginModule
```

Remarks

This provider does not enforce any security at all. So, it is not recommended for use in production environment.

It supports the two configuration options:

- `useUsernameAsIdentity` - If this option is set to true, the username supplied in the callback is set as the name of the principal added to the subject.
- `identity` - The value of this configuration option is used as the identity of the user if one of two conditions is met:
 1. No credentials were supplied
 2. The `useUsernameAsIdentity` option is set to false.
 The default value, if none is specified is "nosec_identity".

Principal class**Syntax**

```
public class Principal
```

Principal(String) constructor**Syntax**

```
Principal ( String name )
```

abort() method

Aborts the authentication process.

Syntax

```
boolean abort ()
```

Returns

true if this method succeeded, or false if this LoginModule should be ignored.

Exceptions

- **LoginException** – encountered while aborting the authentication process.

Usage

This method is called if the LoginContext's overall authentication failed. (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules did not succeed). If this LoginModule's own authentication attempt succeeded (checked by retrieving the private state saved by the login method), then this method cleans up any state that was originally saved.

commit() method

Method to commit the authentication process.

Syntax

```
boolean commit ()
```

Returns

true if this method succeeded, or false if this LoginModule should be ignored.

Exceptions

- **LoginException** – encountered while associating relevant Principals and Credentials with the Subject

Usage

This method is called if the LoginContext's overall authentication succeeded (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules succeeded). If this LoginModule's own authentication attempt succeeded (checked by retrieving the private state saved by the login method), then this method associates relevant Principals and Credentials with the Subject located in the LoginModule. If this LoginModule's own authentication attempted failed, then this method removes/destroys any state that was originally saved.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

initialize(Subject, CallbackHandler, Map< String,?>, Map< String,?>) method

Initializes the LoginModule.

Syntax

```
void initialize ( Subject subject , CallbackHandler callback , Map<
String,?> sharedState , Map< String,?> options )
```

Parameters

- **subject** – Subject to be authenticated
- **callback** – CallbackHandler to retrieve credentials
- **sharedState** – state shared with other configured providers
- **options** – configuration options for this provider

Usage

This method is invoked after the LoginModule has been instantiated. The purpose of this method is to initialize this LoginModule with the configuration properties. If this LoginModule does not understand any of the data stored in sharedState or options parameters, they can be ignored.

login() method

Always returns true.

Syntax

```
boolean login () throws LoginException
```

Returns

true always

Exceptions

- **LoginException** – if the callback handler does not support NameCallback and/or PasswordCallback

logout() method

Destroys any credentials relevant to this LoginModule.

Syntax

```
boolean logout ()
```

Returns

true if this method succeeded, or false if this LoginModule should be ignored.

Exceptions

- **LoginException** – encountered while logging out.

Usage

This typically means ending any session that was created as part of this LoginModule.

If the subject isn't readonly, this also remove any principals and credentials added as part of the earlier authentication.

DEFAULT_IDENTITY_OPTION variable

Default value for the IDENTITY_OPTION property.

Syntax

```
final String DEFAULT_IDENTITY_OPTION
```

DEFAULT_USE_USERNAME_AS_IDENTITY variable

Syntax

```
final boolean DEFAULT_USE_USERNAME_AS_IDENTITY
```

IDENTITY_OPTION variable

Configuration property to specify the identity to be set for the user if no credentials are supplied and if the useUsernameAsIdentity option is set to false.

Syntax

```
final String IDENTITY_OPTION
```

USE_USERNAME_AS_IDENTITY variable***Syntax***

```
final String USE_USERNAME_AS_IDENTITY
```

PasswordExpirationWarning interface

Warning to propagate the password expiration date.

Syntax

```
public interface PasswordExpirationWarning
```

Derived classes

- *com.sybase.security.provider.PasswordExpirationWarningImpl* on page 230

getExpiration() method

Returns the password expiration date.

Syntax

```
Date getExpiration ()
```

Returns

Password expiration date. Null if password does not expire.

Usage

Null is returned if password does not expire. If expiration date is not known, PasswordExpirationWarning is not returned.

PasswordUtils class

Utility class that houses methods to compute encoded password hash and method to compare a specified password against such encoded password hash.

Syntax

```
public class PasswordUtils
```

PasswordException class

Exception thrown by the various methods in PasswordUtils class to signal an error encoding/verifying the password.

Syntax

```
public class PasswordException
```

PasswordException(String) constructor

Constructor that accepts the message to be associated with the exception.

Syntax

```
PasswordException ( String message )
```

Parameters

- **message** – Message to be associated with the exception.

PasswordException(Throwable) constructor

Constructor that accepts the root cause to be associated with the exception.

Syntax

```
PasswordException ( Throwable e )
```

Parameters

- **e** – Root cause to be associated with the exception.

getRootException() method

Gets the root cause of the exception, if any.

Syntax

```
Throwable getRootException ()
```

Returns

root exception

checkPassword(char[], String) method

Checks if the password matches the encoded password hash.

Syntax

```
boolean checkPassword ( char[] password, String encodedPassword ) throws  
PasswordException
```


Parameters

- **password** – password to be checked
- **encodedPassword** – encoded password digest to compare against

Returns

true if they match, false otherwise

Exceptions

- **PasswordException** – if the encoded password hash is not in the expected format or if the digest algorithm ID is not valid/supported.

extractBase64Data(String) method

Decodes the base64 string and returns the byte array.

Syntax

```
byte[] extractBase64Data ( String data ) throws IOException
```

Parameters

- **data** – base64 string to be decoded

Returns

byte array decoded from the input

Exceptions

- **IOException** – if an error occurs during the decoding

extractEncodingAlgorithm(String) method

Extracts the algorithm used to encode the encoded password.

Syntax

```
String extractEncodingAlgorithm ( String encodedPassword )
```

Parameters

- **encodedPassword** – the encoded password from which to extract the algorithm

Returns

the algorithm in all upper case or null if unable to extract it properly

extractRawPassword(String) method

Extracts the password digest without the algorithm tag or the random salt.

Syntax

```
String extractRawPassword ( String encodedPassword )
```

Parameters

- **encodedPassword** – the encoded password from which to strip the algorithm tag and the random salt.

Returns

the password digest

extractSaltData(String) method

Extracts the salt data used to encode the encoded password.

Syntax

```
byte[] extractSaltData ( String encodedPassword ) throws IOException
```

Parameters

- **encodedPassword** – the encoded password from which to extract the salt data

Returns

the salt data

Exceptions

- **IOException** – propagates IOException encountered extracting the salt from base64 encoded form.

fatalJVMEError(Throwable) method

Prints the stack trace of the Throwable and throws an Error.

Syntax

```
void fatalJVMEError ( Throwable e )
```

Parameters

- **e** – Throwable to log

Exceptions

- **Error** –

Usage

This is invoked only in the case when an IOException is encountered while converting String into byte array using the DEFAULT_ENCODING.

generateSaltData() method

Generates random byte data of the specified length.

Syntax

```
byte[] generateSaltData ()
```

Returns

random byte data of the specified length

getBase64String(byte[]) method

Encodes the specified byte array into a Base64 string.

Syntax

```
String getBase64String ( byte[] data ) throws IOException
```

Parameters

- **data** – Byte array to compute the Base64 string from

Returns

Base64 encoding of the input

Exceptions

- **IOException** – if error is encountering during encoding.

getBytesFromArray(char[]) method

Converts a character array to a byte array using the DEFAULT_ENCODING.

Syntax

```
byte[] getBytesFromArray ( char[] ch )
```

Parameters

- **ch** – character array to convert

Returns

byte array derived using the default encoding from the specified input

getBytesFromArrayFrom(String) method

Converts a string to a byte array using the DEFAULT_ENCODING.

Syntax

```
final byte[] getBytesFromArrayFrom(String str)
```

Parameters

- **str** – String to convert into byte array

Returns

byte array

Usage

```
}
```

getMessageDigest(String) method

Instantiates a MessageDigest object with the specified algorithm.

Syntax

```
synchronized MessageDigest getMessageDigest (String algorithm )  
throws NoSuchAlgorithmException
```

Parameters

- **algorithm** – digest algorithm

Returns

MessageDigest instance that uses the specified algorithm.

Exceptions

- **NoSuchAlgorithmException** – if an invalid algorithm identifier is specified.

Usage

If a previous instance of MessageDigest using the same algorithm is found in the cache, it is cloned and returned. If not, a new instance is created and saved in a hash table for later usage. To be thread safe, always a clone of the created digest is returned.

getRawEncodedPassword(char[], byte[], String) method

Computes the raw base64 encoded password digest, without the algorithm tags.

Syntax

String *getRawEncodedPassword* (char[] *password* , byte[] *salt* , String *algorithm*) throws *NoSuchAlgorithmException* , *IOException*

Parameters

- **password** – clear text password to encode
- **salt** – salt to be used for encoding
- **algorithm** – algorithm to use for computing the password digest.

Exceptions

- **NoSuchAlgorithmException** – if the specified algorithm is not valid.
- **IOException** – if an error is encountered in obtaining the base64 string of the computed password digest.

CLEARTEXT_HASH_ALGORITHM variable

Algorithm ID to store the password in clear text (useful for debugging purposes).

Syntax

```
final String CLEARTEXT_HASH_ALGORITHM
```

DEFAULT_ENCODING variable

Default encoding used to convert password string into a byte array.

Syntax

```
final String DEFAULT_ENCODING
```

DEFAULT_HASH_ALGORITHM variable

Default algorithm used to compute password hash.

Syntax

```
final String DEFAULT_HASH_ALGORITHM
```

SALT_LENGTH variable

The salt length.

Syntax

```
final static int SALT_LENGTH
```

PhysicalRole class

PhysicalRole represents a physical role assigned to a LogicalRole within a RolePackage.

Syntax

```
public class PhysicalRole
```

PhysicalRole(String, LogicalRole) constructor

Constructor that accepts the name of the physical role and the LogicalRole it is mapped to.

Syntax

```
PhysicalRole ( String name , LogicalRole lr )
```

Parameters

- **name** – physical role name
- **lr** – LogicalRole mapped to the specified physical role name

addChild(String) method

Unsupported Operation.

Syntax

```
PhysicalRole addChild ( String name ) throws  
AlreadyExistsException
```

Parameters

- **name** – Name of the child item to add

Returns

the newly created item.

Exceptions

- **AlreadyExistsException class** – it a child by that name already exists.
- **UnsupportedOperationException** – always.

PreConfiguredUserLoginModule class

This login module only authenticates pre-configured users.

Syntax

```
public class PreConfiguredUserLoginModule
```

Remarks

It supports three configuration options:

- username - valid user name.
- password - encoded password for the user. The expected format is "{HASH_ALG_ID[:salt]}base64 encoded password digest calculated using the specified algorithm and the salt" [] represents optional data
- roles - comma separated list of roles that are granted to the user.

The user name supplied in the callback is set as the name of the principal added to the subject upon successful authentication. The specified roles are also added as a Principal to the subject.

PreConfigUserPrincipal class

SecNamePrincipal implementation to add a principal with the specified username.

Syntax

```
public class PreConfigUserPrincipal
```

PreConfigUserPrincipal(String) constructor

Constructor that accepts the username to be set as Principal name.

Syntax

```
PreConfigUserPrincipal ( String name )
```

Parameters

- **name** – username that should be set as the Principal name.

PreConfigUserRolePrincipal class

Principal that implements RoleCheck interface to add a principal that the name of the role granted to a user.

Syntax

```
public class PreConfigUserRolePrincipal
```

Remarks

This facilitates the delegation of authorization check to RoleCheckAuthorizer.

PreConfigUserRolePrincipal(String) constructor

Constructor that accepts the role name to be set as Principal name.

Syntax

```
PreConfigUserRolePrincipal ( String name )
```

Parameters

- **name** – role name that should be set as the Principal name.

checkRole(String) method

Syntax

```
boolean checkRole ( String role )
```

equals(Object) method

Syntax

```
boolean equals ( Object o )
```

getName() method

Syntax

```
String getName ( )
```

hashCode() method

Syntax

```
int hashCode ( )
```

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ( )
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ( )
```

Returns

the version

initialize(Subject, CallbackHandler, Map< String,?>, Map< String,?>) method
Initializes the LoginModule.

Syntax

```
void initialize ( Subject subject , CallbackHandler callback , Map<
String,?> sharedState , Map< String,?> options )
```

Parameters

- **subject** – Subject to be authenticated
- **callback** – CallbackHandler to retrieve credentials
- **sharedState** – state shared with other configured providers
- **options** – configuration options for this provider

Usage

This method is invoked after the LoginModule has been instantiated. The purpose of this method is to initialize this LoginModule with the configuration properties. If this LoginModule does not understand any of the data stored in sharedState or options parameters, they can be ignored.

login() method

Syntax

```
boolean login () throws LoginException
```

validateConfiguration(Map< String, String >) method
Validate the supplied configuration and report any errors.

Syntax

```
List< ConfigurationProblem > validateConfiguration ( Map<
String, String > configuration ) throws SecException
```

Parameters

- **configuration** – provider configuration to be validated

Returns

the list of configuration problems.

Exceptions

- **SecException class** – if there is an error validating the specified configuration. *The configuration problems should not be reported using the Exception mechanism.*

Usage

If no errors are discovered an empty set should be returned. If possible, the same runtime checks that are done in the `init()` method should be performed so that the configuration errors can be fixed prior to deploying the configuration. A `SecException` should not be thrown to report the configuration problems, it should only be thrown in case of an error in validating the specified configuration.

PRECONFIG_PASSWORD_OPTION variable

Configuration property to specify the user's password.

Syntax

```
final String PRECONFIG_PASSWORD_OPTION
```

PRECONFIG_ROLES_OPTION variable

Configuration property to specify the comma separated names of the roles granted to the user.

Syntax

```
final String PRECONFIG_ROLES_OPTION
```

PRECONFIG_USERNAME_OPTION variable

Configuration property to specify the user name.

Syntax

```
final String PRECONFIG_USERNAME_OPTION
```

ProfilerImpl class

Syntax

```
public class ProfilerImpl
```

destroyContext(Map< String, Object >) method

Subclasses that need to clean up when a `SecContext` will no longer be used should override this method so they can do their cleanup.

Syntax

```
void destroyContext ( Map< String, Object > context ) throws  
SecException
```

Parameters

- **context** – the context from the SecContext that is being destroyed.

Exceptions

- **SecException class** – if any error occurs while destroying the context. This error will be logged but will not be propagated to the client.

getProfile(Map< String, Object >, SecProfile) method

This method should populate the SecProfile object by calling the set methods on it .

Syntax

```
boolean getProfile ( Map< String, Object > context , SecProfile profile )
```

Parameters

- **context** – CSI context
- **profile** – profile object

Returns

true if the SecProfile is populated; false otherwise

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned false.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

init(Map< String, ?>) method

Implementations that care about initialization properties should override this method.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```

Parameters

- **configuration** – Provider specific configuration data.

Exceptions

- **SecException class** – if a required property is missing, or a provided property had bad syntax or couldn't be used. If a provider throws this exception it will be printed to the log and the provider will be inaccessible.

initContext(Map< String, Object >) method

Subclasses that would like to initialize each SecContext before any other methods are called may do so in this method.

Syntax

```
void initContext ( Map< String, Object > context ) throws  
SecException
```

Parameters

- **context** – the context from the SecContext that is being initialized

Exceptions

- **SecException class** – if any error occurs while creating the context. This will invalidate the context so care should be made about throwing this exception. The exception will be propagated to the client.

listProfiles(Map< String, Object >) method

Returns the list of all known profiles.

Syntax

```
List< Named > listProfiles ( Map< String, Object > context )
```

Parameters

- **context** – CSI context

Returns

List containing Named objects describing the profiles known to this profiler.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

PropertiesConfiguration class

Provides the standard property file-based configuration facility for CSI.

Syntax

```
public class PropertiesConfiguration
```

Remarks

This configuration provider can use configuration data from three different sources in various combinations:

- A file on the filesystem
- A resource loaded from the classpath
- A java.util.Properties object specified programmatically

Loading from a file and resource are each mutually exclusive. The Properties object overrides those values loaded from either the file or resource.

System properties may be used to alter the default behavior when the class is instantiated, or the values may be modified programmatically.

System property values may be embedded into the values side of the properties file. The format of the value must be \${system.property.name}.

PropertiesConfiguration() constructor

Instantiates properties configuration with the default values.

Syntax

```
PropertiesConfiguration ()
```

PropertiesConfiguration(boolean) constructor

Subclasses of PropertiesConfiguration should call this constructor with false as the doBootstrap argument.

Syntax

```
PropertiesConfiguration ( boolean doBootstrap )
```

Usage

This will allow the subclasses constructor to call automaticBootstrap() to perform bootstrapping.

buildConfiguration() method

This method rebuilds the configuration without regards for caching.

Syntax

```
Map< String, String > buildConfiguration () throws SecException
```

Returns

configuration Map

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

getSuggestedExtension() method

Gets the configuration file extension.

Syntax

```
String getSuggestedExtension ()
```

Returns

file extension to fully qualify the file name.

writeNewConfiguration(Map< String, Object >, Map< String, String >) method

Stores the specified configuration to the file it was loaded from.

Syntax

```
void writeNewConfiguration (Map< String, Object > selectors , Map< String, String > configuration) throws SecException
```

Usage

If the configuration source is an URL pointing to a non local file, a resource or a read only file a SecException is thrown.

FILE_NAME_SYSTEM_PROPERTY variable

The system property that should be set in order to set the file name on the filesystem where the properties file should be loaded from.

Syntax

```
final String FILE_NAME_SYSTEM_PROPERTY
```

Remarks

This property has no default and overrides the resource name, if set.

RESOURCE_NAME_SYSTEM_PROPERTY variable

The system property that should be set in order to modify the name of the resource where CSI properties are loaded from.

Syntax

```
final String RESOURCE_NAME_SYSTEM_PROPERTY
```

RESOURCE_NAME_SYSTEM_PROPERTY_DEFAULT variable

The default value of the corresponding system property.

Syntax

```
final String RESOURCE_NAME_SYSTEM_PROPERTY_DEFAULT
```

RoleCheck interface

An interface that principals and credentials can implement to signal that they convey role membership.

Syntax

```
public interface RoleCheck
```

Derived classes

- *com.sybase.security.core.ClientValuePropagatingLoginModule.CVPLMRolePrincipal* on page 70
- *com.sybase.security.core.PreConfiguredUserLoginModule.PreConfigUserRolePrincipal* on page 113

checkRole(String) method

Checks if the specified role is granted.

Syntax

```
boolean checkRole ( String role )
```

Parameters

- **role** – name of the role to check.

Returns

true if the role is granted; false otherwise

RoleCheckAuthorizer class

This authorizer provides only role-based authorization checks.

Syntax

```
public class RoleCheckAuthorizer
```

Remarks

It answers the checkRole method based on the principals and credentials contained in the specified subject. It scans for principals and credentials that implement the com.sybase.security.core.RoleCheck interface in order to perform the role check.

checkRole(Map< String, Object >, String, SecSubject) method

Syntax

```
int checkRole ( Map< String, Object > context , String roleName ,  
SecSubject subject )
```


Returns

ProviderConst.VOTE_ABSTAIN by default, unless overridden in subclasses.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

RoleMapperAdmin class

Allows role mappings to be managed programmatically.

Syntax

```
public class RoleMapperAdmin
```

Derived classes

- *com.sybase.security.core.XMLFileRoleMapper.XMLFileRoleMapperAdmin* on page 141

Remarks

Instances of this class may be retrieved in one of two ways:

1. Using the no-argument constructor, an "empty" container will be created on which you may call `loadFromXML(Reader)` and `toXML(Writer)`. Changes made to objects retrieved in this manner will not affect any other resources indirectly.
2. By calling `SecAdminContext.getRoleMapperAdmin()`, one may retrieve an instance of this class associated with the active role mapping file. The container will be prepopulated with the existing mappings from the file. Changes made to this object will not affect the underlying role mappings until the `activate()` method is called.

SimpleHandler class

this class handles the parsing logic through standard SAX mechanisms.

Syntax

```
private class SimpleHandler
```

Remarks

It associates with parent class instance.

characters(char[], int, int) method

Syntax

```
void characters ( char[] ch , int start , int length )
```

endElement(String, String, String) method

Syntax

```
void endElement ( String uri , String localName , String qName ) throws  
SAXException
```

error(SAXParseException) method

Syntax

```
void error ( SAXParseException ex ) throws SAXException
```

fatalError(SAXParseException) method

Syntax

```
void fatalError ( SAXParseException ex ) throws SAXException
```

startElement(String, String, String, Attributes) method

Syntax

```
void startElement ( String uri , String localName , String qName ,  
Attributes attributes ) throws SAXException
```

warning(SAXParseException) method

Syntax

```
void warning ( SAXParseException ex )
```

elementContent variable

Syntax

StringBuilder elementContent

hasMap variable

Syntax

boolean hasMap

RoleMapperAdmin(RoleMappings , SecAdminContext) constructor

Constructor used by subclasses that can set the admin context.

Syntax

RoleMapperAdmin (RoleMappings *mappings* , SecAdminContext *adminCtx*)

Parameters

- **mappings** – role mappings
- **adminCtx** – unused for this impl, but used for subclasses

RoleMapperAdmin() constructor

Constructs an empty role mapper admin object.

Syntax

RoleMapperAdmin ()

RoleMapperAdmin(Reader, SecAdminContext) constructor

Constructor used by subclasses that can set the admin context.

Syntax

RoleMapperAdmin (Reader *in* , SecAdminContext *adminCtx*) throws
SecException

Parameters

- **adminCtx** – unused for this impl, but used for subclasses SecException if any error is encountered initializing the role mappings from the supplied reader.

activate() method

Will activate the current set of mappings in the role mapper provider from which it was created.

Syntax

```
void activate () throws SecException
```

Exceptions

- **SecException class** – if any errors occur while activating the mappings

Usage

This normally includes writing the mappings out to the role mapping file.

fromXML(Reader) method

This method is deprecated in favor of activate() and loadFromXML.

Syntax

```
void fromXML ( Reader in ) throws SecException
```

Parameters

- **in** – the source for new mappings, null if no new mappings (only write file)

Exceptions

- **SecException class** – if an error occurs trying to read or parse the new mapping source or writing new files

Usage

Replace the existing mappings in this object with those from the supplied Reader object. A call to this method will attempt to store the mappings in the underlying mapping data file as well, if one exists.

ATTENTION: This is a destructive operation and will remove ALL existing mappings!

getMapping(String, String) method

Retrieve a list of physical roles associated with the supplied package and logical role.

Syntax

```
List< String > getMapping ( String pkg , String logicalRole )
```

Parameters

- **pkg** – the package
- **logicalRole** – the logical role

Returns

a list of physical roles

getMappings() method

Gets the stored role mappings.

Syntax

```
RoleMappings getMappings ()
```

Returns

role mappings

getSecAdminContext() method

Gets the SecAdminContext.

Syntax

```
SecAdminContext getSecAdminContext ()
```

Returns

SecAdminContext stored in this instance.

loadFromXML(Reader) method

Replace the existing mappings in this object with those from the supplied Reader object.

Syntax

```
void loadFromXML ( Reader in ) throws SecException
```

Parameters

- **in** – the source for new mappings, null if no new mappings (only write file)

Exceptions

- **SecException class** – if an error occurs trying to read or parse the new mapping source

Usage

ATTENTION: This is a destructive operation and will remove ALL existing mappings!

setMappings(RoleMappings) method

Sets the specified role mappings.

Syntax

```
void setMappings ( RoleMappings mappings )
```

Parameters

- **mappings** – role mappings to be set

Usage

These are not stored to the destination until toXML method is invoked.

setValidate(boolean) method

Sets the validation flag to the specified value to enable/disable XML validation when reading the role mapping information.

Syntax

```
void setValidate ( boolean validate )
```

Parameters

- **validate** – boolean flag to be set to enable/disable XML validation

toXML(Writer) method

Generate an XML document containing the current role mappings.

Syntax

```
void toXML ( Writer out ) throws SecException
```

Parameters

- **out** – a destination for existing mappings

Exceptions

- **SecException class** – if an error occurs while writing the mappings to the destination

Usage

This method call will not change the existing mappings at all.

RoleMappings class

Maintains a list of role package objects.

Syntax

```
public class RoleMappings
```

Remarks

This is the root container object for role mappings.

RoleMappings() constructor

Creates an empty role mapping container.

Syntax

```
RoleMappings ()
```

addChild(String) method**Syntax**

```
RolePackage addChild (String name) throws  
AlreadyExistsException
```

RolePackage class

RolePackage maintains a list of logical role mappings within a package.

Syntax

```
public class RolePackage
```

RolePackage(String, RoleMappings) constructor

Constructor to define the package name for which the role mappings are defined.

Syntax

```
RolePackage (String name , RoleMappings mappings)
```

Parameters

- **name** – name of the package the role mappings are to be scoped
- **mappings** – role mappings

addChild(String) method**Syntax**

```
LogicalRole addChild (String name) throws  
AlreadyExistsException
```

XmlAuditFormatter class

An audit formatter that formats audit data into an XML record.

Syntax

```
public class XmlAuditFormatter
```

Remarks

This implementation is completely threadsafe in that one instance may satisfy requests from multiple threads concurrently.

FactoryHolder class

Holds a set of classes whose access must be synchronized between instances.

Syntax

```
private class FactoryHolder
```

Derived classes

- *com.sybase.security.core.XmlAuditFormatter.SynchronizedFormatHelper* on page 131

FactoryHolder() constructor

Syntax

FactoryHolder () throws SecException

generateDateTimeField(Date) method

Syntax

String generateDateTimeField (Date *date*)

transformDocument(Document) method

Syntax

String transformDocument (Document *document*) throws SecException

_dateFormatter variable

Syntax

```
final SimpleDateFormat _dateFormatter
```


*_documentBuilder variable***Syntax**

```
final DocumentBuilder _documentBuilder
```

*_stringWriter variable***Syntax**

```
final StringWriter _stringWriter
```

*_transformer variable***Syntax**

```
final Transformer _transformer
```

SynchronizedFormatHelper class

This is a well-performing format helper that creates a single set of generation objects and synchronizes access to them as appropriate.

Syntax

```
private class SynchronizedFormatHelper
```

Remarks

This is a fairly simple implementation.

*SynchronizedFormatHelper() constructor***Syntax**

```
SynchronizedFormatHelper () throws SecException
```

*generateDateTimeField(Date) method***Syntax**

```
synchronized String generateDateTimeField ( Date date )
```

*newDocument() method***Syntax**

```
synchronized Document newDocument () throws SecException
```

transformDocument(Document) method

Syntax

`synchronized String transformDocument (Document document) throws
SecException`

ThreadLocalFormatHelper class

This is the best-performing format helper when memory is plentiful.

Syntax

```
private class ThreadLocalFormatHelper
```

Remarks

It uses threadlocal variables to create helper objects for each thread eliminating any need for synchronization. Using simple microbenchmarks, this approach was found to be ~25% faster than the synchronized format helper on a single-cpu machine. On a 4-cpu machine, this implementation was 90% faster than the synchronized format helper. Warning: this implementation exposed significant problems in very old XML implementations that resulted in OutOfMemoryError's. This was not a problem with the implementations in JDK 1.4.2 or higher. This is why we have left in the synchronized implementation.

generateDateTimeField(Date) method

Syntax

`String generateDateTimeField (Date date) throws SecException`

newDocument() method

Syntax

`Document newDocument () throws SecException`

transformDocument(Document) method

Syntax

`String transformDocument (Document document) throws SecException`

FormatHelper interface

Common interface for the formatting helper classes.

Syntax

```
private interface FormatHelper
```

Derived classes

- *com.sybase.security.core.XmlAuditFormatter.SynchronizedFormatHelper* on page 131
- *com.sybase.security.core.XmlAuditFormatter.ThreadLocalFormatHelper* on page 132

generateDateTimeField(Date) method

Syntax

String generateDateTimeField (Date *date*) throws SecException

newDocument() method

Syntax

Document newDocument () throws SecException

transformDocument(Document) method

Syntax

String transformDocument (Document *document*) throws SecException

format(String, String, String, Map< String, Object >, Decision) method

Retrieve the string format of the given audit information.

Syntax

String format (String *resourceClass* , String *resourceID* , String *action* , Map< String, Object > *attributes* , Decision *decision*) throws SecException

Parameters

- **resourceClass** – the scope used when processing the resource ID.
- **resourceID** – the object on which an operation is being performed.
- **action** – the operation that is being performed.
- **attributes** – A map of attributes associated with the audit record.
- **decision** – the result of the security check.

Returns

a string representation of the given audit information.

Exceptions

- **SecException class** – when a error is encountered during audit record formatting.

getFooter() method

Retrieve the footer for the XML document containing log entries.

Syntax

```
String getFooter ()
```

Returns

the footer

getHeader() method

Retrieve the header for the XML document containing log entries.

Syntax

```
String getHeader ()
```

Returns

the header

getStringAttributeValue(Object) method

Build a string attribute value.

Syntax

```
String getStringAttributeValue ( Object value ) throws SecException
```

Parameters

- **value** – object to construct the string attribute value from

Exceptions

- **SecException class** –

Usage

This implementation specially handles Date and X509Certificate instances.

init(Map< String, ?>) method

Implementations that care about initialization properties should override this method.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```

Parameters

- **configuration** – Provider specific configuration data.

Exceptions

- **SecException class** – if a required property is missing, or a provided property had bad syntax or couldn't be used. If a provider throws this exception it will be printed to the log and the provider will be inaccessible.

isReduceThreadContentionSet() method

Checks if this instance is set to reduce thread contention.

Syntax

```
boolean isReduceThreadContentionSet ()
```

Returns

whether or not this instance is set to reduce thread contention

OPTION_REDUCE_THREAD_CONTENTION variable

If set to false, enables the slightly slower traditional synchronization object model.

Syntax

```
final String OPTION_REDUCE_THREAD_CONTENTION
```

OPTION_REDUCE_THREAD_CONTENTION_DEFAULT variable

default value for OPTION_REDUCE_THREAD_CONTENTION

Syntax

```
final boolean OPTION_REDUCE_THREAD_CONTENTION_DEFAULT
```

XmlConfiguration class

Provides the standard XML-based configuration facility for CSI.

Syntax

```
public class XmlConfiguration
```

Remarks

This configuration provider can use configuration data from three different sources in various combinations:

- A file on the filesystem
- A resource loaded from the classpath
- A java.util.Properties object specified programmatically

Security API

Loading from a file and resource are each mutually exclusive. The Properties object overrides those values loaded from either the file or resource.

System properties may be used to alter the default behavior when the class is instantiated, or the values may be modified programmatically. The system property "com.sybase.security.core.XmlConfiguration.XmlValidation" determines if the input XML configuration should be validated against the XML schema definition. It must either be unspecified or must be set to "true" or "false". If it is unspecified then the XML configuration file is not validated against the schema.

System property values may be embedded into any provider option values included in the configuration file. Properties are referenced using the format `${system.property.name}`.

SimpleHandler class

This class handles the parsing logic through standard SAX mechanisms.

Syntax

```
package class SimpleHandler
```

Remarks

It associates with parent class instance.

SimpleHandler() constructor

Default constructor.

Syntax

```
SimpleHandler ()
```

endElement(String, String, String) method

Syntax

```
void endElement ( String uri , String localName , String qName ) throws SAXException
```

error(SAXParseException) method

Syntax

```
void error ( SAXParseException ex ) throws SAXException
```

fatalError(SAXParseException) method

Syntax

```
void fatalError ( SAXParseException ex ) throws SAXException
```

getConfig() method

Retrieves the parsed configuration.

Syntax

```
Map< String, String >getConfig ()
```

startElement(String, String, String, Attributes) method**Syntax**

```
void startElement ( String uri , String localName , String qName ,
Attributes attributes ) throws SAXException
```

warning(SAXParseException) method**Syntax**

```
void warning ( SAXParseException ex )
```

XmlConfiguration() constructor

Instantiates XmlConfiguration with default values.

Syntax

```
XmlConfiguration ()
```

XmlConfiguration(boolean) constructor

Subclasses of XmlConfiguration should call this constructor with false as the doBootstrap argument.

Syntax

```
XmlConfiguration ( boolean doBootstrap )
```

Parameters

- **doBootstrap** – boolean value to indicate if the method should perform automatic bootstrapping in the constructor.

Usage

This will allow the subclasses constructor to call automaticBootstrap() to perform bootstrapping.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

getSuggestedExtension() method

Gets the configuration file extension.

Syntax

```
String getSuggestedExtension ()
```

Returns

file extension to fully qualify the file name.

getValidateXml() method

Retrieves the validate xml flag.

Syntax

```
boolean getValidateXml ()
```

Returns

value of the validateXml flag

setValidateXml(boolean) method

Sets the validateXml flag to the specified value.

Syntax

```
void setValidateXml ( boolean validate )
```


Parameters

- **validate** – specifies whether or not to validate the input xml configuration file.

writeNewConfiguration(Map< String, Object >, Map< String, String >) method

Stores the specified configuration.

Syntax

```
void writeNewConfiguration ( Map< String, Object > selectors , Map<
String, String > configuration ) throws SecException
```

Parameters

- **selectors** – a map of selectors that dynamically select the configuration destination.
- **configuration** – Configuration properties to be saved to the selected destination

Exceptions

- **SecException class** – If the configuration destination is an URL pointing to a non local file, a resource or a read only file

Usage

If the configuration source is an URL pointing to a non local file, a resource or a read only file a SecException is thrown.

CONFIGURATION_XML_VALIDATION_PROPERTY variable

The name of the system property used to choose the XML configuration file validation behavior.

Syntax

```
final String CONFIGURATION_XML_VALIDATION_PROPERTY
```

Remarks

If not specified, the xml file is not validated.

CONFIGURATION_XML_VALIDATION_PROPERTY_DEFAULT variable

The default value for CONFIGURATION_XML_VALIDATION_PROPERTY.

Syntax

```
final String CONFIGURATION_XML_VALIDATION_PROPERTY_DEFAULT
```

FILE_NAME_SYSTEM_PROPERTY variable

The system property that should be set in order to set the file name on the filesystem where the xml file should be loaded from.

Syntax

```
final String FILE_NAME_SYSTEM_PROPERTY
```

Remarks

This property has no default and overrides the resource name, if set.

JAXP_SCHEMA_LANGUAGE variable

tells the XML parser what schema language we're using

Syntax

```
final String JAXP_SCHEMA_LANGUAGE
```

JAXP_SCHEMA_SOURCE variable

allows us to hard-code the XSD file

Syntax

```
final String JAXP_SCHEMA_SOURCE
```

RESOURCE_NAME_SYSTEM_PROPERTY variable

The system property that should be set in order to modify the name of the resource from which CSI configuration is loaded from.

Syntax

```
final String RESOURCE_NAME_SYSTEM_PROPERTY
```

RESOURCE_NAME_SYSTEM_PROPERTY_DEFAULT variable

The default value for RESOURCE_NAME_SYSTEM_PROPERTY.

Syntax

```
final String RESOURCE_NAME_SYSTEM_PROPERTY_DEFAULT
```

W3C_XML_SCHEMA variable

the value that specifies the XML Schema language

Syntax

```
final String W3C_XML_SCHEMA
```

XMLFileRoleMapper class

The RoleMapper provider maintains role mapping data in an XML File format.

Syntax

```
public class XMLFileRoleMapper
```

RoleMapperConfig class

This helper class processes the options for XMLFileRoleMapper provider.

Syntax

```
package class RoleMapperConfig
```

XMLFileRoleMapperAdmin class

This class enhances the generic RoleMapperAdmin to provide persistence for role mapping data in the file specified in the CSI configuration for this provider.

Syntax

```
package class XMLFileRoleMapperAdmin
```

activate() method

Let parent class parse and apply the mapping, then from here we write the new XML out.

Syntax

```
void activate () throws SecException
```

getMapping(String, String) method**Syntax**

```
List< String > getMapping ( String pkg , String logicalRole )
```

Returns

an empty list of Strings by default, unless overridden in subclasses.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```

Returns

the version

getRoleMapperAdmin(SecAdminContext) method

Returns a RoleMapperAdmin object that can be used to manage the role mappings directly associated with this provider.

Syntax

```
RoleMapperAdmin getRoleMapperAdmin ( SecAdminContext sac )
```

Usage

Changes made to this object are isolated from the actual file until the fromXML() method is called

init(Map< String,?>) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```

DISABLE_PASS_THRU variable

This boolean valued option indicates of straight thru mapping from logical to physical roles is enabled.

Syntax

```
final String DISABLE_PASS_THRU
```

Remarks

The default is enabled. Explicitly set this option to "true" to disable.

ROLE_MAP_FILE variable

This option name specifies the file where role Mappings are stored relative to the main CSI configuration file.

Syntax

```
final String ROLE_MAP_FILE
```

provider package

Custom providers allow for the customization and extension of security enforcement as needed, by implementing Provider-side interfaces to author custom providers.

Members

All public members of the provider package.

- **AbstractAttributed class** – This class provides an abstract implementation of interface Attributed.
- **AbstractAttributer class** – Abstract implementation of Attributer.
- **AbstractAuthorizer class** – Abstract implementation of Authorizer.
- **AbstractBootstrapConfiguration class** – Abstract class providing core configuration provider services.
- **AbstractFactoryRetriever class** – Common factory retriever base.
- **AbstractFileConfiguration class** – Abstract class providing configuration-file based services.
- **AbstractLoginModule class** – Provides common base implementations for most common LoginModule functions.
- **AbstractPrincipalContextRetriever class** – Abstract class used as a base for context retriever implementations that retrieve the security context from a principal made available through some environmental object.
- **AbstractProfiler class** – An abstract implementation of the Profiler interface.
- **AbstractRoleMapper class** – Base class for implementing RoleMapper providers.
- **AbstractSecureDataServices class** –
- **AbstractSecureFileConfiguration class** – Abstract class for configuring keystore options in order to facilitate subclass to get key information and decrypt and encrypt specified configuration properties.
- **Attributer interface** – This is the interface which Attributer providers must implement.
- **AttributerRegistration interface** – This is the interface to support self registration functions, which Attributer providers can optionally implement.
- **AttributerRegistration2 interface** – This is the interface to support self registration functions, which Attributer providers can optionally implement.
- **AuditConst interface** – Interface that defines the constants for audit record types.

- **AuditDestination interface** – The AuditDestination is the destination to which the audit information should be logged.
- **AuditFilter interface** – An audit filter may be configured by the administrator to filter out undesired events and only log the events that match the specified filter.
- **AuditFormatter interface** – The audit formatter is used to format an audit record from its component parts.
- **AuditToken interface** – An interface that is used to tag audit token classes.
- **AuthenticationFailureWarningImpl class** – Simple implementation for AuthenticationFailureWarning.
- **Authorizer interface** – This is the required interface for authorization providers.
- **BasicNamed class** – This class provides an implementation for Named.
- **BasicSecIDPrincipal class** – Provides a simple Principal base implementation that implements SecIDPrincipal.
- **BasicSecNamePrincipal class** – Provides a simple Principal base implementation that implements SecNamePrincipal.
- **Bootstrap class** – Provides an object bootstrapping function.
- **CertificateValidation interface** – Interface that provides certificate validation functionality.
- **CertificateValidationException class** – An exception thrown by the Certificate Validation module.
- **ConfigurationParser class** – Utility class that provides a tool to parse the internal configuration from the Map<String, String> format into Provider configuration classes for easy access to the configuration properties for the various providers.
- **ConfigurationProblem class** – Represents the configuration problems discovered during configuration validation.
- **ConfigurationValidationService interface** – Validates the supplied internal CSI configuration according to the stored provider metadata and lets the providers themselves validate the supplied configuration by performing runtime checks where possible.
- **ContextRetriever interface** – Authors wishing to create context retrievers must implement this interface in a class with a public, no argument constructor.
- **ContextRetriever2 interface** – Extended version of context retriever interface that allows the factory that was used to retrieve the context to be supplied.
- **ContextRetrieverPrincipal interface** – Principal interface designed to be paired with AbstractPrincipalContextRetriever and its subclasses.
- **DigitalSignature interface** – Interface to be implemented by secure data service providers that support digital signatures.
- **EncryptionTools class** – This class performs specialized encryption/decryption functions for relatively short strings such as passwords.
- **ExternalConfigurationService interface** – A service that will export the CSI configuration to a specified XML format.

- **FactoryRetriever interface** – Concrete factory retrievers must implement this interface in a class with a public, no argument constructor.
- **MessageDigest interface** – Interface to be implemented by secure data service providers that support message digest.
- **NamedCredentialProvider interface** – This is the interface that the providers can implement so that the configuration validation service can retrieve the names of the NamedCredential(s) the provider adds to an authenticated subject so that it can verify if they conflict with the credentials added by other providers.
- **OptionMapHelper class** – Provides some utility methods for retrieving options from a Map where there are default values and different data types involved.
- **PasswordExpirationWarningImpl class** – Simple implementation for PasswordExpirationWarning.
- **PrefixMap< T > class** – Utility class that extends HashMap that expects keys of type String.
- **Profiler interface** – This is the interface that Profile providers must implement.
- **ProviderConst interface** – Constants that are useful in implementing provider interfaces.
- **ProviderInfo interface** – This interface may optionally implemented by providers of the following types:
- **ProviderServices interface** – Interface that providers can use to access common services without interacting with other providers directly.
- **RoleMapAdministrable interface** – Interface to aid in managing role mappings.
- **RoleMapper interface** – Interface for providers that implement Role Mapping functionality.
- **SecConfigurationValidatingProvider interface** – This is the interface that the providers can implement so that the configuration can be validated and the discovered validation errors can be reported prior to actually using the configuration.
- **SecContextProvider interface** – Interface that should be implemented by providers that support the notion of a context (i.e., session) and need to perform specific actions when a context is created/destroyed.
- **SecIDPrincipal interface** – Tagging interface that can be used to identify a custom Principal as one that should be mapped to the ID attribute of a SecSubject.
- **SecLoginExceptionAuthenticationFailureWarningImpl class** – Simple LoginException implementation that implements the AuthenticationFailureWarning interface.
- **SecLoginExceptionWarningImpl class** – Simple LoginException implementation that implements the SecWarning interface.
- **SecNamePrincipal interface** – Tagging interface that can be used to identify a custom Principal as one that should be mapped to the NAME attribute of a SecSubject.
- **SecProvider interface** – Common interface that should be implemented by all security providers (except authentication providers).

- **SecProviderCapabilities interface** – Optional provider interface that allows providers to respond to capability requests.
- **SecProviderPersistence interface** – This interface gives a provider the opportunity to remove elements from the context Map before context serialization occurs and insert them after deserialization.
- **SecureDataServices interface** – Interface to be implemented by secure data service providers.
- **SecWarningImpl class** – Simple implementation for SecWarning.
- **SynchronizedSecAdminContext class** – This class synchronizes access to the supplied SecAdminContext instance.
- **SynchronizedSecContextImpl class** – This class synchronizes access to the supplied SecContext instance.
- **WarningManager interface** – Interface that providers can use to log and add warnings to the SecContext.

Remarks

Authentication Provider Interfaces The provider-side authentication interfaces are primarily based on Java Authentication and Authorization Services (JAAS). The goal of the design is to allow any implementation of the JAAS pluggable authentication module system to alternatively plug into the Security Framework. Specifically, the `javax.security.auth.spi.LoginModule` interface must be implemented by all authentication providers. The Framework provides a flexible mechanism for defining active authentication providers and their control flags. All of the control flags which are defined in JAAS are fully implemented in the Provider Framework. See the *JAAS documentation* for a complete discussion of the configuration options available with JAAS login modules.

Authorization Provider Interfaces The authorization provider interfaces are defined by the `com.sybase.security.provider.Authorizer` interface. There are two primary worker methods, `checkRole()` and `checkAccess()`. There are also two security context lifecycle methods, `initContext()` and `destroyContext()`, and one provider lifecycle method, `init()`, all of which are inherited from `com.sybase.security.provider.SecContextProvider`.

Attribution Providers The attribution providers are a catch-all for several different tasks:

- Enumeration of resources, resource types, actions, roles
- Retrieval of attributes for resources and subjects

The attribution providers have the same lifecycle methods as the authorization providers. The same rules should be followed with respect to thread-safety; all of the attribution provider methods include a context map which can be used to store state information associated with any given security context.

Profile Providers A profile provider is based on the `Profiler` interface. A profiler retrieves the profile information given a profile name. Multiple profilers can be configured. There are two supported operations `getProfile()` and `listProfiles()`. There are also two security context lifecycle methods, `initContext()` and `destroyContext()`, and one provider lifecycle method,

init(), all of which are inherited from com.sybase.security.provider.SecProvider. If a profiler does not recognize a profile name then it simply returns null and the getProfile() method calls falls through to the next configured profiler. The listProfiles() method is invoked on all the configured profilers and the aggregate list is returned to the client.

AbstractAttributed class

This class provides an abstract implementation of interface Attributed.

Syntax

```
public class AbstractAttributed
```

AbstractAttributed(boolean) constructor

Constructor which allows callers to enable instantiation of the object in read-only mode.

Syntax

```
AbstractAttributed ( boolean allowsReadOnly )
```

Parameters

- **allowsReadOnly** – whether or not this implementation should allow calls to enable readonly mode

addAttribute(String, String) method

Adds an additional value to an attribute.

Syntax

```
void addAttribute ( String name , String value )
```

Parameters

- **name** – the name of the attribute
- **value** – the value to add

Exceptions

- **IllegalStateException** – if called after marked read only

Usage

If the attribute does not exist, it will be created.

addAttributes(String, String[]) method

Adds additional values to an attribute.

Syntax

```
void addAttributes ( String name , String[] values )
```

Parameters

- **name** – the name of the attribute
- **values** – the values to add

Exceptions

- **IllegalStateException** – if called after the attributed object is marked read only

Usage

If the attribute does not exist, it will be created.

commitModifications() method

Commits any changes made to this attributed object.

Syntax

```
void commitModifications () throws SecException
```

getAllAttributes() method

A method to return a map of all the attribute names and their corresponding values.

Syntax

```
Map< String, String[]> getAllAttributes ()
```

Returns

a Map of attribute names and values. Modifications to the returned value will not be propagated to the attributed object.

getAttribute(String) method

Convenience method for getting single-valued attributes.

Syntax

```
String getAttribute ( String name )
```

Parameters

- **name** – The name of the attribute whose value should be retrieved.

Returns

The first value for the named attribute. If there are no attribute values with the given name, it returns null.

getAttributes(String) method

Retrieves an array of values for the specified attribute name.

Syntax

```
String[] getAttributes ( String name )
```

Parameters

- **name** – The name of the attribute whose values should be retrieved.

Returns

All the values associated with the named attribute. If there are no attribute values with the given name returns null.

internalGetAllAttributes() method

A method to return a mutable Map of attribute names and values.

Syntax

```
Map< String, String[]> internalGetAllAttributes ()
```

Returns

a read/write Map of attribute names and values

isReadOnly() method

Returns the status of read only mode.

Syntax

```
boolean isReadOnly ()
```

Returns

true if this instance is marked read only

setAttribute(String, String) method

Convenience method for setting single-valued attributes.

Syntax

```
void setAttribute ( String name , String value )
```

Parameters

- **name** – the name of the attribute
- **value** – the attribute's single value

Exceptions

- **IllegalStateException** – if called after the attributed object is marked read only

setAttributes(String, String[]) method

Sets all of an attribute's values overwriting any values already set.

Syntax

```
void setAttributes ( String name , String[] values )
```

Parameters

- **name** – the name of the attribute
- **values** – the attribute's values

Exceptions

- **IllegalStateException** – if called after the attributed object is marked read only

setReadOnly() method

On Attributed objects supporting this method, permanently disables any operations that would modify the contents of this object.

Syntax

```
void setReadOnly ()
```

Exceptions

- **UnsupportedOperationException** – if the operation is not allowed on this instance

toString() method

A method to return serialized form of all attributes and their values.

Syntax

```
String toString ()
```

Returns

a java.lang.String representation of all attribute names and their corresponding values.

verifyNotReadOnly() method

A method to verify if the object is not read-only.

Syntax

```
void verifyNotReadOnly ()
```

Exceptions

- **IllegalStateException** – if the object is in readonly mode

_attributes variable

A Map of attribute names and values for overriding classes.

Syntax

```
HashMap< String, String[]> _attributes
```

AbstractAttributer class

Abstract implementation of Attributer.

Syntax

```
public class AbstractAttributer
```

Derived classes

- *com.sybase.security.core.NoSecAttributer* on page 98

Remarks

Provides place holder methods for all of the methods so the subclass only need override those that are relevant.

attributeAuthenticatedSubject(Map< String, Object >, SecSubject) method

Syntax

boolean attributeAuthenticatedSubject (Map< String, Object > *context* , SecSubject *subject*) throws SecException

Returns

false by default, unless overridden in subclasses.

attributeResource(Map< String, Object >, SecResource , String) method

Syntax

boolean attributeResource (Map< String, Object > *context* , SecResource *resource* , String *id*) throws SecException

Returns

false by default, unless overridden in subclasses.

attributeSubject(Map< String, Object >, SecSubject , String) method

Syntax

boolean attributeSubject (Map< String, Object > *context* , SecSubject *subject* , String *id*) throws SecException

Returns

false by default, unless overridden in subclasses.

destroyContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

void destroyContext (Map< String, Object > *context*) throws SecException

getSubjectAttributesForResource(Map< String, Object >, SecResource , SecSubject) method

Syntax

Map< String, String[]> getSubjectAttributesForResource (Map< String, Object > *context* , SecResource *res* , SecSubject *subject*)
throws SecException

Returns

null by default, unless overridden in subclasses.

init(Map< String, ?>) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

void init (Map< String, ?> *configuration*) throws SecException

initContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

void initContext (Map< String, Object > *context*) throws SecException

listActions(Map< String, Object >, SecResource) method

Syntax

List< Named > listActions (Map< String, Object > *context* , SecResource *res*) throws SecException

Returns

an empty list by default, unless overridden in subclasses.

listEnvironmentAttributes(Map< String, Object >) method

Syntax

List< Named > listEnvironmentAttributes (Map< String, Object > *context*) throws SecException

Returns

an empty list by default, unless overridden in subclasses.

listResources(Map< String, Object >, String) method

Syntax

List< Named > listResources (Map< String, Object > *context* ,
String *resourceType*) throws SecException

Returns

an empty list by default, unless overridden in subclasses.

listResourceTypes(Map< String, Object >) method

Syntax

List< Named > listResourceTypes (Map< String, Object > *context*)
throws SecException

Returns

an empty list by default, unless overridden in subclasses.

listRoles(Map< String, Object >) method

Syntax

List< Named > listRoles (Map< String, Object > *context*) throws
SecException

Returns

an empty list by default, unless overridden in subclasses.

*listSubjectAttributesForResource(Map< String, Object >, SecResource ,
SecSubject) method*

Syntax

List< Named > listSubjectAttributesForResource (Map< String,
Object > *context* , SecResource *res* , SecSubject *subject*) throws
SecException

Returns

an empty list by default, unless overridden in subclasses.

AbstractAuthorizer class

Abstract implementation of Authorizer.

Syntax

```
public class AbstractAuthorizer
```

Derived classes

- *com.sybase.security.core.NoSecAuthorizer* on page 99
- *com.sybase.security.core.RoleCheckAuthorizer* on page 122

Remarks

Provides place holder methods for all of the methods so the subclass only need override those that are relevant.

checkAccess(Map< String, Object >, String, SecResource , SecSubject , SecEnvironment) method

Syntax

```
int checkAccess ( Map< String, Object > context , String action ,
SecResource resource , SecSubject subject , SecEnvironment env )
throws SecException
```

Returns

ProviderConst.VOTE_ABSTAIN by default, unless overridden in subclasses.

checkRole(Map< String, Object >, String, SecSubject) method

Syntax

```
int checkRole ( Map< String, Object > context , String roleName ,
SecSubject subject ) throws SecException
```

Returns

ProviderConst.VOTE_ABSTAIN by default, unless overridden in subclasses.

destroyContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void destroyContext ( Map< String, Object > context ) throws
SecException
```

init(Map< String,?>) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```

initContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void initContext ( Map< String, Object > context ) throws  
SecException
```

AbstractBootstrapConfiguration class

Abstract class providing core configuration provider services.

Syntax

```
public class AbstractBootstrapConfiguration
```

Derived classes

- *com.sybase.security.core.NamedConfiguration* on page 87
- *com.sybase.security.provider.AbstractFileConfiguration* on page 159

Remarks

These services come in the form of bootstrapping capabilities based upon system properties that contain references to property files. Unlike many of the common security framework functions, each configuration provider must call the `automaticBootstrap` method in their constructor to take advantage of the system-property defined bootstrapping capabilities. Simply including this base class in the inheritance hierarchy does NOT enable the automatic bootstrapping.

AbstractBootstrapConfiguration() constructor

Default constructor.

Syntax

```
AbstractBootstrapConfiguration ()
```

Usage

This constructor will not call `automaticBootstrap` -- this is the responsibility of the implementation class's public constructor.

automaticBootstrap() method

This method, if called, should be called only ONCE during the initialization (constructors) of a class.

Syntax

```
void automaticBootstrap ()
```

setBootstrapProperty(String, String) method

called to set a name/value attribute

Syntax

```
void setBootstrapProperty ( String property , String value ) throws  
SecException
```

_automaticBootstrapComplete variable

Set to true after automatic bootstrapping has been performed.

Syntax

```
boolean _automaticBootstrapComplete
```

Remarks

Used to throw assertion failure if multiple automatic bootstraps are attempted.

_failedBootstrap variable

After the constructor returns, this value will be true if errors occurred while trying to bootstrap the configuration.

Syntax

```
boolean _failedBootstrap
```

_failedBootstrapException variable

After the constructor returns, this value will contain the FIRST exception that caused the bootstrap configuration process to fail.

Syntax

```
Throwable _failedBootstrapException
```

***BOOTSTRAP_CONFIGURATION_PROPERTY_FILE** variable*

If this system property is defined, the configuration provider will search for data in the form of a properties file at the specified location.

Syntax

```
final String BOOTSTRAP_CONFIGURATION_PROPERTY_FILE
```

Remarks

If the URL configuration property is specified then this property will be ignored.

***BOOTSTRAP_CONFIGURATION_PROPERTY_URL** variable*

If this system property is defined, the configuration provider will search for data in the form of a properties file at the specified URL.

Syntax

```
final String BOOTSTRAP_CONFIGURATION_PROPERTY_URL
```

***AbstractFactoryRetriever** class*

Common factory retriever base.

Syntax

```
public class AbstractFactoryRetriever
```

Remarks

This handles logic that is shared between most factory retriever instances.

***retrieveContainerFactory(Object)** method*

Subclasses should override this method.

Syntax

```
abstract SecContextFactory retrieveContainerFactory ( Object  
environmentObject ) throws SecException
```

Parameters

- **environmentObject** – client-supplied environment object

Returns

the appropriate factory object

Exceptions

- **SecException class** – on error

Usage

There is no need to return a cloned factory. The calling code will automatically clone it before returning it to the client.

retrieveFactory(Object) method

This method provides a default implementation of retrieving a Security Context factory.

Syntax

```
SecContextFactory retrieveFactory ( Object environmentObject ) throws
SecException
```

Parameters

- **environmentObject** – an instance of the configured environment.

Returns

an instance of the SecContextFactory

Usage

A default configuration is created for Servlet container contexts. Subclasses are generally required to implement the retrieveContainerFactory method.

returnedFactoryShouldBeCloned() method

Subclasses can override this to skip the factory cloning step, for those cases where it is unnecessary.

Syntax

```
boolean returnedFactoryShouldBeCloned ()
```

Returns

true by default, unless overridden in subclasses.

AbstractFileConfiguration class

Abstract class providing configuration-file based services.

Syntax

```
public class AbstractFileConfiguration
```

Derived classes

- *com.sybase.security.provider.AbstractSecureFileConfiguration* on page 178

Remarks

Subclasses need only implement the abstract `buildConfiguration` method if the default caching strategy is sufficient. This class supports loading configuration data from a specific file or a resource loaded from the classloader.

This class implements `SecConfiguration3` but there is no default implementation for `writeNewConfiguration` method. The sub classes must overwrite this method to store the specified configuration. For backwards compatibility it supplies a default implementation of the `SecConfiguration2.getConfiguration(Map)` method that ignores all configuration selectors except `SecConfiguration2.SELECTOR_FRESH_CONFIGURATION`. When this selector is specified the configuration from the configuration file is returned instead of the cached copy (which is still not updated).

AbstractFileConfiguration(String, File) constructor

Instantiates file-based configuration.

Syntax

```
AbstractFileConfiguration ( String defaultResourceName , File  
defaultFile )
```

Usage

Superclass should supply the appropriate default values for the no-arg constructor.

addGlobalProperties(Map< String, String >) method

return a map of provider prefixed Global configuration properties.

Syntax

```
Map< String, String > addGlobalProperties ( Map< String, String  
> configMap )
```

Parameters

- **configMap** – the map of properties to inspect for provider properties.

Returns

the map of global provider configuration properties.

buildConfiguration() method

Subclasses should implement this method to build the configuration from scratch whenever required.

Syntax

```
abstract Map< String, String > buildConfiguration () throws  
SecException
```

Returns

configuration Map

getConfigurationFile() method

Returns the File object housing the configuration data.

Syntax

```
File getConfigurationFile ()
```

Usage

If the file is a remote URL or some other non-writable source then this method will return null.

getConfigurationStream() method

Retrieves the configuration stream.

Syntax

```
InputStream getConfigurationStream () throws SecException
```

Returns

an InputStream with the contents of the specified configuration data

Exceptions

- **SecException class** – on error

Usage

This may be from a file or a resource depending on the active configuration of this object.

getConfigurationStreamWithArg(File, String) method

Syntax

InputStream getConfigurationStreamWithArg (File *file* , String *resourceName*) throws SecException

getFile() method

Retrieves the file object where configuration data will be retrieved, if any.

Syntax

File getFile ()

Returns

null if the configuration data will not be retrieved from a file

getOverrideProperties() method

Retrieves the value of any override properties.

Syntax

Map< String, String > getOverrideProperties ()

Returns

null if there are no override properties set

getResolvedURL(String) method

return the a fully resolved URL given a relative path entry

Syntax

URL getResolvedURL (String *relative*) throws SecException

getResource() method

Retrieves the resource where configuration data will be retrieved, if any.

Syntax

String getResource ()

Returns

null if the configuration data will not be retrieved from a resource

getURL() method

Retrieves the URL where configuration data will be retrieved, if any.

Syntax

```
URL getURL ()
```

Returns

null if the configuration data will not be retrieved from a URL

*invalidateCache() method***Syntax**

```
void invalidateCache ()
```

setFile(File) method

Modifies this configuration object to retrieve configuration data from the specified file.

Syntax

```
void setFile ( File file )
```

Parameters

- **file** – file to load data from

Usage

Setting this value will clear the value of the resource and url properties.

setResource(String) method

Modifies this configuration object to retrieve configuration data from the specified resource.

Syntax

```
void setResource ( String resource )
```

Parameters

- **resource** – resource to load data from

Usage

Setting this value will clear the value of the file and url properties.

setURL(URL) method

Modifies this configuration object to retrieve configuration data from the specified url.

Syntax

```
void setURL ( URL url )
```

Parameters

- **url** – url to load data from

Usage

Setting this value will clear the value of the file property and resource properties.

writeNewConfiguration(Map< String, Object >, Map< String, String >) method

Stores the updated configuration.

Syntax

```
void writeNewConfiguration ( Map< String, Object > selectors , Map< String, String > configuration ) throws SecException
```

Usage

There is no default implementation. The subclasses must overwrite this method to support this feature.

The supplied configuration should be written to the configuration file as is. The override properties should not be applied. The override properties should only be applied to the configuration when returned from `getConfiguration`.

AbstractLoginModule class

Provides common base implementations for most common LoginModule functions.

Syntax

```
public class AbstractLoginModule
```

Derived classes

- *com.sybase.security.core.CertificateAuthenticationLoginModule* on page 58
- *com.sybase.security.core.CertificateValidationLoginModule* on page 68
- *com.sybase.security.core.ClientValuePropagatingLoginModule* on page 70
- *com.sybase.security.core.NoSecLoginModule* on page 101
- *com.sybase.security.core.PreConfiguredUserLoginModule* on page 112

Remarks

The only method that needs to be implemented under most circumstances is the `login()` method.

***abort()* method**

Aborts the authentication process.

Syntax

```
boolean abort () throws LoginException
```

Returns

true if this method succeeded, or false if this `LoginModule` should be ignored.

Exceptions

- **LoginException** – encountered while aborting the authentication process.

Usage

This method is called if the `LoginContext`'s overall authentication failed. (the relevant `REQUIRED`, `REQUISITE`, `SUFFICIENT` and `OPTIONAL` `LoginModules` did not succeed). If this `LoginModule`'s own authentication attempt succeeded (checked by retrieving the private state saved by the `login` method), then this method cleans up any state that was originally saved.

***cleanupLocalCredentials()* method**

Subclasses must call this at the end of the `login` method in all cases to ensure the stored username/password are properly cleared.

Syntax

```
void cleanupLocalCredentials ()
```

Usage

It is recommended the entire body of the `login` method be wrapper in a `try/finally` with this method in the `finally` clause.

***clearSharedCredentials()* method**

This method is called from `abort` and/or `commit` to remove shared credentials if configured to do so.

Syntax

```
void clearSharedCredentials ()
```

commit() method

Method to commit the authentication process.

Syntax

```
boolean commit () throws LoginException
```

Returns

true if this method succeeded, or false if this LoginModule should be ignored.

Exceptions

- **LoginException** – encountered while associating relevant Principals and Credentials with the Subject

Usage

This method is called if the LoginContext's overall authentication succeeded (the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL LoginModules succeeded). If this LoginModule's own authentication attempt succeeded (checked by retrieving the private state saved by the login method), then this method associates relevant Principals and Credentials with the Subject located in the LoginModule. If this LoginModule's own authentication attempted failed, then this method removes/destroys any state that was originally saved.

getCertificateChain() method

Subclasses may call this method to retrieve the authenticated certificate chain.

Syntax

```
Certificate[] getCertificateChain ()
```

Returns

The authenticated Certificate chain

Usage

This value is normally added to the shared context state by the CertificateValidationLoginModule.

getPassword() method

Subclasses should call this method to retrieve the password.

Syntax

```
char[] getPassword () throws IOException,  
UnsupportedCallbackException
```

Returns

the password

Exceptions

- **IOException** – or **UnsupportedCallbackException** thrown while retrieving password.

Usage

Depending on the configuration of the login module this may result in a call to the callback handler or it may attempt to retrieve the password from the shared context.

getUsername() method

Subclasses should call this method to retrieve the username.

Syntax

```
String getUsername () throws IOException,  
UnsupportedCallbackException
```

Returns

the username.

Exceptions

- **IOException** – or **UnsupportedCallbackException** thrown while retrieving the username.

Usage

Depending on the configuration of the login module this may result in a call to the callback handler or it may attempt to retrieve the username from the shared context.

initialize(Subject, CallbackHandler, Map< String,?>, Map< String,?>) method
Initializes the LoginModule.

Syntax

```
void initialize ( Subject subject , CallbackHandler callback , Map<  
String,?> sharedState , Map< String,?> options )
```

Parameters

- **subject** – Subject to be authenticated
- **callback** – CallbackHandler to retrieve credentials
- **sharedState** – state shared with other configured providers
- **options** – configuration options for this provider

Usage

This method is invoked after the LoginModule has been instantiated. The purpose of this method is to initialize this LoginModule with the configuration properties. If this LoginModule does not understand any of the data stored in sharedState or options parameters, they can be ignored.

isClearPassSet() method

Method to check if the configuration option CLEARPASS_OPTION is true or false.

Syntax

```
boolean isClearPassSet ()
```

Returns

value of CLEARPASS_OPTION

isStorePassSet() method

Method to check if the configuration option STOREPASS_OPTION is true or false.

Syntax

```
boolean isStorePassSet ()
```

Returns

value of STOREPASS_OPTION

isTryFirstPassSet() method

Method to check if the configuration option TRYFIRSTPASS_OPTION is true or false.

Syntax

```
boolean isTryFirstPassSet ()
```

Returns

value of TRYFIRSTPASS_OPTION

isUseFirstPassSet() method

Method to check if the configuration option USEFIRSTPASS_OPTION is true or false.

Syntax

```
boolean isUseFirstPassSet ()
```

Returns

value of USEFIRSTPASS_OPTION

logout() method

Destroys any credentials relevant to this LoginModule.

Syntax

```
boolean logout () throws LoginException
```

Returns

true if this method succeeded, or false if this LoginModule should be ignored.

Exceptions

- **LoginException** – encountered while logging out.

Usage

This typically means ending any session that was created as part of this LoginModule.

If the subject isn't readonly, this also remove any principals and credentials added as part of the earlier authentication.

storeCertificateChain(Certificate[]) method

May be used to store an authenticated and validated certificate chain in the shared state.

Syntax

```
void storeCertificateChain ( Certificate[] certs )
```

Parameters

- **certs** – an authenticated certificate chain.

Usage

This method will only succeed if the following criteria are met:

- Certificate chain is not null
- Certificate chain contains at least one certificate.
- Certificate chain has not already been set in the shared context.

storeSharedCredentials() method

Subclasses must call this at the end of the login method if authentication is successful.

Syntax

```
void storeSharedCredentials ()
```

Usage

This will store the retrieved credentials if storePass flag is set.

_addedPrincipalSet variable

Upon successful authentication in login() method, derived classes should add principals to this set.

Syntax

```
Set< Principal > _addedPrincipalSet
```

_addedPrivateCredentialSet variable

Upon successful authentication in login() method, derived classes should add private credentials to this set.

Syntax

```
Set< Object > _addedPrivateCredentialSet
```

_addedPublicCredentialSet variable

Upon successful authentication in login() method, derived classes should add public credentials to this set.

Syntax

```
Set< Object > _addedPublicCredentialSet
```


_callback variable

initialized with the callback handler in the initialize method

Syntax

```
CallbackHandler _callback
```

_certificateAuthenticationEnabled variable

Initialized based on the configuration options.

Syntax

```
boolean _certificateAuthenticationEnabled
```

_commitSuccess variable

The commit method sets this value to true if the commit() was called successfully.

Syntax

```
boolean _commitSuccess
```

_loginSuccess variable

The login method should set this to true if the login was successful.

Syntax

```
boolean _loginSuccess
```

_options variable

initialized with the options in the initialize method

Syntax

```
Map< String, Object > _options
```

_sharedState variable

initialized with the shared state in the initialize method

Syntax

```
Map< String, Object > _sharedState
```

_subject variable

initialized with the subject in the initialize method

Syntax

```
Subject _subject
```

CLEARPASS_OPTION variable

Key relating to the clearPass option.

Syntax

```
final String CLEARPASS_OPTION
```

Remarks

If this option is true then the login module will clear the username/pw in the shared context when calling either commit or abort.

PASSWORD_SHARED_KEY variable

The shared key that the password is stored in when the storePass option is set to true.

Syntax

```
final String PASSWORD_SHARED_KEY
```

STOREPASS_OPTION variable

Key relating to the storePass option.

Syntax

```
final String STOREPASS_OPTION
```

Remarks

If this option is true then the login module will store the username/pw in the shared context after successfully authenticating.

TRYFIRSTPASS_OPTION variable

Key relating to the useFirstPass option.

Syntax

```
final String TRYFIRSTPASS_OPTION
```

Remarks

If this option is true, the login module will first attempt to retrieve username/pw from the shared context before attempting the callback handler.

USEFIRSTPASS_OPTION variable

Key relating to the tryFirstPass option.

Syntax

```
final String USEFIRSTPASS_OPTION
```

Remarks

If this option is true, the login module will only attempt to retrieve username/pw from the shared context; it will never call the callback handler.

***USERNAME_SHARED_KEY* variable**

The shared key that the username is stored in when the storePass option is set to true.

Syntax

```
final String USERNAME_SHARED_KEY
```

AbstractPrincipalContextRetriever class

Abstract class used as a base for context retriever implementations that retrieve the security context from a principal made available through some environmental object.

Syntax

```
public class AbstractPrincipalContextRetriever
```

Remarks

This implementation supports both HttpServletRequest and EJBContext environmental objects but subclasses can add additional support by overriding the extractContext and extractPrincipal methods. This class has special support for Principal objects that implement the ContextRetrieverPrincipal interface. Often it will be sufficient for subclasses to declare themselves, without adding any methods at all.

extractContext(Principal) method

This method implementation supports principals that implement the ContextRetrieverPrincipal interface.

Syntax

```
SecContext extractContext ( Principal principal )
```

Parameters

- **principal** – the principal from which the context will be extracted

Returns

the extracted security context

Usage

Subclasses may override this method to support additional interfaces or extraction techniques.

extractPrincipal(Object) method

This method supports extracting principals from both HttpServletRequest and EJBContext environment objects.

Syntax

```
Principal extractPrincipal ( Object environmentObject )
```

Parameters

- **environmentObject** – the object from which the principal will be extracted

Returns

the extracted Principal object

Usage

Subclasses may override this method to support additional environment objects.

retrieveContext(Object) method

Context retriever providers must implement this method to allow for retrieval of SecContext objects stored in the environment.

Syntax

```
SecContext retrieveContext ( Object environmentObject )
```

Parameters

- **environmentObject** –

Returns

the SecContext object or null if none found

Exceptions

- **SecException class** –

ejbContextAvailable variable

Syntax

```
final boolean ejbContextAvailable
```

AbstractProfiler class

An abstract implementation of the Profiler interface.

Syntax

```
public class AbstractProfiler
```

Remarks

This class provides default implementations for the Profile interface methods.

destroyContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void destroyContext ( Map< String, Object > context ) throws  
SecException
```

getProfile(Map< String, Object >, SecProfile) method**Syntax**

```
boolean getProfile ( Map< String, Object > context , SecProfile  
profile ) throws SecException
```

Returns

returns false by default, unless overridden in subclasses.

init(Map< String,?>) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void init ( Map< String,?> configuration ) throws SecException
```

initContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void initContext ( Map< String, Object > context ) throws  
SecException
```

listProfiles(Map< String, Object >) method

Syntax

List< Named > listProfiles (Map< String, Object > *context*) throws
SecException

Returns

an empty list by default, unless overridden in subclasses.

AbstractRoleMapper class

Base class for implementing RoleMapper providers.

Syntax

```
public class AbstractRoleMapper
```

Derived classes

- *com.sybase.security.core.XMLFileRoleMapper* on page 141

destroyContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

void destroyContext (Map< String, Object > *context*) throws
SecException

getMapping(String, String) method

Syntax

List< String > getMapping (String *pkg* , String *logicalRole*)

Returns

an empty list of Strings by default, unless overridden in subclasses.

init(Map< String, ?>) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

void init (Map< String, ?> *configuration*) throws SecException

initContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void initContext ( Map< String, Object > context ) throws
SecException
```

AbstractSecureDataServices class**Syntax**

```
public class AbstractSecureDataServices
```

Derived classes

- *com.sybase.security.core.JCESecureDataServices* on page 85

destroyContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void destroyContext ( Map< String, Object > context ) throws
SecException
```

getCipher(Map< String, Object >, SecProfile , String) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
abstract Cipher getCipher ( Map< String, Object > context ,
SecProfile profile , String operation ) throws SecException
```

init(Map< String,?>) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void init ( Map< String, ?> configuration ) throws SecException
```

initContext(Map< String, Object >) method

No default implementation provided, can be overridden in subclasses to implement desired functionality.

Syntax

```
void initContext ( Map< String, Object > context ) throws  
SecException
```

AbstractSecureFileConfiguration class

Abstract class for configuring keystore options in order to facilitate subclass to get key information and decrypt and encrypt specified configuration properties.

Syntax

```
public class AbstractSecureFileConfiguration
```

Derived classes

- *com.sybase.security.core.PropertiesConfiguration* on page 119
- *com.sybase.security.core.XmlConfiguration* on page 135

AbstractSecureFileConfiguration(String, File) constructor

Syntax

```
AbstractSecureFileConfiguration ( String resourceName , File  
configFile )
```

getCipherConfig() method

Syntax

```
EncryptionTools getCipherConfig () throws SecException
```

setCharset(String) method

Syntax

```
void setCharset ( String charset )
```

setCipherProvider(String) method

Syntax

```
void setCipherProvider ( String provider )
```


setCipherTransformation(String) method

Syntax

void setCipherTransformation (String *transform*)

setKeyStoreAlias(String) method

Syntax

void setKeyStoreAlias (String *storeAlias*)

setKeyStoreAliasPassword(String) method

Syntax

void setKeyStoreAliasPassword (String *aliasPasswd*)

setKeyStoreLocation(File) method

Syntax

void setKeyStoreLocation (File *keyStore*)

setKeyStorePassword(String) method

Syntax

void setKeyStorePassword (String *storePasswd*)

setKeyStoreProvider(String) method

Syntax

void setKeyStoreProvider (String *provider*)

setKeyStoreType(String) method

Syntax

void setKeyStoreType (String *storeType*)

setUseCertificate(boolean) method

Syntax

void setUseCertificate (boolean *useCert*)

_charset variable

Syntax

String _charset

_cipherProvider variable

Syntax

String _cipherProvider

_cipherTransform variable

Syntax

String _cipherTransform

_keyStoreAlias variable

Syntax

String _keyStoreAlias

_keyStoreAliasPassword variable

Syntax

String _keyStoreAliasPassword

_keyStoreLocation variable

Syntax

String _keyStoreLocation

_keyStorePassword variable

Syntax

String _keyStorePassword

_keyStoreProvider variable

Syntax

String _keyStoreProvider

_keyStoreType variable**Syntax**

```
String _keyStoreType
```

_useCert variable**Syntax**

```
String _useCert
```

Attributer interface

This is the interface which Attributer providers must implement.

Syntax

```
public interface Attributer
```

Derived classes

- *com.sybase.security.provider.AbstractAttributer* on page 151

Remarks

Providers implementing this interface may also optionally implement ProviderInfo interface to expose additional provider information. SecConfigurationValidatingProvider interface to validate the configuration properties for the provider before they are saved using the Admin API. SecProviderCapabilities interface to expose the capabilities of the provider. AttributerRegistration2 interface to support self registration functions.

attributeAuthenticatedSubject(Map< String, Object >, SecSubject) method

Given a SecSubject object, this method should populate it with the attributes.

Syntax

```
boolean attributeAuthenticatedSubject ( Map< String, Object >
context , SecSubject subject ) throws SecException
```

Parameters

- **subject** – subject that should be attributed.
- **context** – CSI context

Returns

true if the attributer contributed any attributes to the subject; false otherwise

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned false.

attributeResource(Map< String, Object >, SecResource, String) method

If the input resource ID is recognized by this attributer, populate the supplied SecResource with attributes and/or principals as appropriate.

Syntax

```
boolean attributeResource ( Map< String, Object > context ,  
SecResource resource , String id ) throws SecException
```

Parameters

- **resource** – add attributes to this object if the attributer knows anything about the given ID.
- **context** – CSI context
- **id** – the ID of the resource to retrieve

Returns

true if the resource object was recognized and modified

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned false.

Usage

The implementation of this method should additionally populate the resource with any resource types if it recognizes the ID.

This method is indirectly called when the CSI client calls SecContext.getResource(String).

attributeSubject(Map< String, Object >, SecSubject, String) method

Given a SecSubject object, this method should populate it with the attributes.

Syntax

```
boolean attributeSubject ( Map< String, Object > context ,  
SecSubject subject , String id ) throws SecException
```

Parameters

- **subject** – subject that should be attributed (this subject may be just a container to hold the attributes, it may not have been populated with an ID attribute).
- **context** – CSI context
- **id** – The ID to look up the subject when retrieving attributes.

Returns

true if the attributer contributed any attributes to the subject; false otherwise

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned false.

Usage

This method will be called to populate SecSubject attributes when a CSI client first attempts to retrieve attributes. The subject may not have been authenticated so an ID attribute may not be available in the subject. The supplied ID parameter should be used to identify the subject and retrieve the attribute values.

getSubjectAttributesForResource(Map< String, Object >, SecResource , SecSubject) method

Returns all attribute names and values associated with the given resource and subject.

Syntax

```
Map< String, String[]> getSubjectAttributesForResource ( Map<
String, Object > context , SecResource res , SecSubject subject )
throws SecException
```

Parameters

- **context** – CSI context
- **res** – the resource for which attributes are being retrieved
- **subject** – the subject to associate with the asset for attribute retrieval

Returns

map of attributes or null if there are no attributes

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty map.

Usage

This method is indirectly called when a CSI client calls `SecSubject.getAttributesForResource(SecResource)`.

listActions(Map< String, Object >, SecResource) method

Lists all actions that can be performed on the specified resource.

Syntax

```
List< Named > listActions ( Map< String, Object > context ,  
SecResource res ) throws SecException
```

Parameters

- **context** – CSI context
- **res** – the resource for which the actions should be listed.

Returns

a List containing Named objects describing each action.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

Usage

This method is indirectly called when the CSI client calls `SecContext.listActions(SecResource)`.

listEnvironmentAttributes(Map< String, Object >) method

Lists environment attributes that authorization providers may evaluate during `checkAccess` calls.

Syntax

```
List< Named > listEnvironmentAttributes ( Map< String, Object >  
context ) throws SecException
```

Parameters

- **context** – CSI context

Returns

a List containing Named objects describing each environment attribute.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

Usage

This method is indirectly called when the CSI client calls `SecContext.listEnvironmentAttributes()`.

listResources(Map< String, Object >, String) method

Lists all known resources.

Syntax

```
List< Named > listResources ( Map< String, Object > context ,
String resourceName ) throws SecException
```

Parameters

- **context** – CSI context
- **resourceTypeName** – limit results to this resource type. To request a list of resources of all known types, null is specified.

Returns

a List containing Named objects describing each resource.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

Usage

This method is indirectly called when the CSI client calls `SecContext.listResources()` and `SecContext.listResources(String resourceName)`.

listResourceTypes(Map< String, Object >) method

Lists all known resource types.

Syntax

```
List< Named > listResourceTypes ( Map< String, Object > context )
throws SecException
```

Parameters

- **context** – CSI context

Returns

a List containing Named objects describing each resource type. The "name" in the Named object is the value

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

Usage

This method is indirectly called when the CSI client calls `SecContext.listResourceTypes()`.

The return value for this method is a List of Named objects. The important value in the Named object is the "name", which is used as the `resourceTypeName` parameter in the `listResources` method.

listRoles(Map< String, Object >) method

Lists all known roles.

Syntax

```
List< Named > listRoles ( Map< String, Object > context ) throws  
SecException
```

Parameters

- **context** – CSI context

Returns

a List containing Named objects describing each role.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

Usage

This method is indirectly called when the CSI client calls `SecContext.listRoles()`.

listSubjectAttributesForResource(Map< String, Object >, SecResource , SecSubject) method

Lists all attribute names that may be associated with the resource/subject pair.

Syntax

```
List< Named > listSubjectAttributesForResource ( Map< String, Object > context , SecResource res , SecSubject subject ) throws SecException
```

Parameters

- **context** – CSI context
- **res** – the resource to return attribute info for.
- **subject** – the subject to associate with the asset for retrieving the names of the relevant attributes.

Returns

a List containing Named objects describing each resource attribute.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

Usage

This method is indirectly called when the CSI client calls SecSubject.listAttributesForResource(String).

AttributerRegistration interface

This is the interface to support self registration functions, which Attributer providers can optionally implement.

Syntax

```
public interface AttributerRegistration
```

createSubject(SecSubject) method

This method will be called when commitModification method is called on a subject object retrieved by SecContextFactory.newSubject() Attributer can simply return false if it doesn't support this method.

Syntax

```
boolean createSubject ( SecSubject subject ) throws SecException
```

modifySubject(Map< String, Object >, SecSubject , Attributed) method

This method will be called when commitModification method is called on a subject object retrieved through a SecContext instance.

Syntax

```
boolean modifySubject (Map< String, Object > context, SecSubject subject, Attributed modifiedAttributes) throws SecException
```

Parameters

- **context** – the context map
- **subject** – the subject to be modified
- **modifiedAttributes** – modified subject attributes to be committed

Returns

boolean value. "true" indicates subject modification is committed successfully; "false" indicates failed subject modification

Usage

Attributer can simply return false if it doesn't support this method.

AttributerRegistration2 interface

This is the interface to support self registration functions, which Attributer providers can optionally implement.

Syntax

```
public interface AttributerRegistration2
```

createSubject(Map< String, Object >, SecSubject) method

This method will be called when commitModification method is called on a subject object retrieved by SecContextFactory.newSubject() Attributer can simply return false if it doesn't support this method.

Syntax

```
boolean createSubject (Map< String, Object > context, SecSubject subject) throws SecException
```

Parameters

- **context** –
- **subject** –

modifySubject(Map< String, Object >, SecSubject, Attributed) method

This method will be called when commitModification method is called on a subject object retrieved through a SecContext instance.

Syntax

```
boolean modifySubject (Map< String, Object > context, SecSubject
subject, Attributed modifiedAttributes) throws SecException
```

Parameters

- **context** – the context map
- **subject** – the subject to be modified
- **modifiedAttributes** – modified subject attributes to be committed

Returns

boolean value. "true" indicates subject modification is committed successfully; "false" indicates failed subject modification

Usage

Attributer can simply return false if it doesn't support this method.

AuditConst interface

Interface that defines the constants for audit record types.

Syntax

```
public interface AuditConst
```

***ACTION_ACCESS* variable**

Action to represent an access check.

Syntax

```
final String ACTION_ACCESS
```

***ACTION_ACTIVATE* variable**

Action to represent provider activation.

Syntax

```
final String ACTION_ACTIVATE
```

ACTION_AUTHENTICATION variable

Action to represent CSI core authentication (i.e., aggregating the provider authentication results according to the control flags).

Syntax

```
final String ACTION_AUTHENTICATION
```

ACTION_AUTHENTICATION_PROVIDER variable

Action to represent provider authentication.

Syntax

```
final String ACTION_AUTHENTICATION_PROVIDER
```

ACTION_AUTHORIZATION_AUTHORIZATION variable

Action to represent CSI core authorization check.

Syntax

```
final String ACTION_AUTHORIZATION_AUTHORIZATION
```

ACTION_AUTHORIZATION_RESOURCE variable

Action to represent CSI core resource access check.

Syntax

```
final String ACTION_AUTHORIZATION_RESOURCE
```

ACTION_AUTHORIZATION_RESOURCE_PROVIDER variable

Action to represent provider resource access check.

Syntax

```
final String ACTION_AUTHORIZATION_RESOURCE_PROVIDER
```

ACTION_AUTHORIZATION_ROLE variable

Action to represent CSI core role based authorization check.

Syntax

```
final String ACTION_AUTHORIZATION_ROLE
```

ACTION_AUTHORIZATION_ROLE_PROVIDER variable

Action to represent provider role based authorization check.

Syntax

```
final String ACTION_AUTHORIZATION_ROLE_PROVIDER
```

ACTION_CREATE variable

Action to represent object creation in CSI core.

Syntax

```
final String ACTION_CREATE
```

ACTION_CREATE_CIPHER variable

Action to represent a cipher creation.

Syntax

```
final String ACTION_CREATE_CIPHER
```

ACTION_CREATE_DIGEST variable

Action to represent a digest creation.

Syntax

```
final String ACTION_CREATE_DIGEST
```

ACTION_CREATE_PROVIDER variable

Action to represent object creation in a provider.

Syntax

```
final String ACTION_CREATE_PROVIDER
```

ACTION_CREATE_SIGNATURE variable

Action to represent a signature creation.

Syntax

```
final String ACTION_CREATE_SIGNATURE
```

ACTION_LOGOUT variable

Action to represent logout.

Syntax

```
final String ACTION_LOGOUT
```

ACTION_MODIFY variable

Action to represent object modification in CSI core.

Syntax

```
final String ACTION_MODIFY
```

ACTION_MODIFY_PROVIDER variable

Action to represent object modification in a provider.

Syntax

```
final String ACTION_MODIFY_PROVIDER
```

ATTRIBUTE_ALTERNATE_SUBJECT variable

Audit record attribute for alternate subject specified in the authorization check.

Syntax

```
final String ATTRIBUTE_ALTERNATE_SUBJECT
```

ATTRIBUTE_AUTHORIZATION_ACTION variable

Audit record attribute for action specified in the access check.

Syntax

```
final String ATTRIBUTE_AUTHORIZATION_ACTION
```

ATTRIBUTE_AUTHORIZATION_RESOURCE variable

Audit record attribute for resource specified in the access check.

Syntax

```
final String ATTRIBUTE_AUTHORIZATION_RESOURCE
```

ATTRIBUTE_CONTEXT_ID variable

Audit record attribute for context ID.

Syntax

```
final String ATTRIBUTE_CONTEXT_ID
```

ATTRIBUTE_CONTROL_FLAG variable

Audit record attribute representing the control flag for a provider (typically included in the log record for provider activation).

Syntax

```
final String ATTRIBUTE_CONTROL_FLAG
```

ATTRIBUTE_CSI_REQUEST_ID variable

Audit record attribute representing the CSI request that generated the audit record.

Syntax

```
final String ATTRIBUTE_CSI_REQUEST_ID
```

Remarks

This attribute ties multiple audit records generated in a single CSI request.

ATTRIBUTE_FAILURE_REASON variable

Audit record attribute representing the failure reason.

Syntax

```
final String ATTRIBUTE_FAILURE_REASON
```

ATTRIBUTE_PROVIDER_ID variable

Audit record attribute representing the ID of the provider that generated the audit record.

Syntax

```
final String ATTRIBUTE_PROVIDER_ID
```

ATTRIBUTE_ROLE_NAME variable

Audit record attribute for role name (typically used in audit record for role check).

Syntax

```
final String ATTRIBUTE_ROLE_NAME
```

ATTRIBUTE_ROLE_SCOPE variable

Audit record attribute for role scope (typically used in audit record for role check).

Syntax

```
final String ATTRIBUTE_ROLE_SCOPE
```

ATTRIBUTE_SUBJECT_ID variable

Audit record attribute for subject ID.

Syntax

```
final String ATTRIBUTE_SUBJECT_ID
```

ATTRIBUTE_SUPPLIED_CREDENTIALS variable

Audit record attribute for logging credentials supplied for authentication.

Syntax

```
final String ATTRIBUTE_SUPPLIED_CREDENTIALS
```

Remarks

This does not include sensitive information such as passwords but only includes username or subjectDN from the certificate

CLIENT_RESOURCECLASS_PREFIX variable

Prefix added to the audit resource class name for audit records generated by CSI client applications.

Syntax

```
final String CLIENT_RESOURCECLASS_PREFIX
```

CORE_RESOURCECLASS_PREFIX variable

Prefix added to the audit resource class name for audit records generated by CSI core.

Syntax

```
final String CORE_RESOURCECLASS_PREFIX
```

PROVIDER_RESOURCECLASS_PREFIX variable

Prefix added to the audit resource class name for audit records generated by the configured providers.

Syntax

```
final String PROVIDER_RESOURCECLASS_PREFIX
```

RESOURCECLASS_CORE_PROFILE variable

Resource class name used in audit record for actions on a profile.

Syntax

```
final String RESOURCECLASS_CORE_PROFILE
```

RESOURCECLASS_CORE_PROVIDER variable

Resource class name used in audit record for actions on a provider.

Syntax

```
final String RESOURCECLASS_CORE_PROVIDER
```

RESOURCECLASS_CORE_SUBJECT variable

Resource class name used in audit record for actions on a subject.

Syntax

```
final String RESOURCECLASS_CORE_SUBJECT
```

AuditDestination interface

The AuditDestination is the destination to which the audit information should be logged.

Syntax

```
public interface AuditDestination
```


Derived classes

- *com.sybase.security.core.FileAuditDestination* on page 77

Remarks

A simple audit destination might write an audit record to a file. All audit destinations have the ability to use an audit formatter, although they may choose to ignore this if they do their own formatting. Audit destinations may choose to initialize a default formatter if they wish. Audit destinations must be thread-safe, unlike other CSI providers. CSI guarantees that each audit destination will be created only once per factory instance. However, it may be desirable for an audit destination to be aware of other instances of itself instantiated in the same VM and ensure that these instances share access to common resources such as audit files appropriately. Audit destinations may also add additional attributes to the audit record, such as a sequential record ID to guard against log tampering or signature attributes that can be later verified to check for audit record tampering.

audit(String, String, String, Map< String, Object >, Decision) method

Generates and logs an audit record with the specified audit information.

Syntax

```
void audit ( String resourceClass , String resourceID , String action ,
Map< String, Object > attributes , Decision decision ) throws
SecException
```

Parameters

- **resourceClass** – the scope used when processing the resource ID.
- **resourceID** – the object on which an operation is being performed.
- **action** – the operation that is being performed.
- **attributes** – A map of attributes associated with the audit record.
- **decision** – the result of the security check.

Exceptions

- **SecException class** – If an error occurs while issuing the audit request.

setFormatter(AuditFormatter) method

CSI will call this function on the destination in order to suggest what formatter it should use.

Syntax

```
void setFormatter ( AuditFormatter formatter )
```

Parameters

- **formatter** – the audit formatter to set

Usage

The destination may ignore this suggestion.

AuditFilter interface

An audit filter may be configured by the administrator to filter out undesired events and only log the events that match the specified filter.

Syntax

```
public interface AuditFilter
```

Derived classes

- *com.sybase.security.core.DefaultAuditFilter* on page 74

Remarks

The audit filter provider decides what audit records will actually be supplied to the audit destination. This gives applications a way to avoid effort spent collecting audit data when the record won't be audited anyway.

isAuditEnabled(String, String, Decision) method

This method will be called to query if an audit record with the specified attributes should be logged to the audit destination.

Syntax

```
boolean isAuditEnabled ( String resourceClass , String action ,  
Decision decision ) throws SecException
```

Parameters

- **resourceClass** – the scope used when processing the resource ID.
- **action** – the operation that is being performed.
- **decision** – the result of the security check.

Returns

the decision on whether or not an audit record with the specified attributes should be logged to the audit destination.

Exceptions

- **SecException class** – generated when an error is encountered while performing the audit record query.

AuditFormatter interface

The audit formatter is used to format an audit record from its component parts.

Syntax

```
public interface AuditFormatter
```

Derived classes

- *com.sybase.security.core.XmlAuditFormatter* on page 130

Remarks

An example might be an XML audit formatter. An audit formatter will be supplied to the active audit destination upon initialization. The audit destination can use this formatter or not, depending on its needs.

format(String, String, String, Map< String, Object >, Decision) method

Retrieve the string format of the given audit information.

Syntax

```
String format ( String resourceClass , String resourceID , String action ,  
Map< String, Object > attributes , Decision decision ) throws  
SecException
```

Parameters

- **resourceClass** – the scope used when processing the resource ID.
- **resourceID** – the object on which an operation is being performed.
- **action** – the operation that is being performed.
- **attributes** – A map of attributes associated with the audit record.
- **decision** – the result of the security check.

Returns

a string representation of the given audit information.

Exceptions

- **SecException class** – when a error is encountered during audit record formatting.

getFooter() method

Retrieves a footer string that should be written at the end of a file-based audit log when closing out an audit log file permanently.

Syntax

```
String getFooter ()
```

Returns

the footer string for file based audits.

getHeader() method

Retrieves a header string that should be written for file-based audit logs.

Syntax

```
String getHeader ()
```

Returns

the header string for file based audits.

AuditToken interface

An interface that is used to tag audit token classes.

Syntax

```
public interface AuditToken
```

getID() method

If the audit token is associated with a provider, this method returns the unique provider ID.

Syntax

```
String getID ()
```

Returns

the unique provider ID associated with the audit token.

AuthenticationFailureWarningImpl class

Simple implementation for AuthenticationFailureWarning.

Syntax

```
public class AuthenticationFailureWarningImpl
```

AuthenticationFailureWarningImpl() constructor

Default constructor.

Syntax

```
AuthenticationFailureWarningImpl ( )
```

AuthenticationFailureWarningImpl(int) constructor

Creates an instance of `AuthenticationFailureWarningImpl` with the specified failure code.

Syntax

```
AuthenticationFailureWarningImpl ( int code )
```

Parameters

- **code** – authentication failure code

AuthenticationFailureWarningImpl(int, String) constructor

Creates an instance of `AuthenticationFailureWarningImpl` with the specified failure code and message.

Syntax

```
AuthenticationFailureWarningImpl ( int code , String msg )
```

Parameters

- **code** – authentication failure code
- **msg** – message

AuthenticationFailureWarningImpl(String) constructor

Creates an instance of `AuthenticationFailureWarningImpl` with the specified message.

Syntax

```
AuthenticationFailureWarningImpl ( String msg )
```

Parameters

- **msg** – message

getFailureCode() method

Returns the authentication failure code set.

Syntax

```
int getFailureCode ( )
```

Returns

the integer failure code associated with the authentication failure.

getMessage() method

Returns the message indicating the authentication failure reason.

Syntax

```
String getMessage ()
```

Returns

the message indicative of the authentication failure.

_code variable

Authentication failure code.

Syntax

```
int _code
```

Authorizer interface

This is the required interface for authorization providers.

Syntax

```
public interface Authorizer
```

Derived classes

- *com.sybase.security.provider.AbstractAuthorizer* on page 155

Remarks

Providers implementing this interface may also optionally implement ProviderInfo interface to expose additional provider information. SecConfigurationValidatingProvider interface to validate the configuration properties for the provider before they are saved using the Admin API. SecProviderCapabilities interface to expose the capabilities of the provider.

checkAccess(Map< String, Object >, String, SecResource , SecSubject , SecEnvironment) method

Issues a decision for the check access request for this specified subject to perform the specified action on the specified resource under the specified environmental conditions.

Syntax

```
int checkAccess ( Map< String, Object > context , String action ,  
SecResource resource , SecSubject subject , SecEnvironment env )  
throws SecException
```

Parameters

- **context** – runtime information relevant to this security context.
- **action** – the action being considered
- **resource** – the target of that action
- **subject** – who is performing that action
- **env** – the circumstances under which that action is being performed.

Returns

one of the VOTE_* constants from Const

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned ABSTAIN.

checkRole(Map< String, Object >, String, SecSubject) method

Checks if the specified user has the specified role.

Syntax

```
int checkRole ( Map< String, Object > context , String roleName ,
SecSubject subject ) throws SecException
```

Parameters

- **context** – runtime information relevant to this security context.
- **roleName** – the role to check
- **subject** – the subject for whom to check the role.

Returns

one of the VOTE_* constants from Const

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned ABSTAIN.

BasicNamed class

This class provides an implementation for Named.

Syntax

```
public class BasicNamed
```

Comparator< T extends Named > class

A comparator that works for all Named objects (not just BasicNamed).

Syntax

```
public class Comparator< T extends Named >
```

Comparator() method

Syntax

```
Comparator ()
```

compare(T, T) method

A comparator that works for all Named objects (not just BasicNamed).

Syntax

```
int compare ( T t1 , T t2 )
```

Parameters

- **t1** – the first named object to compare.
- **t2** – the second named object to compare.

Returns

an integer representing the result of the comparison (0 indicating identical named objects).

BasicNamed(String, String, String) constructor

Constructor.

Syntax

```
BasicNamed ( String name , String ID , String description )
```

Parameters

- **name** – the human readable name of the object
- **ID** – the unique identifier of the object
- **description** – the description of the object

equals(Object) method

Method to check for equality of BasicNamed objects based on ID.

Syntax

```
boolean equals ( Object o )
```


Returns

boolean value if the passed in Object's ID is identical to that of the current instance.

getDescription() method

Gets the description of this object.

Syntax

```
String getDescription ()
```

Returns

a description of this Named object

getID() method

Gets the identifier of this object.

Syntax

```
String getID ()
```

Returns

the unique identifier for this object

getName() method

Gets the name of this object.

Syntax

```
String getName ()
```

Returns

the name of this object

hashCode() method

Method to return the hashCode of the ID.

Syntax

```
int hashCode ()
```

Returns

an integer representing the hashCode of the ID.

toString() method

Method to return the serialized string representation of the object.

Syntax

```
String toString ()
```

Returns

a string representing the serialized object.

BasicSecIDPrincipal class

Provides a simple Principal base implementation that implements SecIDPrincipal.

Syntax

```
public class BasicSecIDPrincipal
```

Derived classes

- *com.sybase.security.core.CertificateIDPrincipal* on page 64

BasicSecIDPrincipal(String) constructor

Protected Constructor.

Syntax

```
BasicSecIDPrincipal ( String id )
```

Parameters

- **id** – the ID of the principal that should be returned by the getName() method.

equals(Object) method

Method to check for equality of BasicSecIDPrincipal objects based on ID.

Syntax

```
boolean equals ( Object o )
```

Returns

boolean value if the passed in Object's ID is identical to that of the current instance.

getName() method

Retrieves the ID of the principal.

Syntax

```
String getName ()
```

Returns

the ID of the principal

hashCode() method

Method to return the hashCode of the ID.

Syntax

```
int hashCode ()
```

Returns

an integer representing the hashCode of the ID.

_id variable

The ID of the principal.

Syntax

```
final String _id
```

BasicSecNamePrincipal class

Provides a simple Principal base implementation that implements SecNamePrincipal.

Syntax

```
public class BasicSecNamePrincipal
```

Derived classes

- *com.sybase.security.core.ClientValuePropagatingLoginModule.CVPLMUserPrincipal* on page 71
- *com.sybase.security.core.NoSecLoginModule.Principal* on page 101
- *com.sybase.security.core.PreConfiguredUserLoginModule.PreConfigUserPrincipal* on page 113

BasicSecNamePrincipal(String) constructor

Protected Constructor.

Syntax

```
BasicSecNamePrincipal ( String name )
```

Parameters

- **name** – the name of the principal that should be returned by the getName() method.

equals(Object) method

Method to check for equality of BasicSecNamePrincipal objects based on name.

Syntax

```
boolean equals ( Object o )
```

Returns

boolean value if the passed in Object's name is identical to that of the current instance.

getName() method

Retrieves the name of the principal.

Syntax

```
String getName ()
```

Returns

the name of the principal

hashCode() method

Method to return the hashCode of the ID.

Syntax

```
int hashCode ()
```

Returns

an integer representing the hashCode of the Object's name.

_name variable

The name of the principal.

Syntax

```
final String _name
```

Bootstrap class

Provides an object bootstrapping function.

Syntax

```
public class Bootstrap
```

Derived classes

- *com.sybase.security.provider.AbstractBootstrapConfiguration* on page 156

Remarks

This sets JavaBean-style properties from a resource like a File, Properties object or URL. Some special characteristics apply, such as when a File property is set it will be resolved relative to the bootstrap properties source (if possible).

bootstrap(File) constructor

Bootstrap the object instance using data from the supplied File.

Syntax

```
void bootstrap ( File f ) throws IOException, SecException
```

bootstrap(InputStream) constructor

Bootstrap the object instance using data from the supplied InputStream.

Syntax

```
void bootstrap ( InputStream is ) throws IOException, SecException
```

Bootstrap(Object) constructor

The object upon which the bootstrap will be performed is supplied as the only argument of the constructor.

Syntax

```
Bootstrap ( Object o )
```

Parameters

- **o** – if the supplied value is null, the operations will be performed on "this"

bootstrap(Properties) constructor

Bootstrap the object instance using data from the supplied Properties.

Syntax

```
void bootstrap ( Properties props ) throws SecException
```

Exceptions

- **SecException class** – if any errors occurred setting properties. If an error occurred setting a single property, the error is saved and the next property is processed. This ensures that an invalid property name/value does not necessarily break the entire configuration bootstrapping process.

bootstrap(URL) constructor

Bootstrap the object instance using data from the supplied URL.

Syntax

```
void bootstrap ( URL url ) throws IOException, SecException
```

resolveBootstrapResourceAsFile(String) method

Resolve a filename against the bootstrap filename.

Syntax

```
File resolveBootstrapResourceAsFile ( String value )
```

resolveBootstrapResourceAsURL(String) method

Resolve a partial URL against the bootstrap URL.

Syntax

```
URL resolveBootstrapResourceAsURL ( String value ) throws  
ResolveException
```

setBootstrapProperty(String, String) method

called to set a name/value attribute

Syntax

```
void setBootstrapProperty ( String property , String value ) throws  
SecException
```

_bootstrapBaseFile variable

The file from which the bootstrap information was read.

Syntax

```
File _bootstrapBaseFile
```

_bootstrapBaseURL variable

The URL from which the bootstrap information was read.

Syntax

```
URL _bootstrapBaseURL
```

_object variable

The object on which the bootstrap operation will occur.

Syntax

```
final Object _object
```

CertificateValidation interface

Interface that provides certificate validation functionality.

Syntax

```
public interface CertificateValidation
```

isValidCertificateChain(Certificate[]) method

Validates the specified certificate chain.

Syntax

```
boolean isValidCertificateChain ( Certificate[] certs ) throws  
CertificateValidationException
```

Parameters

- **certs** – certificate chain to be validated.

Returns

true if the certificate chain is valid; false otherwise.

Exceptions

- **CertificateValidationException class** – wraps any exceptions encountered while validating the certificate chain in a CertificateValidationException

CertificateValidationException class

An exception thrown by the Certificate Validation module.

Syntax

```
public class CertificateValidationException
```

CertificateValidationException(String) constructor

Constructor for the CertificateValidationException.

Syntax

```
CertificateValidationException ( String msg )
```

Parameters

- **msg** – error encountered during Certificate Validation.

ConfigurationParser class

Utility class that provides a tool to parse the internal configuration from the Map<String, String> format into Provider configuration classes for easy access to the configuration properties for the various providers.

Syntax

```
public class ConfigurationParser
```

AuditProviderConfiguration class

Helper class that holds audit filter, formatter and destination configuration options.

Syntax

```
public class AuditProviderConfiguration
```

AuditProviderConfiguration(ProviderConfiguration , ProviderConfiguration , ProviderConfiguration) constructor

Syntax

```
AuditProviderConfiguration ( ProviderConfiguration filter ,  
ProviderConfiguration formatter , ProviderConfiguration  
destination )
```

_destination variable

Audit destination configuration.

Syntax

```
final ProviderConfiguration _destination
```

_filter variable

Audit Filter configuration.

Syntax

```
final ProviderConfiguration _filter
```

_formatter variable

Syntax

```
final ProviderConfiguration _formatter
```


ConfigurationProperties class

Used to hold the configuration properties.

Syntax

```
public class ConfigurationProperties
```

```
ConfigurationProperties(List< AuditProviderConfiguration >, List<
ProviderConfiguration >, List< ProviderConfiguration >, List< ProviderConfiguration
>, List< ProviderConfiguration >, List< ProviderConfiguration >, List<
ProviderConfiguration >, Map< String, String >) constructor
```

Syntax

```
ConfigurationProperties ( List< AuditProviderConfiguration >
auditConfig , List< ProviderConfiguration > login , List<
ProviderConfiguration > auth , List< ProviderConfiguration > attr ,
List< ProviderConfiguration > roleMappers , List<
ProviderConfiguration > dataServiceProviders , List<
ProviderConfiguration > profilers , Map< String, String >
csiCfgOptions )
```

_attributers variable

Attributer configuration list.

Syntax

```
final List< ProviderConfiguration > _attributers
```

_auditConfigurations variable

Audit configuration list.

Syntax

```
final List< AuditProviderConfiguration > _auditConfigurations
```

_authorizers variable

Authorization provider configuration list.

Syntax

```
final List< ProviderConfiguration > _authorizers
```

_csiConfigOptions variable

configuration specific options

Syntax

```
final Map< String, String > _csiConfigOptions
```

_dataServiceProviders variable

Secure data service provider configuration list.

Syntax

```
final List< ProviderConfiguration > _dataServiceProviders
```

_loginModuleEntries variable

LoginModule configuration list.

Syntax

```
final List< ProviderConfiguration > _loginModuleEntries
```

_profilers variable

Profiler configuration list.

Syntax

```
final List< ProviderConfiguration > _profilers
```

_roleMappers variable

Role Mapper configuration list.

Syntax

```
final List< ProviderConfiguration > _roleMappers
```

ProviderConfiguration class

CSI Provider configuration.

Syntax

```
public class ProviderConfiguration
```

ProviderConfiguration(String, Map< String, String >, Map< String, String >, Set< String >, String, String) constructor

Constructor.

Syntax

```
ProviderConfiguration (String clz, Map< String, String > options,  
Map< String, String > preservedOptions, Set< String > encryptedOptions,  
String controlFlag, String id)
```

Parameters

- **clz** – Classname of the provider

- **options** – Map of options to be supplied to the provider at init time.
- **controlFlag** – control flag configured for the provider (if applicable)

getControlFlag() method

Returns the control flag configured for the provider.

Syntax

```
String getControlFlag ()
```

Returns

control flag value configured for the provider

getId() method

Returns the provider id.

Syntax

```
String getId ()
```

Returns

provider id

Usage

This is guaranteed to be set. If it is not configured, the parser automatically adds one.

getOptionsToEncrypt() method

Returns the configuration properties that should be encrypted.

Syntax

```
Set< String > getOptionsToEncrypt ()
```

Returns

map containing the configuration property names that should be encrypted.

getProviderClass() method

Returns the provider class name.

Syntax

```
String getProviderClass ()
```

Returns

provider class name

getProviderOptions() method

Returns the configuration properties.

Syntax

```
Map< String, String >getProviderOptions ()
```

Returns

configuration properties map

ConfigurationParser() constructor

Syntax

```
ConfigurationParser ()
```

getUsePreservedValues() method

Syntax

```
boolean getUsePreservedValues ()
```

parseConfiguration(Map< String, String >) method

Parse the internal configuration properties and build the provider lists.

Syntax

```
ConfigurationProperties parseConfiguration ( Map< String,  
String > configurationMap )
```

Parameters

- **configurationMap** – internal configuration map

Returns

parsed configuration

setUsePreservedValues(boolean) method

Syntax

```
void setUsePreservedValues ( boolean usePreservedValues )
```

ConfigurationProblem class

Represents the configuration problems discovered during configuration validation.

Syntax

```
public class ConfigurationProblem
```

Remarks

If the problem is related to a specific configuration property the error message is associated with the property. If the problem is associated with a combination of properties or is about a missing property, the error message is associated with the provider.

ConfigurationProblem(Severity , String, String) constructor

Constructs a ConfigurationProblem with the given problem description associated with the specified configuration property and the specified severity.

Syntax

```
ConfigurationProblem ( Severity severity , String  
configurationProperty , String description )
```

Parameters

- **severity** – Severity of the problem.
- **configurationProperty** – provider configuration property that resulted in the generation of the problem.
- **description** – problem description.

Usage

If the problem is associated with the provider or involves one or more configuration properties then the configurationProperty argument should be null.

getConfigurationProperty() method

Retrieve an invalid configuration Property.

Syntax

```
String getConfigurationProperty ()
```

Returns

a string representing the invalid configuration property.

Usage

In the event of a single invalid configuration property, the property is returned. If the configuration problem is associated with a combination of keys, the return value is set to NULL.

getProblemDescription() method

Error message associated with the validation of the property.

Syntax

```
String getProblemDescription ()
```

Returns

the description of the configuration problem.

getSeverity() method

Returns the configuration problem severity.

Syntax

```
Severity getSeverity ()
```

Returns

severity of the problem being reported.

Severity() enumeration

An enumeration representing problem severity.

Enum Constant Summary

- **FATAL** –
- **ERROR** –
- **WARNING** –
- **INFO** –

ConfigurationValidationService interface

Validates the supplied internal CSI configuration according to the stored provider metadata and lets the providers themselves validate the supplied configuration by performing runtime checks where possible.

Syntax

```
public interface ConfigurationValidationService
```

validateConfiguration(Map< String, String >) method

Validates internal configuration and returns the configuration problems.

Syntax

```
Map< String, List< ConfigurationProblem > >
validateConfiguration (Map< String, String > internalConfiguration )
throws SecException
```

Parameters

- **internalConfiguration** – The internal CSI configuration map

Returns

Map containing the set of all validation errors associated with each provider ID. The key should be the provider ID and the corresponding value should be a list of ConfigurationProblem's associated with that provider.

ContextRetriever interface

Authors wishing to create context retrievers must implement this interface in a class with a public, no argument constructor.

Syntax

```
public interface ContextRetriever
```

Derived classes

- *com.sybase.security.provider.AbstractPrincipalContextRetriever* on page 173
- *com.sybase.security.provider.ContextRetriever2* on page 218

retrieveContext(Object) method

Context retriever providers must implement this method to allow for retrieval of SecContext objects stored in the environment.

Syntax

```
SecContext retrieveContext ( Object environmentObject ) throws
SecException
```

Parameters

- **environmentObject** –

Returns

the SecContext object or null if none found

Exceptions

- **SecException class** –

ContextRetriever2 interface

Extended version of context retriever interface that allows the factory that was used to retrieve the context to be supplied.

Syntax

```
public interface ContextRetriever2
```

Remarks

Indeed, immediately after instantiation the factory will call the setFactory method on all context retrievers that implement this interface.

setFactory(SecContextFactory) method

Syntax

```
void setFactory ( SecContextFactory factory )
```

ContextRetrieverPrincipal interface

Principal interface designed to be paired with AbstractPrincipalContextRetriever and its subclasses.

Syntax

```
public interface ContextRetrieverPrincipal
```

Remarks

Principals that have the capability of exposing a stored security context can do so by implementing this interface.

getContext() method

Retrieve the security context associated with this principal.

Syntax

```
SecContext getContext ()
```

DigitalSignature interface

Interface to be implemented by secure data service providers that support digital signatures.

Syntax

```
public interface DigitalSignature
```


Derived classes

- *com.sybase.security.core.JCESecureDataServices* on page 85

getSignature(Map< String, Object >, SecProfile , String) method

Syntax

Signature getSignature (Map< String, Object > *context* ,
SecProfile *profile* , String *operation*) throws SecException

Parameters

- **context** – CSI context
- **profile** – SecProfile object containing the properties to create and initialize a Signature instance.
- **operation** – specifies the operation for which to initialize the Signature object. Valid values are Const.OP_SIGN and Const.OP_VERIFY

Returns

Signature object created based on the properties specified in the profile Map and initialized for the specified operation. Returns null if unable to handle the profile.

Exceptions

- **SecException class** –

EncryptionTools class

This class performs specialized encryption/decryption functions for relatively short strings such as passwords.

Syntax

```
public class EncryptionTools
```

Remarks

All the set*** methods should be invoked first to provide all options required to instantiate a Cipher instance. Next init() should be called to perform necessary initialization. Finally, encrypt()/decrypt() can be invoked to accomplish expected cipher operation. Encryption result will be encoded using base64. UTF-8 will be used for encoding between String and byte array conversion.

EncryptionTools() constructor

Constructor.

Syntax

```
EncryptionTools ()
```

Usage

All the set*** methods should be invoked to provide the options required to instantiate a Cipher instance before invoking init()/encrypt()/decrypt()

decrypt(String) method

Decrypts the string using the configured key/certificate and returns the plain text.

Syntax

```
String decrypt ( String ciphertext ) throws SecException
```

Parameters

- **ciphertext** – String to be decrypted.

Returns

decrypted string

Exceptions

- **SecException class** –

decryptWithFixedKey(String) method

Decrypts the string using the fixed key and returns the plain text.

Syntax

```
String decryptWithFixedKey ( String cipherText ) throws SecException
```

Parameters

- **cipherText** – String to be decrypted.

Returns

decrypted string

Exceptions

- **SecException class** –

encrypt(String) method

Encrypts the specified string using the configured key/certificate and returns the base64 encoded encrypted string.

Syntax

String encrypt (String *plaintext*) throws SecException

Parameters

- **plaintext** – String to be encrypted using the specified key/certificate.

Returns

Base64 encoded encrypted string

Exceptions

- **SecException class** –

Usage

Use the decrypt method to decrypt the string.

encryptWithFixedKey(String) method

Encrypts the specified string using the fixed key and returns the base64 encoded encrypted string.

Syntax

String encryptWithFixedKey (String *plaintext*) throws SecException

Parameters

- **plaintext** – String to be encrypted using the fixed key.

Returns

Base64 encoded encrypted string

Exceptions

- **SecException class** –

Usage

Use the `decryptWithFixedKey` method to decrypt the string.

init() method

Creates and initializes the Cipher instance for use in encryption/decryption.

Syntax

```
void init () throws SecException
```

Exceptions

- **SecException class** – if any of the properties to instantiate a Cipher instance are not initialized or they are initialized with invalid values.

Usage

All the `set***` methods should be invoked to provide the options required to instantiate a Cipher instance before invoking this method.

setCharset(String) method

Syntax

```
void setCharset ( String charset )
```

Parameters

- **charset** –

setCipherProvider(String) method

Syntax

```
void setCipherProvider ( String provider )
```

setCipherTransform(String) method

Syntax

```
void setCipherTransform ( String transform )
```

setEncryptWithLegacyMode(boolean) method

Syntax

```
void setEncryptWithLegacyMode ( boolean legacyMode )
```

Parameters

- **legacyMode** –

setKeyStoreAlias(String) method

Syntax

```
void setKeyStoreAlias ( String storeAlias )
```

setKeyStoreAliasPassword(String) method

Syntax

```
void setKeyStoreAliasPassword ( String aliasPasswd )
```

setKeyStoreLocation(String) method

Syntax

```
void setKeyStoreLocation ( String storeLocation )
```

setKeyStorePassword(String) method

Syntax

```
void setKeyStorePassword ( String storePasswd )
```

setKeyStoreProvider(String) method

Syntax

```
void setKeyStoreProvider ( String provider )
```

setKeyStoreType(String) method

Syntax

```
void setKeyStoreType ( String storeType )
```

setPaddingMaximum(int) method

Syntax

```
void setPaddingMaximum ( int maximum )
```

setPaddingMinimum(int) method

Syntax

```
void setPaddingMinimum ( int minimum )
```

setRandom(SecureRandom) method

Syntax

void setRandom (SecureRandom *random*)

setUseCertKey(String) method

Syntax

void setUseCertKey (String *useCert*)

Parameters

- **useCert** –

ExternalConfigurationService interface

A service that will export the CSI configuration to a specified XML format.

Syntax

```
public interface ExternalConfigurationService
```

Remarks

A service can be written to support one or more XML schemas. These are simply well-established identifiers, but generally should be the XML schema URI. An example of this is <http://www.sybase.com/uep/1.2/configuration>. Service implementations must contain a public, no argument constructor.

supportsSchema(String) method

Returns false if the specified schema ID is not supported.

Syntax

```
boolean supportsSchema ( String primarySchema )
```

Parameters

- **primarySchema** – the schema for which support will be tested

Returns

whether or not the schema is supported by this config service

translateFromInternal(String, Map< String, String >, Map< String, List< ConfigurationProblem >>) method

Translate from Internal CSI configuration to an external configuration format.

Syntax

```
Reader translateFromInternal ( String primarySchema , Map< String,
String > internalConfiguration , Map< String, List<
ConfigurationProblem >> validationProblems ) throws SecException
```

Parameters

- **primarySchema** – the primary schema to be used for exporting the configuration
- **internalConfiguration** – the internal CSI configuration to be translated to the specified format.
- **validationProblems** – Map containing the provider ID as key and the associated configuration problem collection as the value. These should be added to the return XML document.

Returns

A reader object that contains the configuration data and validation problems in the specified format.

Usage

Add the supplied validation problems to the external XML document.

translateToInternal(String, Reader) method

Translate from an external configuration format to the internal CSI configuration format.

Syntax

```
Map< String, String > translateToInternal ( String primarySchema ,
Reader externalConfigData ) throws SecException
```

Parameters

- **primarySchema** – schema used in the supplied XML document
- **externalConfigData** – A reader containing the external configuration data

Returns

A map containing the configuration data with keys and values as Strings (CSI internal configuration format)

FactoryRetriever interface

Concrete factory retrievers must implement this interface in a class with a public, no argument constructor.

Syntax

```
public interface FactoryRetriever
```

Derived classes

- *com.sybase.security.provider.AbstractFactoryRetriever* on page 158

retrieveFactory(Object) method

Factory retriever providers must implement this method to allow for retrieval of SecContextFactory instance stored in the environment.

Syntax

```
SecContextFactory retrieveFactory ( Object environmentObject ) throws  
SecException
```

Parameters

- **environmentObject** –

Returns

the SecContext object or null if none found

Exceptions

- **SecException class** –

MessageDigest interface

Interface to be implemented by secure data service providers that support message digest.

Syntax

```
public interface MessageDigest
```

Derived classes

- *com.sybase.security.core.JCESecureDataServices* on page 85

getMessageDigest(Map< String, Object >, SecProfile) method

Syntax

java.security.MessageDigest getMessageDigest (Map< String, Object > *context* , SecProfile *profile*) throws SecException

Parameters

- **context** – CSI context
- **profile** – SecProfile object containing the properties to create and initialize a MessageDigest instance.

Returns

MessageDigest object created based on the properties specified in the profile Map and initialized for the digest operation. Returns null if unable to handle the profile.

Exceptions

- **SecException class** –

NamedCredentialProvider interface

This is the interface that the providers can implement so that the configuration validation service can retrieve the names of the NamedCredential(s) the provider adds to an authenticated subject so that it can verify if they conflict with the credentials added by other providers.

Syntax

```
public interface NamedCredentialProvider
```

Derived classes

- *com.sybase.security.core.CertificateAuthenticationLoginModule* on page 58
- *com.sybase.security.core.ClientValuePropagatingLoginModule* on page 70

getCredentialNames(Map< String, String >) method

Returns the set of names associated with the NamedCredentials that the provider may add to an authenticated subject.

Syntax

```
Set< String > getCredentialNames ( Map< String, String > configuration )
```

Parameters

- **configuration** – provider configuration properties

Returns

the set of names associated with the NamedCredentials that the provider may add to an authenticated subject

Usage

This is used to detect conflicts among the configured providers when validating the configuration so as to avoid duplicate NamedCredentials added by multiple providers.

OptionMapHelper class

Provides some utility methods for retrieving options from a Map where there are default values and different data types involved.

Syntax

```
public class OptionMapHelper
```

Derived classes

- *com.sybase.security.core.XMLFileRoleMapper.RoleMapperConfig* on page 141

getBooleanOption(Map< String,?>, String, boolean) method

Utility method to retrieve a boolean value from a map in the and provide a default value if the key was not found set.

Syntax

```
boolean getBooleanOption ( Map< String, ?> props , String key ,  
boolean dflt )
```

Parameters

- **props** – the Map in which the property values reside
- **key** – property name
- **dflt** – default value if the property is not found.

Returns

the value from the Map or the default.

getDateOption(Map< String,?>, String, Date) method

Utility method to retrieve a Date value from a map and provide a default value if the key was not found set.

Syntax

```
Date getDateOption ( Map< String, ?> props , String key , Date date )
```

Parameters

- **props** – the Map in which the property values reside
- **key** – property name
- **date** – default value if the property is not found.

Returns

the value from the Map or the default.

getIntOption(Map< String,?>, String, int) method

Utility method to retrieve an integer value from a map in the and provide a default value if the key was not found set.

Syntax

```
int getIntOption ( Map< String, ?> props , String key , int dflt )
```

Parameters

- **props** – the Map in which the property values reside
- **key** – property name
- **dflt** – default value if the property is not found.

Returns

the value from the Map or the default.

getLongOption(Map< String,?>, String, long) method

Utility method to retrieve an long value from a map in the and provide a default value if the key was not found set.

Syntax

```
long getLongOption ( Map< String, ?> props , String key , long dflt )
```

Parameters

- **props** – the Map in which the property values reside

- **key** – property name
- **dflt** – default value if the property is not found.

Returns

the value from the Map or the default.

PasswordExpirationWarningImpl class

Simple implementation for PasswordExpirationWarning.

Syntax

```
public class PasswordExpirationWarningImpl
```

PasswordExpirationWarningImpl(Date) constructor

Constructor to create an instance of the PasswordExpirationWarningImpl with the specified expiration date.

Syntax

```
PasswordExpirationWarningImpl ( Date date )
```

Parameters

- **date** – Password expiration date. Should be set to null, if the password does not expire.

getExpiration() method

Returns the password expiration date.

Syntax

```
Date getExpiration ()
```

Returns

Password expiration date. Null if password does not expire.

Usage

Null is returned if password does not expire. If expiration date is not known, PasswordExpirationWarning is not returned.

getMessage() method

Returns the localized message associated with the warning.

Syntax

```
String getMessage ()
```

Returns

Message associated with the warning.

_date variable

Password expiration date.

Syntax

```
Date _date
```

Remarks

Set to null if the password does not expire

PrefixMap< T > class

Utility class that extends HashMap that expects keys of type String.

Syntax

```
public class PrefixMap< T >
```

Remarks

Allows the caller to set a prefix that is prepended to the keys in the map when accessing the entries in the map using the set/getPrefixProperty methods.

This class simplifies access to properties with lengthy package qualifiers. For example, to access the property "com.sybase.security.TestProperty" you could use a regular properties object but the property name is very long. If you use PrefixProperties you could instantiate the PrefixProperties class and initialize properties, then call setPrefix("com.sybase.security"). Then you can access properties within the package using the method getPrefixProperty("TestProperty");

clonePrefix(String) method

Returns a new PrefixMap containing only properties whose values started with the supplied prefix.

Syntax

```
PrefixMap< T > clonePrefix ( String prefix )
```

Parameters

- **prefix** – prefix of the properties that should be in the new PrefixMap

Returns

the new PrefixMap

Usage

In addition, the property names in the newly created object have the specified prefix removed.

getPrefix() method

Allows the user to retrieve the prefix to use at runtime.

Syntax

```
String getPrefix ()
```

Returns

the defined prefix for this object.

getPrefixProperty(String) method

see Map.get(.

Syntax

```
T getPrefixProperty ( String key )
```

Parameters

- **key** – the specified key before being prefixed.

Returns

the object value corresponding to the calculated prefixed key.

Usage

..) The prefix is prepended to the specified key before using it to retrieve the value from the Map.

setPrefix(String) method

Allows the user to specify the prefix to use at runtime.

Syntax

```
void setPrefix ( String prefix )
```

Parameters

- **prefix** – the new prefix (usually should include a trailing ".")

setPrefixProperty(String, T) method

Allows a user to set an entry in the Map using the prefix values.

Syntax

```
T setPrefixProperty ( String key , T value )
```

Parameters

- **key** – key that should be prepended with the prefix before using it to set the value in the Map.
- **value** – value to be set

Returns

the previous value associated with calculated prefixed key.

Profiler interface

This is the interface that Profile providers must implement.

Syntax

```
public interface Profiler
```

Derived classes

- *com.sybase.security.core.ProfilerImpl* on page 116
- *com.sybase.security.provider.AbstractProfiler* on page 175

Remarks

Providers implementing this interface may also optionally implement the ProviderInfo interface to expose additional provider information.

getProfile(Map< String, Object >, SecProfile) method

This method should populate the SecProfile object by calling the set methods on it .

Syntax

```
boolean getProfile ( Map< String, Object > context , SecProfile profile ) throws SecException
```

Parameters

- **context** – CSI context
- **profile** – profile object

Returns

true if the SecProfile is populated; false otherwise

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned false.

listProfiles(Map< String, Object >) method

Returns the list of all known profiles.

Syntax

```
List< Named > listProfiles ( Map< String, Object > context ) throws  
SecException
```

Parameters

- **context** – CSI context

Returns

List containing Named objects describing the profiles known to this profiler.

Exceptions

- **SecException class** – Thrown if an error occurs. The exception will be logged and after that it is as if the method returned an empty list.

ProviderConst interface

Constants that are useful in implementing provider interfaces.

Syntax

```
public interface ProviderConst
```

Derived classes

- *com.sybase.security.core.CertificateAuthenticationLoginModule* on page 58
- *com.sybase.security.core.CertificateValidationLoginModule* on page 85
- *com.sybase.security.core.JCESecureDataServices* on page 85
- *com.sybase.security.core.NoSecAttributer* on page 98
- *com.sybase.security.core.RoleCheckAuthorizer* on page 122
- *com.sybase.security.core.XMLFileRoleMapper* on page 141
- *com.sybase.security.provider.AbstractSecureFileConfiguration* on page 178
- *com.sybase.security.provider.EncryptionTools* on page 219

ALGORITHM_PARAMETERS variable**Syntax**

```
final String ALGORITHM_PARAMETERS
```

AUDIT_TOKEN_CONFIG_PROPERTY variable

Configuration property that supplies the audit token to all CSI providers.

Syntax

```
final String AUDIT_TOKEN_CONFIG_PROPERTY
```

CERT_VALIDATION_CRL_PREFIX variable

Prefix for the property specifying the crl URI.

Syntax

```
final String CERT_VALIDATION_CRL_PREFIX
```

Remarks

The later part contains the unique index followed by the URI property suffix

CERT_VALIDATION_DEF_ENABLE_REVOCATION_CHECKING variable

Default value for CERT_VALIDATION_ENABLE_REVOCATION_CHECKING.

Syntax

```
final String CERT_VALIDATION_DEF_ENABLE_REVOCATION_CHECKING
```

CERT_VALIDATION_DEF_VALIDATE_CERTPATH variable

Default value for CERT_VALIDATION_VALIDATE_CERTPATH.

Syntax

```
final String CERT_VALIDATION_DEF_VALIDATE_CERTPATH
```

CERT_VALIDATION_EFFECTIVE_DATE variable

When performing certificate validity checks, this is the date/time that is used for the check.

Syntax

```
final String CERT_VALIDATION_EFFECTIVE_DATE
```

Remarks

If this is not provided then the current date/time is used. The format of this is the XML schema date/time format: 2006-01-01T00:00:00Z (YYYY-MM-DDThh:mm:ssZ) Currently only the Zulu timezone is supported (Z).

CERT_VALIDATION_ENABLE_REVOCATION_CHECKING variable

Name of the property to enable OCSP revocation checking.

Syntax

```
final String CERT_VALIDATION_ENABLE_REVOCATION_CHECKING
```

Remarks

Note that this only enables revocation checking in the PKIXParameters. For revocation checking to take effect, OCSP checking should be enabled at the jvm level. Please refer to java security guide on the relevant properties to enable OCSP checking in the JVM.

CERT_VALIDATION_TRUSTED_CERTSTORE_LOCATION variable

The property to specify the keystore containing the trusted CA certificates. If none is specified then the trust store used by the JVM is used i.e., one defined by the system property `javax.net.ssl.trustStoreType`.

Syntax

```
final String CERT_VALIDATION_TRUSTED_CERTSTORE_LOCATION
```

Remarks

If the system property is not defined then `#{java.home}/lib/security/jssecacerts` or if that does not exist `#{java.home}/lib/security/cacerts`

CERT_VALIDATION_TRUSTED_CERTSTORE_PASSWORD variable

The property to specify the password for the keystore containing trusted CA certificates.

Syntax

```
final String CERT_VALIDATION_TRUSTED_CERTSTORE_PASSWORD
```

CERT_VALIDATION_TRUSTED_CERTSTORE_PROVIDER variable

The name of the property to specify the provider of keystore.

Syntax

```
final String CERT_VALIDATION_TRUSTED_CERTSTORE_PROVIDER
```

CERT_VALIDATION_TRUSTED_CERTSTORE_TYPE variable

The name of the property to specify the type of keystore.

Syntax

```
final String CERT_VALIDATION_TRUSTED_CERTSTORE_TYPE
```

CERT_VALIDATION_URI variable

Suffix for the property specifying the crl URI.

Syntax

```
final String CERT_VALIDATION_URI
```

Remarks

The earlier part contains the crl property prefix followed by an unique index

CERT_VALIDATION_VALIDATE_CERTPATH variable

Name of the property to enable certificate path validation.

Syntax

```
final String CERT_VALIDATION_VALIDATE_CERTPATH
```

CERTIFICATE_SHARED_KEY variable

The shared key to identify the certificate chain stored in shared map to be used when certificate authentication is enabled.

Syntax

```
final String CERTIFICATE_SHARED_KEY
```

CHECK_IMPERSONATION variable

Property to configure impersonation check.

Syntax

```
final String CHECK_IMPERSONATION
```

CONFIG_ATTRIBUTER variable

Provider type identifier used used to identify an attributer in internal configuration properties.

Syntax

```
final String CONFIG_ATTRIBUTER
```

CONFIG_ATTRIBUTER_BASE variable

Configuration property prefix to identify properties for a given attributer.

Syntax

```
final String CONFIG_ATTRIBUTER_BASE
```

CONFIG_AUDIT variable

Provider type identifier used to identify an audit provider in internal configuration properties.

Syntax

```
final String CONFIG_AUDIT
```

CONFIG_AUDIT_BASE variable

Configuration property prefix to identify properties for a given audit provider.

Syntax

```
final String CONFIG_AUDIT_BASE
```

CONFIG_AUDIT_DESTINATION variable

Configuration property prefix to identify properties for a given audit destination provider.

Syntax

```
final String CONFIG_AUDIT_DESTINATION
```

CONFIG_AUDIT_FILTER variable

Configuration property prefix to identify properties for a given audit filter provider.

Syntax

```
final String CONFIG_AUDIT_FILTER
```

CONFIG_AUDIT_FILTER_PROVIDER_DEFAULT variable

Default audit filter provider.

Syntax

```
final String CONFIG_AUDIT_FILTER_PROVIDER_DEFAULT
```

CONFIG_AUDIT_FORMATTER variable

Configuration property prefix to identify properties for a given audit formatter provider.

Syntax

```
final String CONFIG_AUDIT_FORMATTER
```

CONFIG_AUDIT_FORMATTER_PROVIDER_DEFAULT variable

Default audit formatter provider.

Syntax

```
final String CONFIG_AUDIT_FORMATTER_PROVIDER_DEFAULT
```

CONFIG_AUTHORIZER variable

Provider type identifier used used to identify an authorizer in internal configuration properties.

Syntax

```
final String CONFIG_AUTHORIZER
```

CONFIG_AUTHORIZER_BASE variable

Configuration property prefix to identify properties for a given authorizer.

Syntax

```
final String CONFIG_AUTHORIZER_BASE
```

CONFIG_CONTROLFLAG variable

Property to specify the control flag for a login module or audit destination provider.

Syntax

```
final String CONFIG_CONTROLFLAG
```

CONFIG_CSI variable***Syntax***

```
final String CONFIG_CSI
```

CONFIG_CSI_CONFIG_OPTION variable***Syntax***

```
final String CONFIG_CSI_CONFIG_OPTION
```

CONFIG_GLOBAL_OPTIONS_BASE variable***Syntax***

```
final String CONFIG_GLOBAL_OPTIONS_BASE
```

CONFIG_GLOBAL_PRESERVED_OPTIONS_BASE variable***Syntax***

```
final String CONFIG_GLOBAL_PRESERVED_OPTIONS_BASE
```

CONFIG_ID variable

Property to specify the provider identifier.

Syntax

```
final String CONFIG_ID
```

CONFIG_LOGINMODULE variable

Provider type identifier used used to identify a login module in internal configuration properties.

Syntax

```
final String CONFIG_LOGINMODULE
```

CONFIG_LOGINMODULE_BASE variable

Configuration property prefix to identify properties for a given login module.

Syntax

```
final String CONFIG_LOGINMODULE_BASE
```

CONFIG_OPTIONS variable

Property to specify the configuration options for a provider.

Syntax

```
final String CONFIG_OPTIONS
```

CONFIG_PROFILER variable

Provider type identifier used used to identify a profile provider in internal configuration properties.

Syntax

```
final String CONFIG_PROFILER
```

CONFIG_PROFILER_BASE variable

Configuration property prefix to identify properties for a given profile provider.

Syntax

```
final String CONFIG_PROFILER_BASE
```

CONFIG_PROVIDER variable

Syntax

```
final String CONFIG_PROVIDER
```

CONFIG_ROLEMAPPER variable

Provider type identifier used used to identify a role mapper in internal configuration properties.

Syntax

```
final String CONFIG_ROLEMAPPER
```

CONFIG_ROLEMAPPER_BASE variable

Configuration property prefix to identify properties for a given role mapping provider.

Syntax

```
final String CONFIG_ROLEMAPPER_BASE
```

CONFIG_SECURE_DATASERVICE_PROVIDER variable

Provider type identifier used used to identify a SecureDataService provider in internal configuration properties.

Syntax

```
final String CONFIG_SECURE_DATASERVICE_PROVIDER
```

CONFIG_SECURE_DATASERVICE_PROVIDER_BASE variable

Configuration property prefix to identify properties for a given SecureDataService provider.

Syntax

```
final String CONFIG_SECURE_DATASERVICE_PROVIDER_BASE
```

CREDENTIAL_NAME variable

Configuration property to specify the name to be associated with the credential added by a login module.

Syntax

```
final String CREDENTIAL_NAME
```

CURRENT_SUBJECT variable

The SecSubject object associated with this context map.

Syntax

```
final String CURRENT_SUBJECT
```

DEF_CHECK_IMPERSONATION variable

Default value for CHECK_IMPERSONATION.

Syntax

```
final boolean DEF_CHECK_IMPERSONATION
```

DEF_ENABLE_CERTIFICATE_AUTHENTICATION variable

Default value for ENABLE_CERTIFICATE_AUTHENTICATION.

Syntax

```
final String DEF_ENABLE_CERTIFICATE_AUTHENTICATION
```

DEF_VALIDATED_CERT_IS_IDENTITY variable

Default value for VALIDATED_CERT_IS_IDENTITY.

Syntax

```
final String DEF_VALIDATED_CERT_IS_IDENTITY
```

ENABLE_CERTIFICATE_AUTHENTICATION variable

Property to enable certificate authentication when a authentication provider (should be specified as an option to the specific authentication provider) is configured in conjunction with CertificateValidationLoginModule.

Syntax

```
final String ENABLE_CERTIFICATE_AUTHENTICATION
```

ENCRYPTED_OPTIONS variable

Prefix to be used when representing the options in internal format that should be encrypted when writing them out.

Syntax

```
final String ENCRYPTED_OPTIONS
```

Remarks

For example: If the value of the property CSI.LoginModule.0.options.options1 is to be encrypted, then the provider options should contain CSI.LoginModule.

```
0.options.options1=valueToBeEncrypted CSI.LoginModule.
```

```
0.encryptedOptions.options1=true
```

The configuration providers should follow this when they encounter options when reading in the configuration file that are encrypted and encrypt the options when they are written out.

FACTORY_LEVEL_SHARED_STATE variable

The shared key to identify the factory level shared map stored in a SecContext's shared context.

Syntax

```
final String FACTORY_LEVEL_SHARED_STATE
```

Remarks

This is meant to be used when information needs to be shared by the SecContext objects (and in turn the provider classes) associated with a SecContextFactory instance.

HTTP_PROPERTYESCOOKIES_SHARED_KEY variable

The shared key to identify the map stored in the shared context.

Syntax

```
final String HTTP_PROPERTYESCOOKIES_SHARED_KEY
```

Remarks

This map contains the key/value pairs that are obtained from the http headers/cookies or other means from the environment and in a format so that they may be set in the http headers/cookies for outbound connections. The value is of type Map<String, String>

PRESERVED_OPTIONS variable

Prefix to be used to store the original preserved option.

Syntax

```
final String PRESERVED_OPTIONS
```

Remarks

Some configuration providers may interpret things like system properties or relative locations when processing configuration options. This prefix is used to store the original untouched property value so that it can be stored in the configuration file or supplied to an admin UI. `CSI.LoginModule.0.options.options1=valueOfSystemProperty CSI.LoginModule.0.preservedOptions.options1=${system.property}`

PROVIDER_SERVICES variable

Shared key to access ProviderServices in the shared context.

Syntax

```
final String PROVIDER_SERVICES
```

SEC_CONTEXT variable

The actual com.sybase.security.SecContext associated with this context map.

Syntax

```
final String SEC_CONTEXT
```

SECURED_PROPERTY_SUFFIX variable

When specified using properties configuration, a property whose value is encrypted should have this suffix.

Syntax

```
final String SECURED_PROPERTY_SUFFIX
```

Remarks

The configuration providers should detect the suffix and decrypt/encrypt the value when reading/writing the property.

SECURITY_CIPHER_ALIAS_PASSWORD variable

The name of the property to specify the password to access a key alias.

Syntax

```
final String SECURITY_CIPHER_ALIAS_PASSWORD
```

SECURITY_CIPHER_KEYSTORE_ALIAS variable

The name of the property to specify the alias to access the certificate/key in the keystore.

Syntax

```
final String SECURITY_CIPHER_KEYSTORE_ALIAS
```

SECURITY_CIPHER_KEYSTORE_LOCATION variable

The name of the property to specify the path to the keystore.

Syntax

```
final String SECURITY_CIPHER_KEYSTORE_LOCATION
```

SECURITY_CIPHER_KEYSTORE_PASSWORD variable

The name of the property to specify the password to access the key store in which certificate/key is stored.

Syntax

```
final String SECURITY_CIPHER_KEYSTORE_PASSWORD
```

SECURITY_CIPHER_KEYSTORE_PROVIDER variable

The name of the property to specify the type of keystore.

Syntax

```
final String SECURITY_CIPHER_KEYSTORE_PROVIDER
```

SECURITY_CIPHER_KEYSTORE_TYPE variable

The name of the property to specify the provider of keystore.

Syntax

```
final String SECURITY_CIPHER_KEYSTORE_TYPE
```

SECURITY_CIPHER_PROVIDER variable

The name of the property to specify the cipher provider.

Syntax

```
final String SECURITY_CIPHER_PROVIDER
```

SECURITY_CIPHER_USE_CERT variable

The name of the property to specify whether the key or certificate associated with an alias in the keystore is to be used.

Syntax

```
final String SECURITY_CIPHER_USE_CERT
```

SECURITY_CIPHER_XFORM variable

Constants for configuration properties used to initialize cipher profile during bootstrap configuration.

Syntax

```
final String SECURITY_CIPHER_XFORM
```

Remarks

The name of the property to specify the cipher transformation

SECURITY_PROFILE_CIPHER_ALIAS_PASSWORD variable

The name of the property to specify the password to access a key alias.

Syntax

```
final String SECURITY_PROFILE_CIPHER_ALIAS_PASSWORD
```

SECURITY_PROFILE_CIPHER_KEystore_ALIAS variable

The name of the property to specify the alias to access the certificate/key in the key store.

Syntax

```
final String SECURITY_PROFILE_CIPHER_KEystore_ALIAS
```

SECURITY_PROFILE_CIPHER_KEystore_LOCATION variable

The name of the property to specify the path to the key store to retrieve certificate or key to initialize the Cipher instance.

Syntax

```
final String SECURITY_PROFILE_CIPHER_KEystore_LOCATION
```

SECURITY_PROFILE_CIPHER_KEystore_PASSWORD variable

The name of the property to specify the password to access the key store in which certificate/key is stored.

Syntax

```
final String SECURITY_PROFILE_CIPHER_KEystore_PASSWORD
```

SECURITY_PROFILE_CIPHER_KEystore_PROVIDER variable

The name of the property to specify the type of key store in which certificate/key is stored.

Syntax

```
final String SECURITY_PROFILE_CIPHER_KEystore_PROVIDER
```

SECURITY_PROFILE_CIPHER_KEystore_TYPE variable

The name of the property to specify the provider of key store in which certificate/key is stored.

Syntax

```
final String SECURITY_PROFILE_CIPHER_KEystore_TYPE
```

SECURITY_PROFILE_CIPHER_PROVIDER variable

The name of the property to specify the provider to be used to get an instance of Cipher.

Syntax

```
final String SECURITY_PROFILE_CIPHER_PROVIDER
```

SECURITY_PROFILE_CIPHER_USE_CERT variable

The name of the property to specify whether the key or certificate associated with an alias in the keystore is to be used.

Syntax

```
final String SECURITY_PROFILE_CIPHER_USE_CERT
```

SECURITY_PROFILE_CIPHER_XFORM variable

The name of the property to specify the transformation to be used to get an instance of Cipher.

Syntax

```
final String SECURITY_PROFILE_CIPHER_XFORM
```

SECURITY_PROFILE_MESSAGEDIGEST_ALGORITHM variable

The name of the property to specify the algorithm to be used to get an instance of MessageDigest.

Syntax

```
final String SECURITY_PROFILE_MESSAGEDIGEST_ALGORITHM
```

SECURITY_PROFILE_MESSAGEDIGEST_PROVIDER variable

The name of the property to specify the provider to be used to get an instance of MessageDigest.

Syntax

```
final String SECURITY_PROFILE_MESSAGEDIGEST_PROVIDER
```

SECURITY_PROFILE_SIGNATURE_ALGORITHM variable

The name of the property to specify the algorithm to be used to get an instance of Signature.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_ALGORITHM
```

SECURITY_PROFILE_SIGNATURE_ALIAS_PASSWORD variable

The name of the property to specify the password to access a key alias to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_ALIAS_PASSWORD
```

SECURITY_PROFILE_SIGNATURE_KEYSTORE_ALIAS variable

The name of the property to specify the alias to access the certificate/key in the key store to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_KEYSTORE_ALIAS
```

SECURITY_PROFILE_SIGNATURE_KEYSTORE_LOCATION variable

The name of the property to specify the path to the key store to retrieve certificate or key to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_KEYSTORE_LOCATION
```

SECURITY_PROFILE_SIGNATURE_KEYSTORE_PASSWORD variable

The name of the property to specify the password to access the key store in which certificate/key is stored to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_KEYSTORE_PASSWORD
```

SECURITY_PROFILE_SIGNATURE_KEYSTORE_PROVIDER variable

The name of the property to specify the type of key store in which certificate/key is stored to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_KEYSTORE_PROVIDER
```

SECURITY_PROFILE_SIGNATURE_KEYSTORE_TYPE variable

The name of the property to specify the provider of key store in which certificate/key is stored to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_KEYSTORE_TYPE
```

SECURITY_PROFILE_SIGNATURE_PROVIDER variable

The name of the property to specify the provider to be used to get an instance of Signature.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_PROVIDER
```

SECURITY_PROFILE_SIGNATURE_USE_CERT variable

The name of the property to specify whether the key or certificate associated with an alias in the keystore is to be used to initialize the Signature instance.

Syntax

```
final String SECURITY_PROFILE_SIGNATURE_USE_CERT
```

VALIDATED_CERT_IS_IDENTITY variable

Name of the property to set certificate as the ID for the authenticated subject.

Syntax

```
final String VALIDATED_CERT_IS_IDENTITY
```

VOTE_ABSTAIN variable

Authorizer providers return VOTE_ABSTAIN to allow other providers to make the decision.

Syntax

```
final int VOTE_ABSTAIN
```

VOTE_NO variable

Authorizer providers return VOTE_NO to deny authorization.

Syntax

```
final int VOTE_NO
```

VOTE_YES variable

Authorizer providers return VOTE_YES to permit authorization.

Syntax

```
final int VOTE_YES
```

WARNING_MANAGER variable***Syntax***

```
final String WARNING_MANAGER
```

ProviderInfo interface

This interface may optionally implemented by providers of the following types:

Syntax

```
public interface ProviderInfo
```

Derived classes

- *com.sybase.security.core.CertificateAuthenticationLoginModule* on page 58
- *com.sybase.security.core.JCESecureDataServices* on page 85
- *com.sybase.security.core.NamedConfiguration* on page 87
- *com.sybase.security.core.NoSecAttributer* on page 98
- *com.sybase.security.core.NoSecAuthorizer* on page 99
- *com.sybase.security.core.NoSecLoginModule* on page 101
- *com.sybase.security.core.PreConfiguredUserLoginModule* on page 112
- *com.sybase.security.core.ProfilerImpl* on page 116
- *com.sybase.security.core.PropertiesConfiguration* on page 119
- *com.sybase.security.core.RoleCheckAuthorizer* on page 122
- *com.sybase.security.core.XmlConfiguration* on page 135
- *com.sybase.security.core.XMLFileRoleMapper* on page 141

Remarks

- Authentication
- Attribution
- Authorization
- Profile
- Secure Data Service
- Configuration

At various points in the CSI infrastructure additional diagnostic information will be logged if providers implement this interface. In addition to the information provided by the interface, the provider classname will be reported as well.

getProviderDescription() method

Allows a provider to return a string describing itself.

Syntax

```
String getProviderDescription ()
```

Returns

the description

getProviderVersion() method

Allows a provider to return a string describing the version of the provider.

Syntax

```
String getProviderVersion ()
```


Returns

the version

ProviderServices interface

Interface that providers can use to access common services without interacting with other providers directly.

Syntax

```
public interface ProviderServices
```

audit(AuditToken, String, String, String, Decision, Map<String, Object >) method

This method is the provider-equivalent of.

Syntax

```
void audit ( AuditToken token, String resourceClass, String
resourceID, String action, Decision decision, Map< String, Object >
attributes ) throws SecException
```

Parameters

- **token** – The provider's audit token. This must be retrieved at provider initialization from the configuration map supplied to the provider. The key to use is `ProviderConst.AUDIT_TOKEN_CONFIG_PROPERTY`
- **resourceClass** – The resource class to audit. This will have the string "provider." prepended to it to identify audit records generated by providers.
- **resourceID** – The resource ID
- **action** – The action
- **decision** – The decision
- **attributes** – Any extra attributes

Exceptions

- **SecException class** – If a critical error occurred while issuing the audit record.

getCertificateValidationInstance(Map<String, Object >) method

Returns a new instance of the certificate validation module configured with the specified properties.

Syntax

```
CertificateValidation getCertificateValidationInstance ( Map<
String, Object > configProps ) throws Exception
```

Parameters

- **configProps** – the configuration properties used by the new instance of the certificate validation module.

Returns

the new instance of the certificate validation module.

Exceptions

- **Exception** – encountered while attempting to obtain the new instance.

Usage

Expected properties are of the format `ProviderConst.CERT_VALIDATION_PREFIX.[index].ProviderConst.CERT_VALIDATION_URI` i.e., `cr1.1.uri cr1.2.uri` etc

getProfile(String) method

Returns the specified profile by invoking the configured profilers without the additional role check performed by the `SecContext`.

Syntax

```
SecProfile getProfile ( String profileName ) throws SecException
```

Parameters

- **profileName** – the name of the profile to retrieve.

Returns

the profile requested.

Exceptions

- **SecException class** – encountered while retrieving the specified profile.

Usage

This is to be used by other providers.

isAuditEnabled(AuditToken , String, String, Decision) method

This method is the provider-equivalent of.

Syntax

```
boolean isAuditEnabled ( AuditToken token , String resourceClass ,  
String action , Decision decision ) throws SecException
```

Parameters

- **token** – The provider's audit token
- **resourceClass** – The resource class to audit
- **action** – The action
- **decision** – The decision

Returns

true if the record would be written, false otherwise

Exceptions

- **SecException class** – If a critical error occurred while checking the audit filters.

RoleMapAdministrable interface

Interface to aid in managing role mappings.

Syntax

```
public interface RoleMapAdministrable
```

Derived classes

- *com.sybase.security.core.XMLFileRoleMapper* on page 141

Remarks

Providers that allow reading and storing the role mapping information in a store should implement this interface.

getRoleMapperAdmin(SecAdminContext) method

Retrieves the RoleMapperAdmin that can be used to manage the role mapping definitions.

Syntax

```
RoleMapperAdmin getRoleMapperAdmin ( SecAdminContext sac )
```

Parameters

- **sac** – SecAdminContext that can be used in managing the security configuration.

Returns

RoleMapperAdmin that can be used to manage the role mapping definitions.

RoleMapper interface

Interface for providers that implement Role Mapping functionality.

Syntax

```
public interface RoleMapper
```

Derived classes

- *com.sybase.security.provider.AbstractRoleMapper* on page 176

getMapping(String, String) method

Returns a list of physical role the specified logical role maps to.

Syntax

```
List< String > getMapping ( String pkg , String logicalRole )
```

Parameters

- **pkg** – package or scope to map roles from
- **logicalRole** – role name to map in this scope

Returns

list of physical roles this logical role maps to

DEFAULT_PACKAGE variable

Use this constant value for the pkg argument when referencing the "Default" package.

Syntax

```
final String DEFAULT_PACKAGE
```

NO_ROLE variable

Use this constant for physicalRole when a logicalRole is explicitly mapped to nothing (disabled).

Syntax

```
final String NO_ROLE
```

SecConfigurationValidatingProvider interface

This is the interface that the providers can implement so that the configuration can be validated and the discovered validation errors can be reported prior to actually using the configuration.

Syntax

```
public interface SecConfigurationValidatingProvider
```

Derived classes

- *com.sybase.security.core.ClientValuePropagatingLoginModule* on page 70
- *com.sybase.security.core.DefaultAuditFilter* on page 74
- *com.sybase.security.core.FileAuditDestination* on page 77
- *com.sybase.security.core.PreConfiguredUserLoginModule* on page 112

validateConfiguration(Map< String, String >) method

Validate the supplied configuration and report any errors.

Syntax

List< ConfigurationProblem > validateConfiguration (Map< String, String > *configuration*) throws SecException

Parameters

- **configuration** – provider configuration to be validated

Returns

the list of configuration problems.

Exceptions

- **SecException class** – if there is an error validating the specified configuration. *The configuration problems should not be reported using the Exception mechanism.*

Usage

If no errors are discovered an empty set should be returned. If possible, the same runtime checks that are done in the init() method should be performed so that the configuration errors can be fixed prior to deploying the configuration. A SecException should not be thrown to report the configuration problems, it should only be thrown in case of an error in validating the specified configuration.

SecContextProvider interface

Interface that should be implemented by providers that support the notion of a context (i.e., session) and need to perform specific actions when a context is created/destroyed.

Syntax

```
public interface SecContextProvider
```

Derived classes

- *com.sybase.security.provider.Attributer* on page 181
- *com.sybase.security.provider.Authorizer* on page 200
- *com.sybase.security.provider.Profiler* on page 233
- *com.sybase.security.provider.RoleMapper* on page 254
- *com.sybase.security.provider.SecureDataServices* on page 263

destroyContext(Map< String, Object >) method

Subclasses that need to clean up when a SecContext will no longer be used should override this method so they can do their cleanup.

Syntax

```
void destroyContext ( Map< String, Object > context ) throws  
SecException
```

Parameters

- **context** – the context from the SecContext that is being destroyed.

Exceptions

- **SecException class** – if any error occurs while destroying the context. This error will be logged but will not be propagated to the client.

initContext(Map< String, Object >) method

Subclasses that would like to initialize each SecContext before any other methods are called may do so in this method.

Syntax

```
void initContext ( Map< String, Object > context ) throws  
SecException
```

Parameters

- **context** – the context from the SecContext that is being initialized

Exceptions

- **SecException class** – if any error occurs while creating the context. This will invalidate the context so care should be made about throwing this exception. The exception will be propagated to the client.

SecIDPrincipal interface

Tagging interface that can be used to identify a custom Principal as one that should be mapped to the ID attribute of a SecSubject.

Syntax

```
public interface SecIDPrincipal
```

Derived classes

- *com.sybase.security.provider.BasicSecIDPrincipal* on page 204

Remarks

This interface does not define any methods to implement.

SecLoginExceptionAuthenticationFailureWarningImpl class

Simple LoginException implementation that implements the AuthenticationFailureWarning interface.

Syntax

```
public class
SecLoginExceptionAuthenticationFailureWarningImpl
```

Remarks

Providers can use this to propagate authentication failure code and associated failure message or exceptions from the login method. These will be added to the context automatically.

SecLoginExceptionAuthenticationFailureWarningImpl(int) constructor

Constructs an instance of SecLoginExceptionAuthenticationFailureWarningImpl with the specified failure code.

Syntax

```
SecLoginExceptionAuthenticationFailureWarningImpl ( int code )
```

Parameters

- **code** – failure code to be set in the SecLoginExceptionAuthenticationFailureWarningImpl

SecLoginExceptionAuthenticationFailureWarningImpl(int, String) constructor

Constructs an instance of `SecLoginExceptionAuthenticationFailureWarningImpl` with the specified failure code and message.

Syntax

```
SecLoginExceptionAuthenticationFailureWarningImpl ( int code ,  
String msg )
```

Parameters

- **code** – failure code to be set in the `SecLoginExceptionAuthenticationFailureWarningImpl`
- **msg** – message to be set in the `SecLoginExceptionAuthenticationFailureWarningImpl`

SecLoginExceptionAuthenticationFailureWarningImpl(String) constructor

Constructs an instance of `SecLoginExceptionAuthenticationFailureWarningImpl` with the specified message.

Syntax

```
SecLoginExceptionAuthenticationFailureWarningImpl ( String  
msg )
```

Parameters

- **msg** – Message to be associated with the warning.

SecLoginExceptionAuthenticationFailureWarningImpl(Throwable) constructor

Constructs an instance of `SecLoginExceptionAuthenticationFailureWarningImpl` wrapping the specified `Throwable`.

Syntax

```
SecLoginExceptionAuthenticationFailureWarningImpl ( Throwable  
e )
```

Parameters

- **e** – `Throwable` to be wrapped.

getFailureCode() method

Returns the failure code for the failed authentication attempt.

Syntax

```
int getFailureCode ()
```


Returns

Authentication failure code

getMessage() method

Returns the localized message associated with the warning.

Syntax

```
String getMessage ()
```

Returns

Message associated with the warning.

_code variable

Authentication failure code.

Syntax

```
int _code
```

SecLoginExceptionWarningImpl class

Simple LoginException implementation that implements the SecWarning interface.

Syntax

```
public class SecLoginExceptionWarningImpl
```

Derived classes

- *com.sybase.security.provider.SecLoginExceptionAuthenticationFailureWarningImpl* on page 257

Remarks

Providers can use this to propagate exceptions from the login method. These will be added to the context automatically.

SecLoginExceptionWarningImpl(String) constructor

Constructs an instance of SecLoginExceptionWarningImpl with the specified message.

Syntax

```
SecLoginExceptionWarningImpl (String msg)
```

Parameters

- **msg** – Message to be associated with the warning.

SecLoginExceptionWarningImpl(Throwable) constructor

Constructs an instance of `SecLoginExceptionWarningImpl` wrapping the specified `Throwable`.

Syntax

```
SecLoginExceptionWarningImpl ( Throwable le )
```

Parameters

- **le** – `Throwable` to be wrapped.

getMessage() method

Returns the localized message associated with the warning.

Syntax

```
String getMessage ( )
```

Returns

Message associated with the warning.

SecNamePrincipal interface

Tagging interface that can be used to identify a custom `Principal` as one that should be mapped to the `NAME` attribute of a `SecSubject`.

Syntax

```
public interface SecNamePrincipal
```

Derived classes

- *com.sybase.security.core.CertificateIDPrincipal* on page 64
- *com.sybase.security.provider.BasicSecNamePrincipal* on page 205

Remarks

This interface does not define any methods to implement.

SecProvider interface

Common interface that should be implemented by all security providers (except authentication providers).

Syntax

```
public interface SecProvider
```

Derived classes

- *com.sybase.security.provider.AuditDestination* on page 194
- *com.sybase.security.provider.AuditFilter* on page 196
- *com.sybase.security.provider.AuditFormatter* on page 197
- *com.sybase.security.provider.DigitalSignature* on page 218
- *com.sybase.security.provider.MessageDigest* on page 226
- *com.sybase.security.provider.SecContextProvider* on page 255

init(Map< String, ?>) method

Implementations that care about initialization properties should override this method.

Syntax

`void init (Map< String, ?> configuration)` throws `SecException`

Parameters

- **configuration** – Provider specific configuration data.

Exceptions

- **SecException class** – if a required property is missing, or a provided property had bad syntax or couldn't be used. If a provider throws this exception it will be printed to the log and the provider will be inaccessible.

SecProviderCapabilities interface

Optional provider interface that allows providers to respond to capability requests.

Syntax

```
public interface SecProviderCapabilities
```

Derived classes

- *com.sybase.security.core.CertificateAuthenticationLoginModule* on page 58
- *com.sybase.security.core.CertificateValidationLoginModule* on page 68

hasCapability(Map< String, Object >, String) method

Called when when building the capability set of a provider.

Syntax

`boolean hasCapability (Map< String, Object > context , String capability)` throws `SecException`

Parameters

- **context** – the context map
- **capability** – the capability to check

Exceptions

- **SecException class** – if some sort of error occurs that should abort the entire capability query.

SecProviderPersistence interface

This interface gives a provider the opportunity to remove elements from the context Map before context serialization occurs and insert them after deserialization.

Syntax

```
public interface SecProviderPersistence
```

Remarks

This interface does not extend SecProvider because LoginModule's may implement it.

activateContext(Map< String, Object >) method

Called when loading a context.

Syntax

```
void activateContext ( Map< String, Object > map ) throws  
SecException
```

Usage

The provider should populate any data items that were removed when deactivateContext was called

deactivateContext(Map< String, Object >) method

Called when storing a context.

Syntax

```
void deactivateContext ( Map< String, Object > map ) throws  
SecException
```

Usage

The provider should remove any items that it has placed into the context map that are not serializable. Additionally it may choose to remove items that are serializable but need not be persisted, to save space.

SecureDataServices interface

Interface to be implemented by secure data service providers.

Syntax

```
public interface SecureDataServices
```

Derived classes

- *com.sybase.security.provider.AbstractSecureDataServices* on page 177

Remarks

Providers implementing this interface may also optionally implement the ProviderInfo interface to expose additional provider information. They may also optionally implement com.sybase.security.provider.MessageDigest interface and (com.sybase.security.provider.DigitalSignature} interface if they wish to support those features. jkrothap

getCipher(Map< String, Object >, SecProfile , String) method

Syntax

```
Cipher getCipher ( Map< String, Object > context , SecProfile profile , String operation ) throws SecException
```

Parameters

- **context** – CSI context
- **profile** – SecProfile object containing the properties to create and initialize a Cipher instance.
- **operation** – specifies the operation for which to initialize the Cipher object. Valid values are Const.OP_ENCRYPT and Const.OP_DECRYPT

Returns

Cipher object created based on the properties specified in the profile Map and initialized for the specified operation. Returns null if unable to handle the profile.

Exceptions

- **SecException class** –

SecWarningImpl class

Simple implementation for SecWarning.

Syntax

```
public class SecWarningImpl
```

Derived classes

- *com.sybase.security.provider.AuthenticationFailureWarningImpl* on page 198
- *com.sybase.security.provider.PasswordExpirationWarningImpl* on page 230

SecWarningImpl() constructor

Default constructor with a null warning message.

Syntax

```
SecWarningImpl ()
```

SecWarningImpl(String) constructor

Constructor with a initialized with a message string.

Syntax

```
SecWarningImpl ( String str )
```

getMessage() method

Returns the localized message associated with the warning.

Syntax

```
String getMessage ()
```

Returns

Message associated with the warning.

_msg variable

The message associated with the warning.

Syntax

```
String _msg
```

SynchronizedSecAdminContext class

This class synchronizes access to the supplied SecAdminContext instance.

Syntax

```
public class SynchronizedSecAdminContext
```

SynchronizedSecAdminContext(SynchronizedSecContextImpl , SecAdminContext) constructor

The Constructor for the class which accepts a SynchronizedSecContextImpl reference and a SecAdminContext reference.

Syntax

```
SynchronizedSecAdminContext ( SynchronizedSecContextImpl sctx ,
SecAdminContext actx )
```

getRoleMapperAdmin() method

RoleMapperAdmin allows one to administer the role mappings for the current configuration.

Syntax

```
RoleMapperAdmin getRoleMapperAdmin ( )
```

Returns

the role mapper admin object tied to the current role mappings

Usage

If the XMLFileRoleMapper provider is included in the CSI configuration, it will read the RoleMapFile option to find an XML file containing role mapping (the option specifies a file relative to the configuration file).

The information in this file maps between "Logical Roles" as referenced in the consuming application (SecContext.checkRole) and "Physical Roles" assigned to subjects in the back-end security stores.

The RoleMapperAdmin class returned from this method is initialized with the role mapping data for the current configuration. Use getMappings to retrieve the RoleMappings data.

The mapping from Logical to Physical roles can be done within different "scopes" or "packages". There is a RoleMapper.DEFAULT_PACKAGE and then there are named packages. If a role is mapped in a specific package, and you do a roleCheck in that package, the role mapping from that package is used. If there is no role mapping for the role you are checking in that package, the DEFAULT_PACKAGE is checked. If there is no mapping there, then it may use straight-thru mapping to the physical role name if the XMLFileRoleMapper is configured to allow this.

This mapping information is maintained in a hierarchy: RoleMappings - contains all the Packages RolePackage - a named package (or the default package) LogicalRole - a mapped logical role PhysicalRole - physical role mapped to a logical role.

com.sybase.security.core.HierarchicalItem for the methods available on each of these items in the hierarchy.

Here is some sample code that iterates across the entire tree of role

```
mappings:
  RoleMapperAdmin rma =
  SecContextFactory.newAdminContext(ctx).getRoleMapperAdmin();
  RoleMappings mappings = rma.getMappings();
  for (RolePackage rp : mappings)
  {
    for (LogicalRole lr : rp)
    {
      for (PhysicalRole pr : lr)
      {
        System.out.println("In package " + package.getName()
          + " the logical role " + lr.getName() +
          " " is mapped to physical role " + pr.getName());
      }
    }
  }
}
```

getSecContext() method

Syntax

SecContext getSecContext ()

Returns

the SecContext associated with this administrative context

loadExternalConfiguration(String) method

Retrieve current configuration used to create the SecContext associated with this administrative context. The configuration will be retrieved fresh from the persistent storage.

Syntax

Reader loadExternalConfiguration (String *primarySchema*) throws
SecException

Parameters

- **primarySchema** – the expected schema for the external configuration

Returns

Reader that returns the configuration in the XML format specified by the primarySchema.

Usage

This may not be the "active" configuration for factories that exist in-memory at the time this method is called. It will always reflect the last persisted configuration. The returned document is a well-formed XML document and includes the configuration metadata as well, which means that it's essential self-validating. Any configuration errors are embedded in the XML document.

storeExternalConfiguration(String, Reader) method

The supplied configuration is validated and written out to persistent storage.

Syntax

Reader storeExternalConfiguration (String *primarySchema* , Reader *reader*) throws SecException

Parameters

- **primarySchema** – the schema for the supplied external configuration to be stored.
- **reader** – the configuration to be persisted

Returns

the specified configuration with any validation problems (if any) added to it.

Usage

If the configuration is not valid, the validation problems are added to the xml document and returned. The configuration is written to the persistent storage whether valid or not. The configuration will not be made effective immediately. A call to SecContextFactory.refresh() should be used to do this.

validateExternalConfiguration(String, Reader) method

The supplied configuration is validated and any validation problems are added to the xml document and returned.

Syntax

Reader validateExternalConfiguration (String *primarySchema* , Reader *reader*) throws SecException

Parameters

- **primarySchema** – the schema for the supplied external configuration to be validated.
- **reader** – the configuration to be validated

Returns

the specified configuration with any validation problems (if any) added to it.

SynchronizedSecContextImpl class

This class synchronizes access to the supplied SecContext instance.

Syntax

```
public class SynchronizedSecContextImpl
```

SynchronizedSecContextImpl(SecContext) constructor

Method to get SynchronizedSecContextImpl - use this only if ctx is not already an instance of SynchronizedSecContextImpl.

Syntax

```
SynchronizedSecContextImpl ( SecContext ctx )
```

Usage

It is safer to use getSynchronizedContext instead.

audit(String, String, String, Decision , Map< String, Object >) method

This method allows a client to issue an audit record to the audit trail.

Syntax

```
void audit ( String resourceClass , String resourceID , String action ,  
Decision decision , Map< String, Object > attributes ) throws  
SecException
```

Parameters

- **resourceClass** – The resource class of the audit record. This string will be prepended by "client." in the audit trail to identify client-supplied audit records.
- **resourceID** – The resource ID
- **action** – The action being performed
- **decision** – The decision
- **attributes** – Any extra attributes to include in the audit request. The default audit record formatting logic specially handles classes such as java.util.Date, java.security.cert.X509Certificate, and most of the classes in the java.util.* collections framework.

Exceptions

- **SecException class** – If a critical error occurred while issuing the audit record. Depending on the configuration of the auditing system, failures to audit to a specific destination may not qualify.
- **IllegalArgumentException** – if a null resource class, resource ID, action or decision are supplied

Usage

There is no guarantee that the record will actually be added to any of the configured audit destinations or that there are even any destinations configured. The client may wish to call

`SecContext.isAuditEnabled` in order to determine if the audit record will actually be written. This can save some time building up extra attributes for the audit record.

checkAuthorization(AuthzRequest) method

Performs an authorization check with somewhat more complex combination of rules.

Syntax

```
AuthzResponse checkAuthorization ( AuthzRequest request ) throws  
SecException
```

Parameters

- **request** – authorization request to be performed.

Returns

authorization response with a decision, and possibly a reason.

Exceptions

- **SecException class** –

Usage

Refer to `AuthzRequest` and `AuthzResponse` for more details.

destroy() method

Release all resources associated with this context.

Syntax

```
void destroy ()
```

Usage

This may include terminating the current session if one has been created either implicitly or explicitly.

getAdminContext() method

Returns an administrative context associated with this context object.

Syntax

```
SecAdminContext getAdminContext () throws SecException
```

Returns

a SecAdminContext that can be used to administer the configuration this SecContext is associated with.

Exceptions

- **SecException class** – if the AdministratorRole global option has been specified for the current configuration, and the user trying to retrieve a SecAdminContext is **not** in the role specified as the value of this option.

getConfigHash() method

Internal use only.

Syntax

```
String getConfigHash ()
```

getEnvironment() method

Syntax

```
SecEnvironment getEnvironment ()
```

Returns

the SecEnvironment object. This object may be re-used with other SecContext instances.

getID() method

Retrieve unique context ID.

Syntax

```
String getID ()
```

Returns

unique id

Usage

This ID may safely be used to refer to a SecContext instance within a hashtable or other such structure. Additionally, these identifiers are guaranteed to be unique across multiple machines because they include (in part) the machine's IP address.

getMaxWarningCount() method

Retrieves maximum number of warnings that will be saved.

Syntax

```
int getMaxWarningCount ()
```

Returns

maximum number of warnings that will be saved

getMessageDigest(String) method

Returns an initialized `java.security.MessageDigest` object instance.

Syntax

```
MessageDigest getMessageDigest ( String profileName ) throws  
SecException
```

Parameters

- **profileName** –

Returns

`MessageDigest` to perform the digest operation.

Exceptions

- **IllegalStateException** – if the specified profile requires a role to access and there is no current authenticated subject or if the context has been destroyed.
- **SecException class** – if no profile with the specified profile name exists or if the specified profile requires a role to access and if the authenticated subject is not in that role. Or if none of the configured `SecureDataService` providers can handle the request.

getProfile(String) method

Returns the profile attributes in a `SecProfile` object.

Syntax

```
SecProfile getProfile ( String id ) throws SecException
```

Parameters

- **id** – unique identifier of the desired profile. The name of the profile is in general unique and can be used to lookup the profile.

Returns

the `SecProfile` wrapper for the specified profile if the caller has a role that is required to access the profile or if the profile does not require any roles to access the information.

Exceptions

- **IllegalStateException** – if the specified profile requires a role to access and there is no current authenticated subject or if the context has been destroyed.
- **SecException class** – if no such profile exists or the specified profile requires a role to access and if the authenticated subject is not in that role.

Usage

If the profile requires a role to access it and the caller does not have any of the required roles, then `SecException` is thrown.

This method is deprecated, there is no need to retrieve specific profile instances. Instead they are referenced by name when necessary.

getResource(String) method

Syntax

```
SecResource getResource ( String id ) throws SecException
```

Parameters

- **id** – unique identifier for the desired resource. Some security providers will also be able to return information on a resource if this "id" parameter contains the name of the resource rather than its ID. Since a resource name is not required/guaranteed to be unique, the resource returned may not be deterministic. Pass in resource names to this method at your own peril.

Returns

A `SecResource` wrapper for the specified resource.

Exceptions

- **SecException class** – if the resource cannot be identified by any providers.
- **IllegalStateException** – if the context has been destroyed.

getSynchronizedContext(SecContext) method

Returns `SynchronizedSecContextImpl`.

Syntax

```
SynchronizedSecContextImpl getSynchronizedContext  
( SecContext ctx )
```

Usage

Can be safely used when ctx is already an instance of SynchronizedSecContextImpl. Method performs the check and returns accordingly.

getWarnings() method

Returns the list of warnings.

Syntax

```
List< SecWarning > getWarnings ()
```

Returns

List of the warnings stored for this context.

Usage

An empty list is returned if no warnings are in the queue. New warnings are appended at the tail of the list so to receive the most recent warnings use List.ListIterator(List.size()).previous() operation. The client may clear or remove warnings from this list as they see fit. The underlying implementation will enforce the maximum warning count by removing the oldest warning when adding a new warning if the queue size is equal to the max warning count. It will also disallow adding new elements by the CSI client.

hasCapability(String) method

Returns whether or not context objects created by this factory have the specified capability or not.

Syntax

```
boolean hasCapability ( String capability ) throws SecException
```

Parameters

- **capability** – The capability that we are checking out.

Exceptions

- **SecException class** – if any providers raised an exception during the checks

isAuditEnabled(String, String, Decision) method

Returns whether or not an audit record with the supplied arguments will be written to any audit destinations.

Syntax

```
boolean isAuditEnabled ( String resourceClass , String action ,  
Decision decision ) throws SecException
```

Parameters

- **resourceClass** – The resource class of the audit record
- **action** – The action to be performed
- **decision** – The decision

Returns

true if the record will be written, false otherwise

Exceptions

- **SecException class** – if an error occurs while checking the audit filters
- **IllegalArgumentException** – if a null resource class, action or decision are supplied

lateInit(Object, SecConfiguration3, Map< String, Object >, Map< String, Object >)
method

Internal use only.

Syntax

```
void lateInit ( Object o, SecConfiguration3 store, Map< String, Object > selectors, Map< String, Object > factoryLevelSharedState ) throws SecException
```

Parameters

- **o** –

listActions(SecResource) method

Lists the known actions for the specified resource.

Syntax

```
Named[] listActions ( SecResource res )
```

Parameters

- **res** – the security resource to list known actions for

Returns

a list of all known security system actions that apply to the specified resource

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listEnvironmentAttributes() method

Lists all known Environment attributes the security system may consider for policy decisions.

Syntax

```
Named[] listEnvironmentAttributes ()
```

Returns

a list of all known Environment attributes.

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listProfiles() method**Syntax**

```
Named[] listProfiles ()
```

Returns

a list of all known profiles.

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listResourceTypes() method**Syntax**

```
Named[] listResourceTypes ()
```

Returns

a list of security system resource types.

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listRoles() method

Returns a list of all known security roles.

Syntax

```
Named[] listRoles ()
```

Returns

a list of all known security roles by Name

Exceptions

- **IllegalStateException** – if the context has been destroyed.

Usage

This role list will be taken from the configured attributers in addition to the list of any physical roles defined by active role mappers.

The roles will be returned sorted by name, then ID, using the `java.lang.String.compareTo()` method.

login() method

Forces the context to be authenticated.

Syntax

```
void login () throws LoginException
```

Exceptions

- **javax.security.auth.login.LoginException** – thrown if any errors encountered while performing authentication.
- **IllegalStateException** – if the context has been destroyed.

Usage

If it's already authenticated, throws a `LoginException`.

reencrypt(String, String, byte[]) method

Convenience method to decrypt a buffer of data using `srcProfile` and encrypt it using `destProfile` in one step.

Syntax

```
byte[] reencrypt ( String srcProfileName , String destProfileName ,  
byte[] data ) throws SecException
```

Parameters

- **srcProfileName** – name of the profile to use for decrypting the data passed in
- **destProfileName** – name of the profile to use for encrypting the decrypted data
- **data** – data encrypted using `srcProfileName`

Returns

data decrypted using srcProfileName and encrypted using destProfileName

Exceptions

- **IllegalStateException** – if one of the profiles require a role to access and there is no current authenticated subject. Or if none of the configured SecureDataService providers can handle the request. Or if the context has been destroyed.
- **SecException class** – if no profiles with the specified names exist. Or one or both the specified profiles require a role to access and if the authenticated subject is not in that role.

setMaxWarningCount(int) method

Sets the maximum number of warnings that will be stored for the context.

Syntax

```
void setMaxWarningCount ( int count )
```

Parameters

- **count** – maximum number of warnings that will be stored

WarningManager interface

Interface that providers can use to log and add warnings to the SecContext.

Syntax

```
public interface WarningManager
```

Derived classes

- *com.sybase.security.provider.ProviderServices* on page 251

Attributed interface

All objects that have Attributes should implement this interface.

Syntax

```
public interface Attributed
```

Derived classes

- *com.sybase.security.provider.AbstractAttributed* on page 147
- *com.sybase.security.SecEnvironment* on page 319
- *com.sybase.security.SecProfile* on page 321
- *com.sybase.security.SecResource* on page 322

- *com.sybase.security.SecSubject* on page 323

Remarks

Attributes are named by Strings and contain one or more String typed values.

addAttribute(String, String) method

Adds an additional value to an attribute.

Syntax

```
void addAttribute ( String name , String value )
```

Parameters

- **name** – the name of the attribute
- **value** – the value to add

Exceptions

- **IllegalStateException** – if called after marked read only

Usage

If the attribute does not exist, it will be created.

addAttributes(String, String[]) method

Adds additional values to an attribute.

Syntax

```
void addAttributes ( String name , String[] values )
```

Parameters

- **name** – the name of the attribute
- **values** – the values to add

Exceptions

- **IllegalStateException** – if called after the attributed object is marked read only

Usage

If the attribute does not exist, it will be created.

commitModifications() method

Commits any changes made to this attributed object.

Syntax

```
void commitModifications () throws SecException
```

getAllAttributes() method

Retrieves a Map containing all the attribute keys and values.

Syntax

```
Map< String, String[]> getAllAttributes ()
```

Returns

a Map object containing all attributes. The key values in the Map will be the attribute names. The values in the Map will each be String[].

getAttribute(String) method

Convenience method for getting single-valued attributes.

Syntax

```
String getAttribute ( String name )
```

Parameters

- **name** – The name of the attribute whose value should be retrieved.

Returns

The first value for the named attribute. If there are no attribute values with the given name, it returns null.

getAttributes(String) method

Retrieves an array of values for the specified attribute name.

Syntax

```
String[] getAttributes ( String name )
```

Parameters

- **name** – The name of the attribute whose values should be retrieved.

Returns

All the values associated with the named attribute. If there are no attribute values with the given name returns null.

isReadOnly() method

Returns the status of read only mode.

Syntax

```
boolean isReadOnly ()
```

Returns

true if this instance is marked read only

setAttribute(String, String) method

Convenience method for setting single-valued attributes.

Syntax

```
void setAttribute ( String name , String value )
```

Parameters

- **name** – the name of the attribute
- **value** – the attribute's single value

Exceptions

- **IllegalStateException** – if called after the attributed object is marked read only

setAttributes(String, String[]) method

Sets all of an attribute's values overwriting any values already set.

Syntax

```
void setAttributes ( String name , String[] values )
```

Parameters

- **name** – the name of the attribute
- **values** – the attribute's values

Exceptions

- **IllegalStateException** – if called after the attributed object is marked read only

setReadOnly() method

On Attributed objects supporting this method, permanently disables any operations that would modify the contents of this object.

Syntax

```
void setReadOnly ()
```

Exceptions

- **UnsupportedOperationException** – if the operation is not allowed on this instance

CertificateCredential class

This class exposes the method to retrieve the certificate after a user has been authenticated so that other components can retrieve it for performing SSO.

Syntax

```
public class CertificateCredential
```

Remarks

This credential is neither refreshable nor destroyable.

CertificateCredential(Certificate) constructor

Creates the credential to hold the specified certificate.

Syntax

```
CertificateCredential ( Certificate cert )
```

Parameters

- **cert** – certificate to store in the credential

getCertificate() method

Returns the certificate stored in the credential.

Syntax

```
Certificate getCertificate ()
```

Returns

certificate stored in the credential

getName() method

Returns the name associated with the credential instance.

Syntax

```
String getName ()
```

Returns

credential name

getValue() method

Returns the value stored in the credential in base64 encoded format appropriate to relay it to the backend system to perform SSO.

Syntax

```
String getValue ()
```

Returns

base64 encoded certificate value

setName(String) method

Associates the specified name with the credential.

Syntax

```
void setName ( String name )
```

Parameters

- **name** – credential name

DEFAULT_CREDENTIAL_NAME variable

Syntax

```
final String DEFAULT_CREDENTIAL_NAME
```

Const interface

This interface defines constants for use with CSI.

Syntax

```
public interface Const
```


Derived classes

- *com.sybase.security.core.CertificateAuthenticationLoginModule* on page 58
- *com.sybase.security.core.CertificateValidationLoginModule* on page 68
- *com.sybase.security.core.JCESecureDataServices* on page 85
- *com.sybase.security.core.XMLFileRoleMapper* on page 141

ATT_CERTIFICATE variable

Key to access the certificate converted into String using `CertificateTools.toAttributeString()` method.

Syntax

```
final String ATT_CERTIFICATE
```

ATT_COMMONNAME variable

Key to retrieve the subject or asset's name.

Syntax

```
final String ATT_COMMONNAME
```

ATT_DESCRIPTION variable

Key to retrieve the description of an Attributed item.

Syntax

```
final String ATT_DESCRIPTION
```

ATT_EMAIL variable

Key to retrieve the subject's email address.

Syntax

```
final String ATT_EMAIL
```

ATT_FIRSTNAME variable

Key to retrieve the subject's first name.

Syntax

```
final String ATT_FIRSTNAME
```

ATT_ID variable

Key to retrieve the ID of an Attributed item.

Syntax

```
final String ATT_ID
```

ATT_LASTNAME variable

Key to retrieve the subject's last name.

Syntax

```
final String ATT_LASTNAME
```

ATT_NAME variable

Key to retrieve the name of the attributed item.

Syntax

```
final String ATT_NAME
```

ATT_PASSWORD variable

Key to access the password for self registration tasks.

Syntax

```
final String ATT_PASSWORD
```

ATT_PHONE variable

Key to retrieve the subject's telephone number.

Syntax

```
final String ATT_PHONE
```

ATT_USERNAME variable

Key to retrieve the username of the subject or in the case of resource-specific attributes, the username stored for that resource.

Syntax

```
final String ATT_USERNAME
```

BUILDTIME variable

Build date/time.

Syntax

```
final String BUILDTIME
```

CAPABILITY_EXPIRED_PASSWORD_CHANGE variable

The capability to change an expired password.

Syntax

```
final String CAPABILITY_EXPIRED_PASSWORD_CHANGE
```

Remarks

Providers should implement this capability themselves.

CAPABILITY_FINE_GRAIN_ACCESS_CONTROL variable

The FineGrainAccessControl capability.

Syntax

```
final String CAPABILITY_FINE_GRAIN_ACCESS_CONTROL
```

Remarks

Providers should implement this capability themselves.

CAPABILITY_PASSWORD_CHANGE variable

The password change capability.

Syntax

```
final String CAPABILITY_PASSWORD_CHANGE
```

Remarks

Providers should implement this capability themselves.

CAPABILITY_PREFIX variable

Prefix reserved for capabilities defined by CSI.

Syntax

```
final String CAPABILITY_PREFIX
```

CAPABILITY_SELF_REGISTRATION variable

The self-registration capability.

Syntax

```
final String CAPABILITY_SELF_REGISTRATION
```

Remarks

Providers should implement this capability themselves.

CAPABILITY_X509_AUTHENTICATION variable

The certificate authentication capability.

Syntax

```
final String CAPABILITY_X509_AUTHENTICATION
```

Remarks

Providers should implement this capability themselves.

CONFIGURATION_PROVIDER_PROPERTY variable

The name of the system property used to override the configuration provider.

Syntax

```
final String CONFIGURATION_PROVIDER_PROPERTY
```

CONFIGURATION_PROVIDER_PROPERTY_DEFAULT variable

The default configuration provider.

Syntax

```
final String CONFIGURATION_PROVIDER_PROPERTY_DEFAULT
```

CONTEXT_RETRIEVER_PROVIDER_PROPERTY variable

The name of the system property that can optionally be used to specify a list of active context retriever providers.

Syntax

```
final String CONTEXT_RETRIEVER_PROVIDER_PROPERTY
```

Remarks

If more than one provider is required, the provider class names must be separated by semicolons.

By default the system will automatically inspect the JVM for available context retrievers so the use of this property is not normally required.

CORE_CAPABILITY_ATTRIBUTION variable

The attribution capability.

Syntax

```
final String CORE_CAPABILITY_ATTRIBUTION
```

CORE_CAPABILITY_AUTHENTICATION variable

The authentication capability.

Syntax

```
final String CORE_CAPABILITY_AUTHENTICATION
```

CORE_CAPABILITY_AUTHORIZATION variable

The authorization capability.

Syntax

```
final String CORE_CAPABILITY_AUTHORIZATION
```

CORE_CAPABILITY_CRYPTOGRAPHY variable

The cryptography capability.

Syntax

```
final String CORE_CAPABILITY_CRYPTOGRAPHY
```

CORE_CAPABILITY_PROVIDER_CLASS variable

This capability is a dynamic capability that can be used to see if a specific provider is active in the provider list for a context.

Syntax

```
final String CORE_CAPABILITY_PROVIDER_CLASS
```

Remarks

To use it, append the class name you want to query for to this capability string when performing the capability check.

FACTORY_RETRIEVER_PROVIDER_PROPERTY variable

The name of the system property that can optionally be used to specify a list of active factory retriever providers.

Syntax

```
final String FACTORY_RETRIEVER_PROVIDER_PROPERTY
```

Remarks

If more than one provider is required, the provider class names must be separated by semicolons.

By default the system will automatically inspect the JVM for available factory retrievers so the use of this property is not normally required.

LOG_WARNINGS_PROPERTY variable

The name of the system property used to enable/disable logging the warnings.

Syntax

```
final String LOG_WARNINGS_PROPERTY
```

LOG_WARNINGS_PROPERTY_DEFAULT variable

The default value for logging warnings.

Syntax

```
final String LOG_WARNINGS_PROPERTY_DEFAULT
```

MAX_WARNINGS_PROPERTY variable

The name of the system property used to override the maximum number of SecWarnings stored in a context.

Syntax

```
final String MAX_WARNINGS_PROPERTY
```

MAX_WARNINGS_PROPERTY_DEFAULT variable

The default maximum number of SecWarnings stored.

Syntax

```
final String MAX_WARNINGS_PROPERTY_DEFAULT
```

OP_DECRYPT variable

The name of the property to specify decryption operation to get an initialized Cipher instance.

Syntax

```
final String OP_DECRYPT
```

OP_ENCRYPT variable

The name of the property to specify encryption operation to get an initialized Cipher instance.

Syntax

```
final String OP_ENCRYPT
```

OP_SIGN variable

The name of the property to specify signing operation to get an initialized Signature instance.

Syntax

```
final String OP_SIGN
```

OP_VERIFY variable

The name of the property to specify verify operation to get an initialized Signature instance.

Syntax

```
final String OP_VERIFY
```

REGEX_AUDIT_ATTRIBUTE_MASK_PROPERTY variable

The name of the system property that can optionally be used to specify the regular expression to match the provider configuration property names whose values should be masked in the audit record.

Syntax

```
final String REGEX_AUDIT_ATTRIBUTE_MASK_PROPERTY
```

REGEX_AUDIT_ATTRIBUTE_MASK_PROPERTY_DEFAULT variable

The default value for the REGEX_AUDIT_ATTRIBUTE_MASK_PROPERTY.

Syntax

```
final String REGEX_AUDIT_ATTRIBUTE_MASK_PROPERTY_DEFAULT
```

RESERVED_CAPABILITY_PREFIX variable

Prefix for standard non-core capabilities, handled by providers.

Syntax

```
final String RESERVED_CAPABILITY_PREFIX
```

RESERVED_CORE_CAPABILITY_PREFIX variable

Prefix for core capabilities, handled by core CSI and not providers.

Syntax

```
final String RESERVED_CORE_CAPABILITY_PREFIX
```

SHORTVERSION variable

Short version string.

Syntax

```
final String SHORTVERSION
```

VERSION variable

Version description (truncated version string).

Syntax

```
final String VERSION
```

VERSIONID variable

Version ID (including the milestone/drop number).

Syntax

```
final String VERSIONID
```

WARNINGS_LOG_LEVEL_PROPERTY variable

The name of the system property used to specify the level at which the warnings should be logged.

Syntax

```
final String WARNINGS_LOG_LEVEL_PROPERTY
```

Remarks

The valid values can be taken from `java.util.logging.Level`.

WARNINGS_LOG_LEVEL_PROPERTY_DEFAULT variable

The default value for the level at which warnings are logged.

Syntax

```
final String WARNINGS_LOG_LEVEL_PROPERTY_DEFAULT
```

Decision class

Represents decisions within the audit subsystems.

Syntax

```
public class Decision
```

Remarks

There are no public methods.

toString() method

Returns the string value of a decision.

Syntax

```
String toString ()
```

valueOf(String) method

Returns the Decision object representing the string argument.

Syntax

```
Decision valueOf (String decision)
```

Parameters

- **decision** –

Returns

Decision object representing the string argument

Exceptions

- **IllegalArgumentException** – if the supplied argument is null or cannot be converted to a Decision.

ABSTAIN variable

Indicates that a decision cannot be conclusively made for the operation.

Syntax

```
final static Decision ABSTAIN
```

DENY variable

Indicates that the decision is to deny the operation (failure or false).

Syntax

```
final static Decision DENY
```

NOTAPPLICABLE variable

Indicates that the operation does not have an applicable decision.

Syntax

```
final static Decision NOTAPPLICABLE
```

PERMIT variable

Indicates that the decision is to permit the operation (success or true).

Syntax

```
final static Decision PERMIT
```

Named interface

This interface defines a base for objects that have names and unique identifiers.

Syntax

```
public interface Named
```

Derived classes

- *com.sybase.security.provider.BasicNamed* on page 201
- *com.sybase.security.SecProfile* on page 321
- *com.sybase.security.SecResource* on page 322
- *com.sybase.security.SecSubject* on page 323

Remarks

The name attribute of such an object may or may not be unique, but usually makes sense to humans.

Security API

The id attribute is unique, but may not be pretty.

getDescription() method

Gets the description of this object.

Syntax

```
String getDescription ()
```

Returns

a description of this Named object

getID() method

Gets the identifier of this object.

Syntax

```
String getID ()
```

Returns

the unique identifier for this object

getName() method

Gets the name of this object.

Syntax

```
String getName ()
```

Returns

the name of this object

NamedCredential interface

This interface associates a name with a credential object added to the JAAS subject.

Syntax

```
public interface NamedCredential
```

Derived classes

- *com.sybase.security.CertificateCredential* on page 281
- *com.sybase.security.core.NamedCredentialImpl* on page 97

Remarks

This can be used by other components interfacing with CSI to retrieve the JAAS credentials and the name of the credentials to perform SSO.

getName() method

Returns Name associated with the credential.

Syntax

```
String getName ()
```

Returns

Name of the credential.

getValue() method

Returns the value stored in the credential.

Syntax

```
String getValue ()
```

Returns

value stored in the credential.

Usage

This should be in the format appropriate to relay to the backend system to perform SSO i.e., base64 encoded certificate, http header/cookie value in the format expected by the backend.

Operation class

Represents operations that can be performed on cryptographic objects.

Syntax

```
public class Operation
```

toString() method

Returns the string value of a operation.

Syntax

```
String toString ()
```

valueOf(String) method**Syntax**

```
Operation valueOf (String operation )
```

Parameters

- **operation** –

Returns

The operation, given the string value

Exceptions

- **IllegalArgumentException** – if the supplied argument is null or cannot be converted to a String

DECRYPT variable

The decryption operation.

Syntax

```
final static Operation DECRYPT
```

ENCRYPT variable

The encryption operation.

Syntax

```
final static Operation ENCRYPT
```

SIGN variable

The sign operation.

Syntax

```
final static Operation SIGN
```

VERIFY variable

The verify operation.

Syntax

```
final static Operation VERIFY
```

SecAdminContext interface

This interface provides access to administrative CSI services.

Syntax

```
public interface SecAdminContext
```

Derived classes

- *com.sybase.security.provider.SynchronizedSecAdminContext* on page 264

Remarks

To obtain a `SecAdminContext`, use the `SecContext.getAdminContext()` method.

getRoleMapperAdmin() method

`RoleMapperAdmin` allows one to administer the role mappings for the current configuration.

Syntax

```
RoleMapperAdmin getRoleMapperAdmin ()
```

Returns

the role mapper admin object tied to the current role mappings

Usage

If the `XMLFileRoleMapper` provider is included in the CSI configuration, it will read the `RoleMapFile` option to find an XML file containing role mapping (the option specifies a file relative to the configuration file).

The information in this file maps between "Logical Roles" as referenced in the consuming application (`SecContext.checkRole`) and "Physical Roles" assigned to subjects in the back-end security stores.

The `RoleMapperAdmin` class returned from this method is initialized with the role mapping data for the current configuration. Use `getMappings` to retrieve the `RoleMappings` data.

The mapping from Logical to Physical roles can be done within different "scopes" or "packages". There is a `RoleMapper.DEFAULT_PACKAGE` and then there are named packages. If a role is mapped in a specific package, and you do a `roleCheck` in that package, the role mapping from that package is used. If there is no role mapping for the role you are checking in that package, the `DEFAULT_PACKAGE` is checked. If there is no mapping there, then it may use straight-thru mapping to the physical role name if the `XMLFileRoleMapper` is configured to allow this.

This mapping information is maintained in a hierarchy: `RoleMappings` - contains all the `Packages` `RolePackage` - a named package (or the default package) `LogicalRole` - a mapped logical role `PhysicalRole` - physical role mapped to a logical role.

`com.sybase.security.core.HierarchicalItem` for the methods available on each of these items in the hierarchy.

Here is some sample code that iterates across the entire tree of role mappings:

```
RoleMapperAdmin rma =
SecContextFactory.newAdminContext(ctx).getRoleMapperAdmin();
```

```
RoleMappings mappings = rma.getMappings();
for (RolePackage rp : mappings)
{
    for (LogicalRole lr : rp)
    {
        for (PhysicalRole pr : lr)
        {
            System.out.println("In package " + package.getName()
                + " the logical role " + lr.getName() +
                " " is mapped to physical role " + pr.getName());
        }
    }
}
```

getSecContext() method

Syntax

SecContext getSecContext ()

Returns

the SecContext associated with this administrative context

loadExternalConfiguration(String) method

Retrieve current configuration used to create the SecContext associated with this administrative context. The configuration will be retrieved fresh from the persistent storage.

Syntax

Reader loadExternalConfiguration (String *primarySchema*) throws
SecException

Parameters

- **primarySchema** – the expected schema for the external configuration

Returns

Reader that returns the configuration in the XML format specified by the primarySchema.

Usage

This may not be the "active" configuration for factories that exist in-memory at the time this method is called. It will always reflect the last persisted configuration. The returned document is a well-formed XML document and includes the configuration metadata as well, which means that it's essential self-validating. Any configuration errors are embedded in the XML document.

storeExternalConfiguration(String, Reader) method

The supplied configuration is validated and written out to persistent storage.

Syntax

```
Reader storeExternalConfiguration ( String primarySchema , Reader
reader ) throws SecException
```

Parameters

- **primarySchema** – the schema for the supplied external configuration to be stored.
- **reader** – the configuration to be persisted

Returns

the specified configuration with any validation problems (if any) added to it.

Usage

If the configuration is not valid, the validation problems are added to the xml document and returned. The configuration is written to the persistent storage whether valid or not. The configuration will not be made effective immediately. A call to `SecContextFactory.refresh()` should be used to do this.

validateExternalConfiguration(String, Reader) method

The supplied configuration is validated and any validation problems are added to the xml document and returned.

Syntax

```
Reader validateExternalConfiguration ( String primarySchema ,
Reader reader ) throws SecException
```

Parameters

- **primarySchema** – the schema for the supplied external configuration to be validated.
- **reader** – the configuration to be validated

Returns

the specified configuration with any validation problems (if any) added to it.

SecConfiguration interface

The interface that describes the configuration interface for CSI.

Syntax

```
public interface SecConfiguration
```

Derived classes

- *com.sybase.security.provider.AbstractBootstrapConfiguration* on page 156
- *com.sybase.security.SecConfiguration2* on page 299

Remarks

Concrete classes implementing this interface should also provide a public, blank constructor in order to make themselves available to the default configuration provider instantiation capabilities. This constructor may optionally throw a `SecException` object in the event of an error.

Providers implementing this interface may also optionally implement the `ProviderInfo` interface to expose additional provider information.

While not deprecated, the `SecConfiguration2` interface is preferred to this one.

getConfiguration() method

This method is used by the CSI infrastructure to retrieve configuration information in a standardized format.

Syntax

`Map< String, String > getConfiguration () throws SecException`

Returns

map of configuration options.

Exceptions

- **SecException class** – on a configuration error

Usage

This method will be called once for the creation of EVERY `SecContext` object. Because of this the following guidelines should be followed by any implementation:

1. The return value of this method should not change if the configuration data has not changed. This allows the CSI infrastructure to properly cache configuration data.
2. If the underlying configuration data has changed, the return value should not be modified and returned. Instead a new object should be created from scratch. CSI infrastructure caches based on object identity, not contents, so if this rule is not followed then updated configuration data will not be noticed by CSI.

SecConfiguration2 interface

The interface that describes the configuration interface for CSI.

Syntax

```
public interface SecConfiguration2
```

Derived classes

- *com.sybase.security.core.NamedConfiguration.CompatibleConfiguration* on page 89
- *com.sybase.security.SecConfiguration3* on page 301

Remarks

Concrete classes implementing this interface should also provide a public, blank constructor in order to make themselves available to the default configuration provider instantiation capabilities. This constructor may optionally throw a `SecException` object in the event of an error.

Although the API contract specifies it should be implemented, in reality the `SecConfiguration.getConfiguration()` method will never be called by the CSI factory. Instead the new `SecConfiguration2.getConfiguration(Map)` will always be called instead.

Providers implementing this interface may also optionally implement the `ProviderInfo` interface to expose additional provider information.

getConfiguration(Map< String, Object >) method

This method is used by the CSI infrastructure to retrieve configuration information in a standardized format.

Syntax

```
Map< String, String > getConfiguration ( Map< String, Object > selectors ) throws SecException
```

Parameters

- **selectors** – a map of selectors that dynamically alter the returned configuration. The specific keys and values that are meaningful depend on the underlying implementation.

Returns

map of configuration options.

Exceptions

- **SecException class** – on a configuration error

Usage

This method will be called once for the creation of every SecContext object. Because of this the following guidelines should be followed by any implementation:

1. The return value of this method should not change if the configuration data has not changed. This allows the CSI infrastructure to properly cache configuration data.
2. If the underlying configuration data has changed, the return value should not be modified and returned. Instead a new object should be created from scratch. CSI infrastructure caches based on object identity, not contents, so if this rule is not followed then updated configuration data will not be noticed by CSI.

DEFAULT_CONFIGURATION_NAME variable

The default configuration name used if none is specified in the selectors.

Syntax

```
final String DEFAULT_CONFIGURATION_NAME
```

SELECTOR_ALTERNATE_REPOSITORY variable

Selector key that is used to supply an alternate repository URL.

Syntax

```
final String SELECTOR_ALTERNATE_REPOSITORY
```

Remarks

This repository will be searched first, unless it is prefixed by the prefix specified by the SELECTOR_PRIMARY_REPOSITORY_PREFIX.

The value for this key should of type java.net.URL or java.util.File.

SELECTOR_CONFIGURATION_NAME variable

Selector key that is used to choose a configuration by name.

Syntax

```
final String SELECTOR_CONFIGURATION_NAME
```

Remarks

If this is unspecified then the configuration provider should use a default name.

The value for this key should be a java.lang.String type.

SELECTOR_FRESH_CONFIGURATION variable

Selector key that is used to specify that the configuration that is returned should be loaded from the underlying source and not a cached copy.

Syntax

```
final String SELECTOR_FRESH_CONFIGURATION
```

Remarks

The value for this key should be a java.lang.Boolean type.

SELECTOR_PRIMARY_REPOSITORY_PREFIX variable

There is no default for this property.

Syntax

```
final String SELECTOR_PRIMARY_REPOSITORY_PREFIX
```

SecConfiguration3 interface

The interface that describes the configuration interface for CSI.

Syntax

```
public interface SecConfiguration3
```

Derived classes

- *com.sybase.security.core.NamedConfiguration* on page 87
- *com.sybase.security.provider.AbstractFileConfiguration* on page 159

Remarks

Concrete classes implementing this interface should also provide a public, blank constructor in order to make themselves available to the default configuration provider instantiation capabilities. This constructor may optionally throw a SecException object in the event of an error.

The configuration provider implementing this interface should support writing the updated configuration.

Providers implementing this interface may also optionally implement the ProviderInfo interface to expose additional provider information.

writeNewConfiguration(Map< String, Object >, Map< String, String >) method

This method is used by the CSI infrastructure to store the configuration information.

Syntax

```
void writeNewConfiguration ( Map< String, Object > selectors , Map< String, String > configuration ) throws SecException
```

Parameters

- **selectors** – a map of selectors that specify the location of the configuration file. The specific keys and values that are meaningful depend on the underlying implementation.
- **configuration** – map of configuration options.

Exceptions

- **SecException class** – if an error occurs storing the configuration.

Usage

This method will be called when a CSI client intends to store the updated configuration.

SecContext interface

This interface defines methods available for performing security checks, and factories for Resource, Subject, and Environment classes.

Syntax

```
public interface SecContext
```

Derived classes

- *com.sybase.security.provider.SynchronizedSecContextImpl* on page 267

audit(String, String, String, Decision, Map< String, Object >) method

This method allows a client to issue an audit record to the audit trail.

Syntax

```
void audit ( String resourceClass , String resourceID , String action ,  
Decision decision , Map< String , Object > attributes ) throws  
SecException
```

Parameters

- **resourceClass** – The resource class of the audit record. This string will be prepended by "client." in the audit trail to identify client-supplied audit records.
- **resourceID** – The resource ID
- **action** – The action being performed
- **decision** – The decision
- **attributes** – Any extra attributes to include in the audit request. The default audit record formatting logic specially handles classes such as java.util.Date, java.security.cert.X509Certificate, and most of the classes in the java.util.* collections framework.

Exceptions

- **SecException class** – If a critical error occurred while issuing the audit record. Depending on the configuration of the auditing system, failures to audit to a specific destination may not qualify.
- **IllegalArgumentException** – if a null resource class, resource ID, action or decision are supplied

Usage

There is no guarantee that the record will actually be added to any of the configured audit destinations or that there are even any destinations configured. The client may wish to call `SecContext.isAuditEnabled` in order to determine if the audit record will actually be written. This can save some time building up extra attributes for the audit record.

checkAuthorization(AuthzRequest) method

Performs an authorization check with somewhat more complex combination of rules.

Syntax

```
AuthzResponse checkAuthorization ( AuthzRequest request ) throws
SecException
```

Parameters

- **request** – authorization request to be performed.

Returns

authorization response with a decision, and possibly a reason.

Exceptions

- **SecException class** –

Usage

Refer to `AuthzRequest` and `AuthzResponse` for more details.

destroy() method

Release all resources associated with this context.

Syntax

```
void destroy ()
```

Usage

This may include terminating the current session if one has been created either implicitly or explicitly.

getAdminContext() method

Returns an administrative context associated with this context object.

Syntax

```
SecAdminContext getAdminContext () throws SecException
```

Returns

a SecAdminContext that can be used to administer the configuration this SecContext is associated with.

Exceptions

- **SecException class** – if the AdministratorRole global option has been specified for the current configuration, and the user trying to retrieve a SecAdminContext is *not* in the role specified as the value of this option.

getEnvironment() method

Syntax

```
SecEnvironment getEnvironment ()
```

Returns

the SecEnvironment object. This object may be re-used with other SecContext instances.

getID() method

Retrieve unique context ID.

Syntax

```
String getID ()
```

Returns

unique id

Usage

This ID may safely be used to refer to a SecContext instance within a hashtable or other such structure. Additionally, these identifiers are guaranteed to be unique across multiple machines because they include (in part) the machine's IP address.

getMaxWarningCount() method

Retrieves maximum number of warnings that will be saved.

Syntax

```
int getMaxWarningCount ()
```

Returns

maximum number of warnings that will be saved

getMessageDigest(String) method

Returns an initialized java.security.MessageDigest object instance.

Syntax

```
MessageDigest getMessageDigest ( String profileName ) throws  
SecException
```

Parameters

- **profileName** –

Returns

MessageDigest to perform the digest operation.

Exceptions

- **IllegalStateException** – if the specified profile requires a role to access and there is no current authenticated subject or if the context has been destroyed.
- **SecException class** – if no profile with the specified profile name exists or if the specified profile requires a role to access and if the authenticated subject is not in that role. Or if none of the configured SecureDataService providers can handle the request.

getProfile(String) method

Returns the profile attributes in a SecProfile object.

Syntax

```
SecProfile getProfile ( String id ) throws SecException
```

Parameters

- **id** – unique identifier of the desired profile. The name of the profile is in general unique and can be used to lookup the profile.

Returns

the SecProfile wrapper for the specified profile if the caller has a role that is required to access the profile or if the profile does not require any roles to access the information.

Exceptions

- **IllegalStateException** – if the specified profile requires a role to access and there is no current authenticated subject or if the context has been destroyed.
- **SecException class** – if no such profile exists or the specified profile requires a role to access and if the authenticated subject is not in that role.

Usage

If the profile requires a role to access it and the caller does not have any of the required roles, then SecException is thrown.

This method is deprecated, there is no need to retrieve specific profile instances. Instead they are referenced by name when necessary.

getResource(String) method

Syntax

```
SecResource getResource ( String id ) throws SecException
```

Parameters

- **id** – unique identifier for the desired resource. Some security providers will also be able to return information on a resource if this "id" parameter contains the name of the resource rather than its ID. Since a resource name is not required/guaranteed to be unique, the resource returned may not be deterministic. Pass in resource names to this method at your own peril.

Returns

A SecResource wrapper for the specified resource.

Exceptions

- **SecException class** – if the resource cannot be identified by any providers.
- **IllegalStateException** – if the context has been destroyed.

getWarnings() method

Returns the list of warnings.

Syntax

```
List< SecWarning > getWarnings ()
```


Returns

List of the warnings stored for this context.

Usage

An empty list is returned if no warnings are in the queue. New warnings are appended at the tail of the list so to receive the most recent warnings use

List.ListIterator(List.size()).previous() operation. The client may clear or remove warnings from this list as they see fit. The underlying implementation will enforce the maximum warning count by removing the oldest warning when adding a new warning if the queue size is equal to the max warning count. It will also disallow adding new elements by the CSI client.

hasCapability(String) method

Returns whether or not context objects created by this factory have the specified capability or not.

Syntax

boolean hasCapability (String *capability*) throws SecException

Parameters

- **capability** – The capability that we are checking out.

Exceptions

- **SecException class** – if any providers raised an exception during the checks

isAuditEnabled(String, String, Decision) method

Returns whether or not an audit record with the supplied arguments will be written to any audit destinations.

Syntax

boolean isAuditEnabled (String *resourceClass* , String *action* , Decision *decision*) throws SecException

Parameters

- **resourceClass** – The resource class of the audit record
- **action** – The action to be performed
- **decision** – The decision

Returns

true if the record will be written, false otherwise

Exceptions

- **SecException class** – if an error occurs while checking the audit filters
- **IllegalArgumentException** – if a null resource class, action or decision are supplied

listActions(SecResource) method

Lists the known actions for the specified resource.

Syntax

```
Named[] listActions ( SecResource res )
```

Parameters

- **res** – the security resource to list known actions for

Returns

a list of all known security system actions that apply to the specified resource

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listEnvironmentAttributes() method

Lists all known Environment attributes the security system may consider for policy decisions.

Syntax

```
Named[] listEnvironmentAttributes ()
```

Returns

a list of all known Environment attributes.

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listProfiles() method

Syntax

```
Named[] listProfiles ()
```

Returns

a list of all known profiles.

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listResourceTypes() method**Syntax**

```
Named[] listResourceTypes ()
```

Returns

a list of security system resource types.

Exceptions

- **IllegalStateException** – if the context has been destroyed.

listRoles() method

Returns a list of all known security roles.

Syntax

```
Named[] listRoles ()
```

Returns

a list of all known security roles by Name

Exceptions

- **IllegalStateException** – if the context has been destroyed.

Usage

This role list will be taken from the configured attributers in addition to the list of any physical roles defined by active role mappers.

The roles will be returned sorted by name, then ID, using the `java.lang.String.compareTo()` method.

login() method

Forces the context to be authenticated.

Syntax

```
void login () throws javax.security.auth.login.LoginException
```

Exceptions

- **javax.security.auth.login.LoginException** – thrown if any errors encountered while performing authentication.
- **IllegalStateException** – if the context has been destroyed.

Usage

If it's already authenticated, throws a LoginException.

reencrypt(String, String, byte[]) method

Convenience method to decrypt a buffer of data using srcProfile and encrypt it using destProfile in one step.

Syntax

```
byte[] reencrypt ( String srcProfileName , String destProfileName ,  
byte[] data ) throws SecException
```

Parameters

- **srcProfileName** – name of the profile to use for decrypting the data passed in
- **destProfileName** – name of the profile to use for encrypting the decrypted data
- **data** – data encrypted using srcProfileName

Returns

data decrypted using srcProfileName and encrypted using destProfileName

Exceptions

- **IllegalStateException** – if one of the profiles require a role to access and there is no current authenticated subject. Or if none of the configured SecureDataService providers can handle the request. Or if the context has been destroyed.
- **SecException class** – if no profiles with the specified names exist. Or one or both the specified profiles require a role to access and if the authenticated subject is not in that role.

setMaxWarningCount(int) method

Sets the maximum number of warnings that will be stored for the context.

Syntax

```
void setMaxWarningCount ( int count )
```

Parameters

- **count** – maximum number of warnings that will be stored

ADMIN_ROLE_NAME variable

Configuration property to specify the name of the security role to be checked prior to allowing interactive administration of security data.

Syntax

```
final String ADMIN_ROLE_NAME
```

Remarks

It should be specified at a global level within the configuration.

SecContextFactory class

This class exposes factory methods that other Sybase products can use for programmatic security.

Syntax

```
public class SecContextFactory
```

Remarks

In the lexicon of security, those products are implementing Policy Enforcement Points (PEPs). The purpose of this package is to provide a pluggable and stackable way to propagate these PEP invocations to the appropriate Policy Decision Points (PDPs) based on a customer's configuration of the security aspects of their installation.

The typical consumer of CSI will perform some or all of the following steps in the following order to retrieve a CSI security context object:

1. Call static `SecContextFactory.newInstance()` to retrieve a factory instance
2. Optionally call `setConfiguration` and/or `setCallbackHandler` methods to alter default Context creation behavior
3. Call `SecContextFactory` object's `getContext()` method to retrieve the security context.

As part of context creation, a configuration must always be established. If the `setConfiguration` method is called with a non-null value, then the object specified will be used when determining configuration of the security context. Otherwise, the global configuration mechanism is used. The global configuration mechanism entails retrieving the appropriate configuration object using the "com.sybase.security.ConfigurationProvider" system property. This property must either be unspecified or must specify the classname of a class that implements the `com.sybase.security.SecConfiguration` interface. If it is unspecified then the JVM is inspected to retrieve the configuration provider service. If one cannot be found then the `com.sybase.security.core.PropertiesConfiguration` class is used for configuration.

clone() method

Syntax

Object clone () throws CloneNotSupportedException

getCallbackHandler() method

Syntax

CallbackHandler getCallbackHandler ()

Returns

the callback handler

getCheckConfigurationCompatibility() method

Syntax

boolean getCheckConfigurationCompatibility ()

Returns

Whether or not the configuration compatibility should be checked when deserializing a security context.

getConfiguration() method

Syntax

SecConfiguration getConfiguration ()

Returns

the configuration instance

getConfigurationName() method

Syntax

String getConfigurationName ()

Returns

name of the default configuration used by the factory.

*getConfigurationProvider() method***Syntax**

```
String getConfigurationProvider ()
```

Returns

the configuration provider class name

getConfigurationSelectors() method

Return the map of configuration selectors.

Syntax

```
Map< String, Object > getConfigurationSelectors ()
```

Returns

the configuration selector map

Usage

Classes can modify the objects in this map as desired to establish default configuration selectors.

getContext() method

This is the method to create a SecContext object with the properties specified in this factory object.

Syntax

```
SecContext getContext () throws SecException
```

Returns

a new SecContext object

Exceptions

- **SecException class** – if an error occurs while retrieving the security context

getDefaultMaxWarningCount() method

Retrieve the current default maximum warning count for security context instances created by this factory.

Syntax

```
int getDefaultMaxWarningCount ()
```

Returns

the maximum warning count

loadContext(InputStream) method

Reads and reinitializes a security context from a stream.

Syntax

```
synchronized SecContext loadContext ( InputStream is ) throws  
SecException
```

Parameters

- **is** – The input stream from which the context should be read

Returns

The new security context

Exceptions

- **SecException class** – on encountering an error

newInstance() method

Retrieves a new instance of the context factory which has a few properties that can be tweaked, altering the results of calling the getContext() method.

Syntax

```
SecContextFactory newInstance ()
```

Returns

the factory instance

newSubject() method

Create a new subject object that can be committed to one or more of the configured attributers.

Syntax

```
SecSubject newSubject () throws SecException
```

refresh() method

This method causes CSI to re-read its global configuration properties and re-load all global provider modules.

Syntax

```
void refresh () throws SecException
```


Exceptions

- **SecException class** – if any errors occur while refreshing the providers.

Usage

It also clears configuration cache for client-specified configurations.

retrieveContext(Object) method

This method may be used to retrieve an already-created context from the local environment.

Syntax

```
SecContext retrieveContext ( Object environmentObject ) throws
SecException
```

Parameters

- **environmentObject** – The object that is provided to aid in retrieving the stored SecContext object.

Returns

the stored SecContext object

Exceptions

- **SecException class** – if no stored SecContext object was found

Usage

This operation may require the inclusion of some environment data from the local environment. For example, in a servlet it might require the HttpServletRequest object whereas in an EJB implementation it might require the EJBContext object. Other situations may not require any environment object at all.

retrieveFactory(Object) method

This method may be used to retrieve a fully initialized factory instance from the local environment.

Syntax

```
SecContextFactory retrieveFactory ( Object environmentObject ) throws
SecException
```

Parameters

- **environmentObject** – The object that is provided to aid in retrieving the stored SecContextFactory object.

Returns

the stored SecContextFactory object

Exceptions

- **SecException class** – if no stored SecContextFactory object was found

Usage

This operation may require some environment data from the local environment, although the environment object is not required at all in most cases.

setCallbackHandler(CallbackHandler) method

Sets the callbackhandler to be used when authenticating a context created by this factory instance.

Syntax

```
void setCallbackHandler ( CallbackHandler ch )
```

Parameters

- **ch** – the callbackhandler

Usage

If this method is not called and the auth.login.defaultCallbackHandler system property is defined, the default callback handler will be instantiated with the value of this property. This corresponds to the behavior outlined in the CallbackHandler javadocs.

setCheckConfigurationCompatibility(boolean) method

Sets whether or not the configuration compatibility should be checked when deserializing a security context.

Syntax

```
void setCheckConfigurationCompatibility ( boolean  
configurationCompatibility )
```

Parameters

- **configurationCompatibility** – The new value of the flag

Usage

The default value of this parameter is true.

setConfiguration(SecConfiguration) method

Sets the SecConfiguration instance to be used when creating a context with this factory instance.

Syntax

```
void setConfiguration ( SecConfiguration config )
```

Parameters

- **config** – the SecConfiguration object

Usage

This property is mutually exclusive with the configuration provider property. If neither of these properties is set, the default static configuration will be used.

setConfigurationName(String) method

Sets the configuration name to be used for creating the contexts.

Syntax

```
void setConfigurationName ( String configName )
```

setConfigurationProvider(String) method

Sets the configuration provider class that should be used when creating a context with this factory instance.

Syntax

```
void setConfigurationProvider ( String provider )
```

Parameters

- **provider** – the configuration provider class name

Usage

This property is mutually exclusive with the configuration property. If neither of these properties is set, the default static configuration will be used.

setContextRetrieverProviders(String) method

Allows a client to specify the context retriever provider list.

Syntax

```
void setContextRetrieverProviders ( String providers )
```

Parameters

- **providers** – semicolon-delimited list of configuration providers

Usage

This value may contain zero or more providers that implement the `com.sybase.security.provider.ContextRetriever` interface. The default value for this property is set by the `com.sybase.security.ContextRetriever` system property. If this property is null then the JVM will be searched for available configuration properties using the JAR services discovery mechanism.

setDefaultMaxWarningCount(int) method

Set the default maximum warning count for security context instances created by this factory.

Syntax

```
void setDefaultMaxWarningCount ( int count )
```

Parameters

- **count** – the new count

storeContext(SecContext , OutputStream) method

Write a security context to an output stream.

Syntax

```
void storeContext ( SecContext ctx , OutputStream os ) throws  
SecException
```

Parameters

- **ctx** – The context to write
- **os** – The stream to which the context will be written

Exceptions

- **SecException class** – on encountering serialization error or if the context is not a CSI core implementation
- **IllegalArgumentException** – if the context is null, output stream is null

SecEnvironment interface

This interface wraps classes that can convey arbitrary attributes of the execution environment to the PDP.

Syntax

```
public interface SecEnvironment
```

SecException class

Various security exceptions that occur during the use of methods in this package will throw this exception.

Syntax

```
public class SecException
```

SecException(String) constructor

Constructs a new SecException exception.

Syntax

```
SecException ( String msg )
```

Parameters

- **msg** – The message describing why this exception was thrown.

SecException(String, List<?extends Throwable >) constructor

Constructs a new SecException with a list of causes.

Syntax

```
SecException ( String msg , List<?extends Throwable > causeList )
```

Parameters

- **msg** – message
- **causeList** – list of causes

SecException(String, Throwable) constructor

Constructs a new SecException exception.

Syntax

```
SecException ( String msg , Throwable cause )
```

Parameters

- **msg** – The message describing why this exception was thrown.
- **cause** – The root cause throwable, if any, that resulted in this error.

SecException(Throwable) constructor

Constructs a new SecException exception.

Syntax

```
SecException ( Throwable cause )
```

Parameters

- **cause** – The root cause throwable, if any, that resulted in this error.

getCause() method

Syntax

```
Throwable getCause ()
```

getCauseList() method

Callers of this method must not rely on the return value of this method being a mutable list.

Syntax

```
List<?extends Throwable > getCauseList ()
```

Returns

list of exception causes, will be zero length if none were stored

initCause(Throwable) method

Syntax

```
synchronized Throwable initCause ( Throwable cause )
```

*initCauseList(List<?extends Throwable >) method***Syntax**

```
synchronized List<?extends Throwable > initCauseList ( List<?
extends Throwable > causeList )
```

_cause variable

Stores the root throwable, if any, of this exception.

Syntax

```
Throwable _cause
```

Remarks

Because this was here in CSI 1.0 and it's "protected" we will continue to set this even though we have a collection for multiple exceptions.

_causeList variable

Stores list of multiple causes.

Syntax

```
List<?extends Throwable > _causeList
```

SecProfile interface

This interface defines methods available for obtaining information on profiles.

Syntax

```
public interface SecProfile
```

setDescription(String) method

Sets the profile description.

Syntax

```
void setDescription ( String description )
```

Parameters

- **description** – profile description

Exceptions

- **IllegalStateException** – if called after marked readonly

setRoles(String[]) method

Sets the roles that are authorized to access the profile.

Syntax

```
void setRoles ( String[] roles )
```

Parameters

- **roles** – roles that are authorized to access this profile

Exceptions

- **IllegalStateException** – if called after marked readonly

SecResource interface

This interface defines classes that can represent a protected resource in a security system.

Syntax

```
public interface SecResource
```

addResourceType(String) method

Allows providers to add a resource type to this resource during resource initialization.

Syntax

```
void addResourceType ( String resourceTypeName )
```

Parameters

- **resourceTypeName** – the resourcetype ID to add to the resource

Exceptions

- **IllegalStateException** – if called after marked readonly

getResourceTypes() method

Retrieve a list of resource type names that this resource implements, in priority order.

Syntax

```
List< String > getResourceTypes ()
```

Returns

list of resource type names

SecSubject interface

This interface defines methods available for obtaining information on subjects.

Syntax

```
public interface SecSubject
```

getAttributesForResource(SecResource) method

Gets the attributes for the specified resource.

Syntax

```
Map< String, String[]> getAttributesForResource ( SecResource
resource )
```

Parameters

- **resource** – the target resource

Returns

a Map object with the named attributes and their values. The values are of type String[].

Usage

Sometimes users have attributes that relate only to a particular security resource (for example Proxy Authentication Information for an Asset representing some back-end system). This method provides an accessor for this information to the application.

getJAASSubject() method

Returns the JAAS subject that stores all the authentication information which includes Principals, public and private credentials added during authentication.

Syntax

```
javax.security.auth.Subject getJAASSubject ()
```

Returns

The authenticated JAAS subject, or an unauthenticated subject if the authentication has not occurred.

isAuthenticated() method

Checks if the subject is authenticated.

Syntax

```
boolean isAuthenticated ()
```

Returns

true if this object represents an authenticated user or false otherwise.

Usage

Returns true if this objects represents an authenticated user or false if it just represents some user information, but that user is not currently authenticated in this application as far as this object knows. This method has the side-effect of attempting to authenticate the user if an attempt to authenticate the user is not already triggered.

listAttributesForResource(SecResource) method

Lists the attribute names relevant for a specified resource.

Syntax

```
Named[] listAttributesForResource ( SecResource resource )
```

Parameters

- **resource** – the target resource

Returns

a list of attribute descriptions.

Usage

For design-time tooling it is sometimes necessary to know what possible attribute names may be returned from `getAttributesForResource`. This method enumerates them.

whenAuthenticated() method

Retrieves the time the user was authenticated.

Syntax

```
java.util.Date whenAuthenticated ()
```

Returns

the time when this user's current authentication was made.

Exceptions

- **IllegalStateException** – if this object does not represent an authenticated subject session.

SecWarning interface

An interface to propagate error information from the framework or the provider to the client.

Syntax

```
public interface SecWarning
```

Derived classes

- *com.sybase.security.core.AuthenticationFailureWarning* on page 56
- *com.sybase.security.core.PasswordExpirationWarning* on page 105
- *com.sybase.security.provider.SecLoginExceptionWarningImpl* on page 259
- *com.sybase.security.provider.SecWarningImpl* on page 263
- *com.sybase.security.SecException* on page 319

getMessage() method

Returns the localized message associated with the warning.

Syntax

```
String getMessage ()
```

Returns

Message associated with the warning.

SSOTokenCredential interface

This represents a credential that contains Single Sign On (SSO) data that can be retrieved from an authenticated Subject.

Syntax

```
public interface SSOTokenCredential
```

Remarks

Authentication providers that intend to provide access to the Single Sign On data to the components in the server should add a public credential that implements this interface after successfully authenticating the user.

The client could then access the SSO token using

```
javax.security.auth.Subject jaasSubject =
csiSubject.getJAASSubject();

        Set<SSOTokenCredential> credentials =
jaasSubject.getPublicCredentials(SSOTokenCredential.class);
```

```
SSOTokenCredential credential = credentials.toArray(new  
SSOTokenCredential[0])[0];
```

getSSOToken() method

Returns the Single Sign On (SSO) token in String format.

Syntax

```
String getSSOToken ()
```

Returns

SSO token

Management API

This section provides information about using the SAP Mobile Platform Management API to custom code an administration client. The audience is advanced developers who are familiar working with APIs, but who may be new to SAP Mobile Platform.

This section describes the features and usage of the Management API, how to get started with client development, and how to program a custom administration client. Also included is information on how to configure SAP Mobile Platform properties using client metadata, how to use properties, and a listing of error codes.

Introducing Management API

Use the Management API in SAP® Mobile Platform to custom code an administration client. The audience is advanced developers who are familiar working with APIs, but who may be new to SAP Mobile Platform.

This guide describes the features and usage of the Administration API, how to get started with client development, and how to program a custom administration client. Also included is information on how to configure SAP Mobile Platform properties using client metadata, how to use properties, and a listing of error codes.

Companion documentation for Management API includes:

- *Fundamentals*
- *System Administration*
- *SAP Mobile WorkSpace - Mobile Business Object Development*
- *Troubleshooting*

Management API Features

SAP Mobile Platform includes a Java API that opens the administration and configuration of SAP Mobile Platform to Java client applications you create. By building a custom client with the Management API, you can build custom application to support SAP Mobile Platform administration features and functionality within an existing IT management infrastructure.

When creating a custom SAP Mobile Platform administration client, the entry point is the `SUPObjectFactory` class. By calling methods of `SUPObjectFactory`, which require different context objects, you can retrieve administration interfaces to perform administration activities. Should errors occur, they are reported through a `SUPAdminException`, which provides the error code and error message. For details of each administration interface, you can refer to the Javadoc shipped with the Management API.

Management API Javadoc

The Management API installation includes javadoc. Use the SAP Mobile Server javadoc for your complete API reference.

As you review the contents of this document, ensure you review the reference details documented in the javadoc delivered with these APIs. By default, javadoc is installed to `SMP_HOME\Servers\UnwiredServer\AdminClientAPI\com.sybase.sup.adminapi\docs\api\index.html`.

The top left navigation pane lists all packages installed with SAP Mobile Platform. The applicable documentation is available with `com.sybase.sup.admin.client` package. Click this link and navigate through the javadoc as required.

Documentation Roadmap for SAP Mobile Platform

SAP® Mobile Platform documents are available for administrative and mobile development user roles. Some administrative documents are also used in the development and test environment; some documents are used by all users.

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Product Documentation Web site regularly for updates: <http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories>, then navigate to the most current version.

Management API Changes in Version 2.3

Changes in the Management API between 2.2 and 2.3 include enhancements for Agentry application support and minor changes for deployment, including topic cache.

For Management API changes prior to 2.2, see *Release Bulletin 2.1 ESD #3* on Product Documentation, the *Administration Client API Changes* topic: <http://dcx.sybase.com/index.html#sup0213/en/com.sybase.infocenter.dc00835.0213/doc/html/aro1345156276263.html>

SUPCluster API Changes

API changes for the `SUPCluster` interface, used to manage the cluster to which the SAP Mobile Server instance belongs.

Table 5. New SUPCluster Methods

Methods	Description
<ul style="list-style-type: none"> • <code>getHttpProxySetting</code> • <code>updateHttpProxySetting</code> 	Manage HTTP proxy settings
<ul style="list-style-type: none"> • <code>getServerLogSetting()</code> • <code>updateServerLogSetting()</code> 	Manage the SAP Mobile Server logs
<ul style="list-style-type: none"> • <code>getBESResponsePortioningLimit</code> • <code>setBESResponsePortioningLimit</code> 	Manage the response portioning limit
<ul style="list-style-type: none"> • <code>isHttpLogEnable()</code> • <code>setHttpLogEnable</code> • <code>getMaxHttpLogFileSize()</code> • <code>setMaxHttpLogFileSize</code> • <code>getHttpLogFileName()</code> • <code>setHttpLogFileName</code> • <code>isSeparateHttpLogFile()</code> • <code>setSeparateHttpLogFile</code> • <code>isReuseHttpLogFile()</code> • <code>setReuseHttpLogFile</code> • <code>getHttpLogArchiveFileName()</code> • <code>setHttpLogArchiveFileName</code> • <code>isArchiveHttpLog()</code> • <code>setArchiveHttpLog</code> • <code>isCompressHttpLogArchive()</code> • <code>setCompressHttpLogArchive</code> 	Manage the HTTP log settings

Table 6. Deprecated SUPCluster Properties

Property	Description
<code>sup.admin.maxThreads</code>	The maximum number of concurrent threads for the management port.
<code>sup.admin.iiops.maxThreads</code>	The maximum number of concurrent threads for the secure management port.

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Managing Clusters*

See also

- *SUPConfiguration API Changes* on page 330
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPConfiguration API Changes

API changes for the `SUPConfiguration` interface, used to manage the SAP Mobile Platform configuration.

Table 7. New SUPConfiguration Methods

Methods	Description
<ul style="list-style-type: none"> • <code>getContext</code> • <code>commitClusterConfiguration</code> • <code>refreshClusterConfiguration</code> 	Manage the cluster configuration
<ul style="list-style-type: none"> • <code>getManagementConfiguration</code> • <code>updateManagementConfiguration</code> 	Manage management configurations
<ul style="list-style-type: none"> • <code>enableOCSPConfiguration</code> • <code>getOCSPConfiguration</code> • <code>updateOCSPConfiguration</code> 	Manage the Online Certificate Status Protocol (OCSP) configuration

Methods	Description
<ul style="list-style-type: none"> • <code>getPerformanceConfiguration</code> • <code>updatePerformanceConfiguration</code> 	Manage the performance configuration
<ul style="list-style-type: none"> • <code>getConfigurationCache</code> • <code>updateConfigurationCache</code> • <code>commitConfigurationCache</code> • <code>refreshConfigurationCache</code> 	Manage the cache configuration
<ul style="list-style-type: none"> • <code>getReplicationConfiguration()</code> • <code>updateReplicationConfiguration()</code> 	Manage replication configuration
<ul style="list-style-type: none"> • <code>getMessagingConfiguration()</code> • <code>updateMessagingConfiguration()</code> 	Manage messaging configuration
<ul style="list-style-type: none"> • <code>getSolutionManagerConfiguration()</code> • <code>updateSolutionManagerConfiguration()</code> 	Manage the Solution Manager configuration
<ul style="list-style-type: none"> • <code>getDCNConfiguration()</code> • <code>updateDCNConfiguration()</code> 	Manage data change notification (DCN) configuration
<ul style="list-style-type: none"> • <code>getWebContainerCommonConfiguration</code> • <code>updateWebContainerCommonConfiguration</code> • <code>commitWebContainerConfiguration</code> • <code>refreshWebContainerConfiguration</code> 	Manage the Hybrid Web Container configuration
<ul style="list-style-type: none"> • <code>getHTTPListenerConfigurations</code> • <code>addHTTPListenerConfiguration</code> • <code>deleteHTTPListenerConfiguration</code> • <code>updateHTTPListenerConfiguration</code> 	Manage HTTP listener configurations

Methods	Description
<ul style="list-style-type: none"> • <code>getSSLSecurityProfileConfigurations()</code> • <code>addSSLSecurityProfileConfiguration()</code> • <code>deleteSSLSecurityProfileConfiguration()</code> • <code>updateSSLSecurityProfileConfiguration()</code> 	Manage the SSL security profile configuration
<ul style="list-style-type: none"> • <code>getKeyStoreConfiguration</code> • <code>updateKeyStoreConfiguration</code> 	Manage the keystore configuration
<ul style="list-style-type: none"> • <code>getTrustStoreConfiguration</code> • <code>updateTrustStoreConfiguration</code> 	Manage the truststore configuration
<ul style="list-style-type: none"> • <code>refresh</code> • <code>commit</code> 	Manage the server configurations locally

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Managing Clusters*

See also

- *SUPCluster API Changes* on page 329
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPServerConfiguration API Changes

API changes for `SUPServerConfiguration`, used to administer the SAP Mobile Server. With this release, most `SUPServerConfiguration` methods have been deprecated, except the `OutboundEnabler` and JVM-related API.

Table 8. New SUP Server Configuration Methods

Methods	Description
<ul style="list-style-type: none"> • <code>startOutboundEnablers</code> • <code>stopOutboundEnablers</code> • <code>getOutboundEnabler</code> • <code>getOutboundEnablers</code> • <code>saveOutboundEnabler</code> • <code>updateOutboundEnabler</code> • <code>deleteOutboundEnabler</code> • <code>deleteOutboundEnablers</code> • <code>addOutboundEnablerCertificates</code> • <code>getOutboundEnablerCertificates</code> • <code>deleteOutboundEnablerCertificateFile</code> • <code>deleteOutboundEnablerCertificateFiles</code> 	Manage outbound enablers.

Table 9. Deprecated SUP Server Configuration Administration Methods

Methods	Description
<ul style="list-style-type: none"> • <code>getReplicationSyncServerConfiguration()</code> • <code>updateReplicationSyncServerConfiguration()</code> 	Manage the replication synchronization server configuration
<ul style="list-style-type: none"> • <code>getMessagingSyncServerConfiguration()</code> • <code>updateMessagingSyncServerConfiguration()</code> 	Manage the messaging synchronization server configuration
<ul style="list-style-type: none"> • <code>getConsolidatedDatabaseConfiguration()</code> 	Manage the viewing of the consolidated database configuration

Methods	Description
<ul style="list-style-type: none"> • <code>getAdministrationListenerConfiguration()</code> • <code>updateAdministrationListenerConfiguration()</code> 	<p>Manage the administration listener configuration</p>
<ul style="list-style-type: none"> • <code>getHTTPListenerConfigurations()</code> • <code>addHTTPListenerConfiguration()</code> • <code>deleteHTTPListenerConfiguration()</code> • <code>updateHTTPListenerConfiguration()</code> 	<p>Manage the HTTP listener configuration</p>
<ul style="list-style-type: none"> • <code>getSecureHTTPListenerConfigurations()</code> • <code>addSecureHTTPListenerConfiguration()</code> • <code>deleteSecureHTTPListenerConfiguration()</code> • <code>updateSecureHTTPListenerConfiguration()</code> 	<p>Manage the HTTPS listener configuration</p>
<ul style="list-style-type: none"> • <code>getSSLSecurityProfileConfigurations()</code> • <code>addSSLSecurityProfileConfiguration()</code> • <code>deleteSSLSecurityProfileConfiguration()</code> • <code>updateSSLSecurityProfileConfiguration()</code> 	<p>Manage the SSL security profile configuration</p>
<ul style="list-style-type: none"> • <code>getKeyStoreConfiguration()</code> • <code>updateKeyStoreConfiguration()</code> 	<p>Manage the key store configuration</p>
<ul style="list-style-type: none"> • <code>getTrustStoreConfiguration()</code> • <code>updateTrustStoreConfiguration()</code> 	<p>Manage the truststore configuration</p>

Methods	Description
<ul style="list-style-type: none"> • <code>getApplePushNotificationConfigurations()</code> • <code>addApplePushNotificationConfiguration()</code> • <code>deleteApplePushNotificationConfiguration()</code> • <code>updateApplePushNotificationConfiguration()</code> 	Manage Apple native notification configuration

Table 10. Deprecated SUPServerConfiguration Methods

Methods	Description
<ul style="list-style-type: none"> • <code>ReplicationSyncServer</code> • <code>ReplicationNotifier_Push</code> • <code>ReplicationPushNotificationGateway</code> 	Manage the replication synchronization server
<ul style="list-style-type: none"> • <code>MessagingSyncServer</code> 	Manage the messaging synchronization server
<ul style="list-style-type: none"> • <code>AdministrationListener</code> • <code>SecureAdministrationListener</code> • <code>HTTPListener</code> • <code>SecureHTTPListener</code> 	Manage the administration listener
<ul style="list-style-type: none"> • <code>SSLSecurityProfile</code> • <code>TrustStore</code> • <code>ReplicationNotifier_Pull</code> • <code>OCSF</code> 	Manage metadata-based server components
<ul style="list-style-type: none"> • <code>getSSLSecurityProfileConfiguration</code> • <code>addSSLSecurityProfileConfiguration</code> • <code>deleteSSLSecurityProfileConfiguration</code> • <code>updateSSLSecurityProfileConfiguration</code> 	Manage the security profile configurations
<ul style="list-style-type: none"> • <code>getKeyStoreConfiguration</code> • <code>updateKeyStoreConfiguration</code> 	Manage the keystore configuration

Methods	Description
<ul style="list-style-type: none"> • <code>getTrustStoreConfiguration</code> • <code>updateTrustStoreConfiguration</code> 	Manage the truststore configuration
<ul style="list-style-type: none"> • <code>getHTTPListenerConfigurations</code> • <code>addHTTPListenerConfiguration</code> • <code>deleteHTTPListenerConfiguration</code> • <code>updateHTTPListenerConfiguration</code> 	Manage HTTP listener configurations

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Controlling SAP Mobile Server (SUPServer Interface)*

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPRelayServer API Changes

API changes for the SUPRelayServer interface, used to manage the Relay Server or Relay Server cluster for the SAP Mobile Server.

Table 11. New SUPRelayServer Methods

New Methods	Description
<ul style="list-style-type: none"> • <code>getOutboundEnablerProxy</code> • <code>getOutboundEnablerProxies</code> • <code>saveOutboundEnablerProxy</code> • <code>updateOutboundEnablerProxy</code> • <code>deleteOutboundEnablerProxy</code> • <code>deleteOutboundEnablerProxies</code> 	Manage outbound enabler proxy configurations

New Methods	Description
<ul style="list-style-type: none"> • <code>getRelayServer</code> • <code>getRelayServers</code> • <code>saveRelayServer</code> • <code>updateRelayServer</code> • <code>deleteRelayServer</code> • <code>deleteRelayServers</code> 	Manage relay server configurations
<ul style="list-style-type: none"> • <code>getServerFarm</code> • <code>saveServerFarm</code> • <code>updateServerFarm</code> • <code>deleteServerFarm</code> • <code>deleteServerFarms</code> 	Manage configurations of server farms
<ul style="list-style-type: none"> • <code>getServerNode</code> • <code>saveServerNode</code> • <code>updateServerNode</code> • <code>deleteServerNode</code> • <code>deleteServerNodes</code> 	Manage configurations of server nodes

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Managing Relay Servers*

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330
- *SUPServerConfiguration API Changes* on page 333
- *SUPDomain API Changes* on page 338
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPDomain API Changes

API changes for SUPDomain, used to manage domains and their properties.

Table 12. New SUPDomain Methods

Methods	Description
<ul style="list-style-type: none"> • exportPackage • importPackage 	Manage the importing and exporting of packages
<ul style="list-style-type: none"> • getDefaultSecurityConfiguration • setDefaultSecurityConfiguration 	Manage the retrieval and setting of the default security configuration for a domain

Documented in: *Developer Guide: SAP Mobile Server Runtime, Managing Domains*

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPMobileWorkflow API Changes

API changes for SUPMobileWorkflow, used to administer mobile workflow packages. This API has been deprecated. Use the SUPMobileHybridApp API instead.

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPMobileHybridApp API Changes

API changes for SUPMobileHybridApp, used to manage Hybrid Apps and their properties.

Note: SUPMobileHybridApp replaces SUPMobileWorkflow, which has been deprecated.

Table 13. New SUPMobileHybridApp Methods

Methods	Description
<ul style="list-style-type: none"> • exportMobileHybridApp • importMobileHybridApp 	Manage the importing and exporting of Hybrid Apps
<ul style="list-style-type: none"> • setDefaultMobileHybridAppForDevices • unsetDefaultMobileHybridAppforDevices • getMobileHybridAppCustomIcons 	Manage modifying Hybrid Apps
<ul style="list-style-type: none"> • assignMobileHybridAppToTemplates () • unassignMobileHybridAppFromTemplates () • getTemplateHybridAppAssignments () • getTemplateMobileHybridAppStatus () • setDefaultMobileHybridAppforTemplates () • unsetDefaultMobileHybridAppforTemplates () 	Manage Hybrid App templates
<ul style="list-style-type: none"> • getMobileHybridAppClientVariables • setMobileHybridAppClientVariables 	Manage client variables for Hybrid Apps

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Managing Mobile Workflows*

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338

- *SUPMobileWorkflow API Changes* on page 338
- *SUPApplication API Changes* on page 340
- *SUPSecurityConfiguration API Changes* on page 341

SUPApplication API Changes

SUPApplication changes help manage applications, application connections, and application connection templates.

Table 14. New SUPApplication Methods

Methods	Description
<code>createApplication</code>	Create an Agentry application with or without a package
<code>deployAgentryApplication</code>	Deploy an Agentry application to the Agentry server.
<ul style="list-style-type: none"> • <code>startAgentryApplication</code> • <code>stopAgentryApplication</code> • <code>restartAgentryApplication</code> 	Manage the application instance on the Agentry server.
<ul style="list-style-type: none"> • <code>getAgentryApplicationLogSetting</code> • <code>rollAgentryApplicationlog</code> 	Manage the Agentry application log settings.
<ul style="list-style-type: none"> • <code>getAgentryApplication DefinitionFiles</code> • <code>getAgentryApplication LocalesFiles</code> • <code>getAgentryApplication ResourceFiles</code> • <code>getAgentryApplicationConfiguration</code> • <code>updateAgentryApplicationConnection-Setting</code> 	Manage the Agentry application settings.
<code>GetApplicationType</code>	Retrieve the Agentry client SDK type.
<code>downloadAgentryApplication</code>	Download the Agentry client application.
<code>setAgentryApplicationStatus</code>	Enable or disable the Agentry application on the Agentry server.

Methods	Description
<ul style="list-style-type: none"> • <code>getAgentryActiveUsers</code> • <code>disconnectAgentryActiveUsers</code> 	Manage Agentry application users.
<ul style="list-style-type: none"> • <code>getOperationSystemInfo</code> • <code>getAgentryApplicationNodeStatus</code> 	Get the information from the Agentry server node.

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Managing Applications and Application Connections and Templates*

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPSecurityConfiguration API Changes* on page 341

SUPSecurityConfiguration API Changes

API updates for `SUPSecurityConfiguration`, used to define security providers for authentication, authorization, and auditing.

Table 15. New SUPSecurityConfiguration Methods

Methods	Description
<ul style="list-style-type: none"> • <code>createLogicalRole</code> • <code>deleteLogicalRole</code> • <code>getRoleMappings</code> • <code>updateLogicalRole</code> 	Manage role mappings

Documented in: *Developer Guide: SAP Mobile Server Runtime*, see *Configuring Security Configurations*

See also

- *SUPCluster API Changes* on page 329
- *SUPConfiguration API Changes* on page 330

Management API

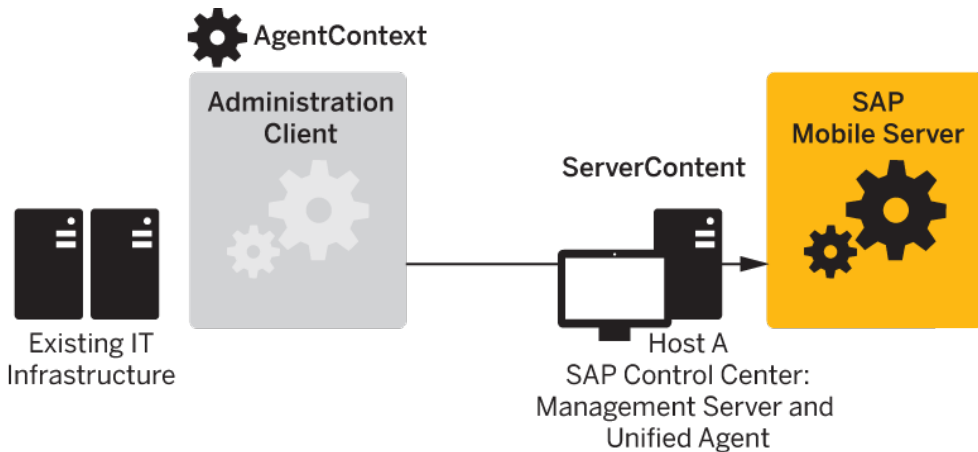
- *SUPServerConfiguration API Changes* on page 333
- *SUPRelayServer API Changes* on page 336
- *SUPDomain API Changes* on page 338
- *SUPMobileWorkflow API Changes* on page 338
- *SUPMobileHybridApp API Changes* on page 339
- *SUPApplication API Changes* on page 340

Management API

The client you create connects to SAP Mobile Server through SAP Control Center and SAP Unified Agent.

For example, as this illustration shows, connections are established using an `AgentContext` and a `ServerContext`:

Figure 1: Establishing Connections Using AgentContext and ServerContext



You do not need to create an instance of `AgentContext`. If none is defined, a default one is created by the `ServerContext` using its host value and default agent port (9999).

Note: Calling `deleteXxxx()` on a non-existent object causes the API to return silently and not throw an exception.

Contexts

A context is a lightweight, immutable object that is used to retrieve a specific administration interface instance. You create a connection to the SAP Mobile Server when you invoke an API (such as `ping`) on a supported interface (such as `SUPServer`), but not when context objects (such as `AgentContext` or `ServerContext`) are initialized. There is no need to maintain the states of contexts because state changes are not supported.

The administration client API includes these contexts:

Context	Description
AgentContext	Optional. Connects to the Unified Agent that acts as a proxy and manages the connection to the SAP Mobile Server instance identified in the ServerContext.
DefaultAdmin-Context	The super class of other concrete context classes.
AdminContext	The AdminContext is an interface that all context classes implement.
ServerContext	Required to connect to the SAP Mobile Server instance. If you do not specify an AgentContext, the ServerContext creates one for you using default values. See <i>Connecting to an SAP Mobile Server Instance</i> . Use this context to retrieve the ClusterContext.
ClusterContext	Required to manage a specific cluster. Use this context to retrieve the DomainContext.
DomainContext	Required to manage a specific domain. Use this context to retrieve the PackageContext.
PackageContext	Required to deploy and manage a package. Use this context to retrieve the MBOContext.
MBOContext	Required to manage a mobile business object. Use this context to retrieve the OperationContext.
OperationContext	Required to manage an operation.
SecurityContext	Required to manage the security for the platform.

For details on these classes, and the methods that implement them, see the Javadocs for `com.sybase.sup.admin.client`.

See also

- *Connecting to an SAP Mobile Server Instance* on page 348

Administration Interfaces

The Management API uses several interfaces that contain operations which can be invoked by custom code to perform management of the SAP Mobile Server.

The Management API includes these administration interfaces:

Interface	Includes Methods That
SUPServer	Command and control operations for an SAP Mobile Server instance, for example start , stop , and ping .
SUPCluster	Manage cluster security, monitoring configuration and domain creation for a cluster instance, and so on.
SUPDomain	Manage domains, deploy packages to a domain, set security configurations for a domain, and so on.
SUPPackage	Configure packages by setting up subscriptions, configuring cache groups, configuring endpoint properties, and so on.
SUPMobileBusinessObject	View mobile business object properties, operations, errors, endpoints, and so on.
SUPOperation	View operation properties, errors, endpoints, and so on.
SUPApplication	Manage applications, application connections, and application connection templates
SUPMonitor	Perform monitoring functions like viewing histories, summaries, details, and performance data for various platform components, and export data as required.
SUPServerLog	View, filter, delete and refresh logs, configure appenders, and so on, for SAP Mobile Server and its embedded services like replication and messaging synchronization.
SUPDomainLog	Configure domain log settings and view, filter, delete domain logs entries, and so on.
SUPConfiguration	Configure an SAP Mobile Server cluster.
SUPServerConfiguration	Configure JVM performance, and manage and configure Outbound Enablers.
SUPSecurityConfiguration	Create, manage, and configure a security configuration with at least one authentication provider. You can add other providers (authentication, authorization, attribution, and audit) as required.
SUPMobileWorkflow	This interface has been deprecated. Use the SUPMobileHybridApp interface instead. Manage and configure deployed mobile workflow packages.
SUPMobileHybridApp	Manage and configure deployed Hybrid App packages.
SUPRelayServer	Manage and configure Relay Servers used in SAP Mobile Platform.
SUPDeviceUser	This interface has been deprecated.

For details on these classes, and the methods that implement them, see the Javadocs for `com.sybase.sup.admin.client`.

See also

- *Client Metadata* on page 590

SUPObjectFactory

Once a context has been instantiated, pass it to a specific method of `SUPObjectFactory` to retrieve an administration interface. You can then start administration by calling methods of the interface.

The methods in the `SUPObjectFactory` class can accept an instance of `AdminContext` as a parameter. For example, to get an administration interface of `SUPServer`, you must create an instance of `ServerContext` with the correct information and pass it to `SUPObjectFactory.getSUPServer()`.

`SUPObjectFactory` provides a `shutdown()` method to cleanly shut down an application that uses the API. See the Javadocs for details.

Metadata

Metadata-based configuration is used by these administration components:

- SAP Mobile Server configuration properties
- SAP Mobile Server log configuration properties
- Security configurations and the providers used in those configurations
- Endpoint connection properties

See also

- *Client Metadata* on page 590

Exceptions and Error Codes

The Management API throws only one checked exception, `SUPAdminException`.

An error code is associated with each thrown `SUPAdminException`, so that developers can easily diagnose what happened when the exception is thrown.

Note: See *Developer Guide: SAP Mobile Server Runtime > Management API > Error Code Reference* for a list of predefined error codes.

Best Practices

Observe these best practices.

- **Thread safety** – The admin API client library is not thread safe, so external synchronization is required when calling the APIs concurrently from multiple threads.
- **Performance** – When managing multiple SAP Mobile Platform clusters, for performance considerations, it is best to connect to SAP Control Center co-located with the managed SAP Mobile Platform cluster. Although connecting to another SAP Control Center (as long as they share the same credentials) to perform management is supported, performance may not be as good as the former approach.
- **Commit configuration** – The `SUPServerConfiguration`, `SUPSecurityConfiguration`, and `SUPConfiguration` APIs use a local cache and upload changes later to the SAP Mobile Server. You must perform a commit to upload changes to the SAP Mobile Server and then refresh.
- **Error handling** – For error handling, use the error code returned in the exception. Also, by calling the `getErrorCode()` method of `SUPAdminException`, a string representation of the structured error code can be retrieved, which can further the centralized error code handling.

Getting Started with Client Development

An SAP Mobile Platform development cycle includes several steps.

1. *Required Files*

The following files are required in your class path.

2. *Starting Required Services*

Before beginning development, you must start required SAP Mobile Platform services.

3. *Connecting to an SAP Mobile Server Instance*

`AgentContext` and `ServerContext` are lightweight, immutable Java objects.

4. *Developing Client Contexts, Objects, and Operations*

Once you have an instance of `ServerContext`, you can create other contexts from it.

Prerequisites

Review this list to understand what prerequisites to consider before starting the development of a custom administration tool within an existing enterprise-level administration framework.

- A development environment that supports Java development, for example, Eclipse.
- Optionally, if you want to install SAP Control Center, it must be installed on the same host as SAP Mobile Server.

Required Files

The following files are required in your class path.

- sup-admin-pub-client.jar
- sup-admin-pub-common.jar
- castor-1.2.jar
- commons-beanutils-core-1.7.0.jar
- commons-lang-2.2.jar
- commons-logging-1.1.1.jar
- commons-pool-1.4.jar
- sup-at-lite.jar
- sup-mms-admin-api-lite.jar
- uaf-client.jar
- log4j-1.2.6.jar
- commons-codec-1.3.jar
- log4j.properties (the file residing in the sample folder can be a template)

By default, the sup-admin-pub-client.jar, and sup-admin-pub-common.jar files are installed to the `SMP_HOME\Servers\UnwiredServer\AdminClientAPI\com.sybase.sup.adminapi` folder. All other jar files can be found in the `SMP_HOME\Servers\UnwiredServer\AdminClientAPI\com.sybase.sup.adminapi\lib` folder.

Note: If you have Xerces J-Parser installed and have xerces.jar (the parser class files) in your class path, the xerces.jar library may cause a class conflict with SAP Mobile Platform. This problem only occurs in certain circumstances when JDK 6 is used with Xerces. If this problem occurs, you must remove this jar from your class path.

Starting Required Services

Before beginning development, you must start required SAP Mobile Platform services.

Prerequisites

Ensure the required services are installed on the same host.

Task

By starting required services, you start the servers and dependent services. For a complete list of SAP Mobile Platform services, see *System Administration > System Reference > SAP Mobile Platform Windows Services*.

1. Click the **Start SAP Mobile Platform Services** desktop shortcut to start SAP Mobile Server and the dependent services.

2. Use the Services Control Panel to verify that the Windows service named SAP Control Center X.X is started. If it has not, start it by selecting the service and clicking **Start**.

Connecting to an SAP Mobile Server Instance

`AgentContext` and `ServerContext` are lightweight, immutable Java objects.

Creating either of these objects does not immediately establish a connection to either SAP Control Center or the SAP Mobile Server.

1. (Optional) Create an `AgentContext` object.

The default constructor creates an instance with `host="localhost"`, `port="9999"`, `user=""` and `password=""`. The constructor in this sample creates an instance with `host"<host name>"`, `port="9999"`, `user="supAdmin"` and `password="supPwd"`:

```
AgentContext agentContext = new AgentContext();
agentContext = new AgentContext("<host name>", 9999, "supAdmin",
"supPwd");
```

2. Create a `ServerContext` object.

Every `ServerContext` instance has an `AgentContext` instance. When you instantiate `ServerContext`, you can pass an instance of `AgentContext` to the constructor. If you do not specify an `AgentContext`, the constructor automatically creates an `AgentContext` with the same host, user name, and password values as those defined in the `ServerContext`.

It also assigns 9999 as the port number for `AgentContext`, for these reasons:

- SAP Mobile Server and SAP Control Center are installed on the same host, and they share the same security provider.
- By default, SAP Control Center listens on port 9999. The administration API connects to SAP Control Center using this port.

This sample creates a `ServerContext` that uses values of `supAdmin` and `supPwd` for the user name and password, and uses secure port (2001) by specifying "true" in the last parameter:

```
ServerContext serverContext = new ServerContext();
serverContext = new ServerContext("<host name>", 2001, "supAdmin",
"supPwd", true);
```

The usage of secure port does not require server certificate installation on the client-side. It is assumed that server is configured with a valid and secure certificate for transport level security, and client authentication is done via the security provider assigned to the "admin" security configuration.

See also

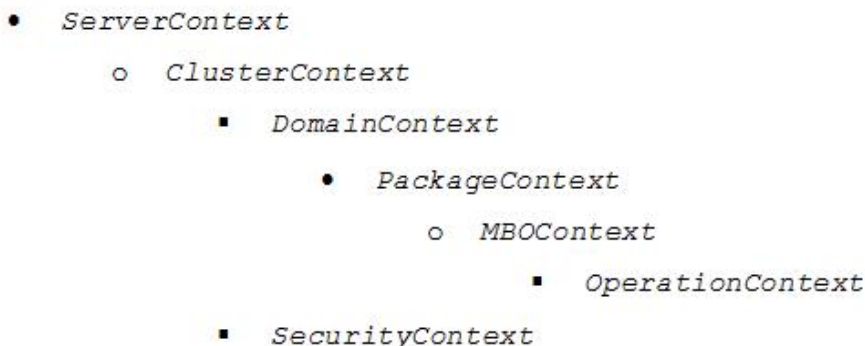
- *Contexts* on page 342

Developing Client Contexts, Objects, and Operations

Once you have an instance of `ServerContext`, you can create other contexts from it.

1. Create required client artifacts.

- Create the context objects you require. The following diagram illustrates the subclasses of `AdminContext` and their logical hierarchy.



The following code fragment creates multiple contexts for cluster, security, domain, package, mobile business objects, and operations:

```
ClusterContext clusterContext =
serverContext.getClusterContext("<cluster name>");
SecurityContext securityContext =
clusterContext.getSecurityContext("<security configuration
name>");
DomainContext domainContext =
clusterContext.getDomainContext("<domain name>");
PackageContext packageContext =
domainContext.getPackageContext("<package name>");
MBOContext mboContext = packageContext.getMBOContext("<MBO
name>");
OperationContext operationContext =
mboContext.getOperationContext("<operation name>");
```

- Call methods of `SUPObjectFactory` to create the administration interface required. For example, to create an object of `SUPServer`, pass an instance of `ServerContext` to `SUPObjectFactory` by calling:

```
SUPObjectFactory.getSUPServer(serverContext);
```

- ### 2. Once the administration session ends, clean the resources held by the API by calling `SUPObjectFactory.shutdown()`. This method is provided only to help your administration application exit cleanly, and is not designed to be called after each administration operation.

For example:

```
SUPObjectFactory.shutdown();
```

- ### 3. Build the client application.

Code Samples

Use the javadoc for the Management API package with the interface code samples to understand how to program a custom administration client.

Code samples are organized by the interface used.

Controlling SAP Mobile Server (SUPServer Interface)

The `SUPServer` interface allows you to manage the SAP Mobile Server.

Operations you can perform with this interface include:

- Starting an administration session for an SAP Mobile Server instance.
- Retrieving SAP Mobile Server properties and status.
- Performing command and control actions like starting and stopping.

Session Start-up

Starts the management of an SAP Mobile Server instance.

Syntax

```
public static SUPServer getSUPServer(ServerContext serverContext)
throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Session Start-up**

```
SUPServer supServer =
SUPObjectFactory.getSUPServer(serverContext);
```

Usage

When an instance of `SUPServer` is returned from the `SUPObjectFactory`, call its method. The state of the connection to the SAP Mobile Server is automatically managed; an explicit connection to the SAP Mobile Server is not required.

Server Properties Retrieval

Retrieves the general properties of the SAP Mobile Server instance.

Syntax

```
ServerVO getProperties() throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Examples

- **Getting properties** – gets the properties for a server instance named `ServerVO`:

```
ServerVO svo = supServer.getProperties();
```

Status Verification

Checks if the SAP Mobile Server instance is available.

Syntax

```
void ping() throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Ping** – pings an SAP Mobile Server to see if it is available:

```
supServer.ping();
```

Server Start-up

Starts an SAP Mobile Server instance.

Syntax

```
void start() throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Startup**

```
supServer.start();
```

Server Shutdown

Stops an SAP Mobile Server instance.

Syntax

```
void stop() throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Shutdown**

```
supServer.stop();
```

Server Restart

Restarts an SAP Mobile Server instance.

Syntax

```
void restart() throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Restart**

```
supServer.restart();
```

Managing Clusters

The `SUPCluster` interface allows you manage the cluster to which the SAP Mobile Server instance belongs. You can also manage SAP Mobile Server configurations for all nodes in the cluster

Operations you can perform with this interface include:

- Listing member servers, suspending and resuming member servers
- Listing, creating, and deleting domains
- Listing, creating, and deleting security configurations
- Listing, creating, updating, and deleting monitoring configurations, deleting monitoring data

- Listing, creating, updating, and deleting domain administrators
- Listing, updating, and deleting administration users
- Retrieving licensing information.

The SAP Mobile Server configuration consists of the following components, all of which are metadata-based:

- Communication
 - Administration Listener
 - HTTP / HTTPS Listener
 - SSL Security Profile
 - Key Store
 - Trust Store
- Messaging
- Replication

Note: The `SUPCluster` interface also contains methods for managing monitoring profiles in a cluster, and monitoring data store policies and domain log data store policies. These methods are described in *Developer Guide: SAP Mobile Server Runtime > Management API > Code Samples > Monitoring SAP Mobile Platform Components*.

Start Cluster Management

Starts the management of an SAP Mobile Server cluster.

Syntax

```
public static SUPCluster getSUPCluster(ClusterContext
clusterContext) throws SUPAdminException;
```

Examples

- **Cluster startup** – starts the management of the specified cluster.

```
clusterContext = serverContext.getClusterContext("<cluster
name>");
SUPCluster supCluster =
SUPObjectFactory.getSUPCluster(clusterContext);
```

Usage

When an instance of `SUPCluster` is returned from the `SUPObjectFactory`, call its method.

SAP Mobile Servers Retrieval

Retrieves a list of servers that are members in a cluster.

Syntax

```
Collection<ServerVO> getServers() throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

Examples

- **Getting member servers** – lists the servers that are members of a cluster:

```
Collection<ServerVO> svos = supCluster.getServers();
```

Resume an SAP Mobile Server

Resumes an SAP Mobile Server in a cluster.

Syntax

```
void resume(String name) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

Examples

- **Resume a server** – resumes an SAP Mobile Server in a cluster:

```
supCluster.resume("<member server name>");
```

Suspend an SAP Mobile Server

Suspends a member server in a cluster.

Prerequisite: The server must belong to a Relay Server.

Syntax

```
void suspend(String name) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

Examples

- **Suspend a server** – suspends an SAP Mobile Server in a cluster:

```
supCluster.suspend("<member server name>");
```

Retrieval of Domains

Retrieves the domains in a cluster.

Syntax

```
Collection<String> getDomains() throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Examples

- **Retrieval of domains** – retrieves the domains in a cluster.

```
Collection<String> domains = supCluster.getDomains();
```

Creation of Domains

This method has been deprecated. Creates domains in a cluster.

Syntax

```
void createDomain(String name) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Creation of domains** – creates, in the cluster, the domain specified by "<domain name>".

```
supCluster.createDomain("<domain name>");
```

Deletion of Domains

Deletes domains from a cluster.

Syntax

```
void deleteDomains(Collection<String> names) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Deletion of domains** – deletes, from the cluster, the domains in the specified array.

```
supCluster.deleteDomains(Arrays.asList(new String[] {  
    "<domain name 1>", "<domain name 2>" }));
```

Retrieval of Security Configurations

Retrieves a list of security configurations in a cluster.

Syntax

```
Collection<String> getSecurityConfigurations() throws  
SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Examples

- **Retrieval of security configurations** – lists the security configurations in a cluster.

```
Collection<String> securityConfigurations=  
supCluster.getSecurityConfigurations();
```

Creation of a Security Configuration

Creates a security configuration in a cluster.

Syntax

```
void createSecurityConfiguration(String name) throws  
SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Creation of a security configuration** – creates a security configuration of the specified name in the cluster:

```
supCluster.createSecurityConfiguration("<security configuration  
name>");
```

Deletion of a Security Configuration

Deletes a security configuration from the cluster.

Syntax

```
void deleteSecurityConfigurations(Collection<String> names) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

Examples

- **Deletion of a security configuration** – deletes a security configuration from the cluster.

```
supCluster.deleteSecurityConfigurations(securityConfigurations);
```

Retrieval of Domain Administrators

Retrieves a list of domain administrators in a cluster.

Syntax

```
Collection<DomainAdministratorVO> getDomainAdministrators() throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

Examples

- **Retrieval of domain administrators** – retrieves a list of domain administrators in a cluster:

```
//List domain administrators
for (DomainAdministratorVO davo :
supCluster.getDomainAdministrators()) {
    System.out.println(davo.getLoginName());
}
```

Creation of a Domain Administrator

Creates a domain administrator in the cluster.

Syntax

```
void createDomainAdministrator(DomainAdministratorVO
domainAdministrator) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Creation of a domain administrator** – creates a domain administrator in the cluster:

```
//Create a domain administrator
DomainAdministratorVO davo = new DomainAdministratorVO();
davo.setLoginName("<new domain administrator login name>");
supCluster.createDomainAdministrator(davo);
```

Update of a Domain Administrator

Updates a domain administrator in the cluster.

Syntax

```
void updateDomainAdministrator(DomainAdministratorVO
domainAdministrator) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Update of a domain administrator** – updates a domain administrator in the cluster by setting the login name and company name:

```
//Update a domain administrator
davo = new DomainAdministratorVO();
davo.setLoginName("<domain administrator login name>");
davo.setCompanyName("SAP");
supCluster.updateDomainAdministrator(davo);
```

Deletion of a Domain Administrator

Deletes a domain administrator from the cluster.

Syntax

```
void deleteDomainAdministrator(DomainAdministratorVO
domainAdministrator) throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Deletion of a domain administrator** – deletes the specified domain administrator from the cluster:

```
//Delete a domain administrator
davo = new DomainAdministratorVO();
davo.setLoginName("<domain administrator login name>");
supCluster.deleteDomainAdministrator(davo);
```

Retrieval and Setting of Authentication Cache Timeout

Retrieves and sets the authentication cache timeout from a cluster.

Syntax

```
Long timeout getAuthenticationCacheTimeout () throws
SUPAdminException;

void setAuthenticationCacheTimeout(user, timeout);
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

Examples

- **Retrieve and set authentication cache timeout** – retrieves and sets the specified authentication cache timeout from a cluster:

```
Long timeout = supCluster.getAuthenticationCacheTimeout("admin");
supCluster.setAuthenticationCacheTimeout("admin", 200L);
timeout = supCluster.getAuthenticationCacheTimeout("admin");
assertEquals(new Long(200), timeout);
```

Retrieval and Setting of Cluster Properties

Retrieves and sets the properties of a cluster.

Syntax

```
ClusterPropertiesVO getClusterProperties () throws SUPAdminException;

void setClusterProperties(vo)
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

Examples

- **Retrieve or set cluster properties**

```
//Get cluster properties
ClusterPropertiesVO vo = supCluster.getClusterProperties();
//change cluster properties
vo.setClusterSyncDataSharedPathEnabled(true);
vo.setClusterSyncDataSharedPath("\\\\myhost\\newSharedPath");
//Set cluster properties
supCluster.setClusterProperties(vo);
```

Retrieval and Setting of Maximum Allowed Authentication Failures

Retrieves and sets the maximum number of allowed authentication failures.

Syntax

```
Integer getMaximumAllowedAuthenticationFailure(String
securityConfiguration) throws SUPAdminException;
```

```
void setMaximumAllowedAuthenticationFailure(String
securityConfiguration, Integer maximumAllowed) throws
SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

Examples

- **Retrieve or set cluster properties**

```
//Get maximum allowed authentication failures
Integer threshold=
supCluster.getMaximumAllowedAuthenticationFailure("admin");
//Set maximum allowed authentication failures
supCluster.setMaximumAllowedAuthenticationFailure("admin", 20);
```

Retrieval and Setting of Authentication Lock Duration

Retrieves and sets the duration for authentication lock.

Syntax

```
Integer getAuthenticationLockDuration(String securityConfiguration)
throws SUPAdminException;
```

```
void setAuthenticationLockDuration(String securityConfiguration,
Integer duration) throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Examples

- **Retrieve or set authentication lock duration**

```
Integer duration =
supCluster.getAuthenticationLockDuration("admin");
supCluster.setAuthenticationLockDuration("admin", 3000);
```

Set HTTP Proxy Setting

Set the HTTP proxy setting.

Syntax

```
setHttpProxySetting
    throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Set HTTP Proxy Setting**

```
HttpProxySettingVO vo = cluster.getHttpProxySetting();
vo.setMessagingProxyMaxConnectionsPerHost(5500);
vo.setMessagingProxyThreadCounts(200);
vo.setReplicationProxyThreadCounts(2340);
vo.setMessagingProxyConnectionTimeout(100);
vo.setMessagingProxyConnectionIdleTime(150);
vo.setMessagingProxyResponseTimeout(200);
cluster.updateHttpProxySetting(vo);
```

Retrieve HTTP Proxy Setting

Retrieve the HTTP proxy setting.

Syntax

```
HttpProxySettingVO getHttpProxySetting()
    throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Examples

- **Get HTTP Proxy Setting**

```
HttpProxySettingVO hp = supCluster.getHttpProxySetting();
System.out.println(hp.getMessagingProxyMaxConnectionsPerHost());
System.out.println(hp.getMessagingProxyThreadCounts());
System.out.println(hp.getReplicationProxyThreadCounts());
System.out.println(hp.getMessagingProxyConnectionIdleTime());
System.out.println(hp.getMessagingProxyConnectionTimeout());
System.out.println(hp.getMessagingProxyResponseTimeout());
```

Update HTTP Proxy Setting

Update the HTTP proxy setting.

Syntax

```
void updateHttpProxySetting(HttpProxySettingVO proxySetting)
    throws SUPAdminException;
```

Parameters

- **proxySetting** – The proxy setting that you want to update.

Returns

If successful, returns silently. If unsuccessful, throws `SUPAdminException`.

Examples

- **Update HTTP Proxy Setting**

```
HttpProxySettingVO hp = new HttpProxySettingVO();
hp.setMessagingProxyThreadCounts(20);
hp.setReplicationProxyThreadCounts(17);
hp.setMessagingProxyMaxConnectionsPerHost(3000);
supCluster.updateHttpProxySetting(hp);
```

- **Update HTTP Proxy Setting With Messaging Properties**

```
HttpProxySettingVO vo = cluster.getHttpProxySetting();
vo.setMessagingProxyMaxConnectionsPerHost(5500);
vo.setMessagingProxyThreadCounts(200);
vo.setReplicationProxyThreadCounts(2340);
vo.setMessagingProxyConnectionTimeout(100);
vo.setMessagingProxyConnectionIdleTime(150);
vo.setMessagingProxyResponseTimeout(200);
cluster.updateHttpProxySetting(vo);
```


Update HTTP Log Setting

Update the HTTP log setting.

Syntax

```
setHttpLogSetting
    throws SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

Examples

- **Update HTTP Log Setting**

```
ServerLogSettingVO vo =cluster.getServerLogSetting();
vo.setHttpLogEnable(true);
vo.setArchiveHttpLog(!vo.isArchiveHttpLog());
vo.setCompressHttpLogArchivement(!
vo.isCompressHttpLogArchivement());
vo.setHttpLogArchiveFileName("httpLog.txt");
vo.setMaxHttpLogFileSize("100K");
vo.setReuseHttpLogFile(!vo.isReuseHttpLogFile());
vo.setSeperateHttpLogFile(!vo.isSeperateHttpLogFile());
cluster.updateServerLogSetting(vo);
```

Retrieve Cluster Context for Web Container Configuration

Retrieve the cluster configuration from server and cache it locally. Only a SAP Mobile Platform administrator or the SAP Mobile Platform help desk can perform this.

This refreshes the cluster configurations, which includes following components: Management, Replication, Messaging, SolutionManager, RelayServer, SecurityProfile, KeyStore, TrustStore, OCSP, and Performance. The returned `ConfigurationValidationStatus` contains the validation status of the cluster configuration at server side.

Syntax

```
ConfigurationValidationStatus refreshClusterConfiguration()
    throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Commit Server Changes in Local Cache Refresh

Deliver the local server configuration to the server to be saved. Only a SAP Mobile Platform administrator can perform this.

All the metadata based configurations are delivered, including cluster configuration, Web container configuration, and configuration cache. The returned `ConfigurationValidationStatus` contains the validation status of the delivered SAP Mobile Platform configuration at server side.

Syntax

```
ConfigurationValidationStatus commit()  
    throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Refresh Server Changes from Local Cache

Retrieve all SAP Mobile Platform configurations from server and cache it locally. Only a SAP Mobile Platform administrator or the SAP Mobile Platform help desk can perform this.

This refreshes all meta-data based configurations, including: cluster configuration, web container configuration, and configuration cache. The returned `ConfigurationValidationStatus` contains the validation status of the SAP Mobile Platform configuration at server side.

Syntax

```
ConfigurationValidationStatus refresh()  
    throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

Retrieve Server Context for Cluster

Retrieve the server context associated with this cluster configuration.

Syntax

```
ClusterContext getContext();
```

Returns

Returns the server context.

Retrieve Management Configuration

Retrieve the management related configuration.

Syntax

```
SUPConfigurationComponentVO getManagementConfiguration()  
    throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException .

Examples

- **Retrieve Management configuration**

```
SUPConfiguration supConf = SUPObjectFactory  
    .getSUPConfiguration(clusterContext);  
supConf.refreshClusterConfiguration();  
SUPConfigurationComponentVO confVO = supConf  
    .getManagementConfiguration();  
System.out.println(confVO.getProperties());
```

Update Management Configurations

Update the properties of the management related configuration.

Syntax

```
void updateManagementConfiguration(SUPConfigurationComponentVO  
configurationComponent)  
    throws SUPAdminException;
```

Parameters

- **portVO** –

Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException

Examples

- **Update Management Configuration**

```
supConf.refreshClusterConfiguration();  
SUPConfigurationComponentVO confVO = supConf  
    .getManagementConfiguration();  
confVO.getProperties().put("sup.admin.port", "2003");
```

```
supConf.updateManagementConfiguration (confVO) ;  
supConf.commitClusterConfiguration () ;
```

Retrieve Security Profile Configuration

Retrieve a list of SSL security profile configurations.

Syntax

```
Collection<SUPConfigurationComponentVO>  
getSSLSecurityProfileConfigurations ()  
    throws SUPAdminException;
```

Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

Examples

- **Retrieve Security Profiles**

```
SUPConfiguration supConf = SUPObjectFactory  
    .getSUPConfiguration (clusterContext) ;  
supConf.refreshClusterConfiguration () ;  
for (SUPConfigurationComponentVO scvo :  
supConf.getSSLSecurityProfileConfigurations ()) {  
    System.out.println (scvo.getID ()) ;  
    System.out.println (scvo.getType ()) ;  
    System.out.println (scvo.getProperties ()) ;  
}
```

Add Security Profile Configuration

Add a new SSL security profile configuration.

Syntax

```
void addSSLSecurityProfileConfiguration (SUPConfigurationComponentVO  
serverComponent)  
    throws SUPAdminException;
```

Parameters

Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

Examples

- ```
scvo = new SUPConfigurationComponentVO (
SUPConfigurationComponentType.SSLSecurityProfile) ;
```

```
Map<String, String> properties = scvo.getProperties();
properties.put("sup.security.profile.name", name);
properties.put("securityCharacteristics", "intl");
properties.put("certificateLabel", "sample1");
scvo.setProperties(properties);
supConf.addSSLSecurityProfileConfiguration(scvo);
commit(supConf);
```

### **Delete Security Profile Configuration**

Delete a SSL security profile configuration.

#### **Syntax**

```
void deleteSSLSecurityProfileConfiguration(String serverComponentID)
 throws SUPAdminException;
```

#### **Parameters**

- **serverComponentID** – The ID of the security profile configuration you want to delete.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Update Security Profile Configuration**

Update a SSL security profile configuration.

#### **Syntax**

```
void
updateSSLSecurityProfileConfiguration(SUPConfigurationComponentVO
serverComponent)
 throws SUPAdminException;
```

#### **Parameters**

- **serverComponent** – The server component that you want to update.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Update Security Profile Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO scvo = supConf
 .getSSLSecurityProfileConfigurations().iterator().next();
scvo.getProperties().put("certificateLabel", "mycertificate");
```

```
supConf.updateSSLSecurityProfileConfiguration(scvo);
supConf.commitClusterConfiguration();
```

### **Retrieve Keystore Configuration**

Retrieve the properties of the key store configuration.

#### **Syntax**

```
SUPConfigurationComponentVO getKeyStoreConfiguration()
 throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException .

#### **Examples**

- **Retrieve**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getKeyStoreConfiguration();
System.out.println(confVO.getProperties());
```

### **Update Keystore Configuration**

Update the key store configuration.

#### **Syntax**

```
void updateKeyStoreConfiguration(SUPConfigurationComponentVO
serverComponent)
 throws SUPAdminException;
```

#### **Parameters**

- **serverComponent** – The server component you want to update.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### **Examples**

- **Update Keystore Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getKeyStoreConfiguration();
```

```
confVO.getProperties().put("sup.sync.sslkeystore_password", "
changeit");
supConf.updateKeyStoreConfiguration(confVO);
supConf.commitClusterConfiguration();
```

### **Retrieve Trust Store Configuration**

Retrieve the properties of the trust store configuration.

#### **Syntax**

```
SUPConfigurationComponentVO getTrustStoreConfiguration()
 throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException .

#### **Examples**

- **Update TrustStore Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getTrustStoreConfiguration();
System.out.println(confVO.getProperties());
```

### **Update Trust Store Configuration**

Update the trust store configuration.

#### **Syntax**

```
void updateTrustStoreConfiguration(SUPConfigurationComponentVO
serverComponent)
 throws SUPAdminException;
```

#### **Parameters**

- **serverComponent** – The server component that you want to update.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### **Examples**

- **Update TrustStore Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
```

```
.getTrustStoreConfiguration();
confVO.getProperties().put("sup.sync.sslkeystore_password", "
changeit");
supConf.updateTrustStoreConfiguration(confVO);
supConf.commitClusterConfiguration();
```

### **Enable/Disable OCSP Configuration**

Enable or disable OCSP configuration.

#### **Syntax**

```
void enableOCSPConfiguration(Boolean flag)
 throws SUPAdminException;
```

#### **Parameters**

- **flag** – Set the flag to TRUE to enable and to FALSE to disable.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### **Examples**

- **Enable OCSP Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
supConf.enableOCSPConfiguration(true);
supConf.commitClusterConfiguration();
```

### **Retrieve OCSP Configuration**

Retrieve the OCSP configuration. Only a SAP Mobile Platform administrator or the SAP Mobile Platform help desk can perform this.

#### **Syntax**

```
SUPConfigurationComponentVO getOCSPConfiguration() throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieve OCSP Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
```



```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getOCSPConfiguration();
System.out.println(confVO.getProperties());
```

### **Update OCSP Configuration**

Update the OCSP configuration. Only a SAP Mobile Platform administrator can perform this.

#### **Syntax**

```
void updateOCSPConfiguration(SUPConfigurationComponentVO
serverComponent)
 throws SUPAdminException;
```

#### **Parameters**

- **serverComponent** – The server component that you want to update.

#### **Returns**

If successful, returns silently. If unsuccessful, throws `SUPAdminException`.

#### **Examples**

- **Update OCSP Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getOCSPConfiguration();
confVO.getProperties().put("ocsp.responderURL", "http://
apple.com");
supConf.updateOCSPConfiguration(confVO);
supConf.commitClusterConfiguration();
```

### **Retrieve Performance Configuration**

Retrieve the performance configuration.

#### **Syntax**

```
SUPConfigurationComponentVO getPerformanceConfiguration()
 throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieve Performance Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getPerformanceConfiguration();
System.out.println(confVO.getProperties());
```

### Update Performance Configuration

Update the properties of the performance configuration.

### Syntax

```
void updatePerformanceConfiguration(SUPConfigurationComponentVO
serverComponent)
 throws SUPAdminException;
```

### Parameters

- **serverComponent** – The sever component that you want to update.

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### Examples

- **Update Performance Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getPerformanceConfiguration();
confVO.getProperties().put("sup.msg.inbound_count", "70");
supConf.updatePerformanceConfiguration(confVO);
supConf.commitClusterConfiguration();
```

### Retrieve Cache Configuration

Retrieve the distribution configuration. Only a SAP Mobile Platform administrator and SAP Mobile Platform help desk can perform this.

### Syntax

```
SUPConfigurationComponentVO getConfigurationCache()
 throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieve Configuration Cache**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshConfigurationCache();
SUPConfigurationComponentVO confVO = supConf
 .getConfigurationCache();
System.out.println(confVO.getProperties());
```

## Update Cache Configuration

Update the properties of the cache configuration. Only a SAP Mobile Platform administrator can do this.

## Syntax

```
void updateConfigurationCache(SUPConfigurationComponentVO
serverComponent)
 throws SUPAdminException;
```

## Parameters

- **serverComponentID** – The ID of the server component you want to update.
- **serverComponent** – The server component you want to update.

## Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Update Configuration Cache**

```
supConf.refreshConfigurationCache();
SUPConfigurationComponentVO confVO =
supConf.getConfigurationCache();
confVO.getProperties().put("cache.core.pool.size", "100");
supConf.updateConfigurationCache(confVO);
supConf.commitConfigurationCache();
```

## Retrieve Web Container Configuration

Retrieve the common web container configuration.

## Syntax

```
SUPConfigurationComponentVO getWebContainerCommonConfiguration()
 throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieve Common WebContainer Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshWebContainerConfiguration();
SUPConfigurationComponentVO confVO = supConf

 .getWebContainerCommonConfiguration();
System.out.println(confVO.getProperties());
```

### **Update Web Container Configuration**

Update the properties of the administration listener configuration.

### **Syntax**

```
void
updateWebContainerCommonConfiguration(SUPConfigurationComponentVO
webContainerComponent)
 throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Update Common WebContainer Configuration**

```
supConf.refreshWebContainerConfiguration();
SUPConfigurationComponentVO confVO = supConf

 .getWebContainerCommonConfiguration();
confVO.getProperties().put("maxFormContentSize", "2500000");
confVO.getProperties().put("gzipFilter", "true");
supConf.updateWebContainerCommonConfiguration(confVO);
supConf.commitWebContainerConfiguration();
```

### **Update Messaging Configuration**

Update the properties of the messaging related configuration.

### **Syntax**

```
void updateMessagingConfiguration(SUPConfigurationComponentVO
configurationComponent)
 throws SUPAdminException;
```

## Parameters

- **configurationComponent** – The messaging component configuration properties to update to.

## Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Update Messaging Configuration**

```
SUPConfigurationComponentVO confVO =
supConf.getMessagingConfiguration();
confVO.getProperties().put("msg.http.server.proxy.ports",
"5002");
supConf.updateMessagingConfiguration(confVO);
supConf.commitClusterConfiguration();
```

## Retrieve Messaging Configuration

Retrieve the messaging related configuration.

## Syntax

```
SUPConfigurationComponentVO getMessagingConfiguration()
throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, throws SUPAdminException.

## Examples

- **Get Messaging Configuration**

```
SUPConfiguration supConf =
SUPObjectFactory.getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO =
supConf.getMessagingConfiguration();
System.out.println(confVO.getProperties());
```

### Update Replication Configuration

Update the properties of the replication related configuration.

#### Syntax

```
void updateReplicationConfiguration (SUPConfigurationComponentVO
configurationComponent)
 throws SUPAdminException;
```

#### Parameters

- **configurationComponent** – The replication configuration component properties to update to.

#### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### Examples

- **Update SUP Replication Configuration**

```
supConf.updateReplicationConfiguration (confVO)
confVO.getProperties().put ("sup.sync.port", "2480");
confVO.getProperties().put ("sup.sync.httpsport", "2481");
confVO.getProperties().put ("sup.sync.protocol", "http,https");
confVO.getProperties().put ("sup.sync.e2ee_type", "RSA");
confVO.getProperties().put ("sup.sync.e2ee_private_key",
"Repository/Certificate/e2ee_private_key.key");
confVO.getProperties().put ("sup.sync.e2ee_private_key_password",
"sybase1");
confVO.getProperties().put ("sup.sync.e2ee_public_key",
"Repository/Certificate/e2ee_public_key.key");
confVO.getProperties().put ("sup.sync.certificate", "Repository/
Certificate/https_server_identity.crt");
confVO.getProperties().put ("sup.sync.certificate_password",
"sybase1");
confVO.getProperties().put ("sup.sync.public.certificate",
"Repository/Certificate/https_public_cert.crt");
confVO.getProperties().put ("relayserver.trusted_certs", "");
confVO.getProperties().put ("sup.user.options", "-zf");
supConf.updateReplicationMessagingConfiguration (confVO);
supConf.commitClusterConfiguration ();
```

### Retrieve Replication Configuration

Retrieve the replication related configuration.

#### Syntax

```
SUPConfigurationComponentVO getReplicationConfiguration()
 throws SUPAdminException;
```

**Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, throws `SUPAdminException`.

**Examples**

- **Retrieve SUP Replication Configuration**

```
SUPConfiguration supConf =
 SUPObjectFactory.getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO =
 supConf.getReplicationConfiguration();
System.out.println(confVO.getProperties());
```

**Retrieve HTTP Listener Configuration**

Retrieve a list of HTTP listener configurations.

**Syntax**

```
Collection<SUPConfigurationComponentVO>
getHTTPListenerConfigurations()
 throws SUPAdminException;
```

**Returns**

If successful, returns a collection of HTTP listener configurations. If unsuccessful, throws `SUPAdminException`.

**Examples**

- **Retrieve HTTP Listener Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshWebContainerConfiguration();
for(SUPConfigurationComponentVO scvo :
 supConf.getHTTPListenerConfigurations()){
 System.out.println(scvo.getID());
 System.out.println(scvo.getType());
 System.out.println(scvo.getProperties());
}
```

### **Add HTTP Listener Configuration**

Add a new HTTP listener configuration.

#### **Syntax**

```
void addHTTPListenerConfiguration(SUPConfigurationComponentVO
webContainerComponent)
 throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### **Examples**

- **Add HTTP Listener Configuration**

```
supConf.refreshWebContainerConfiguration();
SUPConfigurationComponentVO confVO =
supConf
 .getHTTPListenerConfigurations().iterator().next();
confVO.getProperties().put("port", "8100");
supConf.addHTTPListenerConfiguration(confVO);
supConf.commitWebContainerConfiguration();
```

### **Delete HTTP Listener Configuration**

Delete a HTTP listener configuration .

#### **Syntax**

```
void deleteHTTPListenerConfiguration(String
httpListenerComponentID)
 throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Delete HTTP Listener Configuration**

```
supConf.refreshWebContainerConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getHTTPListenerConfigurations().iterator().next();
supConf.deleteHTTPListenerConfiguration(confVO.getID());
supConf.commitWebContainerConfiguration();
```



## **Update HTTP Listener Configuration**

Update a HTTP listener configuration.

### **Syntax**

```
void updateHTTPListenerConfiguration(SUPConfigurationComponentVO
webContainerComponent)
 throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Update HTTP Listener Configuration**

```
supConf.refreshWebContainerConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getHTTPListenerConfigurations().iterator().next();
confVO.getProperties().put("maxThreads", "50");
supConf.updateHTTPListenerConfiguration(confVO);
supConf.commitWebContainerConfiguration();
```

## **Retrieve Solution Manager Configuration**

Retrieve the solution manager configuration.

### **Syntax**

```
SUPConfigurationComponentVO getSolutionManagerConfiguration()
 throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, throws SUPAdminException.

### **Examples**

- **Retrieve Solution Manager Configuration**

```
SUPConfiguration supConf = SUPObjectFactory
 .getSUPConfiguration(clusterContext);
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getSolutionManagerConfiguration();
System.out.println(confVO.getProperties());
```

### Update Solution Manager Configuration

Update the properties of the solution manager configuration.

#### Syntax

s

```
void updateSolutionManagerConfiguration(SUPConfigurationComponentVO
configurationComponent)
 throws SUPAdminException;
```

#### Parameters

- **configurationComponent** – The solution manager configuration properties to upgrade to.

#### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### Examples

- **Update Solution Manager Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO = supConf
 .getSolutionManagerConfiguration();
confVO.getProperties().put("com.sap.solutionmanager.url",
"http://solutionManagerHost");
supConf.updateSolutionManagerConfiguration(confVO);
supConf.commitClusterConfiguration();
```

### Retrieve DCN Configuration

Retrieve the DCN Configuration.

#### Syntax

```
SUPConfigurationComponentVO getDCNConfiguration()
 throws SUPAdminException;
```

#### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, throws SUPAdminException.

#### Examples

- **Get DCN Configuration**

```
SUPConfiguration supConf =
SUPObjectFactory.getSUPConfiguration(clusterContext);
```

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO =
supConf.getDCNConfiguration();
System.out.println(confVO.getProperties());
```

### **Update DCN Configuration**

Update the DCN configuration properties.

#### **Syntax**

```
void updateDCNConfiguration(SUPConfigurationComponentVO
configurationComponent)
 throws SUPAdminException;
```

#### **Parameters**

- **configurationComponent** – The DCN configuration properties to update to.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### **Examples**

- **Update DCN Configuration**

```
supConf.refreshClusterConfiguration();
SUPConfigurationComponentVO confVO =
supConf.getDCNConfiguration();
confVO.getProperties().put("sup.dcn.http.get.enabled", "true");
supConf.updateDCNConfiguration(confVO);
supConf.commitClusterConfiguration();
```

### **Set Server Log Setting**

Update the server log settings.

#### **Syntax**

```
void setServerLogLevelSetting
 throws SUPAdminException;
```

#### **Parameters**

- **setting** – The server log setting to set.

#### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### Examples

- **Set Server Log Setting**

```
SUPCluster cluster = this.getSUPCluster();
ServerLogSettingVO vo = cluster.getServerLogSetting();
Collection<ServerLogComponentLogLevel> cvos = vo
 .getComponentLogLevels();
for (ServerLogComponentLogLevel cvo : cvos) {
 cvo.setLogLevel(LOG_LEVEL.WARN);
}
vo.setComponentLogLevels(cvos);
cluster.updateServerLogSetting(vo);
```

### Retrieve Server Log Setting

Retrieve the server log settings.

### Syntax

```
ServerLogSettingVO getServerLogSetting()
 throws SUPAdminException;
```

### Returns

If successful, returns the specified type (can be null). If unsuccessful, throws SUPAdminException.

### Examples

- **Get Server Log Setting**

```
SUPCluster supCluster =
 SUPObjectFactory.getSUPCluster(clusterContext);
ServerLogSettingVO sls = supCluster.getServerLogSetting();
System.out.println(sls);
```

### Update Server Log Setting

Update the server log settings.

### Syntax

```
void updateServerLogSetting(ServerLogSettingVO setting)
 throws SUPAdminException;
```

### Parameters

- **setting** – The server log setting to update.

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Update Server Log Setting**

```

ServerLogSettingVO sls = supCluster.getServerLogSetting();
sls.setMaxBackupLogs(15);
sls.setMaxLogFileSize("20mb");
sls.setNewLogWhenServerStarted(false);
Collection<ServerLogComponentLogLevel> loglevels =
sls.getComponentLogLevels();
for(ServerLogComponentLogLevel level : loglevels){
 if(level.getComponent().equals(SERVER_LOG_COMPONENT.Other))
 {
 level.setLogLevel(LOG_LEVEL.WARN);
 break;
 }
}
supCluster.updateServerLogSetting(sls);

```

## Retrieval of Relay Servers

This method is deprecated. Use the SUPRelayServer interface to manage and configure Relay Servers. Retrieves a list of Relay Servers configured for an SAP Mobile Server cluster.

## Syntax

```
List<RelayServerVO> getRelayServers() throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieval of Relay Servers** – retrieves a list of relay servers in a cluster:

```

// Get all relay servers configured for the Mobile Server cluster.
List<RelayServerVO> relayServers = supCluster.getRelayServers();
for (RelayServerVO relayServer : relayServers) {
 // Print relay server info
 System.out.println("=====Begin Relay Server
Info=====");
 System.out.println("Host: " + relayServer.getHost());
 System.out.println("HTTP port: " + relayServer.getPort());
 System.out.println("HTTP port: " +
relayServer.getSecurePort());
 System.out.println("URL suffix: " +
relayServer.getUrlSuffix());
 // Print farm info of this relay server
 System.out.println("=====Farms within this relays
server=====");
 for (FarmVO farm : relayServer.getFarms()) {
 System.out.println(" " + farm);
 // print server node info of this farm

```

```
System.out.println("===Server nodes within this farm===");
for (ServerNodeVO serverNode : farm.getServerNodes()) {
 System.out.println(" Server node: " + serverNode);
 // print Outbound Enabler info of this server node
 System.out.println(" Outbound enabler: "
 + serverNode.getOutboundEnabler());
}
}
System.out.println("====End Relay Server Info====");
}
```

### **Licensing Information Retrieval**

Retrieves information about software and device licensing on SAP Mobile Server.

#### **Syntax**

```
LicensingInfoVO getLicensingInfo() throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval** – retrieves licensing information for SAP Mobile Server.

```
// Get Licensing info.
LicensingInfoVO infoVO = supCluster.getLicensingInfo();
System.out.println(infoVO.getAvailableDeviceLicenseCount());
System.out.println(infoVO.getLicenseType());
System.out.println(infoVO.getProductionEdition());
System.out.println(infoVO.getUsedDeviceLicenseCount());
System.out.println(infoVO.getDeviceLicenseExpireDate());
System.out.println(infoVO.getServerLicenseExpireDate());
```

---

**Note:** For more information on SAP Mobile Platform licensing, see *System Administration > Systems Maintenance and Monitoring > Platform Licenses*.

---

### **Retrieval and Setting of Trace Configuration**

Retrieves and sets the trace configuration settings.

#### **Syntax**

```
Collection<TraceConfigVO> getTraceConfigs() throws
SUPAdminException;

void setTraceConfigs(configs) throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieve and set trace configuration settings**

```
Collection<TraceConfigVO> configs = supCluster.getTraceConfigs();
for (TraceConfigVO config : configs) {
 if
 (TRACE_LOG_MODULE.JMS_BRIDGE.equals(config.getModule())) {
 config.setLevel(TRACE_LOG_LEVEL.DEBUG);
 }
}
supCluster.setTraceConfigs(configs);
System.out.println(configs);
```

## Setting Time Zone

When the time zone of the administration client is different from that of the SAP Mobile Server, you must format the time zone.

- If a time or date string representation is returned to the client, it must be formatted using the SAP Mobile Server's time zone. This requires the API implementation to perform the formatting; the client is not required to perform it.
- If a time or date string representation is passed to the API, it must be formatted in the SAP Mobile Server's time zone. This requires the client to perform the formatting before passing the time or date to API.
- If a time or date is of `java.util.Date`, `java.util.Calendar`, or `java.sql.Timestamp`, it can be used as it is.

## Syntax

```
TimeZone getTimeZone throws SUPAdminException;
void setTimeZone(timezone) throws SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Setting the Time Zone** – This example shows how to meet the time zone requirements.

```
TimeZone tz = supCluster.getTimeZone();
ClusterContext clusterContext = supCluster.getContext();
clusterContext.setTimeZone(tz);
```

## Management API

```
DomainContext domainContext =
clusterContext.getDomainContext("<domain name>");
```

### **Usage**

Execute these methods before making any timezone related API calls.

### **SAP License Audit**

For SAP® built applications, generate an XML file that contains usage audit data that is then uploaded to SAP License Audit.

### **Syntax**

```
String auditMeasurement =
supCluster.generateSAPAuditMeasurement(userName);
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Generate audit measurement file:** –

```
String auditMeasurement =
supCluster.generateSAPAuditMeasurement("John Doe");
```

## **Managing Relay Servers**

The `SUPRelayServer` interface allows you to manage the relay server or relay server cluster for the SAP Mobile Server.

Operations you can perform with this interface include:

- Starting an administration session for a relay server.
- Creating, updating and deleting a relay server configuration.
- Creating, updating and deleting a relay server farm configuration.
- Creating, updating and deleting relay server node information.

### **Start Relay Server Management**

Starts the management of a relay server.

### **Syntax**

```
SUPRelayServer getSUPRelayServer(ClusterContext clusterContext)
throws SUPAdminException;
```



## Parameters

- **clusterContext** – The specified relay server to manage.

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, throws `SUPAdminException`.

## Examples

- **Relay server startup** – starts the management of the specified relay server.

```
SUPRelayServer suprs =
SUPObjectFactory.getSUPRelayServer(contextFactory
 .getClusterContext());
```

## Usage

When an instance of `SUPRelayServer` is returned from the `SUPObjectFactory`, call its method.

## Retrieve Relay Server Configuration

Retrieves one or more relay server configuration.

You can retrieve a list of relay server configurations, or a specific relay server configuration based on a given ID, or a given host name and port number.

## Syntax

```
java.util.List<RelayServerVO> getRelayServers()
 throws
com.sybase.sup.admin.exception.SUPAdminException;

RelayServerVO getRelayServer(java.lang.Integer relayServerId)
 throws
com.sybase.sup.admin.exception.SUPAdminException;

RelayServerVO getRelayServer(java.lang.String host,
 java.lang.Integer port)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

## Parameters

- **relayServerId** – The ID of the relay server configuration you want to retrieve.
- **host** – The host name of the relay server configuration you want to retrieve.
- **port** – The port of the relay server configuration you want to retrieve.

### Returns

If successful, retrieves a list of relay server configurations, or a specific relay server configuration. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieval of a list of relay servers** – retrieves a list of all relay server configurations.

```
Iterator<RelayServerVO> iter =
suprs.getRelayServers().iterator();
 if (!iter.hasNext())
 return;
 RelayServerVO rsvo = iter.next();
 int id = rsvo.getID();
```

- **Retrieval of a relay server with a given ID** – retrieves the relay server configuration with the given ID.

```
RelayServerVO rsvo_byId = suprs.getRelayServer(id);
```

- **Retrieval of a relay server with a given host and port** – retrieves the relay server configuration with the given host name and port.

```
RelayServerVO rsvo_hostPort =
suprs.getRelayServer(rsvo.getHost(),
 rsvo.getPort());
```

### Create Relay Server Configuration

Creates a relay server configuration.

### Syntax

```
saveRelayServer(RelayServerVO relayServer) throws
SUPAdminException;
```

### Parameters

- **relayServer** – The relay server configuration to be created.

### Returns

If successful, creates a new relay server configuration. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Creation of a relay server** – Creates a relay server configuration with server credentials, server farms, and server nodes.

```
RelayServerVO rsvo = new RelayServerVO();

RelayServerVOBuilder bld = new RelayServerVOBuilder(rsvo);
 bld.host("myrelayservers.sybase.com").
```

```

 port(1234).
 credential("supAdmin").password("s3pAdmin").back().

credential("supAdmin2").password("s3pAdmin2").back().
 deleteFarm("supAdmin2.SupAdminAutoTestRBS").
 farm("supAdmin2.SupAdminAutoTestMBS").
 description("Description from API - Messaging").
 type(SERVER_FARM_TYPE.MESSAGING).
 name("supAdmin2.SupAdminAutoTestMBS - API").
 deleteServerNode("node-name").
 serverNode("node1").
 token("node1-token").
 back().
 back().
 farm("farm - API").
 description("Description from API - Replication").
 type(SERVER_FARM_TYPE.REPLICATION).
 serverNode("node2").
 token("node2-token").
 back().
 back().
 back();

rsvo = bld.build();
suprs.saveRelayServer(rsvo);

```

### **Update Relay Server Configuration**

Updates an existing relay server configuration with a given ID.

#### **Syntax**

```

updateRelayServer(java.lang.Integer relayServerId,
 RelayServerVO relayServer) throws
SUPAdminException;

```

#### **Parameters**

- **relayServerID** – The new ID of the relay server configuration.
- **relayServer** – The relay server configuration to be updated.

#### **Returns**

If successful, updates the relay server configuration. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Update of a relay server configuration** – Updates the relay server with the given ID.

```

@Test
public void updateRelayServer() throws Exception {
 List<RelayServerVO> relayServers =

```

```
supRelayServer.getRelayServers();
 if (relayServers.size() == 0) {
 System.out.println("No relay server defined.");
 return;
 }
 RelayServerVO relayServer = relayServers.get(0);
 relayServer.setHost("myRelayServer2.sybase.com");
 relayServer.setPort(8080);
 relayServer
 .setDescription("The relay server host and port has been
updated.");
 supRelayServer.updateRelayServer(relayServer.getID(),
relayServer);
}
```

### **Delete Relay Server Configurations**

Deletes one or more relay server configurations.

You can delete the following:

- a group of relay servers based on a list
- a group of relay servers based on a list of IDs
- a specific relay server based on an ID
- a specific relay server based on a host name and port number

### **Syntax**

```
void deleteRelayServers(java.util.List<RelayServerVO> relayServers)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

```
void deleteRelayServer(java.lang.String host, java.lang.Integer port)
 throws SUPAdminException
```

```
deleteRelayServers(java.util.Set<java.lang.Integer> relayServerIds)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

```
deleteRelayServer(java.lang.Integer relayServerId)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

### **Parameters**

- **relayServers** – The list of relay server configurations that you want to delete.
- **relayServerIds** – The list of relay server configuration IDs that you want to delete.
- **relayServerId** – The ID of the relay server configuration you want to delete.
- **host** – The host name of the relay server configuration you want to delete.
- **port** – The port of the relay server configuration you want to delete.

## Returns

If successful, deletes a list of relay servers, or a specific relay server. If unsuccessful, returns SUPAdminException.

## Examples

- **Deletion of list of relay servers** – deletes all relay server configurations in the list.

```
suprs.deleteRelayServers(suprs.getRelayServers());
```

- **Deletion of list of relay servers by ID** – deletes all relay server configurations with the given IDs.

```
Set<Integer> relayServerIDsToDelete = new HashSet<Integer>();
relayServerIDsToDelete.add(1);
relayServerIDsToDelete.add(2);
supRelayServer.deleteRelayServers(relayServerIDsToDelete);
```

- **Deletion of a relay server with a given ID** – deletes the relay server configuration with the given ID.

```
List<RelayServerVO> rss = suprs.getRelayServers();
Iterator<RelayServerVO> iter = rss.iterator();
suprs.deleteRelayServer(iter.next().getID());
```

- **Deletion of a relay server with a given host and port** – deletes the relay server with the given host name and port.

```
supRelayServer.deleteRelayServer("myRelayServer.sybase.com", 80);
```

## Retrieve Server Node

Retrieve a server node configuration with the given ID.

## Syntax

```
ServerNodeVO getServerNode(Integer serverNodeId)
 throws SUPAdminException;
```

## Parameters

- **serverNodeId** – The ID of the server node configuration.

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieve Server Node** – Get the server node with a primary key of 1.

```
@Test
public void testGetServerNode() throws Exception {
```

```
ServerNodeVO serverNode = suprs.getServerNode(1);
if (serverNode != null) {
 String serverNodeName = serverNode.getName();
 System.out.println(serverNodeName);
}
}
```

### **Create Server Node**

Create a server node configuration if there is none. Otherwise, the existing one will be updated.

### **Syntax**

```
void saveServerNode(ServerNodeVO serverNode)
 throws SUPAdminException;
```

### **Parameters**

- **serverNode** – The server configuration to be created or updated.

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Add Server Node** – Add a server node into a farm with a primary key of 1 in the database.

```
@Test
public void testSaveServerNode() throws Exception {
 ServerNodeVO serverNode = new ServerNodeVO();
 serverNode.setName("node1");
 serverNode.setToken("myOuboundEnablerToken");
 ServerFarmVO serverFarm = supRelayServer.getServerFarm(1);
 serverNode.setServerFarm(serverFarm);
 supRelayServer.saveServerNode(serverNode);
}
```

### **Delete Server Node**

Delete a set of server node configurations with the given IDs or a single server node with the given ID.

### **Syntax**

```
void deleteServerNodes(java.util.Set<Integer> serverNodeIds)
 throws SUPAdminException;
void deleteServerNode(Integer serverNodeId)
 throws SUPAdminException;
```

## Parameters

- **serverNodeIds** – A set of server node configuration IDs.
- **serverNodeId** – A server node configuration ID.

## Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Delete a Server Node**

```
@org.junit.Test
public void testDeleteServerNode() throws Exception {
 RelayServerVO relayServer = supRelayServer.getRelayServer(
 "myrelayserver.sybase.com", 80);
 if (relayServer == null) {
 System.out.println("myrelayserver.sybase.com:80 does not
configured");
 return;
 }
 List<ServerFarmVO> farms = relayServer.getServerFarms();
 if (farms.size() == 0) {
 System.out.println("No farm defined.");
 return;
 }
 ServerFarmVO farm = farms.get(0);
 List<ServerNodeVO> nodes = farm.getServerNodes();
 if (nodes.size() == 0) {
 System.out.println("No server node defined in farm "
+ farm.getName());
 return;
 }
 ServerNodeVO nodeToDelete = nodes.get(0);
 supRelayServer.deleteServerNode(nodeToDelete.getID());
}
```

## Update Server Node

Update an existing server node configuration with the given ID.

## Syntax

```
void updateServerNode(java.lang.Integer serverNodeId,
ServerNodeVO serverNode)
 throws SUPAdminException;
```

## Parameters

- **serverNodeId** – The server node configuration ID.
- **serverNode** – The value object containing the property values to be updated.

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### Examples

- **Update Server Node** – Update the name and token value of the server node whose ID is 1.

```
@org.junit.Test
public void testUpdateServerNode() throws Exception {
 ServerNodeVO node = supRelayServer.getServerNode(1);
 if (node == null) {
 System.out.println("There is no server node with ID
1.");
 }
 node.setName("node1_updated");
 node.setToken("token_updated");
 supRelayServer.updateServerNode(node.getID(), node);
}
```

### Retrieve Server Farm Configuration

Retrieve the server farms that belong to this Relay Server.

### Syntax

```
ServerFarmVO getServerFarm(Integer serverFarmID)
 throws SUPAdminException;
```

### Parameters

- **serverFarmId** – The ID of the server farm you want to retrieve.

### Returns

If successful, returns a list of server farms. If unsuccessful, throws SUPAdminException.

### Examples

- **Retrieve Server Farm** – Get server farm with a primary key of 1 in the database.

```
@org.junit.Test
public void testGetServerFarm() throws Exception {
 ServerFarmVO serverFarm = supRelayServer.getServerFarm(1);
 if (serverFarm != null) {
 String farmName = serverFarm.getName();
 System.out.println(farmName);
 SERVER_FARM_TYPE type = serverFarm.getType();
 System.out.println(type);
 }
}
```



## Create Server Farm Configuration

Creates a server farm configuration if none exists. Otherwise, the existing one is updated.

### Syntax

```
void saveServerFarm(ServerFarmVO serverFarm)
 throws SUPAdminException;
```

### Parameters

- **serverFarm** – The server farm configuration to be created or updated..

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException

### Examples

- **Add a Server Farm** – Add a new farm specifying a server node.

```
public void testSaveServerFarmReplication() throws
SUPAdminException {
 // prepare
 if (relayServer.getRelayServer(Constant.RS_HOST,
Constant.RS_PORT) == null) {
 addRelayServer();
 }
 ServerFarmVO svo;
 if ((svo = findServerFarmVO(
relayServer.getRelayServer(Constant.RS_HOST, Constant.RS_PORT),
serverFarmReplicationName, SERVER_FARM_TYPE.REPLICATION)) !=
null) {
 relayServer.deleteServerFarm(svo.getID());
 }
}
```

## Update Server Farm Configuration

Update an existing server farm configuration with the given ID.

### Syntax

```
void updateServerFarm(Integer serverFarmId, ServerFarmVO serverFarm)
 throws SUPAdminException;
```

### Parameters

- **serverFarmId** – The server farm configuration ID.
- **serverFarm** – The value object containing the property values to be updated.

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### Examples

- **Retrieve and Update Farm Configuration** – Retrieve a farm and update the name and type values.

```
@org.junit.Test
public void testUpdateServerFarm() throws Exception {
 List<RelayServerVO> relayServers =
 supRelayServer.getRelayServers();
 if (relayServers.size() == 0) {
 System.out.println("The SUP cluster do not have any
 relay server configured.");
 return;
 }
 RelayServerVO relayServer = relayServers.get(0);
 List<ServerFarmVO> farms = relayServer.getServerFarms();
 if (farms.size() == 0) {
 System.out.println("Relay server " +
 relayServer.getHost() + ":"
 + relayServer.getPort() + " do not have any
 farm.");
 return;
 }
 ServerFarmVO farm = farms.get(0);
 farm.setName("farm_name_updated");
 farm.setType(SERVER_FARM_TYPE.WEBSERVICE);
 farm.setDescription("This is a farm that name and type has
 been updated.");
 supRelayServer.updateServerFarm(farm.getID(), farm);
}
```

### Delete Server Farm Configuration

Delete a server farm configuration or a set of server farm configurations with the given ID or set of IDs.

### Syntax

```
void deleteServerFarm(Integer serverFarmId)
 throws SUPAdminException;
void deleteServerFarms(java.util.Set<Integer> serverFarmIds)
 throws SUPAdminException;
```

### Parameters

- **serverFarmId** – A server farm configuration ID you want to delete.
- **serverFarmIds** – The set of server farm configuration IDs you want to delete.

## Returns

If successful, returns silently. If unsuccessful, throws `SUPAdminException`.

## Examples

- **Delete a Farm**

```

@org.junit.Test
public void testDeleteFarm() throws Exception {
 RelayServerVO relayServer = supRelayServer.getRelayServer(
 "myrelayserver.sybase.com", 80);
 if (relayServer == null) {
 System.out.println("myrelayserver.sybase.com:80 does not
configured");
 return;
 }
 List<ServerFarmVO> farms = relayServer.getServerFarms();
 if (farms.size() == 0) {
 System.out.println("No farm defined.");
 return;
 }
 ServerFarmVO farm = farms.get(0);
 supRelayServer.deleteServerFarm(farm.getID());
}

```

## Retrieve Outbound Enabler Proxy Configuration

Retrieves one or more outbound enabler proxy configurations.

You can retrieve a list of outbound enabler proxy configurations, or a specific outbound enabler proxy configuration based on a given ID, or a given host name and port number.

## Syntax

```

java.util.List<OutboundEnablerProxyVO> getOutboundEnablerProxies ()
 throws
com.sybase.sup.admin.exception.SUPAdminException;

OutboundEnablerProxyVO getOutboundEnablerProxy(java.lang.Integer
outboundEnablerProxyId)
 throws
com.sybase.sup.admin.exception.SUPAdminException;

OutboundEnablerProxyVO getOutboundEnablerProxy(java.lang.String
host,
 java.lang.Integer port)
 throws
com.sybase.sup.admin.exception.SUPAdminException;

```

## Parameters

- **outboundEnablerProxyId** – The ID of the outbound enabler proxy configuration you want to retrieve.

- **host** – The host name of the outbound enabler proxy configuration you want to retrieve.
- **port** – The port of the outbound enabler proxy configuration you want to retrieve.

### Returns

If successful, retrieves a list of outbound enabler proxy configurations, or a specific outbound enabler proxy configuration. If unsuccessful, returns SUPAdminException.

### Examples

- **Retrieval of a list of outbound enabler proxies** – retrieves a list of all outbound enabler proxy configurations.

```
@Test
public void getOutboundEnablerProxyVOByHostPort() throws
Exception {
 Iterator<OutboundEnablerProxyVO> iterator = supRelayServer
 .getOutboundEnablerProxies().iterator();
 if (!iterator.hasNext()) {
 System.out.println("No Outbound Enabler Proxy
configured.");
 return;
 }
 OutboundEnablerProxyVO proxy1 = iterator.next();
 OutboundEnablerProxyVO proxy2 =
supRelayServer.getOutboundEnablerProxy(
 proxy1.getHost(), proxy1.getPort());
 assertEquals(proxy1, proxy2);
}
```

- **Retrieval of an outbound enabler proxy with a given ID** – retrieves the outbound enabler proxy configuration with the given ID.

```
OutboundEnablerVO rsvo_byId = suprs.getOutboundEnablerProxy(id);
```

- **Retrieval of an outbound enabler proxy with a given host and port** – retrieves the outbound enabler proxy configuration with the given host name and port.

```
OutboundEnablerVO rsvo_hostPort =
suprs.getOutboundEnablerProxy(rsvo.getHost(),
 rsvo.getPort());
```

### Create Outbound Enabler Proxy Configuration

Creates an outbound enabler proxy.

### Syntax

```
void saveOutboundEnablerProxy(OutboundEnablerProxyVO
outboundEnablerProxy)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

## Parameters

- **outboundEnablerProxy** – The outbound enabler proxy configuration to be created.

## Returns

If successful, creates a new outbound enabler proxy configuration. If unsuccessful, returns SUPAdminException.

## Examples

- **Creation of an outbound enabler proxy** – Creates an outbound server proxy configuration.

```

Iterator<OutboundEnablerVO> iter = supServerConf
 .getOutboundEnablers().iterator();
if (!iter.hasNext())
 return;

 OutboundEnablerVO oe2 = new OutboundEnablerVO();
 OutboundEnablerVOBuilder bld = new
OutboundEnablerVOBuilder (oe2);
 bld.host ("proxy1").
 port (4321)
 credential ("proxy-user-1").
 password ("proxy-password-1")
 back().
 credential ("proxy-user-2").
 password ("proxy-password-2")
 back().

suprs.saveOutboundEnablerProxy (bld.build());

```

## Update Outbound Enabler Proxy

Updates an existing outbound enabler proxy configuration with a given ID.

## Syntax

```

void updateOutboundEnablerProxy (java.lang.Integer
outboundEnablerProxyId,
 OutboundEnablerProxyVO
outboundEnablerProxy)
 throws
com.sybase.sup.admin.exception.SUPAdminException;

```

## Parameters

- **outboundEnablerProxyID** – The new ID of the outbound enabler proxy configuration.
- **outboundEnablerProxy** – The outbound enabler proxy configuration to be updated.

### **Returns**

If successful, updates the outbound enabler proxy configuration. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Update of an outbound enabler proxy configuration** – updates the outbound enabler proxy with the given ID.

```
Iterator<OutboundEnablerVO> iter = supServerConf
 .getOutboundEnablers().iterator();
 if (!iter.hasNext())
 return;

 OutboundEnablerVO oel = iter.next();
 oel.setCertificateFile("4321");

 supServerConf.updateOutboundEnabler(oel.getID(), oel);
```

### **Delete Outbound Enabler Proxy Configuration**

Deletes one or more outbound enabler proxy configurations.

You can delete the following:

- a group of outbound enabler proxies based on a list
- a group of outbound enabler proxies based on a list of IDs
- a specific outbound enabler proxy based on an ID
- a specific outbound enabler proxy based on a host name and port number

### **Syntax**

```
void
deleteOutboundEnablerProxies (java.util.List<OutboundEnablerProxyVO>
outboundEnablerProxies)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

```
void deleteOutboundEnablerProxies (java.util.Set<java.lang.Integer>
outboundEnablerProxyIds)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

```
void deleteOutboundEnablerProxy (java.lang.Integer
outboundEnablerProxyId)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

```
void deleteOutboundEnablerProxy (java.lang.String host,
 java.lang.Integer port)
 throws
com.sybase.sup.admin.exception.SUPAdminException;
```

## Parameters

- **outboundEnablerProxies** – The list of outbound enabler proxy configurations that you want to delete.
- **outboundEnablerProxyIds** – The list of outbound enabler proxy configurations IDs that you want to delete.
- **outboundEnablerProxyId** – The ID of the outbound enabler proxy configuration you want to delete.
- **host** – The host name of the outbound enabler proxy configuration you want to delete.
- **port** – The port of the outbound enabler proxy configuration you want to delete.

## Returns

If successful, deletes a list of outbound enabler proxies, or a specific outbound enabler proxy. If unsuccessful, returns SUPAdminException.

## Examples

- **Deletion of a List of Outbound Enabler Proxies** – Deletes all outbound enabler proxy configurations in the list.

```
supRelayServer.deleteOutboundEnablerProxies(supRelayServer
 .getOutboundEnablerProxies());
```

- **Deletion of a List of Outbound Enabler Proxies by ID** – Deletes all outbound enabler proxy configurations with the given IDs.

```
Set<Integer> proxyIDsToDelete = new HashSet<Integer>();
proxyIDsToDelete.add(1);
proxyIDsToDelete.add(2);
supRelayServer.deleteOutboundEnablerProxies(proxyIDsToDelete);
```

- **Deletion of an Outbound Enabler Proxy with a Given ID** – Deletes the outbound enabler proxy configuration with the given ID.

```
Iterator<OutboundEnablerProxyVO> iter = supRelayServer
 .getOutboundEnablerProxies().iterator();
if (!iter.hasNext())
 return;
supRelayServer.deleteOutboundEnablerProxy(iter.next().getID());
```

- **Deletion of an Outbound Enabler Proxy with a Given Host and Port** – Deletes the outbound enabler proxy with the given host name and port.

```
supRelayServer.deleteOutboundEnablerProxy("myProxy.sybase.com",
 80);
```

### **Managing Domains**

You can manage domains of SAP Mobile Servers through the SUPDomain interface.

Operations you can perform with this interface include:

- Enabling or disabling a SAP Mobile Platform domain.
- Packages: listing, creating, deleting, importing, exporting packages.
- Endpoints: listing, creating, deleting, updating endpoints.
- Security configuration: getting and setting associated security configurations.
- Domain administrators: listing administrators.
- Data maintenance: cleaning up accumulated data artifacts.
- Applications: viewing applications and application connections at the domain level.

#### **Start Domain Management**

Starts the management of a domain.

#### **Syntax**

```
public static SUPDomain getSUPDomain(DomainContext domainContext)
throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Start domain management** – starts the management of the specified domain:

```
DomainContext domainContext =
serverContext.getDomainContext("<domain name>");
SUPDomain supDomain =
SUPObjectFactory.getSUPDomain(domainContext);
```

#### **Usage**

To manage SAP Mobile Server domains, you must first create an instance of SUPDomain.

To perform SAP Mobile Platform domain administration operations, you must be assigned an SAP Mobile Platform administrator or SAP Mobile Platform domain administrator role.

#### **Enable a Domain**

Enables a domain.

#### **Syntax**

```
void enable(Boolean flag) throws SUPAdminException;
```



**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Enable a domain**

```
supDomain.enable(true); //Enable domain
```

**Disable a Domain**

Disables a domain.

**Syntax**

```
void enable(Boolean flag) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Disable a domain**

```
supDomain.enable(false); //Disable domain
```

**Package Retrieval**

Retrieves a list of packages in a domain.

**Syntax**

```
Collection<String> getPackages() throws SUPAdminException;
```

**Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Package retrieval** – retrieves a list of packages in a domain:

```
for (String packageName : supDomain.getPackages()) {
 System.out.println(packageName);
}
```

### **Package Deployment**

Deploys a package to a domain.

#### **Syntax**

```
deployPackage(java.lang.String fileName, DEPLOY_MODE deployMode,
java.lang.String securityConfiguration,
java.util.Collection<RoleMappingVO> roleMappings,
java.util.Map<java.lang.String,java.lang.String> endpointMappings,
java.util.Map<java.lang.String,java.lang.String>
eisManagedCacheEndpointMappings)
void deployPackage(String fileName, DEPLOY_MODE deployMode, String
securityConfiguration, Collection<RoleMappingVO> roleMappings,
Map<String, String> endpointMappings) throws SUPAdminException;

void purgeSyncCacheGroup(SyncCachePurgeOptionVO optionVO,
java.lang.Boolean synchronous)
throws SUPAdminException;
```

#### **Parameters**

The deployment mode determines how the deployment process handles the objects in a deployment unit and package. Which value you choose depends on whether or not a package of the same name already exists on SAP Mobile Server. Allowed values are:

- **UPDATE** – updates the target package with updated objects. After deployment, objects in the server's package with the same name as those being deployed are updated. By default, deploymentMode is UPDATE.
- **VERIFY** – do not deploy package. Only return errors, if any. Used to determine the results of the UPDATE deploy mode.

If the deployment mode is specified both in the descriptor file and the command-line, the command-line deploymentMode option override the deployment mode specified in the descriptor file.

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Package deployment** – deploys a package to a domain:

```
Collection<RoleMappingVO> roleMappingVOs = new
ArrayList<RoleMappingVO>();
RoleMappingVO rmvo1 = new RoleMappingVO();
rmvo1.setSourceRole("Role1");
rmvo1.setRoleMappingType(ROLE_MAPPING_TYPE.AUTO);
RoleMappingVO rmvo2 = new RoleMappingVO();
rmvo2.setSourceRole("Role2");
```

```

rmvo2.setRoleMappingType(ROLE_MAPPING_TYPE.AUTO);
RoleMappingVO rmvo3 = new RoleMappingVO();
rmvo3.setSourceRole("Role3");
rmvo3.setRoleMappingType(ROLE_MAPPING_TYPE.AUTO);

roleMappingVOs.add(rmvo1);
roleMappingVOs.add(rmvo2);
roleMappingVOs.add(rmvo3);

Map<String, String> endpointMappings = new HashMap<String,
String>();
endpointMappings.put("sampledb", "sampledb2");

supDomain.deployPackage("<deployment unit file name>",
 DEPLOY_MODE.UPDATE,
 "<security configuration name>", roleMappingVOs,
 endpointMappings);

```

### **Package Deletion**

Deletes a package from a domain.

#### **Syntax**

```
void deletePackage(String name) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Package deletion** – deletes the specified package from the domain:

```
supDomain.deletePackage("<package name>");
```

### **Package Import**

Imports a package to a domain.

#### **Syntax**

```
void importPackage(java.lang.String fileName) throws
SUPAdminException
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Package import** – imports a package with the specified package file name to the domain:

```
void importPackage("<exported package file name>", true);
```

### Usage

You can only import package into the same domain as the one you exported from. The API requires that the domain where the package was exported from exists on the server when the import is done. Also, you are required to create domains in the same order in both the export and import server environments, which ensures that an internal ID assigned to the domain in both environment matches.

You can verify the internal ID assigned to a domain by looking at the prefix used in the package folder in the zip.

### Package Export

Exports a package from a domain.

### Syntax

```
void exportPackage(java.lang.String fileName, java.lang.String name); throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Package Export** – exports a package with the specified file name and package name from a domain:

```
supAdmin.exportPackage("<file name>", "<package name>");
```

### Endpoint Retrieval

Retrieves a list of server connection endpoints in the domain. The supported endpoint types are JDBC, SAP®, and WEBSERVICE.

### Syntax

```
Collection<EndpointVO> getEndpoints(ENDPOINT_TYPE type) throws SUPAdminException;
```

```
EndpointVO getEndpoint(ENDPOINT_TYPE type, java.lang.String name)
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Endpoint retrieval** – retrieves a list of endpoints for each endpoint type:

```

for(EndpointVO evo : supDomain.getEndpoints(ENDPOINT_TYPE.JDBC)) {
 System.out.println(evo.getName());
 System.out.println(evo.getExtraProps());
}

for(EndpointVO evo : supDomain.getEndpoints(ENDPOINT_TYPE.SAP)) {
 System.out.println(evo.getName());
 System.out.println(evo.getExtraProps());
}

for(EndpointVO evo :
supDomain.getEndpoints(ENDPOINT_TYPE.WEBSERVICE)) {
 System.out.println(evo.getName());
 System.out.println(evo.getExtraProps());
}

```

---

**Note:** For detailed information on each of these endpoint types, see *Developer Guide: SAP Mobile Server Runtime > Management API > Property Reference > EIS Data Source Connection Properties Reference*.

---

## **Endpoint Creation**

Creates a server connection endpoint of the specified endpoint type.

### **Syntax**

```
void createEndpoint(ENDPOINT_TYPE type, String name, String
template, Map<String, String> properties) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Endpoint creation** – creates an endpoint for each endpoint type, and sets its properties:

```

Map<String, String> properties = new HashMap<String, String>();

// For Sybase ASA
properties.put("commitProtocol", "<commit protocol>");
properties.put("dataSourceClass", "<data source class>");
properties.put("databaseURL", "<database URL>");
properties.put("user", "<user name>");
properties.put("password", "<password>");
supDomain.createEndpoint(ENDPOINT_TYPE.JDBC, "<endpoint name>",
"<template name>", properties);

properties.clear();
properties.put("jco.client.user", "<jco client user>");
properties.put("jco.client.passwd", "<jco client password>");
properties.put("jco.client.ashost", "<jco client AS host>");
properties.put("jco.client.client", "<jco client>");
supDomain.createEndpoint(ENDPOINT_TYPE.SAP, "<endpoint name>",
"<template name>", properties);

```

```
properties.clear();
properties.put("address", "<address>");
properties.put("user", "<user name>");
properties.put("password", "<password>");
supDomain.createEndpoint(ENDPOINT_TYPE.WEBSERVICE, "<endpoint
name>", "<template name>", properties);
```

### **Endpoint Deletion**

Deletes a specific server connection endpoint of the specified type.

### **Syntax**

```
void deleteEndpoint(ENDPOINT_TYPE type, String name) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Endpoint deletion** – deletes an endpoint of each endpoint type:

```
supDomain.deleteEndpoint(ENDPOINT_TYPE.JDBC, "<endpoint name>");
supDomain.deleteEndpoint(ENDPOINT_TYPE.SAP, "<endpoint name>");
supDomain.deleteEndpoint(ENDPOINT_TYPE.WEBSERVICE, "<endpoint
name>");
```

### **Endpoint Update**

Updates the properties of a specific server connection endpoint.

### **Syntax**

```
void updateEndpoint(EndpointVO endpoint) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Endpoint update**

```
EndpointVO evo = new EndpointVO();
evo.setName("sampledb2");
evo.setType(ENDPOINT_TYPE.JDBC);
Map<String, String> properties = new HashMap<String, String>();
properties.put("commitProtocol", "pessimistic");
properties.put("dataSourceClass",
"com.sybase.jdbc3.jdbc.SybDataSource");
properties.put("databaseURL", "jdbc:sybase:Tds:localhost:5500/
sampledb2?ServiceName=sampledb2");
```

```
evo.setExtraProps(properties);
supDomain.updateEndpoint(evo);
```

### **Endpoint Template Retrieval**

Retrieves a list of endpoint templates in the domain. The supported endpoint template types are JDBC, SAP®, and WEBSERVICE.

#### **Syntax**

```
Collection<EndpointVO> getEndpointTemplates(ENDPOINT_TYPE type)
throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Endpoint template retrieval** – retrieves a list of endpoint templates for each endpoint type:

```
for (EndpointVO evo : supDomain
 .getEndpointTemplates(ENDPOINT_TYPE.JDBC)) {
 System.out.println(evo.getName());
 System.out.println(evo.getExtraProps());
}

for (EndpointVO evo :
 supDomain.getEndpointTemplates(ENDPOINT_TYPE.SAP)) {
 System.out.println(evo.getName());
 System.out.println(evo.getExtraProps());
}

for (EndpointVO evo : supDomain
 .getEndpointTemplates(ENDPOINT_TYPE.WEBSERVICE)) {
 System.out.println(evo.getName());
 System.out.println(evo.getExtraProps());
}
```

---

**Note:** For detailed information on each of these endpoint types, see *Developer Guide: SAP Mobile Server Runtime > Management API > Property Reference > EIS Data Source Connection Properties Reference*.

---

### **Endpoint Template Creation**

Creates a server connection endpoint template for the specified endpoint type.

#### **Syntax**

```
void createEndpointTemplate(ENDPOINT_TYPE type, String name, String
template, Map<String, String> properties) throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Endpoint creation** – creates an endpoint for each endpoint type, and sets its properties:

```
Map<String, String> properties = new HashMap<String, String>();
properties.put("commitProtocol", "<commit protocol>");
properties.put("dataSourceClass", "<data source class>");
properties.put("databaseURL", "<database URL>");
properties.put("user", "<user name>");
properties.put("password", "<password>");
supDomain.createEndpointTemplate(ENDPOINT_TYPE.JDBC,
 "myJDBC_template",
 "Sybase_ASA_template", properties);

properties.clear();
properties.put("jco.client.user", "<jco client user>");
properties.put("jco.client.passwd", "<jco client password>");
properties.put("jco.client.ashost", "<jco client AS host>");
properties.put("jco.client.client", "<jco client>");
supDomain.createEndpointTemplate(ENDPOINT_TYPE.SAP,
 "mySAP_template",
 "sap_template", properties);

properties.clear();
properties.put("address", "<address>");
properties.put("user", "<user name>");
properties.put("password", "<password>");
supDomain.createEndpointTemplate(ENDPOINT_TYPE.WEBSERVICE,
 "myWS_template", "webservice_template", properties);
```

### Endpoint Template Deletion

Deletes a specific server connection endpoint template of the specified type.

### Syntax

```
void deleteEndpointTemplate(ENDPOINT_TYPE type, String name) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Endpoint template deletion** – deletes an endpoint template of each endpoint type:

```
supDomain.deleteEndpointTemplate(ENDPOINT_TYPE.JDBC,
 "<endpoint template name>");
supDomain.deleteEndpointTemplate(ENDPOINT_TYPE.SAP,
 "<endpoint template name>");
```



```
supDomain.deleteEndpointTemplate(ENDPOINT_TYPE.WEBSERVICE,
 "<endpoint template name>");
```

### **Endpoint Template Update**

Updates the properties of a specific server connection endpoint template.

#### **Syntax**

```
void updateEndpointTemplate(EndpointVO endpoint) throws
 SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Endpoint update**

```
EndpointVO evo = new EndpointVO();
evo.setName("<endpoint template name>");
evo.setType(ENDPOINT_TYPE.JDBC);
Map<String, String> properties = new HashMap<String, String>();
properties.put("commitProtocol", "pessimistic");
properties.put("dataSourceClass",
 "com.sybase.jdbc3.jdbc.SybDataSource");
properties.put("databaseURL", "jdbc:sybase:Tds:localhost:5500/
 sampledb2?ServiceName=sampled2");
evo.setExtraProps(properties);
supDomain.updateEndpointTemplate(evo);
```

### **Retrieval of Security Configurations**

Retrieves a list of security configurations for a domain.

#### **Syntax**

```
Collection<String> getSecurityConfigurations() throws
 SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval of security configurations** – retrieves a list of security configurations for a domain:

```
for (String securityConfiguration : supDomain
 .getSecurityConfigurations()) {
```

```
System.out.println(securityConfiguration);
}
```

### **Set Default Security Configuration**

Sets the default security configuration for a domain.

#### **Syntax**

```
void setDefaultSecurityConfiguration(String security) throws
SUPAdminException;
```

#### **Returns**

If successful, returns the default security configuration for a domain. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Set default security domain** – gets and sets the default security configuration for a domain.

```
supDomain.setDefaultSecurityConfiguration(defsec);
```

### **Retrieval of Default Security Configuration**

Retrieves the default security configuration for a domain.

#### **Syntax**

```
String getDefaultSecurityConfiguration() throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval of default security configuration** – retrieves the default security configuration for a domain:

```
supDomain.getDefaultSecurityConfiguration();
```

### **Update of Security Configurations**

Updates security configurations in the domain. You must be assigned a SAP Mobile Platform administrator role to perform this operation.

#### **Syntax**

```
void setSecurityConfigurations(Collection<String> names) throws
SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Update of security configurations** – updates the security configurations specified in an array:

```
supDomain.setSecurityConfigurations(Arrays.asList(new String[] {
 "<security configuration 1>", "<security configuration
2>" }));
```

## Retrieve Scheduled Purge Task Status

Checks to see whether domain-level cleanup is scheduled for the specified purge task type.

## Syntax

```
Boolean isScheduledPurgeTaskEnable(SCHEDULE_PURGE_TASK task) throws
SUPAdminException;
```

## Returns

If successful, returns true or false. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Purge task status** – retrieves the scheduled data purge task status for synchronization cache, subscription, client log, and error history purge tasks.

```
supDomain.isScheduledPurgeTaskEnable(SCHEDULE_PURGE_TASK.CLIENT_L
OG);
supDomain.isScheduledPurgeTaskEnable(SCHEDULE_PURGE_TASK.ERROR_HI
STORY);
supDomain.isScheduledPurgeTaskEnable(SCHEDULE_PURGE_TASK.SUBSCRIP
TION);
supDomain.isScheduledPurgeTaskEnable(SCHEDULE_PURGE_TASK.SYNC_CAC
HE_GROUP);
```

## Enable or Disable Scheduled Purge Tasks

Enables or disables domain-level cleanup using the current scheduled purge task values.

## Syntax

```
void enableScheduledPurgeTask(SCHEDULE_PURGE_TASK task, Boolean
enabled) throws SUPAdminException;
```

## Returns

If successful, enables or disables cleanup. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Enables or disables purge tasks** – enables or disables the scheduled data purge tasks for synchronization cache, subscription, client log, or error history.

```
supDomain.enableScheduledPurgeTask(SCHEDULE_PURGE_TASK.CLIENT_LOG, true);
supDomain.enableScheduledPurgeTask(SCHEDULE_PURGE_TASK.ERROR_HISTORY, false);
supDomain.enableScheduledPurgeTask(SCHEDULE_PURGE_TASK.SUBSCRIPTION, false);
supDomain.enableScheduledPurgeTask(SCHEDULE_PURGE_TASK.SYNC_CACHE_GROUP, true);
```

### Get Purge Task Schedule

Gets the cleanup schedule for the selected purge task type. Getting the purge task schedule is typically used with setting the purge task schedule.

### Syntax

```
ScheduleVO getPurgeTaskSchedule(SCHEDULE_PURGE_TASK task) throws SUPAdminException;
```

### Returns

If successful, returns true or false. If unsuccessful, returns SUPAdminException.

### Examples

- **Get purge task schedule** – gets and sets the purge task schedule for synchronization cache, subscription, client log, or error history.

```
ScheduleVO reschedule =
supDomain.getPurgeTaskSchedule(SCHEDULE_PURGE_TASK.CLIENT_LOG);
reschedule =
supDomain.getPurgeTaskSchedule(SCHEDULE_PURGE_TASK.ERROR_HISTORY);
reschedule =
supDomain.getPurgeTaskSchedule(SCHEDULE_PURGE_TASK.SUBSCRIPTION);
reschedule =
supDomain.getPurgeTaskSchedule(SCHEDULE_PURGE_TASK.SYNC_CACHE_GROUP);
```

### Set Purge Task Schedule

Sets the domain-level cleanup schedule for the selected purge task. Setting the purge task schedule is typically used with getting the purge task schedule.

### Syntax

```
void setPurgeTaskSchedule(SCHEDULE_PURGE_TASK task, ScheduleVO schedule) throws SUPAdminException;
```

## Returns

If successful, returns the schedule for the selected type. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Set purge task schedule** – gets and sets the purge task schedule for synchronization cache, subscription, client log, or error history.

```
ScheduleVO schedule = new ScheduleVO();
schedule.setDaysOfWeek(EnumSet.of(DAY_OF_WEEK.MONDAY, DAY_OF_WEEK.FRIDAY));
schedule.setStartDate(new Date());
schedule.setStartTime(new Date());
schedule.setEndDate(new Date());
schedule.setEndTime(new Date());
schedule.setFreq(SCHEDULE_FREQ.INTERVAL);
schedule.setInterval(50);

supDomain.setPurgeTaskSchedule(SCHEDULE_PURGE_TASK.CLIENT_LOG,
schedule);
supDomain.setPurgeTaskSchedule(SCHEDULE_PURGE_TASK.ERROR_HISTORY,
schedule);
supDomain.setPurgeTaskSchedule(SCHEDULE_PURGE_TASK.SUBSCRIPTION,
schedule);
supDomain.setPurgeTaskSchedule(SCHEDULE_PURGE_TASK.SYNC_CACHE_GROUP,
schedule);
```

## Purge Synchronization Cache at the Domain Level

Purges synchronization cache at the domain level. The purge can be done synchronously or asynchronously.

## Syntax

```
void purgeSyncCacheGroup(Boolean synchronous) throws
SUPAdminException;
```

## Returns

If successful, purges synchronization cache using the schedule. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Purge sync cache** – purges the synchronization cache using defined settings.

```
supDomain.purgeSyncCacheGroup(false);
```

### **Manage Synchronization Cache Options**

Sets synchronization cache purge at the domain level. The purge can be done synchronously or asynchronously.

#### **Syntax**

```
void setSyncCacheGroupOption(SyncCachePurgeOptionVO option) throws
SUPAdminException
```

```
SyncCachePurgeOptionVO getSyncCachePurgeOption() throws
SUPAdminException
```

#### **Returns**

If successful, synchronizes cache. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Manage synchronization cache purge group option** – Manage the synchronization cache group option.

```
public void testSetPurgeSyncCacheOptionVO() throws
SUPAdminException{
 _log.debug("test set purge sync cache option vo");
 SUPDomain domain = this.getSUPDomain("default");
 SyncCachePurgeOptionVO vo = new SyncCachePurgeOptionVO();
 vo.setNewPartitionThreshold(100);
 vo.setOldPartitionThreshold(200);
 domain.setSyncCacheGroupOption(vo);

}
```

- **Manage synchronization cache purge option** – Manage the synchronization cache purge option.

```
public void testGetPurgeSyncCacheOptionVO() throws
SUPAdminException{
 _log.debug("test get purge sync cache option vo");
 SUPDomain domain = this.getSUPDomain("default");
 SyncCachePurgeOptionVO vo = domain.getSyncCachePurgeOption();

}
```

### **Purge Client Log**

Purges the client log at the domain level. The purge can be done synchronously or asynchronously.

#### **Syntax**

```
void purgeClientLog(ClientLogPurgeOptionVO purgeOption, Boolean
synchronous) throws SUPAdminException;
```

## **Returns**

If successful, purges the client log using the schedule. If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Purge client log** – purges the client log using current settings.

```
ClientLogPurgeOptionVO purgeOption = new
ClientLogPurgeOptionVO ();
purgeOption.setDaysToPreserve (10);
supDomain.purgeClientLog (purgeOption, false);
```

## **Get Client Log Purge Options**

Obtains the current client log purge settings at the domain level.

## **Syntax**

```
ClientLogPurgeOptionVO getClientLogPurgeOption() throws
SUPAdminException;
```

## **Returns**

If successful, gets the current client log purge settings. If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Gets client log options** – gets the current client log purge options.

```
ClientLogPurgeOptionVO roption =
supDomain.getClientLogPurgeOption ();
```

## **Set Client Log Purge Options**

Sets the client log purge options at the domain level using the current settings.

## **Syntax**

```
void setClientLogPurgeOption (ClientLogPurgeOptionVO option) throws
SUPAdminException;
```

## **Returns**

If successful, sets the current client log purge settings. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Sets client log options** – sets the current client log purge settings, which includes preserving data for the last 15 days.

```
ClientLogPurgeOptionVO option = new ClientLogPurgeOptionVO();
option.setDaysToPreserve(15);
supDomain.setClientLogPurgeOption(option);
```

### Purge Error History

Purges the error history at the domain level. The purge can be done synchronously or asynchronously.

### Syntax

```
void purgeErrorHistory(ErrorHistoryPurgeOptionVO purgeOption,
Boolean synchronous) throws SUPAdminException;
```

### Returns

If successful, purges the error history using the schedule. If unsuccessful, returns SUPAdminException.

### Examples

- **Purge error history** – purges the error history using defined settings.

```
ErrorHistoryPurgeOptionVO purgeOption = new
ErrorHistoryPurgeOptionVO();
purgeOption.setDaysToPreserve(10);
supDomain.purgeErrorHistory(purgeOption, false);
```

### Get Error History Purge Options

Gets the current error history purge option settings at the domain level.

### Syntax

```
ErrorHistoryPurgeOptionVO getErrorHistoryPurgeOption() throws
SUPAdminException;
```

### Returns

If successful, gets the current error history purge settings. If unsuccessful, returns SUPAdminException.

### Examples

- **Gets error history purge options** – gets the current error history purge settings.



```
ErrorHistoryPurgeOptionVO roption =
supDomain.getErrorHistoryPurgeOption();
```

### Set Error History Purge Options

Sets the error history purge options at the domain level using current settings.

#### Syntax

```
void setErrorHistoryPurgeOption(ErrorHistoryPurgeOptionVO option)
throws SUPAdminException;
```

#### Returns

If successful, sets the current error history purge settings. If unsuccessful, returns SUPAdminException.

#### Examples

- **Set error history purge options** – sets the current error history purge settings.

```
ErrorHistoryPurgeOptionVO option = new
ErrorHistoryPurgeOptionVO();
option.setDaysToPreserve(15);
supDomain.setErrorHistoryPurgeOption(option);
```

### Purge Subscription

Purges subscriptions at the domain level. The purge can be done synchronously or asynchronously.

#### Syntax

```
void purgeSubscription(SubscriptionPurgeOptionVO purgeOption,
Boolean synchronous) throws SUPAdminException;
```

#### Returns

If successful, purges subscriptions using the schedule. If unsuccessful, returns SUPAdminException.

#### Examples

- **Purge subscription** – purges subscriptions using defined settings.

```
SubscriptionPurgeOptionVO purgeOption = new
SubscriptionPurgeOptionVO();
purgeOption.setDaysInactive(10);
supDomain.purgeSubscription(purgeOption, false);
```

### Get Subscription Purge Options

Obtains the current subscription purge options at the domain level.

#### **Syntax**

```
SubscriptionPurgeOptionVO getSubscriptionPurgeOption() throws
SUPAdminException;
```

#### **Returns**

If successful, gets the subscription purge settings. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Gets subscription purge options** – gets the current subscription purge settings.

```
SubscriptionPurgeOptionVO roption =
supDomain.getSubscriptionPurgeOption();
```

### Set Subscription Purge Options

Sets the subscription purge options at the domain level.

#### **Syntax**

```
void setSubscriptionPurgeOption(SubscriptionPurgeOptionVO option)
throws SUPAdminException;
```

#### **Returns**

If successful, sets the current subscription purge settings. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Sets subscription purge options** – sets the subscription purge options, including setting 15 as the number of inactive days.

```
SubscriptionPurgeOptionVO option = new
SubscriptionPurgeOptionVO();
option.setDaysInactive(15);
supDomain.setSubscriptionPurgeOption(option);
```

## Managing Packages

You can manage MBO packages and their properties through the `SUPPackage` interface. Operations you can perform with this interface include:

- **Security configuration** – getting or setting security configuration.
- **Synchronization group** – getting or setting synchronization group properties.
- **Synchronization tracing** – enabling or disabling synchronization tracing.
- **Message-based sync subscription management** – these subscriptions determine what synchronization messages mobile device users receive on messaging-based devices.
- **Replication-based sync subscription and template management** – these subscriptions determine what synchronization messages mobile device users receive on replication-based devices.
- **Package role mapping** – getting/setting package level role mappings. You can define role mapping for the package to map logical roles in the package to physical roles on the SAP Mobile Server.
- **Applications** – viewing applications, adding or removing application to/from a package, viewing application users.
- **Uncategorized** – enabling and disabling packages, listing MBOs, managing cache groups, listing personalization keys, and retrieving endpoint properties.

### Start Package Management

Starts the management of an SAP Mobile Server package.

### Syntax

```
public static SUPPackage getSUPPackage(PackageContext
packageContext) throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Start package management**

```
domainContext = serverContext.getDomainContext("<domain name>");
packageContext = domainContext.getPackageContext("<package
name>");
SUPPackage suppkg =
SUPObjecTFactory.getSUPPackage(packageContext);
```

### Usage

To manage SAP Mobile Server packages, you must first create an instance of `SUPPackage`.

### **Enable a Package**

Enables a package.

#### **Syntax**

```
void enable(Boolean flag) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Enable a package** – enables a package and retrieves a list of mobile business objects and personalization keys in the package.

```
//Enable a package.
suppkg.enable(true); //Enable package

//Retrieve a list of MBOs
for (String mboName : suppkg.getMobileBusinessObjects()) {
 System.out.println(mboName);
}
//Retrieve a list of personalization keys
for(PersonalizationKeyVO pvo : suppkg.getPersonalizationKeys()){
 System.out.println(pvo.getKey());
}
```

### **Disable a Package**

Disables a package.

#### **Syntax**

```
void enable(Boolean flag) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Disable a package**

```
//Disable a package.
suppkg.enable(false); //Disable package
```

**Enable Synchronization Tracing**

This method has been deprecated. Enables synchronization tracing.

**Syntax**

```
void setSyncTracingStatus(Boolean flag) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

**Examples**

- **Enable synchronization tracing**

```
suppkg.setSyncTracingStatus(true); //Enable synchronization tracing
```

**Disable Synchronization Tracing**

This method has been deprecated. Disables synchronization tracing.

**Syntax**

```
void setSyncTracingStatus(Boolean flag) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

**Examples**

- **Disable synchronization tracing**

```
suppkg.setSyncTracingStatus(false); //Disable synchronization tracing
```

**Retrieval of Security Configuration**

Retrieves the security configuration associated with a package.

**Syntax**

```
String getSecurityConfiguration() throws SUPAdminException;
```

**Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval of security configurations**

```
String securityConfiguration = suppkg.getSecurityConfiguration();
```

### **Set Security Configuration**

Sets the security configuration for a package.

### **Syntax**

```
void setSecurityConfiguration(String name) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Set security configuration**

```
suppkg.setSecurityConfiguration("<security configuration name>");
```

### **Retrieval of Synchronization Group Properties**

Retrieves a list of synchronization group properties for a package.

### **Syntax**

```
Collection<SyncGroupVO> getSyncGroups() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval of synchronization group properties**

```
for (SyncGroupVO sgvo : suppkg.getSyncGroups()) {
 System.out.println(sgvo.getName());
}
```

### **Set Synchronization Group Properties**

Sets properties for a synchronization group in a package.

### **Syntax**

```
void setSyncGroupChangeDetectionInterval(String syncGroup, Integer
checkInterval) throws SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Set synchronization group properties** – updates the check interval for the specified synchronization group:

```
suppkg.setSyncGroupChangeDetectionInterval("<sync group name>",
1000);
```

## Retrieval of Messaging Package Subscriptions

Retrieves messaging package subscriptions.

## Syntax

```
Collection<MBSSubscriptionVO> getMBSSubscriptions() throws
SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval of messaging package subscriptions**

```
Collection<MBSSubscriptionVO> mbsSubs =
suppkg.getMBSSubscriptions();
MBSSubscriptionVO mbsSub = suppkg.getMBSSubscription("<client
id>");
```

---

**Note:** For more information on managing messaging package subscriptions, see *SAP Control Center for SAP Mobile Platform > Deploy > MBO Packages > MBO Subscription Management > Managing Subscriptions*.

---

## Deletion of Messaging Package Subscriptions

Deletes messaging package subscriptions.

## Syntax

```
void removeMBSSubscriptions(Collection<String> clientIds) throws
SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Deletion of messaging package subscriptions**

```
suppkg.removeMBSSubscriptions(clientIds);
```

### **Suspend Package Subscriptions**

Suspends messaging package subscriptions, or DOE-C package subscriptions.

### **Syntax**

```
void suspendMBSSubscriptions(Collection<String> clientIds) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Suspend messaging (or DOE-C) package subscriptions**

```
suppkg.suspendMBSSubscriptions(clientIds);
```

### **Resume Package Subscriptions**

Resumes messaging package subscriptions, or DOE-C package subscriptions.

### **Syntax**

```
void resumeMBSSubscriptions(Collection<String> clientIds) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Resume messaging (or DOE-C) package subscriptions**

```
suppkg.resumeMBSSubscriptions(clientIds);
```

### **Reset Messaging Package Subscriptions**

Resets messaging package subscriptions.

### **Syntax**

```
void resetMBSSubscriptions(Collection<String> clientIds) throws SUPAdminException;
```



## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Reset messaging package subscriptions**

```
suppkg.resetMBSSubscriptions(clientIds);
```

## Retrieval of Replication Package Subscriptions

Retrieves replication package subscriptions.

## Syntax

```
Collection<RBSSubscriptionVO> getRBSSubscriptions(String syncGroup,
String user) throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval of replication package subscriptions**

```
for (RBSSubscriptionVO rbsSub : suppkg
 .getRBSSubscriptions("<sync group name>")) {
 System.out.println(rbsSub.getSyncGroup() + ":"
 + rbsSub.getClientId());
}
for (RBSSubscriptionVO rbsSub : suppkg.getRBSSubscriptionVOs(
 "<sync group name>", "<user name>")) {
 System.out.println(rbsSub.getSyncGroup() + ":"
 + rbsSub.getClientId());
}
```

---

**Note:** For more information on managing messaging package subscriptions, see *SAP Control Center for SAP Mobile Platform > Deploy > MBO Packages > MBO Subscription Management > Managing Subscriptions*.

---

## Update of Replication Package Subscriptions

Updates replication package subscriptions.

## Syntax

```
void updateRBSSubscription(RBSSubscriptionVO rbsSub) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update of replication package subscriptions** – updates subscriptions of replication packages and sets the properties:

```
RBSSubscriptionVO rbsSub = new RBSSubscriptionVO();
//Client id, sync group, package and domain can uniquely
//identify a RBS subscription
rbsSub.setClientId("<client id>");
rbsSub.setSyncGroup("<sync group>");
//Bellow are the modifiable properties of a RBS subscription
//Please refer to Java doc for detailed information.
rbsSub.setAdminLocked(false);
rbsSub.setPushEnabled(true);
rbsSub.setSyncIntervalMinutes(5);
suppkg.updateRBSSubscription(rbsSub);
```

### Removal of Replication Package Subscriptions

Removes a subscription or a list of subscriptions for a package.

### Syntax

```
void removeRBSSubscription(String syncGroup, String clientId) throws
SUPAdminException;

void removeRBSSubscriptions(String syncGroup) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Removal of replication package subscriptions** – shows how to remove a list of subscriptions, or a single subscription, for a replication package:

```
//Remove one subscription
suppkg.removeRBSSubscription("<sync group name>", "<client id>"

//Remove a list of subscriptions
suppkg.removeRBSSubscriptions(Arrays.asList(new String[] {
 "<client id 1>", "<client id 2>" }));
suppkg.removeRBSSubscriptions("<sync group>");
suppkg.removeRBSSubscriptions("<sync group>", "<user name>");
```

## **Purge RBS and MBS Subscriptions**

Purges replication-based and message-based synchronization (RBS and MBS) subscriptions at the package level using the number of inactive days. The purge can be done synchronously or asynchronously.

### **Syntax**

```
void purgeSubscription(SubscriptionPurgeOptionVO purgeOption,
Boolean synchronous) throws SUPAdminException;
```

### **Returns**

If successful, purges RBS and MBS subscriptions based on the number of inactive days specified. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Purge subscriptions** – purges RBS and MBS subscriptions.

```
SubscriptionPurgeOptionVO purgeOption = new
SubscriptionPurgeOptionVO();
purgeOption.setDaysInactive(10);
suppkg.purgeSubscription(purgeOption, false);
```

## **Create Subscription Templates**

Creates a subscription template for replication packages.

### **Syntax**

```
RBSSubscriptionVO createRBSSubscriptionTemplate(String syncGroup,
Boolean isPushEnabled, Boolean isAdminLocked, Integer
minimumSyncMinutes) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Creation of a subscription template** – creates a subscription template for replication packages:

```
suppkg.createRBSSubscriptionTemplate("<sync group name>", false,
false, 5);
```

## **Retrieval of Role Mappings**

Retrieves role mappings for a package.

Role mappings map logical roles in the package to physical roles on the SAP Mobile Server.

### **Syntax**

```
Collection<RoleMappingVO> getRoleMappings() throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval of role mappings**

```
Collection<RoleMappingVO> roleMappingVOs =
suppkg.getRoleMappings();
```

---

**Note:** See the *SAP Control Center for SAP Mobile Platform > Administer > Security Configurations > Creating a Security Configuration > Roles and Mappings*.

---

### **Set Role Mappings**

Sets role mappings for a package.

### **Syntax**

```
void setRoleMappings(Collection<RoleMappingVO> rmvos) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Set role mappings**

```
roleMappingVOs = new ArrayList<RoleMappingVO>();
RoleMappingVO rmvo1 = new RoleMappingVO();
rmvo1.setSourceRole("Role1");
rmvo1.setRoleMappingType(ROLE_MAPPING_TYPE.AUTO);
RoleMappingVO rmvo2 = new RoleMappingVO();
rmvo2.setSourceRole("Role2");
rmvo2.setRoleMappingType(ROLE_MAPPING_TYPE.AUTO);
RoleMappingVO rmvo3 = new RoleMappingVO();
rmvo3.setSourceRole("Role3");
rmvo3.setRoleMappingType(ROLE_MAPPING_TYPE.AUTO);

roleMappingVOs.add(rmvo1);
roleMappingVOs.add(rmvo2);
roleMappingVOs.add(rmvo3);

suppkg.setRoleMappings(roleMappingVOs);
```

## **Cache Groups**

A cache group specifies the data refresh behavior for every mobile business object (MBO) within that group.

You can perform these management tasks for cache groups:

- Retrieving a list of cache groups
- Managing schedule properties of a cache group
- Listing the MBOs associated with a cache group
- Purging or clearing a cache group

### **Cache Groups Retrieval**

Retrieves a list of cache groups for a package.

### **Syntax**

```
Collection<CacheGroupVO> getCacheGroups() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval of cache groups**

```
for (CacheGroupVO cgvo : suppkg.getCacheGroups()) {
 System.out.println(cgvo.getName());
}
```

### **Schedule Properties Retrieval**

Retrieves the schedule properties of a cache group for a package.

### **Syntax**

```
CacheGroupScheduleVO getCacheGroupSchedule(String cacheGroupName)
throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval of schedule properties** – retrieves a list of cache groups for a package:

```
CacheGroupScheduleVO cgsvo = suppkg
 .getCacheGroupSchedule("<cache group name>");
```

### Set Schedule Properties

Sets the schedule properties of a cache group for a package.

### Syntax

```
void setCacheGroupSchedule(String cacheGroupName,
 CacheGroupScheduleVO cacheGroupSchedule) throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Set schedule properties** – retrieves a list of cache groups for a package:

```
CacheGroupScheduleVO cgsvo = new CacheGroupScheduleVO();
cgsvo.setFrequency(SCHEDULE_FREQ.DAILY);
```

```
EnumSet<DAY_OF_WEEK> daysOfWeek =
EnumSet.noneOf(DAY_OF_WEEK.class);
daysOfWeek.add(DAY_OF_WEEK.MONDAY);
daysOfWeek.add(DAY_OF_WEEK.THURSDAY);
cgsvo.setDayOfWeek(daysOfWeek);
```

```
//start date: 2009-12-03
//start time: 18:31:45
//end date: 2009-12-23
//end time: 21:34:47
Calendar cal = Calendar.getInstance();
cal.set(Calendar.YEAR, 2009);
cal.set(Calendar.MONTH, 11);
cal.set(Calendar.DAY_OF_MONTH, 3);
Date startDate = cal.getTime();
cgsvo.setStartDate(startDate);
```

```
cal.set(Calendar.YEAR, 2009);
cal.set(Calendar.MONTH, 11);
cal.set(Calendar.DAY_OF_MONTH, 23);
Date endDate = cal.getTime();
cgsvo.setEndDate(endDate);
```

```
cal.set(Calendar.HOUR_OF_DAY, 18);
cal.set(Calendar.MINUTE, 31);
cal.set(Calendar.SECOND, 45);
Date startTime = cal.getTime();
cgsvo.setStartTime(startTime);
```

```
cal.set(Calendar.HOUR_OF_DAY, 21);
cal.set(Calendar.MINUTE, 34);
cal.set(Calendar.SECOND, 47);
Date endTime = cal.getTime();
```

```
cgsvo.setEndTime(endTime);

suppkg.setCacheGroupSchedule("<cache group name>", cgsvo);
```

- **Set cache group interval**

```
CacheGroupScheduleVO cgsvo = new CacheGroupScheduleVO();
cgsvo.setFrequency(SCHEDULE_FREQ.INTERVAL);
cgsvo.setInterval(CacheGroupScheduleVO.NEVER_EXPIRE);
suppkg.setCacheGroupSchedule("<cache group name>", cgsvo);
```

### Associated Mobile Business Objects

Retrieves a list of the mobile business objects associated with a cache group.

### Syntax

```
Collection<String> getCacheGroupMBOs(String cacheGroupName) throws
SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### Examples

- **Getting associated mobile business objects**

```
for(String mboName : suppkg.getCacheGroupMBOs("<cache group
name>")) {
 System.out.println(mboName);
}
```

### Cache Group Purge

Physically deletes rows in the cache group that are marked as logically deleted and are older than the specified date.

### Syntax

```
void purgeCacheGroup(String cacheGroupName, Date date) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Cache group purge** – physically deletes data that is marked as deleted and older than the dateThreshold:

```
Calendar cal = Calendar.getInstance();
cal.clear();
```

## Management API

```
cal.set(Calendar.YEAR, 2009);
cal.set(Calendar.MONTH, 11);
cal.set(Calendar.DAY_OF_MONTH, 3);
Date dateThreshold = cal.getTime();
// Physically delete data that is marked as deleted and older than
the
// dateThreshold
suppkg.purgeCacheGroup("<cache group name>", dateThreshold);
```

### **Usage**

Ensure that all devices have synchronized at least once before the specified purge date.

### **Mobile Business Objects**

Packages contain mobile business objects that are deployed to SAP Mobile Server to facilitate access to back-end data and transactions from mobile devices.

---

**Note:** See the *SAP Control Center for SAP Mobile Platform > Get Started > About SAP Control Center for SAP Mobile Platform > MBO Package Management Overview*.

---

### **Mobile Business Object Retrieval**

Retrieves a list of mobile business objects for a package.

### **Syntax**

```
Collection<String> getMobileBusinessObjects() throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Mobile business object retrieval**

```
//Retrieve a list of MBOs
for (String mboName : suppkg.getMobileBusinessObjects()) {
 System.out.println(mboName);
}
```

### **Personalization Keys**

Personalization keys are created by the MBO developer for use as client parameters (user data, such as user name and password), to be validated by the EIS.



### Personalization Key Retrieval

Retrieves a list of personalization keys for a package.

#### **Syntax**

```
Collection<PersonalizationKeyVO> getPersonalizationKeys() throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Personalization key retrieval**

```
//Retrieve a list of personalization keys
for(PersonalizationKeyVO pvo : suppkg.getPersonalizationKeys()){
 System.out.println(pvo.getKey());
}
```

#### **Client Logs**

Client logs record errors, history, and informational messages for mobile clients. Logs include data change notification logs, device notification logs, error logs, messaging logs, replication logs, and subscription logs.

You can perform these management tasks for client logs:

- Retrieving client logs
- Deleting client logs
- Exporting client logs

#### **Retrieval of Client Logs**

Retrieves the client logs specified in the search and sort criteria.

#### **Syntax**

```
PaginationResult<LogEntryVO>
getClientLogs(ClientLogSearchCriteriaVO searchCriteria, Integer
skip, Integer take, ClientLogSortVO sortInfo) throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### Examples

- **Client log retrieval**

```
//Prepare the search and sort criteria
ClientLogSearchCriteriaVO searchCriteria = new
ClientLogSearchCriteriaVO();
searchCriteria.setUserName("*sup*");
searchCriteria.setLevel("*N?O");
searchCriteria.setOperation("*up*");
ClientLogSortVO sortInfo = new ClientLogSortVO();
sortInfo.setAscending(false);
sortInfo.setSortField(ClientLogSortVO.SortField.device);

//Get client Log
PaginationResult<LogEntryVO> result = suppkg.getClientLogs(
searchCriteria, 0, 5, sortInfo);
```

### Deletion of Client Logs

Deletes client logs.

### Syntax

```
void deleteClientLogs(List<Long> messageIDs) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Client log deletion**

```
//Delete Client Log
List<Long> messageIDs = new ArrayList<Long>();
messageIDs.add(310004L);
suppkg.deleteClientLogs(messageIDs);

Map<CLIENT_LOG_FIELD, String> map = new HashMap<CLIENT_LOG_FIELD,
String>();
map.put(CLIENT_LOG_FIELD.USER, "supAdmin");
map.put(CLIENT_LOG_FIELD.START_TIME, "2011-07-07");
map.put(CLIENT_LOG_FIELD.END_TIME, "2011-07-08");
suppkg.deleteClientLogs(map);
```

### Export of Client Logs

Exports client logs.

#### **Syntax**

```
void exportClientLogs(File file, ClientLogSearchCriteriaVO
searchCriteria, Integer skip, Integer take, ClientLogSortVO
sortInfo) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Client log export**

```
//Export client Log
suppkg.exportClientLogs(new File("F:/tmp/out.txt"),
searchCriteria, 0,
3, sortInfo);
```

### Purge Client Log

Purges the client log at the package level. The purge can be done synchronously or asynchronously.

#### **Syntax**

```
void purgeClientLog(ClientLogPurgeOptionVO purgeOption, Boolean
synchronous) throws SUPAdminException;
```

#### **Returns**

If successful, purges the client log using current settings. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Purge client log** – purges the client log, except for data from the last 10 days.

```
ClientLogPurgeOptionVO purgeOption = new
ClientLogPurgeOptionVO();
purgeOption.setDaysToPreserve(10);
suppkg.purgeClientLog(purgeOption, false);
```

### **Purge Synchronization Cache at the Package Level**

Purges synchronization cache at the package level. The purge can be done synchronously or asynchronously.

#### **Syntax**

```
void purgeSyncCacheGroup(Boolean synchronous) throws
SUPAdminException;
```

```
void purgeSyncCacheGroup(SyncCachePurgeOptionVO
optionVO, java.lang.Boolean synchronous) throws SUPAdminException
```

#### **Returns**

If successful, purges synchronization cache using current settings. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Purge sync cache** – purges the synchronization cache using defined settings.

```
suppkg.purgeSyncCacheGroup(false);
```

### **Purge Error History**

Purges the error history at the package level. The purge can be done synchronously or asynchronously.

#### **Syntax**

```
void purgeErrorHistory(ErrorHistoryPurgeOptionVO purgeOption,
Boolean synchronous) throws SUPAdminException;
```

#### **Returns**

If successful, purges the error history using current settings. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Purge error history** – purges the error history, except for data from the last 10 days.

```
ErrorHistoryPurgeOptionVO purgeOption = new
ErrorHistoryPurgeOptionVO();
purgeOption.setDaysToPreserve(10);
suppkg.purgeErrorHistory(purgeOption, false);
```

## **Purge Subscription**

Purges subscriptions at the package level. The purge can be done synchronously or asynchronously.

### **Syntax**

```
void purgeSubscription(SubscriptionPurgeOptionVO purgeOption,
Boolean synchronous) throws SUPAdminException;
```

### **Returns**

If successful, purges subscriptions using current settings. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Purge subscription** – purges subscriptions, except for data from the last 10 days.

```
SubscriptionPurgeOptionVO purgeOption = new
SubscriptionPurgeOptionVO ();
purgeOption.setDaysInactive(10);
suppkg.purgeSubscription(purgeOption, false);
```

## **Add Applications to the Package**

Adds existing applications to the package.

### **Syntax**

```
void addApplications(Collection<String> appIds) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Addition of applications to the package**

```
Collection<String> apps = new ArrayList<String>();
apps.add("app1");
suppkg.addApplications(apps);
```

## **Remove Applications from the Package**

Removes existing applications from the package.

### **Syntax**

```
void removeApplications (Collection<String> appIds) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Remove applications from the package**

```
Collection<String> apps = new ArrayList<String>();
apps.add("appl");
suppkg.removeApplications(apps);
```

### Retrieval of a List of Applications

Retrieves a list of applications for a package.

### Syntax

```
Collection<ApplicationVO> getApplications() throws
SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieval of role mappings**

```
Collection<ApplicationVO> apps = suppkg.getApplications();
```

### Retrieval of a List of Package Users

Retrieves a list of package users for a package.

### Syntax

```
PaginationResult<PackageUserVO> getPackageUsers(PackageUser_SortVO
filter, Long offset, Integer length) throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieval of role mappings**

```
PackageUser_SortVO filter = new PackageUser_SortVO();
filter.setSortField(PACKAGE_USER.REGISTRATION_TIME);
filter.setSortOrder(SORT_ORDER.ASCENDING);
```

```
PaginationResult<PackageUserVO> apps =
suppkg.getPackageUsers(filter, 0L, 100);
```

## **Managing Mobile Business Objects**

You can manage mobile business objects and their properties through the `SUPMobileBusinessObject` interface. Operations you can perform with this interface include:

- **Mobile business objects** – retrieving properties and data refresh history, and listing operations.
- **Endpoints** – retrieving properties.

### **Start Mobile Business Object Management**

Starts the management of a mobile business object.

#### **Syntax**

```
public static SUPMobileBusinessObject
getSUPMobileBusinessObject(MBOContext mboContext) throws
SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

#### **Examples**

- **Start mobile business object management**

```
domainContext = clusterContext.getDomainContext("<domain name>");
packageContext = domainContext.getPackageContext("<package
name>");
mboContext = packageContext.getMBOContext("<MBO name>");
SUPMobileBusinessObject supmbo =
SUPObjectFactory.getSUPMobileBusinessObject(mboContext);
```

#### **Usage**

To manage SAP Mobile Server mobile business objects, you must first create an instance of `SUPMobileBusinessObject`.

#### **Import MBO Package**

Imports a package to a domain. Only a SAP Mobile Platform administrator or a SAP Mobile Platform domain administrator can perform this.

#### **Syntax**

```
void importPackage(String fileName) throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Package import** – imports a package with the specified package file name to the domain:

```
supDomain.importPackage(fileName);
```

### Properties Retrieval

Retrieves properties for a mobile business object.

### Syntax

```
MobileBusinessObjectVO getProperties() throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Properties retrieval** – retrieves properties for a mobile business object, including name, package, creation date, and roles used:

```
MobileBusinessObjectVO mbovo = supmbo.getProperties();
System.out.println(mbovo.getName());
System.out.println(mbovo.getPackage());
System.out.println(mbovo.getCreationDate());
System.out.println(mbovo.getUsedRoles());
```

### Endpoints

Endpoint connection information allows applications to retrieve data from back-end production systems.

---

**Note:** For more information, see *System Administration > EIS Connection Management > Data Source Connections > Changing Connections to Production Data Sources*.

---

### Endpoint Properties Retrieval

Retrieves the properties of an endpoint used by a mobile business object.

### Syntax

```
EndpointVO getEndpoint() throws SUPAdminException;
```



## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Endpoint properties retrieval**

```
EndpointVO evo = supmbo.getEndpoint();
System.out.println(evo.getName());
System.out.println(evo.getType());
for(Map.Entry<String, String> entry :
evo.getExtraProps().entrySet()){
 System.out.println(entry.getKey() + " --> " +
entry.getValue());
}
```

## Retrieval of Data Refresh Error History

Retrieves the data refresh error history for a mobile business object.

## Syntax

```
Collection<DataRefreshErrorVO> getDataRefreshErrors(Date startDate,
Date endDate) throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **History retrieval**

```
for(DataRefreshErrorVO drevo : supmbo.getDataRefreshErrors(null,
null)){
 System.out.println(drevo.getErrorMessage());
}
```

## Deletion of Data Refresh Error History

Deletes the data refresh error history for a mobile business object.

## Syntax

```
void deleteDataRefreshErrors(Date startDate, Date endDate) throws
SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **History deletion**

```
supmbo.deleteDataRefreshErrors(null, null);
```

### Operations Retrieval

Retrieves a list of the operations of a mobile business object.

### Syntax

```
Collection<String> getOperations() throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Operations retrieval**

```
for (String op : supmbo.getOperations()) {
 System.out.println(op);
}
```

## Managing Operations

You can manage operations and endpoints used by those operations through the `SUPOperation` interface. Operations you can perform with this interface include:

- **Operations** – retrieving properties.
- **Endpoints** – retrieving properties.

### Start Operations Management

Starts the management of an SAP Mobile Server operation.

### Syntax

```
public static SUPOperation getSUPOperation(OperationContext
operationContext) throws SUPAdminException
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Start operation management**

```
domainContext = serverContext.getDomainContext("<domain name>");
packageContext = domainContext.getPackageContext("<package
name>");
mboContext = packageContext.getMBOContext("<MBO name>");
operationContext = mboContext.getOperationContext("<operation
name>");
SUPOperation supOperation =
SUPObjectFactory.getSUPOperation(operationContext);
```

## **Usage**

To manage SAP Mobile Server operations, you must first create an instance of `SUPOperation`.

## **Operation Properties Retrieval**

Retrieves the properties of an operation.

## **Syntax**

```
OperationVO getProperties() throws SUPAdminException;
```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Operation properties retrieval**

```
OperationVO ovo = supOperation.getProperties();
```

## **Endpoint Properties Retrieval**

Retrieves the properties of an endpoint used by an operation.

## **Syntax**

```
EndpointVO getEndpoint() throws SUPAdminException;
```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Endpoint properties retrieval**

```
EndpointVO evo = supOperation.getEndpointVO();

System.out.println(evo.getExtraProps());
```

### Retrieval of Playback Error History

Retrieves the playback error history of an operation.

### Syntax

```
Collection<PlaybackErrorVO> getPlaybackErrors(Date startDate, Date
endDate) throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Playback history retrieval**

```
for(PlaybackErrorVO pbevo : supOperation.getPlaybackErrors(null,
null)) {
 System.out.println(pbevo.getErrorMessage());
}
```

## Managing Applications and Application Connections and Templates

You can manage applications, application connections, and application connection templates through the `SUPApplication` method. Operations you can perform with this interface include:

- **Managing applications** – creating, deleting, and updating applications. Retrieving a list of applications or application users. Deleting application users. Assigning or unassigning domains to an application. Adding or removing packages from an application, or retrieving a list of packages from an application.
- **Managing application connections** – retrieving, cloning, registering, updating, locking, unlocking, and deleting application connections.
- **Managing application connection templates** – managing, listing, and updating application connection templates.

### Start Application Management

Starts the management of SAP Mobile Server applications, application connections, and application connection templates.

**Syntax**

```
public static SUPApplication getSUPApplication(ClusterContext
clusterContext) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Start application management**

```
app = SUPObjectFactory.getSUPApplication(clusterContext);
```

**Usage**

To manage SAP Mobile Server applications, you must first create an instance of `SUPApplication`.

**Managing Applications**

Use the `SUPApplication` interface to manage applications. Operations you can perform with this interface include:

- Creating an application
- Deleting an application
- Updating an application
- Retrieving a list of applications
- Retrieving a list of application users
- Deleting application users
- Assigning or unassigning domains from an application
- Retrieving domains assigned to an application
- Adding packages to or removing packages from an application
- Retrieving a list of packages from an application

**Application Creation**

Creates an application.

**Syntax**

```
void createApplication(String appID, String displayName, String
description) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Create application**

```
supApplication.createApplication("app1", "app1display", "app1
description");
```

### **Application Deletion**

Deletes applications.

### **Syntax**

```
void deleteApplications(Collection<String> appIDs) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Delete application**

```
Collection<String> appIDs = new ArrayList<String>();
appIDs.add("app1");

supApplication.deleteApplications(appIDs);
```

### **Application Update**

Updates the application's display name and description.

### **Syntax**

```
void updateApplication(String appId, String displayName, String
description) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Update**

```
supApplication.updateApplication("app1", "updated display
name", "updated desc");
```

### Retrieval of a List of Applications

Retrieves a list of applications that satisfy the filter. The return result is paginated.

#### **Syntax**

```
PaginationResult<ApplicationVO>
getApplications(ApplicationFilterSortVO filter,
Long offset, Integer length) throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```
ApplicationFilterSortVO filter = new ApplicationFilterSortVO();
FilterExpression<APPLICATION> resultExpression = new
FilterExpression<APPLICATION>();
FilterExpression<APPLICATION> expression1 = new
FilterExpression<APPLICATION>();
FilterExpression<APPLICATION> expression2 = new
FilterExpression<APPLICATION>();
expression1 = expression1.eq(APPLICATION.APPLICATION_USER,
"WM2@admin");
expression2 = expression2.eq(APPLICATION.APPLICATION_USER,
"abc@admin");
resultExpression = expression1.or(expression2);

filter.setFilterExpression(resultExpression);
filter.setSortField(APPLICATION.APPLICATION_ID);
filter.setSortOrder(SORT_ORDER.ASCENDING);
PaginationResult<ApplicationVO> apps =
supApplication.getApplications(filter, 01, 100);
```

### Retrieval of a List of Application Users

Retrieves a list of application users.

#### **Syntax**

```
PaginationResult<ApplicationVO>
getApplicationUsers(ApplicationUser_FilterSortVO filter, Long
offset, Integer length) throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### Examples

- **Retrieval**

```
ApplicationUserFilterSortVO filter = new
ApplicationUserFilterSortVO();

filter.setFilterExpression(null);
filter.setSortField(APPLICATION_USER.APPLICATION_ID);
filter.setSortOrder(SORT_ORDER.ASCENDING);
PaginationResult<ApplicationUserVO> apps =
supApplication.getApplicationUsers(filter, 01,
100);
```

### Application Users Deletion

Deletes a list of application users.

### Syntax

```
void deleteApplicationUsers(Collection<ApplicationUserVO> users)
throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Deletion**

```
Collection<ApplicationUserVO> users = new
ArrayList<ApplicationUserVO>();
ApplicationUserVO user1 = new ApplicationUserVO();
user1.setApplicationId("app1");
user1.setSecurityConfiguration("admin");
user1.setUserName("user1");
users.add(user1);
supApplication.deleteApplicationUsers(users);
```

### Export Hybrid App

Export a Hybrid App from the server. Only a SAP Mobile Platform administrator or a SAP Mobile Platform domain administrator can perform this.

### Syntax

```
void exportMobileHybridApp(String fileName, MobileHybridAppIDVO
hybridAppID)
throws SUPAdminException;
```



## Parameters

- **fileName** – The name of the file which stores the exported Hybrid App.
- **hybridAppID** – The Hybrid App ID.

## Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Export Hybrid App**

```
SUPMobileHybridApp hybridApp =
 SUPObjectFactory.getSUPMobileHybridApp(clusterContext);
hybridApp.exportMobileHybridApp(fileName, hybridAppID);
```

- ***Import Hybrid App***

Import an exported Hybrid App to the server. Only a SAP Mobile Platform administrator or a SAP Mobile Platform domain administrator can perform this.

## Syntax

```
void importMobileHybridApp(String fileName)
 throws SUPAdminException;
```

## Parameters

- **fileName** – The file name of the exported Hybrid App.

## Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Import Hybrid App**

```
SUPMobileHybridApp hybridApp =
 SUPObjectFactory.getSUPMobileHybridApp(clusterContext);
hybridApp.importMobileHybridApp(fileName);
```

- ***Export Application***

Export an application.

## Syntax

```
void exportApplication(String fileName, String id)
 throws SUPAdminException;
```

### **Parameters**

- **fileName** – The name of the file which stores the exported application.
- **id** – The application ID.

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Export Application**

```
SUPApplication app =
SUPObjectFactory.getSUPApplication(clusterContext);
app.exportApplication(fileName, appID);
```

#### *Import Application*

Import an application archive.

### **Syntax**

```
void importApplication(String fileName)
 throws SUPAdminException;
```

### **Parameters**

- **fileName** – The file name of the exported application archive.

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException. If the import fails, the thrown exception will contain the proper error message.

### **Examples**

- **Import Application**

```
SUPApplication app =
SUPObjectFactory.getSUPApplication(clusterContext);
app.importApplication(fileName);
```

#### *Assign Domains to an Application*

Assigns domains to the specified application.

### **Syntax**

```
void assignDomainsToApplication(String appID, Collection<String>
domains) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Assign Domains**

```
Collection<String> domains = new ArrayList<String>();
domains.add("default");
domains.add("domain1");
supApplication.assignDomainsToApplication("app1", domains);
```

**Unassign Domains from an Application**

Unassigns domains from the specified application.

**Syntax**

```
void unassignDomainsToApplication(String appID, Collection<String>
domains) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Unassign domains**

```
Collection<String> domains = new ArrayList<String>();
domains.add("default");
domains.add("domain1");
supApplication.unassignDomainsFromApplication("app1", domains);
```

**Retrieval of Assigned Domains**

Retrieves the domains assigned to an application.

**Syntax**

```
Collection<String> getApplicationDomainAssignments(String appId)
throws SUPAdminException;
```

**Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval**

```
Collection<String> domains =
 supApplication.getApplicationDomainAssignments("app1");
```

### **Add Packages to an Application**

Adds packages to the specified application.

### **Syntax**

```
void addApplicationPackages(String appID, String domain,
Collection<String> pkgs) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Add packages**

```
String domain = "default";
Collection<String> pkgs = new ArrayList<String>();
pkgs.add("pkg1");
supApplication.addApplicationPackages("app1", domain, pkgs);
```

### **Remove Packages from an Application**

Removes packages from the specified application.

### **Syntax**

```
void removeApplicationPackages(String appID, String domain,
Collection<String> pkgs) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Removal**

```
String domain = "default";
Collection<String> pkgs = new ArrayList<String>();
pkgs.add("pkg1");
supApplication.removeApplicationPackages("app1", domain, pkgs);
```

### Retrieval of a List of Packages from an Application

Retrieves a list of packages from an application that satisfy the filter. The return result is paginated

#### Syntax

```
PaginationResult<ApplicationPackageVO>
getApplicationPackages(Application_FilterSortVO filter, Long offset,
Integer length) throws SUPAdminException;
```

#### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### Examples

- **Retrieval**

```
Package_FilterSortVO filter = new Package_FilterSortVO();
FilterExpression<APPLICATION_PACKAGE> expression1 = new
FilterExpression<APPLICATION_PACKAGE>();
expression1 = expression1.eq(APPLICATION_PACKAGE.APPLICATION_ID,
"appl");
filter.setFilterExpression(expression1);
filter.setSortField(APPLICATION_PACKAGE.DOMAIN);
filter.setSortOrder(SORT_ORDER.ASCENDING);
PaginationResult<ApplicationPackageVO> apps =
supApplication.getApplicationPackages(filter, 01, 100);
```

### Managing Agency Applications

Use the SUPApplication interface to manage Agency applications.

#### Create Agency Shell Application

Creates and deploys an Agency application without a package.

#### Syntax

```
void creatApplication(String applicationId, String displayName,
String description,
CLIENTSDK_TYPE type, boolean enabled) throws
SUPAdminException;
```

#### Parameters

- **applicationId** – the unique ID for the application.
- **displayName** – the display name for the application.
- **description** – (optional) the string that describes the application.

- **type** – must be of type AGENTRY. All other types are ignored.
- **enable** – enable the application immediately after its creation.

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Create application** – the client invokes `SUPApplication`:

```
createApplication("agentryappl", "agentryapplDisplay",
 "DescribeAgentry", CLIENT_SDK_TYPE.Agentry, true);
```

### See also

- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Create Agentry Application with Package

Creates and deploys an Agentry application with a package.

### Syntax

```
void createApplication (String appID, String
 displayName, String description, CLIENT_SDK_TYPE type, Boolean
enable, String
 appPackage, Boolean immediate, Date time)
 throws SUPAdminException;
```

## Parameters

- **applicationId** – the unique ID for the application.
- **displayName** – the display name for the application.
- **description** – (optional) the string that describes the application.
- **type** – must be of type AGENTRY. All other types are ignored.
- **appPackage** – the application package file for the application. Agentry packages must be saved to the server's `agp/agpz/zip` directory.
- **immediate** – include this parameter to immediately activate the application.
- **time** – the time to activate the application, when you do not immediately activate it.
- **enable** – include this parameter to immediately enable the application. Otherwise, it remains disabled.

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Create application with package**

```
createApplication("agentryapp1", "agentryapp1Display",
 "DescribeAgentry", "c:\\app.agentry2.agp", true, null)
```

## **See also**

- *Create Agentry Shell Application* on page 455
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Deploy Agentry Application

Deploys an Agentry application to the Agentry server. SAP Mobile Server parses the application manifest to decide which deployment mode to use; new or update.

You can use this method only to deploy Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
void deployAgentryApplication (String appId, String
 fileLocation ,boolean
 immediate, Date time)
 throws SUPAdminException;
```

### Parameters

- **applicationId** – the unique ID for the application.
- **packageZIP** – the application package file for the application. Agentry packages must be saved to the server's `agp/agpz/zip` directory.
- **immediate** – include this parameter to immediately activate the application.
- **time** – the time to activate the application, when you do not immediately activate it.

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Deploy application**

```
deployAgentryApplication("testApp", "c:\
\app.agentry2.agp", true, null)
```

### See also

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466



- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Download Agentry Application

Downloads the application definition, locale, and resource files according to the file list.

This method can be used only by Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
void
 downloadAgentryApplication(String fileName, String
appId, List<String> files)
 throws SUPAdminException;
```

### Parameters

- **appId** – the unique ID for the application.
- **fileName** – The name of the application file to be downloaded.
- **files** – the list of files to be downloaded (application definition files, locale files, resource files).

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Create application**

```
ArrayList<String> files=new ArrayList<String>();
files.add("enLocale");
files.add("cnLocale");
supApplication.downloadAgentryApplication("agentryappl.zip",
"agentry.appl", files);
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458

## Management API

- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Get Client SDK Type

Gets the type of SDK client for the application: Object API, Hybrid App, OData, REST API, or Agentry.

### Syntax

```
CLIENTSDK_TYPE getClientSDKType(String appId)
throws SUPAdminException;
```

### Parameters

- **applicationId** – the unique ID for the application.

### Returns

The client SDK type according to application ID used.

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Get SDK type**

```
SupApplication.getClientSDKType("agentry.app1");
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458

- *Download Agentry Application* on page 459
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Enable or Disable Agentry Application

Enables or disables an Agentry application that is deployed on the Agentry server.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
void setAgentryApplicationStatus (String appId, Boolean flag)
 throws SUPAdminException;
```

### Parameters

- **appId** – the unique ID for the application.
- **flag** – if set to true, enables the application; otherwise, it remains disabled.

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Enable the application**

```
SupApplication.setAgentryApplicationStatus
("synclo.application", true)
```

### See also

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456

- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### *Start, Stop, or Restart Agentry Application*

Starts, stops, or restarts an Agentry application on the Agentry server node instance.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### **Syntax**

```
void startAgentryApplication (String appId)
 throws SUPAdminException;
```

```
void stopAgentryApplication (String appId)
 throws SUPAdminException;
```

```
void restartAgentryApplication (String appId)
 throws SUPAdminException;
```

### **Parameters**

- **appId** – the unique ID for the application.

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Start the application**

```
SupApplication.startAgentryApplication ("synclo.application")
```

- **Stop the application**

```
SupApplication.stopAgentryApplication ("synclo.application")
```

- **Restart the Application**

```
SupApplication.restartAgentryApplication ("synclo.application")
```

## **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### *Disconnect Active Agentry Application Users*

Disconnects active users.

This method can be used only by Agentry applications; for other application types, the method throws a not supported exception.

## Syntax

```
void disconnectAgentryActiveUsers (String appId, List<String>
userIds)
 throws SUPAdminException;
```

### **Parameters**

- **appId** – the unique ID for the application.
- **userID** – the unique ID for the active application user.

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Disconnect active application user**

```
SupApplication.disconnectAgentryActiveUsers
("synclo.application", users);
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### **List Active Agentry Application Users**

Retrieves a list of active Agentry application users.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### **Syntax**

```
void List<AgentryActiveUserVO> getAgentryActiveUsers (String
appId) throws SUPAdminException;
```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Active Agentry user list**

```
SupApplication.getAgentryActiveUsers
("syclo.application")
```

## **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

## **Roll Agentry Application Log**

As the log file reaches its limit, rolls the log file over.

This method can be used only by Agentry applications; for other application types, the method throws a not supported exception.

## **Syntax**

```
void rollAgentryApplicationLog (String appId)
 throws SUPAdminException;
```

### Parameters

- **appId** – the unique ID for the application.

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Roll log**

```
SupApplication.rollAgentryApplicationlog ("syclo.application")
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Get Agentry Application Log Properties

Gets the configuration properties for the Agentry application server instance log files.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
void AgentryApplicationLogSettingVO
getAgentryApplicationLogSetting(String appId)
throws SUPAdminException;
```



## Parameters

- **applicationId** – the unique ID for the application.

## Returns

The Agentry application server instance log settings object.

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieve log settings**

```
SupApplication.getAgentryApplicationLogSetting
("syclo.application");
```

## **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Update Agentry Application Log Properties

Updates the configuration properties for the Agentry application server instance log files. Settings must be retrieved before they can be updated.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### **Syntax**

```
void updateAgentryApplicationLogSetting(String appId,
AgentryApplicationLogSettingVO
 setting)
 throws SUPAdminException;
```

### **Parameters**

- **applicationId** – the unique ID for the application.

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Update settings**

```
SupApplication.updateAgentryApplicationLogSetting
("synclo.application", setting);
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Get Agentry Application Configuration

Retrieves the configuration properties of the Agentry application. You can also list properties by category or list only the categories.

This method can be used only by Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
void AgentryApplicationConfigurationVO
getAgentryApplicationConfiguration (String
 appId) throws SUPAdminException;
```

### Parameters

- **appId** – the unique ID for the application.

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### Examples

- **Get oroperties**

```
AgentryApplicationConfigurationVO
setting=SupApplication.getAgentryApplicationConfiguration
("syclo.application");
```

- **Get properties by category**

```
setting.getPropertiesByCategory ("ANGEL Front End");
```

- **Get category**

```
setting.getCategories ("ANGEL Front End");
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464

- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Update Application Configuration

Updates the application's configuration. The client must get the properties before they can be updated.

This method can only be used only by Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
void updateAgentryApplicationConfiguration (String
appId, AgentryApplicationConfigurationVO
vo) throws SUPAdminException;
```

### Parameters

- **appId** – the unique ID for the application.
- **AgentryApplicationConfigurationVO** – the application configuration object.

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update retrieved properties**

```
SupApplication.updateAgentryApplicationConnectionSetting
("syclo.application", setting);
```

### **See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461

- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### Get Status of Agentry Application from Server Node

Gets Agentry application related properties, such as application node status and application server information, for each node.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
Collection<AgentryApplicationNodeStatusVO>
getAgentryApplicationNodeStatuses (String
 appId)
 throws SUPAdminException;
```

### Parameters

- **appId** – the unique ID for the application.

### Returns

Returns list of application node status values.

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Get status of node**

```
public class AgentryApplicationNodeStatusVO {
 // retrieve agentry server info
 public Map<String,String> getAgentryServerInfo();

 //Get the node name
 public String getNodeName();
```

```
//Get the status for an agentry application host by this node
public APPLICATION_STATUS getStatus();
```

```
//Get the message
public String getMessage()
```

### See also

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

### List Agentry Application Definition Files

Returns a list of application definition files for the Agentry application.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
List<String> getAgentryApplicationDefinitionFiles(String appId)
throws SUPAdminException;
```

### Parameters

- **applicationId** – the unique ID for the application.

### Returns

A list of definition files by application ID used.

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Local Files* on page 473
- *List Agentry Application Resource Files* on page 474

**List Agentry Application Local Files**

Returns a list of application locale files for the Agentry application.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

**Syntax**

```
List<String> getAgentryApplicationLocalesFiles (String appId)
 throws SUPAdminException;
```

**Parameters**

- **applicationId** – the unique ID for the application.

**Returns**

A list of locale files used by the application.

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**See also**

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458

- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462
- *Disconnect Active Agentry Application Users* on page 463
- *List Active Agentry Application Users* on page 464
- *Roll Agentry Application Log* on page 465
- *Get Agentry Application Log Properties* on page 466
- *Update Agentry Application Log Properties* on page 467
- *Get Agentry Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agentry Application from Server Node* on page 471
- *List Agentry Application Definition Files* on page 472
- *List Agentry Application Resource Files* on page 474

### List Agentry Application Resource Files

Returns a list of application resource files for the Agentry application.

This method can be used only for Agentry applications; for other application types, the method throws a not supported exception.

### Syntax

```
List<String> getAgentryApplicationResourceFiles(String appId)
throws SUPAdminException;
```

### Parameters

- **applicationId** – the unique ID for the application.

### Returns

A list of resource files used by the application.

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### See also

- *Create Agentry Shell Application* on page 455
- *Create Agentry Application with Package* on page 456
- *Deploy Agentry Application* on page 458
- *Download Agentry Application* on page 459
- *Get Client SDK Type* on page 460
- *Enable or Disable Agentry Application* on page 461
- *Start, Stop, or Restart Agentry Application* on page 462



- *Disconnect Active Agency Application Users* on page 463
- *List Active Agency Application Users* on page 464
- *Roll Agency Application Log* on page 465
- *Get Agency Application Log Properties* on page 466
- *Update Agency Application Log Properties* on page 467
- *Get Agency Application Configuration* on page 469
- *Update Application Configuration* on page 470
- *Get Status of Agency Application from Server Node* on page 471
- *List Agency Application Definition Files* on page 472
- *List Agency Application Local Files* on page 473

### **Managing Application Connections**

Use the SUPApplication interface to manage registration of application connections.

Operations you can perform with this interface include:

- Retrieving a list of application connections
- Cloning application connections
- Registering or re-registering an application connection
- Updating application connection settings
- Deleting an application connection
- Locking or unlocking an application connection

#### **Retrieve Application Connections**

Retrieves a list of application connections that satisfy the given filter. You can filter application connections by: APPLICATION\_ID, APPLICATION\_CONNECTION\_ID, NUMERIC\_ID, SECURITY\_CONF, and USER\_ID. The return result is paginated.

#### **Syntax**

```
PaginationResult<ApplicationConnectionVO>
getApplicationConnections(AppConnection_FilterSortVO filter, Long
offset, Integer length) throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```
AppConnectionFilterSortVO filter = new
AppConnectionFilterSortVO();
FilterExpression<APPCONNECTION> fe = new FilterExpression<
```

```
APPCONNECTION >());
FilterExpression< APPCONNECTION > fe1 =
fe.eq(APPCONNECTION.USER_ID, "user");
filter.setFilterExpression(fe1);

PaginationResult<ApplicationConnectionVO> result = app
 .getApplicationConnections(filter, 0L, 10);
for (ApplicationConnectionVO appConn : result.getItems()) {
 System.out.println(appConn.getId());
}
```

### Cloning Application Connections

Registers an application connection by cloning an existing application connection.

### Syntax

```
java.util.Collection<java.lang.Integer>
cloneApplicationConnections(java.util.Collection<AppConnectionClone
RequestVO> cloneRequests,
AppConnectionSettingVO settings) throws SUPAdminException
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Clone application connection**

```
AppConnectionCloneRequestVO accrvo = new
AppConnectionCloneRequestVO();
Map<APPCONNECTION_CLONE, Object> req1 = new
HashMap<APPCONNECTION_CLONE, Object>();
req1.put(APPCONNECTION_CLONE.EXISTING_NUMERIC_ID, "8");
req1.put(APPCONNECTION_CLONE.ACTIVATION_CODE, "345");
req1.put(APPCONNECTION_CLONE.EXPIRATION_HOUR, "3");
req1.put(APPCONNECTION_CLONE.USER_ID, "river");
accrvo.setRequest(req1);

Collection<AppConnectionCloneRequestVO> reqs = new
ArrayList<AppConnectionCloneRequestVO>();
reqs.add(accrvo);

AppConnectionSettingVO acsvo = new AppConnectionSettingVO();
Map<APPCONNECTION_SETTING_FIELD, Object> setting = new
HashMap<APPCONNECTION_SETTING_FIELD, Object>();
setting.put(APPCONNECTION_SETTING_FIELD.SECURITY_CONF, "admin2");
setting.put(APPCONNECTION_SETTING_FIELD.ALLOW_ROAMING, "true");
acsvo.setSetting(setting);
app.cloneApplicationConnections(reqs, acsvo);
```

### Register an Application Connection

Registers a batch of application connections.

#### **Syntax**

```
java.util.Collection<java.lang.Integer>
registerApplicationConnections (java.lang.String
templateName, java.util.Collection<AppConnectionRegistrationRequestV
O> registrationRequests,
AppConnectionSettingVO settings) throws SUPAdminException
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Register application connection**

```
AppConnectionRegistrationRequestVO acrrvo1 = new
AppConnectionRegistrationRequestVO ();
AppConnectionRegistrationRequestVO acrrvo2 = new
AppConnectionRegistrationRequestVO ();

Map<APPCONNECTION_REGISTRATION, Object> req1 = new
HashMap<APPCONNECTION_REGISTRATION, Object> ();
req1.put (APPCONNECTION_REGISTRATION.USER_ID,
contextFactory.getProperty ("sup.app.user.1"));
req1.put (APPCONNECTION_REGISTRATION.ACTIVATION_CODE, "1234");
req1.put (APPCONNECTION_REGISTRATION.EXPIRATION_HOUR, "1");
acrrvo1.setRequest (req1);

Map<APPCONNECTION_REGISTRATION, Object> req2 = new
HashMap<APPCONNECTION_REGISTRATION, Object> ();
req2.put (APPCONNECTION_REGISTRATION.USER_ID,
contextFactory.getProperty ("sup.app.user.2"));
req2.put (APPCONNECTION_REGISTRATION.ACTIVATION_CODE, "5678");
req2.put (APPCONNECTION_REGISTRATION.EXPIRATION_HOUR, "1");
acrrvo2.setRequest (req2);

Collection<AppConnectionRegistrationRequestVO> reqs = new
ArrayList<AppConnectionRegistrationRequestVO> ();
reqs.add (acrrvo1);
reqs.add (acrrvo2);

AppConnectionSettingVO settings = new AppConnectionSettingVO ();
Map<APPCONNECTION_SETTING_FIELD, Object> setting = new
HashMap<APPCONNECTION_SETTING_FIELD, Object> ();
setting.put (APPCONNECTION_SETTING_FIELD.SECURITY_CONF,
contextFactory.getProperty ("sup.secconf.1"));
setting.put (APPCONNECTION_SETTING_FIELD.ALLOW_ROAMING, "true");
setting.put (APPCONNECTION_SETTING_FIELD.SERVER_NAME,
"localhost");
```

```
settings.setSetting(setting);
app.registerApplicationConnections(templateName, reqs, settings);
```

### Re-register an Application Connection

Re-registers an application connection.

### Syntax

```
java.util.Collection<java.lang.Integer>
reregisterApplicationConnections(java.util.Collection<AppConnection
ReregistrationRequestVO>
reregistrationRequests, AppConnectionSettingVO settings) throws
SUPAdminException
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Re-registration**

```
AppConnectionReregistrationRequestVO acrrvo1 = new
AppConnectionReregistrationRequestVO();

Map<APPCONNECTION_REREGISTRATION, Object> req1 = new
HashMap<APPCONNECTION_REREGISTRATION, Object>();
req1.put(APPCONNECTION_REREGISTRATION.EXISTING_NUMERIC_ID, "5");
req1.put(APPCONNECTION_REREGISTRATION.ACTIVATION_CODE, "15");
req1.put(APPCONNECTION_REREGISTRATION.EXPIRATION_HOUR, "2");
req1.put(APPCONNECTION_REREGISTRATION.USER_ID, "hel");
acrrvo1.setRequest(req1);

Collection<AppConnectionReregistrationRequestVO> reqs = new
ArrayList<AppConnectionReregistrationRequestVO>();
reqs.add(acrrvo1);

AppConnectionSettingVO settings = new AppConnectionSettingVO();
Map<APPCONNECTION_SETTING_FIELD, Object> setting = new
HashMap<APPCONNECTION_SETTING_FIELD, Object>();
setting.put(APPCONNECTION_SETTING_FIELD.SERVER_NAME, "helxp-
vm1");
setting.put(APPCONNECTION_SETTING_FIELD.SERVER_PORT, "8888");
setting.put(APPCONNECTION_SETTING_FIELD.FARM_ID, "1");
setting.put(APPCONNECTION_SETTING_FIELD.DOMAIN, "default");
setting.put(APPCONNECTION_SETTING_FIELD.SECURITY_CONF, "admin1");
settings.setSetting(setting);
app.reregisterApplicationConnections(reqs, settings);
```

### Retrieve Application Connection Setting

Retrieve the application connection setting for the specified field.

## **Syntax**

```
Object getApplicationConnectionSettings(Integer numericId,
APPCONNECTION_SETTING_FIELD field)
 throws SUPAdminException;
```

```
AppConnectionSettingVO
getApplicationConnectionSettings(java.lang.Integer numericId) throws
SUPAdminException
```

## **Parameters**

- **field** – the field for the setting.

## **Returns**

If successful, returns the field setting. If unsuccessful, throws SUPAdminException. If the field is invalid, returns null.

## **Examples**

- **Retrieve Notification Behavior** – Retrieves the NOTIFICATION\_MODE setting:

```
public enum
com.sybase.sup.admin.enumeration.application.APPCONNECTION_SETTING_
FIELD {
...
/**
 * Identifies the notification behaviour of the application
connection.
 * <p>
 *

 * Type is {@link APPCONNECTION_SETTING_FIELD_TYPE}.INTEGER 0: Only
Native
 * Notifications 1: Only Online/Payload Push 2: Online/Payload Push
with
 * Native Notifications
 */
@EnumAnn(alias = "Notification Mode")
NOTIFICATION_MODE (APPCONNECTION_SETTING_CATEGORY.APPLICATION,
APPCONNECTION_SETTING_FIELD_TYPE.INTEGER, 2907) {
},
...
}
```

## **Application Connection Settings Update**

Updates the settings of a list of application connections.

## **Syntax**

```
void
updateApplicationConnectionSettings(java.util.Collection<java.lang.
Integer> numericIds, AppConnectionSettingVO settings) throws
SUPAdminException
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update**

```
PaginationResult<ApplicationConnectionVO> result = app
 .getApplicationConnections(filter, 0L, NULL);
Collection<Integer> appConnIds = new ArrayList<Integer>();

for (ApplicationConnectionVO appConn : result.getItems()) {
 appConnIds.add(appConn.getNumericId());
}

AppConnectionSettingVO settings = new AppConnectionSettingVO();
Map<APPCONNECTION_SETTING_FIELD, Object> setting = new
 HashMap<APPCONNECTION_SETTING_FIELD, Object>();
setting.put(APPCONNECTION_SETTING_FIELD.SECURITY_CONF, "admin");
settings.setSetting(setting);
app.updateApplicationConnectionSettings(appConnIds, settings);
```

### Application Connection Deletion

Deletes a list of application connections.

### Syntax

```
void deleteApplicationConnections(Collection<Integer> numericIds)
 throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Create registration template** – deletes the specified registration templates ("Default" and "testTemplate2"):

```
Collection<Integer> appConnIds = new ArrayList<Integer>();
appConnIds.add(7);
appConnIds.add(8);

app.deleteApplicationConnections(appConnIds);
```

### Lock or Unlock Application Connection

Locks or unlocks a list of application connections.

### Syntax

```
void lockApplicationConnections(Collection<String>
 applicationConnectionIds) throws SUPAdminException;
```

```
void unlockApplicationConnections(Collection<String>
applicationConnectionIds) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Lock or Unlock Application Connection**

```
PaginationResult<ApplicationConnectionVO> result =
app.getApplicationConnections(filter, 0L, NULL);
Collection<String> appConnIds = new ArrayList<String>();

for (ApplicationConnectionVO appConn : result.getItems()) {
 appConnIds.add(appConn.getId());
}

app.lockApplicationConnection(appConnIds);
app.unlockApplicationConnection(appConnIds);
```

### **Usage**

This API requires the application connection ID of the application connection (and not the numeric ID of the application connection).

### **Managing Application Connection Templates**

Use the SUPApplication interface to manage application connection templates.

Operations you can perform with this interface include:

- Retrieving a list of application connection templates
- Creating an application connection template
- Updating application connection template settings
- Deleting an application connection template

#### **Assign a Hybrid App to Application Connection Templates**

Assign a Hybrid App to a list of application connection templates.

### **Syntax**

```
void assignMobileHybridAppToTemplates(MobileHybridAppIDVO
hybridAppID, List<Integer> templateIDs)
throws SUPAdminException;
```

### **Parameters**

- **hybridAppID** – The Hybrid App ID.

- **templateIDs** – The list of application connection template IDs to assign to the Hybrid App.

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Assign Hybrid App to Templates**

```
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setWID(1);
hybridAppID.setVersion(1);
List<Integer> templateIDs = new ArrayList<Integer>();
templateIDs.add(1);
hybridApp.assignMobileHybridAppToTemplates(hybridAppID,
templateIDs);
```

### **Unassign a Hybrid App from Application Connection Templates**

Unassigns a Hybrid App from a list of application connection templates.

### **Syntax**

```
void unassignMobileHybridAppFromTemplates(MobileHybridAppIDVO
hybridAppID, List<Integer> templateIDs)
 throws SUPAdminException;
```

### **Parameters**

- **hybridAppID** – The Hybrid App ID.
- **templateIDs** – The list of application connection templates to unassign from the Hybrid App.

### **Returns**

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Unassign Hybrid App from Templates**

```
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setWID(1);
hybridAppID.setVersion(1);
List<Integer> templateIDs = new ArrayList<Integer>();
templateIDs.add(1);
hybridApp.unassignMobileHybridAppFromTemplates(hybridAppID,
templateIDs);
```



**Retrieve List of Hybrid Apps Assigned to an Application Connection Template**

Retrieve the Hybrid App assignment information for an application connection template. This can only be done by the SAP Mobile Platform help desk or a SAP Mobile Platform administrator.

**Syntax**

```
List<MobileHybridAppAssignmentVO>
getTemplateHybridAppAssignments(int templateID)
 throws SUPAdminException;
```

**Parameters**

- **templateID** – The application connection template ID.

**Returns**

If successful, returns a list of Hybrid Apps. If unsuccessful, throws SUPAdminException.

**Examples**

- **Get Template Hybrid App Assignments**

```
int templateID = 1;
List<MobileHybridAppAssignmentVO> list =
hybridApp.getTemplateHybridAppAssignments(templateID);
System.out.println(list.size());
```

**Retrieve Status of Application Connection Templates for a Hybrid App**

Retrieves a template status for a Hybrid App. This can only be done by the SAP Mobile Platform helpdesk or a SAP Mobile Platform administrator.

**Syntax**

```
List<TemplateMobileHybridAppStatusVO>
getTemplateMobileHybridAppStatus(MobileHybridAppIDVO hybridAppID)
 throws SUPAdminException;
```

**Parameters**

- **hybridAppID** – The Hybrid App ID.

**Returns**

If successful, returns a list of application connection statuses. If unsuccessful, throws SUPAdminException.

### Examples

- **Get Template Hybrid App Status**

```
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setWID(1);
hybridAppID.setVersion(1);
List<TemplateMobileHybridAppStatusVO> list =
hybridApp.getTemplateMobileHybridAppStatus(hybridAppID);
System.out.println(list.size());
```

### Set Default Hybrid App for an Application Connection Template

Set the default Hybrid App for a list of application connection templates.

### Syntax

```
void setDefaultMobileHybridAppforTemplates (MobileHybridAppIDVO
hybridAppID, List<Integer> templateIDs)
 throws SUPAdminException;
```

### Parameters

- **hybridAppID** – The Hybrid App ID.
- **templateIDs** – The list of application connection templates to set the default Hybrid App.

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### Examples

- **Set Default Hybrid App for Templates**

```
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setWID(1);
hybridAppID.setVersion(1);
List<Integer> templateIDs = new ArrayList<Integer>();
templateIDs.add(1);
hybridApp.setDefaultMobileHybridAppforTemplates (hybridAppID,
templateIDs);
```

### Unset Default Hybrid App for an Application Connection Template

Remove the default Hybrid App for a list of application connection templates.

### Syntax

```
void unsetDefaultMobileHybridAppforTemplates (List<Integer>
templateIDs)
 throws SUPAdminException
```

## Parameters

- **templateIDs** – The list of application connection templates to remove the default Hybrid App.

## Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Examples

- **Unset Default Hybrid App for Templates**

```
List<Integer> templateIDs = new ArrayList<Integer>();
templateIDs.add(1);
hybridApp.unsetDefaultMobileHybridAppforTemplates(templateIDs);
```

## Application Connection Template Retrieval

Retrieves a list of application connection templates.

## Syntax

```
PaginationResult<ApplicationConnectionTemplateVO>
getApplicationConnectionTemplates(AppConnectionTemplateFilterSortVO
filter, Long offset, Integer length) throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieval**

```
AppConnectionTemplateFilterSortVO filter = new
AppConnectionTemplateFilterSortVO();
FilterExpression<APPCONNECTION_TEMPLATE> fe = new
FilterExpression<APPCONNECTION_TEMPLATE>();

FilterExpression<APPCONNECTION_TEMPLATE> fe1 =
fe.eq(APPCONNECTION_TEMPLATE.DOMAIN, "default");

FilterExpression<APPCONNECTION_TEMPLATE> fe2 =
fe.eq(APPCONNECTION_TEMPLATE.SECURITY_CONF, "admin");

fe = fe1.and(fe2);
filter.setFilterExpression(fe);
PaginationResult<ApplicationConnectionTemplateVO> result = app
 .getApplicationConnectionTemplates(filter, 0L, 10);
for (ApplicationConnectionTemplateVO appConnT :
 result.getItems()) {
```

```
System.out.println(appConnT.getName());
}
```

### Application Connection Template Creation

Creates an application connection templates with the specified settings.

#### Syntax

```
void
createApplicationConnectionTemplate(ApplicationConnectionTemplateVO
applicationConnectionTemplate, Map settings) throws
SUPAdminException;
```

#### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### Examples

- **Creation**

```
AppConnectionSettingVO acsvo = new AppConnectionSettingVO();
Map<APPCONNECTION_SETTING_FIELD, Object> setting = new
HashMap<APPCONNECTION_SETTING_FIELD, Object>();
setting.put (APPCONNECTION_SETTING_FIELD.SECURITY_CONF,
"mySecurity");
acsvo.setSetting(setting);

app.createApplicationConnectionTemplate("MyTemplate",
"Short description", acsvo);
```

### Update of Application Connection Template Settings

Updates application connection template settings.

#### Syntax

```
void updateApplicationConnectionTemplateSettings(java.lang.String
templateName, AppConnectionSettingVO settings) throws
SUPAdminException
```

#### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### Examples

- **Update**

```
AppConnectionSettingVO settings = new AppConnectionSettingVO();
Map<APPCONNECTION_SETTING_FIELD, Object> setting = new
HashMap<APPCONNECTION_SETTING_FIELD, Object>();
setting.put (APPCONNECTION_SETTING_FIELD.SECURITY_CONF, "admin");
```

```

setting.put(APPCONNECTION_SETTING_FIELD.ACTIVATION_CODE_LENGTH,
"9");
setting.put(APPCONNECTION_SETTING_FIELD.ALLOW_ROAMING, "true");
settings.setSetting(setting);
app.updateApplicationConnectionTemplateSettings("template 1",
settings);

```

### **Application Connection Template Deletion**

Deletes a list of application connection templates.

### **Syntax**

```

void deleteApplicationConnectionTemplates(List<String>
templateNames) throws SUPAdminException;

```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Deletion**

```

Collection<String> names = new ArrayList<String>();
names.add("MyTemplate");

app.deleteApplicationConnectionTemplates(names);

```

### **Managing Application Customization Resource Bundles**

Use the SUPApplication interface to manage customization resource bundles for OData SDK Android and iOS applications.

### **Retrieve an Application Customization Resource Bundle ID**

Retrieves a customization resource bundle identifier from its binaries.

### **Syntax**

```

String getCustomizationResourceBundleId(byte[]
customizationResourceBundleBytes)
throws SUPAdminException;

```

### **Returns**

If successful, returns the ID of the supplied customization resource bundle binaries. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval**

```
File file = new File("C:\\CustomizationResourceBundle.jar");
InputStream is = new FileInputStream(file);
byte[] bytes = new byte[is.available()];
is.read(bytes);
app.getCustomizationResourceId(bytes);
```

### **Deploy an Application Customization Resource Bundle**

Deploys a customization resource bundle to an application.

### **Syntax**

```
void deployCustomizationResourceBundle(java.lang.String
applicationId,
byte[] customizationResourceBundleBytes)
throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Deploy**

```
File file = new File("C:\\CustomizationResourceBundle.jar");
InputStream is = new FileInputStream(file);
byte[] bytes = new byte[is.available()];
is.read(bytes);
app.deployCustomizationResourceBundle("<application ID>", bytes);
```

### **Export an Application Customization Resource Bundle**

Exports a customization resource bundle from an application to a JAR file.

### **Syntax**

```
void exportCustomizationResourceBundle(java.io.File file,
java.lang.String applicationId,
java.lang.String customizationResourceId)
throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Export**

```
File file = new File("C:\\ExportedCRB.jar");
app.exportCustomizationResourceBundle(file, "<application ID>",
"<customization resource bundle ID>");
```

### Assign an Application Customization Resource Bundle

Assigns a customization resource bundle to all qualified application connections or application connection templates for the application ID.

### Syntax

```
void assignCustomizationResourceBundle(java.lang.String
applicationId,
java.lang.String customizationResourceBundleId,
CUSTOMIZATION_RESOURCE_BUNDLE_REFERRAL referral)
throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Assign to all application connections with the defined application ID**

```
app.assignCustomizationResourceBundle("<application ID>",
"<customization resource bundle ID>",
CUSTOMIZATION_RESOURCE_BUNDLE_REFERRAL.APPCONNECTION);
```

- **Assign to all application connection templates with the defined application ID**

```
app.assignCustomizationResourceBundle("<application ID>",
"<customization resource bundle ID>",
CUSTOMIZATION_RESOURCE_BUNDLE_REFERRAL.APPCONNECTION_TEMPLATE);
```

### Unassign an Application Customization Resource Bundle

Unassigns a customization resource bundle from all applicable application connections or application connection templates for the application ID.

### Syntax

```
void unassignCustomizationResourceBundle(java.lang.String
applicationId,
java.lang.String customizationResourceBundleId,
CUSTOMIZATION_RESOURCE_BUNDLE_REFERRAL referral)
throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Unassign from all application connections with the defined application ID**

```
app.unassignCustomizationResourceBundle("<application ID>",
 "<customization resource bundle ID>",
 CUSTOMIZATION_RESOURCE_BUNDLE_REFERRAL.APPCONNECTION);
```

- **Unassign from all application connection templates with the defined application ID**

```
app.unassignCustomizationResourceBundle("<application ID>",
 "<customization resource bundle ID>",
 CUSTOMIZATION_RESOURCE_BUNDLE_REFERRAL.APPCONNECTION_TEMPLATE);
```

### **Delete an Application Customization Resource Bundle**

Deletes a customization resource bundles from an application. You cannot delete a customization resource bundle if it is assigned to an application connection or application connection template; you must unassign it first.

### **Syntax**

```
void deleteCustomizationResourceBundle(java.lang.String
applicationId,
 java.lang.String customizationResourceId)
 throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Delete**

```
app.deleteCustomizationResourceBundle("<application ID>",
 "<customization resource bundle ID>");
```

### **Managing Application Push Configuration**

Methods for managing application push configurations.



### Create a push configuration

Creates a push configuration if there is not such a one, otherwise updates the existing one.

#### Syntax

```
public void
saveApplicationPushConfiguration (ApplicationPushConfigurationVO
 pushConfiguration) throws SUPAdminException;
```

#### Parameters

- **pushConfiguration** – Application push configuration to create or update.

#### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

#### Examples

- **Create an application push configuration**

```
app.saveApplicationPushConfiguration("<ApplicationPushConfigurat
ionVO>");
```

### Delete a list of push configurations

Delete a list of push configurations from the specified application.

#### Syntax

```
public void
deleteApplicationPushConfiguration (List<ApplicationPushConfiguratio
nVO> list)
 throws SUPAdminException;
```

#### Parameters

- **list** – Application push configuration objects.

#### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

### Examples

- **Delete a list of push configurations**

```
app.deleteApplicationPushConfiguration("List<ApplicationPushConfigurationVO>");
```

### Retrieve a list of push application configurations

Retrieves a list of push application configurations.

### Syntax

```
public List<ApplicationPushConfigurationVO>
getApplicationPushConfigurations(String applicationId,
APPLICATION_PUSH_CONFIGURATION_TYPE type)
 throws SUPAdminException;
```

### Parameters

- **applicationId** – Application Identifier
- **type** – See APPLICATION\_PUSH\_CONFIGURATION\_TYPE

### Returns

If successful, returns a list of application push configurations. If unsuccessful, throws SUPAdminException.

### Examples

- **Retrieve a list of push application configurations** – Retrieves a list of push application configurations for app.

```
app.getApplicationPushConfigurations("<application ID>",
"<APPLICATION_PUSH_CONFIGURATION_TYPE>");
```

## Monitoring SAP Mobile Platform Components

SUPMonitor provides most of the operations related to monitoring of SAP Mobile Platform components. SUPCluster provides additional operations.

### Start Monitoring Management

Starts the management of an SAP Mobile Server monitoring operations.

### Syntax

```
public static SUPMonitor getSUPMonitor(ClusterContext
clusterContext) throws SUPAdminException;
```

## **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Start monitoring management**

```
clusterContext = serverContext.getClusterContext("<cluster
name>");
SUPMonitor supMonitor =
SUPObjectFactory.getSUPMonitor(clusterContext);
```

## **Usage**

To manage SAP Mobile Server monitoring operations, you must create an instance of `SUPMonitor`.

## **Retrieval of Monitoring Profiles Using SUPCluster**

Retrieves the monitoring profiles in a cluster.

## **Syntax**

```
Collection<MonitoringProfileVO> getMonitoringProfiles() throws
SUPAdminException;
```

```
MonitoringProfileVO getMonitoringProfile(String name) throws
SUPAdminException;
```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Retrieval**

```
Collection<MonitoringProfileVO> mpvos = supCluster
 .getMonitoringProfiles();
MonitoringProfileVO mpvo = supCluster
 .getMonitoringProfile("<monitoring configuration name>");
System.out.println(mpvo.getName());
```

## **Creation of a Monitoring Profile Using SUPCluster**

Creates a monitoring profile in a cluster.

## **Syntax**

```
void createMonitoringProfile(MonitoringProfileVO mpvo) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Create monitoring profile**

```
//Create a monitoring profile
MonitoringProfileVO mpvo_new = new MonitoringProfileVO();
mpvo_new.setName("<monitoring configuration new name>");
mpvo_new.setDurationType(MONITORING_DURATION_TYPE.SCHEDULED);
mpvo_new.setEnabled(true);

MonitoredDomain md = new MonitoredDomain("<domain name>");
md.setName("<domain name>");
MonitoredPackage mp1 = new MonitoredPackage("<package name 1>");
MonitoredPackage mp2 = new MonitoredPackage("<package name 2>");
md.setMonitoredPackages(Arrays
 .asList(new MonitoredPackage[] { mp1, mp2 }));
mpvo_new.setMonitoredDomains(Arrays.asList(new MonitoredDomain[]
 { md }));

ScheduleVO svo = new ScheduleVO();
svo.setEndDate(new Date());
svo.setEndTime(new Date());
svo.setStartDate(new Date(0));
svo.setStartTime(new Date(0));
svo.setInterval(1234);
svo.setFreq(SCHEDULE_FREQ.INTERVAL);
EnumSet<DAY_OF_WEEK> dayofweeks =
EnumSet.noneOf(DAY_OF_WEEK.class);
svo.setDaysofweek(dayofweeks);
dayofweeks.add(DAY_OF_WEEK.MONDAY);
mpvo_new.setSchedule(svo);
supCluster.createMonitoringProfile(mpvo_new);
```

### Update of a Monitoring Profile Using SUPCluster

Updates a monitoring profile in a cluster.

### Syntax

```
void updateMonitoringProfile(MonitoringProfileVO monitoringProfile)
throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update monitoring profile**

```
// Update monitoring profile
MonitoringProfileVO mpvo = supCluster
```

```

 .getMonitoringProfile("<monitoring configuration name>");
mpvo.getSchedule().setFreq(SCHEDULE_FREQ.INTERVAL);
mpvo.getSchedule().setInterval(200000);
supCluster.updateMonitoringProfile(mpvo);

```

### **Usage**

A monitoring profile you create with this method replaces a profile with the same name on the SAP Mobile Server.

### **Deletion of a Monitoring Profile Using SUPCluster**

Deletes a monitoring profiles from a cluster.

### **Syntax**

```
void deleteMonitoringProfile(String name) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Delete monitoring profile**

```

// Delete monitoring profile
supCluster.deleteMonitoringProfile("<monitoring configuration
name>");

```

### **Deletion of Monitoring Data Using SUPCluster**

Deletes monitoring data.

### **Syntax**

```
void deleteMonitoringData(Date startTime, Date endTime) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Delete monitoring data** – deletes monitoring data for the specified time period (between the startTime and the endTime):

```

Date startTime = new Date(0);
Date endTime = new Date();
supCluster.deleteMonitoringData(startTime, endTime);

```

### **Construct a Path to the Monitored Object**

To retrieve monitoring data, you must provide an instance or collection of `MonitoredObject` to specify the data that gets returned.

`MonitoredObject` contains subclasses in this logical hierarchy:

- `MonitoredCluster`
  - `MonitoredDomain`
    - `MonitoredPackage`
      - `MonitoredSyncGroup`
      - `MonitoredCacheGroup`
    - `MonitoredMBO`
      - `MonitoredOperation`

With this hierarchy, an object can be identified using a path-like structure. Such a path acts as a context against which monitoring data is searched and returned. Follow these rules when constructing a path:

- Start with `MonitoredCluster`.
- Except for `MonitoredCluster`, if `Monitored*` appears in a path, then the class logically above it is in the path.
- `MonitoredSyncGroup` and `MonitoredCache` are mutual exclusive in a path.

### **Retrieval of a Large Volume of Monitoring Data**

Retrieves a specified portion of a large volume of monitoring data (for example, user access histories).

#### **Syntax**

```
Long getSecurityLogHistoryCount(Collection<MonitoredObject>
monitoredObjects, Boolean accessResult, Date startTime, Date
endTime) throws SUPAdminException;
```

```
Collection<SecurityLogHistoryVO>
getSecurityLogHistory(Collection<MonitoredObject> monitoredObjects,
Boolean accessResult, Date startTime, Date endTime, Long offset,
Integer length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval**

```

MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });

long count = supMonitor.getSecurityLogHistoryCount(mos, null,
 null, null);
Collection<SecurityLogHistoryVO> slhvos =
 supMonitor.getSecurityLogHistory(mos, null,
 null, null, null, null, null);
for (SecurityLogHistoryVO slhvo : slhvos) {
 System.out.println(slhvo.getUserName());
}
long offset = slhvos.size();
while (offset < count) {
 slhvos = supMonitor.getSecurityLogHistory(mos, null,
 null, null, offset, null, null);
 for (SecurityLogHistoryVO slhvo : slhvos) {
 System.out.println(slhvo.getUserName());
 }
 offset += slhvos.size();
}

```

## Usage

When monitoring a large volume of data, a paginated API allows you to get a total row count for retrieving the data in chunks. `Offset` specifies where the returned data starts for this call. `Length` specifies the maximum number of records returned for this call.

## Specify Result Sorting

You can specify an instance of `SortedField` to sort the returned result on the given field in the given order (ascending or descending).

Each type of monitoring data has a different set of sortable fields.

- Data change notification
  - DOMAIN
  - NOTIFICATION\_TIME
  - PACKAGE
  - PROCESS\_TIME
  - PUBLICATION
- Device notification
  - DEVICE\_ID
  - DOMAIN
  - NOTIFICATION\_TIME
  - PACKAGE

## Management API

- PUBLICATION
- SUBSCRIPTION\_ID
- USER\_NAME
- Messaging summary
  - DOMAIN\_NAME
  - LAST\_TIME\_IN
  - LAST\_TIME\_OUT
  - PACKAGE
  - SUBSCRIPTION\_COMMAND\_COUNT
  - TOTAL\_ERRORS
  - TOTAL\_MESSAGES\_RECEIVED
  - TOTAL\_MESSAGES\_SENT
  - TOTAL\_OPERATION\_REPLAYS
  - TOTAL\_PAYLOAD\_RECEIVED
  - TOTAL\_PAYLOAD\_SENT
- Messaging details
  - DEVICE
  - DOMAIN\_NAME
  - ERROR
  - FINISH\_TIME
  - MBO
  - MESSAGE\_TYPE
  - OPERATION\_NAME
  - PACKAGE
  - PAYLOAD\_SIZE
  - PROCESS\_TIME
  - START\_TIME
  - USER
- Replication summary
  - DOMAIN\_NAME
  - PACKAGE
  - START\_TIME
  - SYNC\_TIME
  - TOTAL\_BYTES\_RECEIVED
  - TOTAL\_BYTES\_SENT
  - TOTAL\_ERRORS
  - TOTAL\_OPERATION\_REPLAYS
  - TOTAL\_ROWS\_SENT
- Replication details



- BYTES\_TRANSFERRED
- DEVICE
- DOMAIN\_NAME
- ERROR
- FINISH\_TIME
- OPERATION\_NAME
- OPERATION\_NAME
- PACKAGE
- START\_TIME
- SYNC\_PHASE
- TOTAL\_BYTES\_SENT
- TOTAL\_ROWS\_SENT
- USER
- Security access
  - DEVICE\_ID
  - DOMAIN
  - OUTCOME
  - PACKAGE
  - SECURITY\_CONFIGURATION
  - TIME
  - USER

### **Retrieval of Security Log History**

Retrieves a security log history for specified monitored objects, determines how many records are available, and specifies how to retrieve and sort the data.

### **Syntax**

```
Long getSecurityLogHistoryCount(Collection<MonitoredObject>
monitoredObjects, Boolean accessResult, Date startTime, Date
endTime) throws SUPAdminException;
```

```
Collection<SecurityLogHistoryVO>
getSecurityLogHistory(Collection<MonitoredObject> monitoredObjects,
Boolean accessResult, Date startTime, Date endTime, Long offset,
Integer length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

**Examples**• **Retrieval**

```

// Prepare monitored objects
MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
mc.addMonitoredDomain(new MonitoredDomain("test"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });

// Prepare time range
Date startTime = new Date(0);
Date endTime = new Date();

// Should only return successful access
Boolean accessResult = true;

// Starting from 10th record
Long offset = 10L;
// Try to retrieve 10000 records
Integer target = 10000;

// Specify sorting field and sorting order
SortedField<SortedField.SECURITY_ACCESS> sf = new
SortedField<SortedField.SECURITY_ACCESS>(
 SECURITY_ACCESS.DOMAIN, SORT_ORDER.ASCENDING);

// See how many records are available
long count = supMonitor.getSecurityLogHistoryCount(mos,
accessResult,
 startTime, endTime);
long available = Math.min(count - offset, target);
if (available < 1) {
 System.out.println("No monitoring data found at offset " +
offset);
 return;
} else {
 System.out.println("There " + available
 + " records monitoring data at offset " + offset);
}

// Specify the preferred record number to be fetched from server
in one
// call.
// Management server has imposed a upper limit of 500 for sake of
// performance.
Integer length = new Integer(new Long(Math.min(500, available))
 .intValue());
Collection<SecurityLogHistoryVO> slhvos =
supMonitor.getSecurityLogHistory(mos,
 accessResult, startTime, endTime, offset, length, sf);
// All the available records can be fetched at one call.
if (slhvos.size() == available) {
 System.out.println("Fetched " + available + " of " + available
 + " records of monitoring data.");
}

```

```

 return;
 }
 long read = slhvos.size();
 offset += read;
 while (read < available) {
 slhvos = supMonitor.getSecurityLogHistory(mos, accessResult,
 startTime, endTime, offset, length, sf);
 System.out.println("Fetched " + slhvos.size() + " of " +
 available
 + " records of monitoring data.");
 read += slhvos.size();
 offset += read;
 }
}

```

### **Retrieval of Current Messaging Requests**

Retrieves current messaging requests for the specified domains and packages.

#### **Syntax**

```

Collection<MessagingRequestVO>
getMessagingRequests(Collection<MonitoredObject> monitoredObjects)
throws SUPAdminException;

```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```

MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
for (MessagingRequestVO mrvo :
 supMonitor.getMessagingRequests(mos)) {
 System.out.println(mrvo.getPackageName());
}

```

### **Retrieval of Detailed Messaging History**

Retrieves a detailed messaging history for the specified domains and packages.

#### **Syntax**

```
Collection<MessagingHistoryDetailVO>
getMessagingHistoryDetail(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime, Long offset, Integer
length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval** – retrieves a detailed messaging history for the specified domains and packages (the "test\_mbs:1.0" and "test\_mbs:2.0" packages from the "default" domain, and the "test\_mbs:3.0" and "test\_mbs:4.0" packages from the "test" domain):

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getMessagingHistoryDetail(mos,
null, null, null, null, null));
```

---

**Note:** See *Developer Guide: SAP Mobile Server Runtime > Management API > Code Samples > Monitoring SAP Mobile Platform Components > Retrieval of a Large Volume of Monitoring Data* for handling the large volume of data that this method may retrieve.

---

### **Retrieval of Summary Messaging History**

Retrieves a summary of the messaging history for the specified domains and packages.

#### **Syntax**

```
Collection<MessagingHistorySummaryVO>
getMessagingHistorySummary(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime, Long offset, Integer
length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval** – retrieves a summary of the messaging history for the specified domains and packages (the "test\_mbs:1.0" and "test\_mbs:2.0" packages from the "default" domain, and the "test\_mbs:3.0" and "test\_mbs:4.0" packages from the "test" domain):

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getMessagingHistorySummary(mos,
 null, null, null, null, null));
```

**Note:** See *Developer Guide: SAP Mobile Server Runtime > Management API > Code Samples > Monitoring SAP Mobile Platform Components > Retrieval of a Large Volume of Monitoring Data* for handling the large volume of data that this method may retrieve.

## Messaging Performance Retrieval

Retrieves the messaging performance data for the specified domains and packages.

## Syntax

```
MessagingPerformanceVO
getMessagingPerformance(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime) throws
SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval** – retrieves the messaging performance data for the specified domains and packages (the "test\_mbs:1.0" and "test\_mbs:2.0" packages from the "default" domain, and the "test\_mbs:3.0" and "test\_mbs:4.0" packages from the "test" domain):

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
```

```
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
MessagingPerformanceVO mpvo =
 supMonitor.getMessagingPerformance(mos,
 null, null);
System.out.println(mpvo.getMboForMaxProcessTime());
```

### **Messaging Statistics Retrieval**

Retrieves the messaging statistics for a cluster, a domain, a package, or a specific mobile business object.

### **Syntax**

```
MessagingStatisticsVO getMessagingStatistics(MonitoredObject
monitoredObject, Date startTime, Date endTime) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Cluster-level messaging statistics** – retrieves the messaging statistics for all domains in a cluster:

```
MonitoredCluster mc = new MonitoredCluster();

// Retrieve cluster-level messaging statistics (statistics for all
domains).
supMonitor.getMessagingStatistics(mc, null, null);
```

- **Domain-level messaging statistics** – retrieves the messaging statistics for all packages in a domain:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md = new MonitoredDomain("default");

// Retrieve domain-level messaging statistics (statistics for all
packages).
mc.addMonitoredDomain(md);
supMonitor.getMessagingStatistics(mc, null, null);
```

- **Package-level messaging statistics** – retrieves the messaging statistics for all MBOs in a package:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md = new MonitoredDomain("default");
```

```

MonitoredPackage mp = new MonitoredPackage("test_mbs:1.0");

// Retrieve package-level messaging statistics (statistics for all
MBOs).
md.addMonitoredPackage(mp);
supMonitor.getMessagingStatistics(mc, null, null);

```

- **MBO messaging statistics** – retrieves the messaging statistics for a specific mobile business object:

```

MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md = new MonitoredDomain("default");
MonitoredPackage mp = new MonitoredPackage("test_mbs:1.0");
// Monitored cache does not contribute to messaging statistics,
but in
// order to retain the validity of the monitored object path, it
should be
// part of the path.
MonitoredCacheGroup mcg = new MonitoredCacheGroup("Default");
MonitoredMBO mmbo = new MonitoredMBO("Customer");

// Retrieve messaging statistics for a specific MBO.
mcg.addMonitoredMBO(mmbo);
supMonitor.getMessagingStatistics(mc, null, null);

```

### **Retrieval of Current Replication Requests**

Retrieves current replication requests for the specified domains and packages.

#### **Syntax**

```

Collection<ReplicationRequestVO>
getReplicationRequests(Collection<MonitoredObject>
monitoredObjects) throws SUPAdminException;

```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```

MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays

```

```
.asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getReplicationRequests(mos));
```

### **Retrieval of Detailed Replication History**

Retrieves a detailed replication history for the specified domains and packages.

#### **Syntax**

```
Collection<ReplicationHistoryDetailVO>
getReplicationHistoryDetail(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime, Long offset, Integer
length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
.asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getReplicationHistoryDetail(mos,
null, null, null, null, null));
```

### **Retrieval of Summary Replication History**

Retrieves a summary of replication history for the specified domains and packages.

#### **Syntax**

```
Collection<ReplicationHistorySummaryVO>
getReplicationHistorySummary(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime, Long offset, Integer
length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.



## Examples

- **Retrieval**

```

MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getReplicationHistorySummary(mos,
 null, null, null, null));

```

## Replication Performance Retrieval

Retrieves replication performance data for the specified domains and packages.

### Syntax

```

ReplicationPerformanceVO
getReplicationPerformance(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime) throws
SUPAdminException;

```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieval**

```

MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
ReplicationPerformanceVO rpvo =
supMonitor.getReplicationPerformance(mos, null, null);
System.out.println(rpvo.getMaxSyncTime());

```

### **Replication Statistics Retrieval**

Retrieves the replication statistics for a cluster, a domain, a package, a mobile business object, or a specific user.

### **Syntax**

```
ReplicationStatisticsVO getReplicationStatistics(MonitoredObject
monitoredObject, Date startTime, Date endTime) throws
SUPAdminException;
```

```
ReplicationStatisticsVO getReplicationStatistics(java.lang.String
user, java.util.Date startTime, java.util.Date endTime)
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Cluster-level replication statistics** – retrieves the replication statistics for all domains in a cluster:

```
MonitoredCluster mc = new MonitoredCluster();

//Retrieve cluster-level replication statistics (for all domains).
supMonitor.getReplicationStatistics(mc, null, null);
```

- **Domain-level replication statistics** – retrieves the replication statistics for all packages in a domain:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md = new MonitoredDomain("default");

//Retrieve domain-level replication statistics (for all packages).
mc.addMonitoredDomain(md);
supMonitor.getReplicationStatistics(mc, null, null);
```

- **Package-level replication statistics** – retrieves the replication statistics for all MBOs in a package:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md = new MonitoredDomain("default");
MonitoredPackage mp = new MonitoredPackage("test_mbs:1.0");

//Retrieve package-level replication statistics (for all MBOs) .
md.addMonitoredPackage(mp);
supMonitor.getReplicationStatistics(mc, null, null);
```

- **MBO replication statistics** – retrieves the replication statistics for a specific mobile business object:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md = new MonitoredDomain("default");
```

```

MonitoredPackage mp = new MonitoredPackage("test_mbs:1.0");
// Monitored cache does not contribute to replication statistics,
however
// to retain the validity of the monitored object path, it should
be part of the path.
MonitoredCacheGroup mcg = new MonitoredCacheGroup("Default");
MonitoredMBO mmbo = new MonitoredMBO("Customer");

//Retrieve replication statistics for a specific MBO.
mcg.addMonitoredMBO(mmbo);
supMonitor.getReplicationStatistics(mc, null, null);

```

### **Retrieval of Data Change Notification History**

Retrieves data change notification history for a monitored cluster.

#### **Syntax**

```

Collection<DataChangeNotificationHistoryVO>
getDataChangeNotificationHistory(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime, Long offset, Integer
length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;

```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```

MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getDataChangeNotificationHistory(mo
s,
 null, null, null, null, null));

```

### **Retrieval of Data Change Notification Performance**

Retrieves data change notification performance for monitored objects in a cluster.

#### **Syntax**

```

DataChangeNotificationPerformanceVO
getDataChangeNotificationPerformance(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime) throws
SUPAdminException;

```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval**

```
MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
DataChangeNotificationPerformanceVO npvo = supMonitor
 .getDataChangeNotificationPerformance(mos, null, null);
System.out.println(npvo.getMinProcessingTime());
```

### **Retrieval of Device Notification History**

Retrieves device notification history for the monitored objects in a cluster.

### **Syntax**

```
Collection<DeviceNotificationHistoryVO>
getDeviceNotificationHistory(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime, Long offset, Integer
length, SortedField<? extends Enum> sortedField) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval** – retrieves device notification history for the "default" domain in a cluster:

```
MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
System.out.println(supMonitor.getDeviceNotificationHistory(mos,
null, null, null, null, null));
```

### **Retrieval of Device Notification Performance**

Retrieves device notification performance for the monitored objects in a cluster.

### **Syntax**

```
DeviceNotificationPerformanceVO
getDeviceNotificationPerformance(Collection<MonitoredObject>
```

```
monitoredObjects, Date startTime, Date endTime) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval** – retrieves device notification performance for the monitored "default" domain in a cluster:

```
MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
DeviceNotificationPerformanceVO dnpvo = supMonitor
 .getDeviceNotificationPerformance(mos, null, null);
System.out.println(dnpvo.getDistinctDevices());
```

### **Retrieval of Cache Group Performance**

Retrieves cache group performance data of the monitored objects within a specified time range.

### **Syntax**

```
Collection<CacheGroupPerformanceVO>
getCacheGroupPerformance(Collection<MonitoredObject>
monitoredObjects, Date startTime, Date endTime) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval** – retrieves cache group performance data for the specified domains and packages:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredDomain md_tst = new MonitoredDomain("test");
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:1.0"));
md_def.addMonitoredPackage(new MonitoredPackage("test_mbs:2.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:3.0"));
md_tst.addMonitoredPackage(new MonitoredPackage("test_mbs:4.0"));
mc.addMonitoredDomain(md_def);
mc.addMonitoredDomain(md_tst);
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
for (CacheGroupPerformanceVO cpvo : supMonitor
```

```
.getCacheGroupPerformance(mos, null, null)) {
 System.out.println(cpvo.getMaxCacheHits());
}
```

### **Retrieval of Cache Group Statistics**

Retrieves cache group statistics for a package or for an MBO within the specified time range.

#### **Syntax**

```
Collection<CacheGroupPackageStatisticsVO>
getCacheGroupPackageStatistics(MonitoredObject monitoredObject, Date
startTime, Date endTime) throws SUPAdminException;
```

```
Collection<CacheGroupMBOStatisticsVO>
getCacheGroupMBOStatistics(MonitoredObject monitoredObject, Date
startTime, Date endTime) throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Package** – retrieves cache group statistics for the specified package in a domain:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
MonitoredPackage mp = new MonitoredPackage("jdbc:1.0");
md_def.addMonitoredPackage(mp);
mc.addMonitoredDomain(md_def);
for (CacheGroupPackageStatisticsVO cgpsvo : supMonitor
 .getCacheGroupPackageStatistics(mc, null, null)) {
 System.out.println(cgpsvo.getRowCount());
}

mp.addMonitoredCacheGroup(new MonitoredCacheGroup("default"));
for (CacheGroupPackageStatisticsVO cgpsvo : supMonitor
 .getCacheGroupPackageStatistics(mc, null, null)) {
 System.out.println(cgpsvo.getRowCount());
}
```

- **MBO** – retrieves cache group statistics for the specified package, cache group, and MBO:

```
MonitoredCluster mc = new MonitoredCluster();
MonitoredDomain md_def = new MonitoredDomain("default");
mc.addMonitoredDomain(md_def);
for (CacheGroupMBOStatisticsVO cgmsvo : supMonitor
 .getCacheGroupMBOStatistics(mc, null, null)) {
 System.out.println(cgmsvo.getAccessCount());
}

MonitoredPackage mp = new MonitoredPackage("jdbc:1.0");
md_def.addMonitoredPackage(mp);
for (CacheGroupMBOStatisticsVO cgmsvo : supMonitor
```

```

 .getCacheGroupMBOStatistics(mc, null, null)) {
 System.out.println(cgmsvo.getAccessCount());
 }

 MonitoredCacheGroup mcg = new MonitoredCacheGroup("default");
 mp.addMonitoredCacheGroup(mcg);
 for (CacheGroupMBOStatisticsVO cgmsvo : supMonitor
 .getCacheGroupMBOStatistics(mc, null, null)) {
 System.out.println(cgmsvo.getAccessCount());
 }

 MonitoredMBO mmbo = new MonitoredMBO("Customer");
 mcg.addMonitoredMBO(mmbo);
 for (CacheGroupMBOStatisticsVO cgmsvo : supMonitor
 .getCacheGroupMBOStatistics(mc, null, null)) {
 System.out.println(cgmsvo.getAccessCount());
 }
 }

```

### **Retrieval of Queue Monitoring Data and Statistics**

Retrieves a list of the monitoring statistics of Java Message Service (JMS) queues of the SAP Mobile Server within the specified time range.

#### **Syntax**

```

Collection<MessagingQueueStatisticsVO>
getMessagingQueueStatistics(Date startTime, Date endTime) throws
SUPAdminException;

```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```

for (MessagingQueueStatisticsVO mqsvo : supMonitor
 .getMessagingQueueStatistics(null, null)) {
 System.out.println(mqsvo.getQueueName());
}

```

#### **Monitoring Data Export**

Export access history of the monitored objects during the specified time range.

Exporting monitoring data is similar to retrieving monitoring data, with these differences:

- Exporting monitoring data requires an instance of `java.io.File`.
- You specify length to set the number of rows of records to be exported to a specified file. There is no server-side limitation on length.

### **Syntax**

```
void exportSecurityLogHistory(File file, Collection<MonitoredObject>
monitoredObjects, Boolean accessResult, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportMessagingQueueStatistics(File file, Date startTime, Date
endTime) throws SUPAdminException;

void exportMessagingRequests(File file, Collection<MonitoredObject>
monitoredObjects) throws SUPAdminException;

void exportMessagingHistorySummary(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportMessagingHistoryDetail(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportMessagingPerformance(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime) throws SUPAdminException;

void exportMessagingStatistics(File file, String user, Date
startTime, Date endTime) throws SUPAdminException;

void exportReplicationRequests(File file,
Collection<MonitoredObject> monitoredObjects) throws
SUPAdminException;

void exportReplicationHistorySummary(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportReplicationHistoryDetail(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportReplicationPerformance(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime) throws SUPAdminException;

void exportReplicationStatistics(File file, MonitoredObject
monitoredObject, Date startTime, Date endTime) throws
SUPAdminException;

void exportMessagingOperationStatistics(file, mc, null, null) throws
SUPAdminException;

void exportReplicationOperationStatistics(file, mc, null, null)
```



```

throws SUPAdminException;

void exportDataChangeNotificationHistory(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportDataChangeNotificationPerformance(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime) throws SUPAdminException;

void exportDeviceNotificationHistory(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime, Long offset, Integer length, SortedField<? extends Enum>
sortedField) throws SUPAdminException;

void exportDeviceNotificationPerformance(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime) throws SUPAdminException;

void exportCacheGroupPerformance(File file,
Collection<MonitoredObject> monitoredObjects, Date startTime, Date
endTime) throws SUPAdminException;

void exportCacheGroupPackageStatistics(File file, MonitoredObject
monitoredObject, Date startTime, Date endTime) throws
SUPAdminException;

void exportCacheGroupMBOSTatistics(File file, MonitoredObject
monitoredObject, Date startTime, Date endTime) throws
SUPAdminException;

```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## **Examples**

- **Export Security Log History** – exports records for a monitored domain to access.log:

```

File file = new File("D:\\tmp\\access.log");
MonitoredCluster mc = new MonitoredCluster();
mc.addMonitoredDomain(new MonitoredDomain("default"));
Collection<MonitoredObject> mos = Arrays
 .asList(new MonitoredObject[] { mc });
// when the method returns, the access.log contains the exported
records.
supMonitor.exportSecurityLogHistory(file, mos, null, null, null,
 null, null, null);

```

## **Managing SAP Mobile Server Logs**

You can enable logging, filter logs, and manage log appenders and buckets through the `SUPServerLog` interface. Operations you can perform with this interface include:

- Starting administration of logging.
- Constructing filters for a log.
- Filtering and retrieving log entries.
- Deleting a log.
- Managing log appenders and buckets.

### **Start Log Management**

Starts the management of logging for SAP Mobile Server.

### **Syntax**

```
public static SUPServerLog getSUPServerLog(ServerContext
serverContext);
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Start log management**

```
SUPServerLog supServerLog =
SUPObjectFactory.getSUPServerLog(serverContext);
```

### **Usage**

When an instance of `SUPServerLog` is returned from the `SUPObjectFactory`, call its method.

### **Log Filter Construction**

You can define and compose filters to form a log fetching pattern. All the filters are subclasses of `FieldFilter`. There are two types of filters: those that act directly on log fields, and those that connect other filters.

These are the supported filters in `FieldFilter` for server logging:

- Direct Field Filters
  - `FieldEqualityFilter`
  - `FieldRangeFilter`
  - `FieldRegexpFilter`
  - `FieldSetFilter`

- FieldWildcardFilter
- Connecting Filters
  - LogicalAndFilter
  - LogicalNotFilter
  - LogicalOrFilter

You cannot directly instantiate filters through a new operator. You must acquire them by calling methods of SUPServerLog.

```
FieldEqualityFilter bucket_eq = supServerLog.getFieldEqualityFilter(
 SERVER_LOG_FIELD.BUCKET, "MMS");

FieldSetFilter thread_set = supServerLog.getFieldSetFilter(
 SERVER_LOG_FIELD.THREAD_NAME, Arrays.asList(new String[] {
 "main", "dispatcher" }));

FieldWildcardFilter logger_wild =
supServerLog.getFieldWildcardFilter(
 SERVER_LOG_FIELD.LOGGER_NAME, "com.sybase.sup*");

FieldRangeFilter time_range = supServerLog.getFieldRangeFilter(
 SERVER_LOG_FIELD.TIMESTAMP, new Date(0), new Date());
FieldRegexpFilter regexp = supServerLog.getFieldRegexpFilter(
 SERVER_LOG_FIELD.THREAD_NAME, "^RMI");

LogicalNotFilter notFilter = supServerLog
 .getLogicalNotFilter(bucket_eq);
LogicalOrFilter orFilter = supServerLog.getLogicalOrFilter(Arrays
 .asList(new FieldFilter[] { time_range, regexp }));
LogicalAndFilter andFilter = supServerLog.getLogicalAndFilter(Arrays
 .asList(new FieldFilter[] { thread_set, logger_wild }));

FieldFilter filter = supServerLog.getLogicalAndFilter(Arrays
 .asList(new FieldFilter[] { notFilter, orFilter,
andFilter }));

supServerLog.setLogFilter(filter);
```

### **Log Entry Retrieval**

Filters and retrieves entries from an SAP Mobile Server log.

#### **Syntax**

```
void setLogPosition(LogPositionVO logPosition) throws
SUPAdminException;
```

```
Collection<LogEntryVO> getLogEntries(Integer start, Integer end)
throws SUPAdminException;
```

```
Collection<LogEntryVO> getLogEntries(Integer start, Integer end,
Boolean includingBackup) throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Filter from the start of a log** – returns log entries from the start of the log (the 100th through 250th entries after the start of the log):

```
supServerLog.setLogPosition(LogPositionVO.START);
for (LogEntryVO levo : supServerLog.getLogEntries(100, 250)) {
 System.out.println(levo.getBucket());
}
for (LogEntryVO levo : supServerLog.getLogEntries(100, 250, true))
{
 System.out.println(levo.getBucket());
}
```

- **Filter from the end of a log** – returns log entries from the end of the log (the 100th to 250th entries before the end of the log):

```
supServerLog.setLogPosition(LogPositionVO.END);
for (LogEntryVO levo : supServerLog.getLogEntries(-100, -250)) {
 System.out.println(levo.getBucket());
}
for (LogEntryVO levo : supServerLog.getLogEntries(-100, -250,
true)) {
 System.out.println(levo.getBucket());
}
```

### Log Deletion

Truncates a server log.

### Syntax

```
void deleteLog() throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Deletion**

```
supServerLog.deleteLog();
```

## **Managing Log Appenders and Buckets**

SAP Mobile Platform server logs are managed through metadata-based configuration and consist of one or more log appenders. Each log appender has one or more log buckets. They are represented by `LogAppenderVO` and `LogBucketVO` respectively.

These rules apply when managing server log appenders and buckets:

- Each instance of `SUPServerLog` is a local object that holds values for all metadata based configuration. All of its methods perform against those values. The values are refreshed when `commit()` and `refresh()` are called.
- After getting an instance of `SUPServerLog`, call `refresh()` to populate the values, before calling any other methods.
- Changes made through these methods are cached locally unless you call the `commit()` method. `Commit()` sends all cached values (changed or not) to SAP Mobile Server.

### **Populate Server Log Configuration**

Populates the server log configuration values to SAP Mobile Server.

### **Syntax**

```
ConfigurationValidationStatus refresh() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Populate server log configuration**

```
supServerLog.refresh();
```

### **LogAppenderVO and LogBucketVO**

The `LogAppenderVO` and `LogBucketVO` classes have two read-only properties that you must initialize at construction time.

- **ID** – a unique ID within the locally cached log configuration.
- **Type** – specifies the type of appender or bucket. The types of appenders and buckets are described in *Developer Guide: SAP Mobile Server Runtime > Management API > Client Metadata > Server Log Configuration*.

### **Retrieval of a List of Active Log Appenders**

Retrieves a list of active log appenders.

### **Syntax**

```
Collection<LogAppenderVO> getActiveLogAppenders() throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Deletion**

```
supServerLog.refresh();
for(LogAppenderVO lavo: supServerLog.getActiveLogAppenders()){
 System.out.println(lavo.getType());
 System.out.println(lavo.getProperties());
}
```

### **Update of an Active Log Appender**

Updates an active log appender.

### **Syntax**

```
void updateActiveLogAppender(String logAppenderID, LogAppenderVO
logAppender) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Update**

```
supServerLog.refresh();
LogAppenderVO lavo =
supServerLog.getActiveLogAppenders().iterator().next();
LogAppenderVO lavo_new = new LogAppenderVO(lavo.getID(),
lavo.getType());
Map<String, String> properties = new HashMap<String, String>();
properties.put("async", "true");
lavo_new.setProperties(properties);
supServerLog.updateActiveLogAppender(lavo_new.getID(), lavo_new);
supServerLog.commit();
```

### **Retrieval of a List of Active Log Buckets**

Retrieves a list of active log appenders.

**Syntax**

```
Collection<LogAppenderVO> getActiveLogAppenders() throws
SUPAdminException;
```

**Returns**

If successful, returns a list of objects of the specified type (can be null). If unsuccessful, returns SUPAdminException

**Examples**

- **Retrieve Active Log Buckets**

```
supServerLog.refresh();
LogAppenderVO lavo =
supServerLog.getActiveLogAppenders().iterator().next();
for(LogBucketVO lbvo : lavo.getChildren()){
 System.out.println(lbvo.getType());
 System.out.println(lbvo.getProperties());
}
```

**Update of an Active Log Bucket**

Updates an active log bucket of an active log appender with the specified properties.

**Syntax**

```
void updateActiveLogBucket(String logAppenderID, String logBucketID,
LogBucketVO logBucket) throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

**Examples**

- **Update**

```
supServerLog.refresh();
LogAppenderVO lavo =
supServerLog.getActiveLogAppenders().iterator().next();
LogBucketVO lbvo = lavo.getChildren().iterator().next();
LogBucketVO lbvo_new = new LogBucketVO(lbvo.getID(),
lbvo.getType());
Map<String, String> properties = new HashMap<String, String>();
properties.put("LogLevel", "INFO");
lbvo_new.setProperties(properties);
supServerLog.updateActiveLogBucket (lavo.getID(),
lbvo_new.getID(), lbvo_new);
supServerLog.commit();
```

### **Retrieval and Export of Trace Entries**

Retrieves and exports trace entries to allow you to identify and resolve server-side issues while debugging a device application.

#### **Syntax**

```
PaginationResult<TraceEntryVO> getTraceEntries() throws
SUPAdminException;

void exportTraceEntries(exportFile, filter, sort) through
SUPAdminException
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieve and export trace entries**

```
TraceFilterVO filter = new TraceFilterVO();
Calendar c = Calendar.getInstance();
c.set(2011, 10, 1);

filter.setStartTime(c.getTime());
filter.setEndTime(new Date());
filter.setLevel(TRACE_LOG_LEVEL.DEBUG);
Collection<TRACE_LOG_MODULE> modules =
Arrays.asList(TRACE_LOG_MODULE.MO);
filter.setModules(modules);
TraceSortVO sort = null;
PaginationResult<TraceEntryVO> entries =
supServerLog.getTraceEntries(
filter, 0L, 100, sort);

System.out.println(entries.getTotalAvailableRecords());

File exportFile = new File("D:\\temp\\jmsBridge.zip");
supServerLog.exportTraceEntries(exportFile, filter, sort);
```

## **Managing Domain Logs**

You can define log filtering and fetching behavior and change log settings for a domain through the SUPDomainLog interface.

### **Start Managing Domain Logs**

Starts the management of logging for a domain.



**Syntax**

```
public static SUPDomainLog getSUPDomainLog(DomainContext
domainContext);
```

**Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Security configuration**

```
SUPDomainLog domainLog =
SUPObjectFactory.getSUPDomainLog(domainContext);
```

**Usage**

When an instance of `SUPDomainLog` is returned from the `SUPObjectFactory`, call its method.

**Retrieval of a List of Domain Log Profiles**

Retrieves a list of domain log profiles.

**Syntax**

```
Collection<DomainLogProfileVO> getDomainLogProfiles() throws
SUPAdminException;
```

**Returns**

If successful, returns a list of objects of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

**Examples**

- **Retrieval**

```
for (DomainLogProfileVO dlpvo : domainLog.getDomainLogProfiles())
{
 System.out.println(dlpvo.getName());
}
```

**Creation of a Domain Log Profile**

Creates a domain log profile.

**Syntax**

```
trap5.setValues(Arrays.asList(new DOMAIN_LOG_CATEGORY[]
{ DOMAIN_LOG_CATEGORY.DATA_SYNC,
DOMAIN_LOG_CATEGORY.GENERAL_DCN }));
```

**Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

**Examples**

```

• String profileName = "profile1";
Collection<DomainLogTrapVO<? extends Enum>> traps = new
ArrayList<DomainLogTrapVO<? extends Enum>>();
DomainLogTrapVO<DOMAIN_LOG_PROFILE_PACKAGE_TRAP> trap1 = new
DomainLogTrapVO<DOMAIN_LOG_PROFILE_PACKAGE_TRAP>(
DOMAIN_LOG_PROFILE_PACKAGE_TRAP.APPLICATION_ID);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_SECURITY_TRAP> trap2 = new
DomainLogTrapVO<DOMAIN_LOG_PROFILE_SECURITY_TRAP>(
DOMAIN_LOG_PROFILE_SECURITY_TRAP.SECURITY_CONF);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_ENDPOINT_TRAP> trap3 = new
DomainLogTrapVO<DOMAIN_LOG_PROFILE_ENDPOINT_TRAP>(
DOMAIN_LOG_PROFILE_ENDPOINT_TRAP.ENDPOINT);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_APPCONNECTION_TRAP> trap4 =
new DomainLogTrapVO<DOMAIN_LOG_PROFILE_APPCONNECTION_TRAP>(
DOMAIN_LOG_PROFILE_APPCONNECTION_TRAP.APPLICATION_CONNECTION_ID);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_PAYLOAD_TRAP> trap5 = new
DomainLogTrapVO<DOMAIN_LOG_PROFILE_PAYLOAD_TRAP>(
DOMAIN_LOG_PROFILE_PAYLOAD_TRAP.PAY_LOAD);

trap1.setEnabled(true);
trap1.setValues(Arrays.asList(new String[] { "app1:1.0",
"app2:2.0" }));

trap2.setEnabled(true);
trap2.setValues(Arrays.asList(new String[] { "admin", "test" }));

trap3.setEnabled(true);
EndpointTrapVO etvo1 = new EndpointTrapVO();
etvo1.setName("sampledb");
etvo1.setType(ENDPOINT_TYPE.JDBC);
EndpointTrapVO etvo2 = new EndpointTrapVO();
etvo2.setName("sap_crm:1.0");
etvo2.setType(ENDPOINT_TYPE.DOEC);
trap3.setValues(Arrays.asList(new EndpointTrapVO[] { etvo1,
etvo2 }));

trap4.setEnabled(true);
trap4.setValues(Arrays.asList(new String[] { "emulator1",
"bb2" }));

trap5.setEnabled(true);
trap5.setValues(Arrays
 .asList(new DOMAIN_LOG_CATEGORY[] {
 DOMAIN_LOG_CATEGORY.DATA_SYNC,
 DOMAIN_LOG_CATEGORY.GENERAL_DCN }));

```

```
traps.add(trap1);
traps.add(trap2);
traps.add(trap3);
traps.add(trap4);
traps.add(trap5);

domainLog.createDomainLogProfile(profileName, description, traps,
 false);
```

### **Update of a Domain Log Profile**

Updates a domain log profile.

#### **Syntax**

```
void updateDomainLogProfile(String profileName, String description,
 Collection<DomainLogTrapVO> traps) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- ```
String profileName = "profile1";
String description = "domain log profile description updated";

Collection<DomainLogTrapVO<? extends Enum>> traps = new
    ArrayList<DomainLogTrapVO<? extends Enum>>();
DomainLogTrapVO<DOMAIN_LOG_PROFILE_PACKAGE_TRAP> trap1 = new
    DomainLogTrapVO<DOMAIN_LOG_PROFILE_PACKAGE_TRAP>(
    DOMAIN_LOG_PROFILE_PACKAGE_TRAP.APPLICATION_ID);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_SECURITY_TRAP> trap2 = new
    DomainLogTrapVO<DOMAIN_LOG_PROFILE_SECURITY_TRAP>(
    DOMAIN_LOG_PROFILE_SECURITY_TRAP.SECURITY_CONF);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_ENDPOINT_TRAP> trap3 = new
    DomainLogTrapVO<DOMAIN_LOG_PROFILE_ENDPOINT_TRAP>(
    DOMAIN_LOG_PROFILE_ENDPOINT_TRAP.ENDPOINT);

DomainLogTrapVO<DOMAIN_LOG_PROFILE_APPCONNECTION_TRAP> trap4 =
    new DomainLogTrapVO<DOMAIN_LOG_PROFILE_APPCONNECTION_TRAP>(
    DOMAIN_LOG_PROFILE_APPCONNECTION_TRAP.APPLICATION_CONNECTION_ID);

trap1.setEnabled(true);
trap2.setEnabled(true);
trap3.setEnabled(true);
trap4.setEnabled(true);

traps.add(trap1);
traps.add(trap2);
traps.add(trap3);
```

```
traps.add(trap4);

domainLog.updateDomainLogProfile(profileName, description,
traps);
```

Deletion of a Domain Log Profile

Deletes a domain log profile.

Syntax

```
void deleteDomainLogProfiles(Collection<String> profileNames) throws
SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

Examples

- ```
domainLog.deleteDomainLogProfiles(Arrays.asList(new String[]
{ profileName }));
```

### **Retrieval of a List of Domain Log Filters**

Retrieves a list of domain log filters.

### **Syntax**

```
Collection<DomainLogFilterVO> getDomainLogFilters() throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval**

```
for (DomainLogFilterVO dlfvo : domainLog.getDomainLogFilters()) {
 System.out.println (dlfvo.getName());
}
```

### **Creation or Update of a Correlation Log Filter**

Persists the domain log filters for later usage.

**Syntax**

```
void saveDomainLogFilters(Collection<DomainLogFilterVO> filters)
throws SUPAdminException;
```

**Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

**Examples**

- ```
DomainLogFilterVO dlfvo = new
DomainLogFilterVO(DOMAIN_LOG_CATEGORY.ALL);
FilterExpression<DOMAIN_LOG_FILTER> fe = new FilterExpression<
DOMAIN_LOG_FILTER >();
FilterExpression< DOMAIN_LOG_FILTER > fel = new FilterExpression<
DOMAIN_LOG_FILTER >();
fel = fe.eq(DOMAIN_LOG_FILTER.APPLICATION_CONNECTION_ID,
"emulator1").and(
fe.eq(DOMAIN_LOG_FILTER.DOMAIN,
"default")).or(fe.eq(DOMAIN_LOG_FILTER.PACKAGE, "sap_crm:1.0"));
dlfvo.setFilterExpression(fel);
domainLog.saveDomainLogFilters(Arrays.asList(new
DomainLogFilterVO[]{dlfvo}));
```

Deletion of a Domain Log Filter

Deletes a domain log filter.

Syntax

```
void deleteDomainLogFilters(Collection<String> filterNames) throws
SUPAdminException;
```

Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

Examples

- ```
domainLog.deleteDomainLogFilters(Arrays
.asList(new String[] { "filter1" }));
```

**Retrieval of a List of Log Entries**

Retrieves the domain log entries with the given filters, time range, offset and length.

**Syntax**

```
List<DomainLogEntryVO>
getDomainLogEntry(Collection<DomainLogFilterVO> filters, Date
```

## Management API

```
StartTime, Date endTime, Long offset, Integer length) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval**

```
DomainLogFilterVO dlfvo =
domainLog.getDomainLogFilter("filter1");
List<DomainLogEntryVO> logEntries = domainLog.getDomainLogEntry(
 Arrays.asList(new DomainLogFilterVO[] { dlfvo }), null,
 null, null, null);
for (DomainLogEntryVO dlevo : logEntries) {
 for (Map.Entry<String, Object> entry :
 dlevo.getEntry().entrySet()) {
 System.out.println(entry.getKey() + ":" +
 entry.getValue());
 }
}
```

### **Deletion of Domain Log Entries**

Deletes the domain log entries within the specified time range.

### **Syntax**

```
void deleteLog(Date startTime, Date endTime) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Deletion**

```
domainLog.deleteLog(new Date(0), new Date());
```

### **Retrieval of Log Store Policy**

Retrieves the properties of the domain log store policy.

### **Syntax**

```
DomainLogStorePolicyVO getDomainLogStorePolicy() throws
SUPAdminException;
```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Retrieval**

```
DomainLogStorePolicyVO dlspvo = supCluster
 .getDomainLogStorePolicy();
System.out.println(dlspvo.getCurrentDomainLogDataSource());
System.out.println(dlspvo.getAvailableDomainLogDataSource());
System.out.println(dlspvo.getDomainLogFlushBatchSize());
System.out.println(dlspvo.getLazyWriteEnabled());
System.out.println(dlspvo.getLazyWriteRowThreshold());
System.out.println(dlspvo.getLazyWriteTimeThreshold());
System.out.println(dlspvo.getPurgeTimeThreshold());
```

## **Usage**

These methods are only accessible to the Platform Administrator.

## **Update of Log Store Policy**

Updates the properties of the domain log store policy.

## **Syntax**

```
void setDomainLogAutoPurgeTimeThreshold(Integer days) throws
SUPAdminException;
```

```
void setDomainLogDataSource(String datasource) throws
SUPAdminException;
```

```
void setDomainLogFlushBatchSize(Integer rows) throws
SUPAdminException;
```

```
void setDomainLogLazyWriteRowThreshold(Integer rowcount) throws
SUPAdminException;
```

```
void setDomainLogLazyWriteStatus(Boolean flag) throws
SUPAdminException;
```

```
void setDomainLogLazyWriteTimeThreshold(Integer minutes) throws
SUPAdminException;
```

## **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update**

```
supCluster.setDomainLogAutoPurgeTimeThreshold(7);
supCluster.setDomainLogDataSource("newDomainLogDB");
supCluster.setDomainLogFlushBatchSize(100);
supCluster.setDomainLogLazyWriteRowThreshold(200);
supCluster.setDomainLogLazyWriteStatus(true);
supCluster.setDomainLogLazyWriteTimeThreshold(100);
```

### Usage

These methods are only accessible to the Platform Administrator.

### Export of Log Entries

Exports the domain log entries to a file.

### Syntax

```
File exportDomainLogEntry(file, Date StartTime, Date EndTime,
Integer length) throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Export**

```
File file = new File("D:\\domainlog.txt");
domainLog.exportDomainLogEntry(file, Date StartTime, Date
EndTime, Integer length);
```

## Configuring SAP Mobile Platforms

Administration of the SAP Mobile Server configuration is provided through the `SUPServerConfiguration` interface.

The SAP Mobile Server configuration consists of Java Virtual Machine (JVM) startup options and Outbound Enabler Proxy management, which are metadata-based configuration. All other components have been deprecated and are now performed on the cluster.

The metadata-based configurations have these characteristics:

- Each of these components is represented by `ServerComponentVO`.



- The properties of `ServerComponentVO` differentiate these components. See *Developer Guide: SAP Mobile Server Runtime > SAP Mobile Server Management API > Client Metadata*.
- Each instance of `SUPServerConfiguration` is a local object which holds values of all metadata-based configurations. All of its methods perform against those values. The values are refreshed when you call the `commit()` and `refresh()` methods. After you receive an instance of `SUPServerConfiguration`, call the `refresh()` method to populate the values, before calling any other methods.
- Changes made through these methods are cached locally unless the `commit()` method is called. `Commit()` sends all the cached values (whether changed or not) to the SAP Mobile Server. A server restart may be required for some changes to take effect.

### **ServerComponentVO**

The `ServerComponentVO` class has a read-only property that you must initialize at construction time.

The type property specifies the server component type. The server component types are described in *Developer Guide: SAP Mobile Server Runtime > Management API > Client Metadata > Server Configuration*.

### **Start Management of SAP Mobile Server Configuration**

Starts the management of SAP Mobile Server configuration information.

### **Syntax**

```
public static SUPServerConfiguration
getSUPServerConfiguration(ServerContext serverContext) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **SAP Mobile Server configuration**

```
SUPServerConfiguration supServerConf = SUObjectFactory
 .getSUPServerConfiguration(serverContext);
```

### **Usage**

When an instance of `SUPServerConfiguration` is returned from the `SUObjectFactory`, call its method.

### **Populate Server Configuration**

Retrieves the server configuration from the SAP Mobile Server and caches it locally. This method refreshes all metadata-based configuration. The returned

`ConfigurationValidationStatus` contains the validation status of the security configuration on the server.

### **Syntax**

```
ConfigurationValidationStatus refresh() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Populate server configuration**

```
supServerConf.refresh();
```

### **Usage**

When you call `SUPServerConfiguration.refresh()`, any data in the local cache is overwritten.

Each call to `commit()` and `refresh()` expire all previous `ServerComponentVOs`, because all the IDs are regenerated.

### **Commit Local Changes to SAP Mobile Server**

Commits local changes to SAP Mobile Server. The returned `ConfigurationValidationStatus` contains the validation status of the delivered security configuration on the SAP Mobile Server.

### **Syntax**

```
ConfigurationValidationStatus commit() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Update**

```
ConfigurationValidationStatus cvs = supServerConf.commit();
if(cvs.isValid()){
 //succeed.
}
else{
```

```
//fail.
}
```

### **Retrieval of Replication Sync Server Configuration**

This method has been deprecated. Retrieves the properties of the replication synchronization server configuration.

#### **Syntax**

```
ServerComponentVO getReplicationSyncServerConfiguration() throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getReplicationSyncServerConfiguration();
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
```

### **Update of Replication Sync Server Configuration**

This method has been deprecated. Updates the properties of the replication synchronization server configuration.

#### **Syntax**

```
void updateReplicationSyncServerConfiguration(ServerComponentVO
serverComponent) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Update**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getReplicationSyncServerConfiguration();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
```

```
Map<String, String> properties = scvo.getProperties();
properties.put("sup.sync.protocol", "http");
properties.put("ml.threadcount", "50");
scvo_new.setProperties(properties);
supServerConf.updateReplicationSyncServerConfiguration(scvo_new);
supServerConf.commit();
```

### **Retrieval of Messaging Sync Server Configuration**

This method has been deprecated. Retrieves the properties of the messaging synchronization configuration from the SAP Mobile Server.

### **Syntax**

```
ServerComponentVO getMessagingSyncServerConfiguration() throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieval**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getMessagingSyncServerConfiguration();
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
```

### **Update of Messaging Sync Server Configuration**

This method has been deprecated. Updates the properties of the messaging synchronization configuration on the SAP Mobile Server.

### **Syntax**

```
void updateMessagingSyncServerConfiguration(ServerComponentVO
serverComponent) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Update** – updates the messaging synchronization configuration on the SAP Mobile Server by specifying the ID, Type, and Properties:

```

supServerConf.refresh();
ServerComponentVO scvo = supServerConf
 .getMessagingSyncServerConfiguration();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
 scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("msg.admin.webservices.port", "5100");
properties.put("msg.http.server.ports", "5001,80");
scvo_new.setProperties(properties);
supServerConf.updateMessagingSyncServerConfiguration(scvo_new);
supServerConf.commit();

```

### **Retrieval of Consolidated Database Configuration**

This method has been deprecated. Retrieves the properties of the consolidated database configuration.

#### **Syntax**

```

ServerComponentVO getConsolidatedDatabaseConfiguration() throws
SUPAdminException;

```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```

supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getConsolidatedDatabaseConfiguration();
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());

```

### **Retrieval of Administration Listener Configuration**

This method has been deprecated. Retrieves the configuration of the administration listener.

#### **Syntax**

```

ServerComponentVO getAdministrationListenerConfiguration() throws
SUPAdminException;

```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### Examples

- **Retrieval**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getAdministrationListenerConfiguration();
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
```

### Update of Administration Listener Configuration

This method has been deprecated. Updates the properties of the administration listener configuration.

### Syntax

```
void updateAdministrationListenerConfiguration(String
serverComponentID, ServerComponentVO serverComponent) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Update**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getAdministrationListenerConfiguration();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.socket.listener.port", "2000");
properties.put("sup.socket.listener.enabled", "true");
scvo_new.setProperties(properties);
supServerConf.updateAdministrationListenerConfiguration(scvo_new.
getID(), scvo_new);
supServerConf.commit();
```

### Retrieval of HTTP Listener Configuration

This method has been deprecated. Retrieves a list of HTTP listener configurations.

### Syntax

```
Collection<ServerComponentVO> getHTTPListenerConfigurations() throws
SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval**

```
supServerConf.refresh();
for(ServerComponentVO scvo :
supServerConf.getHTTPListenerConfigurations()){
 System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
}
```

## Addition of HTTP Listener Configuration

This method has been deprecated. Adds a new HTTP listener configuration.

## Syntax

```
void addHTTPListenerConfiguration(ServerComponentVO serverComponent)
throws SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Add configuration**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getHTTPListenerConfigurations()
 .iterator().next();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.socket.listener.port", "8000");
properties.put("sup.socket.listener.enabled", "true");
scvo_new.setProperties(properties);
supServerConf.addHTTPListenerConfiguration(scvo_new);
supServerConf.commit();
```

## Deletion of HTTP Listener Configuration

This method has been deprecated. Deletes the configuration for an HTTP listener.

### **Syntax**

```
void deleteHTTPListenerConfiguration(String serverComponentID)
throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Deletion**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getHTTPListenerConfigurations()
 .iterator().next();
supServerConf.deleteHTTPListenerConfiguration(scvo.getID());
supServerConf.commit();
```

### **Update of HTTP Listener Configuration**

This method has been deprecated. Updates the configuration of an HTTP listener.

### **Syntax**

```
void updateHTTPListenerConfiguration(String serverComponentID,
ServerComponentVO serverComponent) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Update**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getHTTPListenerConfigurations()
 .iterator().next();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.socket.listener.port", "8000");
properties.put("sup.socket.listener.enabled", "true");
scvo_new.setProperties(properties);
supServerConf.updateHTTPListenerConfiguration(scvo_new.getID(),
scvo_new);
supServerConf.commit();
```



### **Retrieval of HTTPS Listener Configuration**

This method has been deprecated. Retrieves a list of HTTPS listener configurations.

#### **Syntax**

```
Collection<ServerComponentVO> getSecureHTTPListenerConfigurations()
throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```
supServerConf.refresh();
for (ServerComponentVO scvo :
supServerConf.getSecureHTTPListenerConfigurations()) {
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
}
```

### **Addition of HTTPS Listener Configuration**

This method has been deprecated. Adds a new HTTPS listener configuration.

#### **Syntax**

```
void addSecureHTTPListenerConfiguration(ServerComponentVO
serverComponent) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Add configuration**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getSecureHTTPListenerConfigurations()
.iterator().next();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.socket.listener.port", "8001");
properties.put("sup.socket.listener.enabled", "true");
```

```
scvo_new.setProperties(properties);
supServerConf.addSecureHTTPListenerConfiguration(scvo_new);
supServerConf.commit();
```

### **Deletion of HTTPS Listener Configuration**

This method has been deprecated. Deletes the configuration for a secure HTTP (HTTPS) listener.

### **Syntax**

```
void deleteSecureHTTPListenerConfiguration(String serverComponentID)
throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Deletion**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getSecureHTTPListenerConfigurations ()
 .iterator().next();
supServerConf.deleteSecureHTTPListenerConfiguration(scvo.getID())
;
supServerConf.commit();
```

### **Update of HTTPS Listener Configuration**

This method has been deprecated. Updates the configuration of an HTTP listener.

### **Syntax**

```
void updateSecureHTTPListenerConfiguration(String serverComponentID,
ServerComponentVO serverComponent) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Update**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getHTTPListenerConfigurations ()
 .iterator().next();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
```

```
Map<String, String> properties = scvo.getProperties();
properties.put("sup.socket.listener.port", "8000");
properties.put("sup.socket.listener.enabled", "true");
scvo_new.setProperties(properties);
supServerConf.updateHTTPListenerConfiguration(scvo_new.getID(),
scvo_new);
supServerConf.commit();
```

### **Retrieval of SSL Security Profile Configuration**

This method has been deprecated. Retrieves the list of all the SSL security profiles and their properties.

#### **Syntax**

```
Collection<ServerComponentVO> getSSLSecurityProfileConfigurations()
throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Retrieval**

```
supServerConf.refresh();
for(ServerComponentVO scvo :
supServerConf.getSSLSecurityProfileConfigurations()){
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
}
```

### **Addition of SSL Security Profile Configuration**

This method has been deprecated. Adds configuration for an SSL security profile.

#### **Syntax**

```
void addSSLSecurityProfileConfiguration(ServerComponentVO
serverComponent) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Add configuration** – adds configuration for an SSL security profile, including the authentication profile, profile name, and key alias:

```
supServerConf.refresh();
ServerComponentVO scvo = supServerConf
 .getSSLSecurityProfileConfigurations().iterator().next();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
 scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.security.profile.auth", "domestic");
properties.put("sup.security.profile.name",
 "<SSL security profile name>");
properties.put("sup.security.profile.key.alias",
 "<SSL security key alias>");
scvo_new.setProperties(properties);
supServerConf.addSSLSecurityProfileConfiguration(scvo_new);
supServerConf.commit();
```

### **Deletion of SSL Security Profile Configuration**

This method has been deprecated. Deletes the configuration for an SSL security profile.

#### **Syntax**

```
void deleteSSLSecurityProfileConfiguration(String serverComponentID)
throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Deletion**

```
supServerConf.refresh();
ServerComponentVO scvo = supServerConf
 .getSSLSecurityProfileConfigurations().iterator().next();
supServerConf.deleteSSLSecurityProfileConfiguration(scvo.getID())
;
supServerConf.commit();
```

### **Update of SSL Security Profile Configuration**

This method has been deprecated. Updates the configuration of an SSL security profile.

#### **Syntax**

```
void updateSSLSecurityProfileConfiguration(String serverComponentID,
ServerComponentVO serverComponent) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Update** – updates the configuration of an SSL security profile, including the authentication profile, profile name, and key alias:

```
supServerConf.refresh();
ServerComponentVO scvo = supServerConf
 .getSSLSecurityProfileConfigurations().iterator().next();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
 scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.security.profile.auth", "domestic");
properties.put("sup.security.profile.name",
 "<SSL security profile name>");
properties.put("sup.security.profile.key.alias",
 "<SSL security key alias>");
scvo_new.setProperties(properties);
supServerConf.updateSSLSecurityProfileConfiguration(scvo_new.getID(),
 scvo_new);
supServerConf.commit();
```

## Key Store Configuration Retrieval

This method has been deprecated. Retrieves the properties of the key store configuration.

### Syntax

```
ServerComponentVO getKeyStoreConfiguration() throws
SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieval**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getKeyStoreConfiguration();
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
```

## Key Store Configuration Update

This method has been deprecated. Updates the configuration of the key store.

### Syntax

```
void updateKeyStoreConfiguration(ServerComponentVO serverComponent)
throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update** – updates the configuration of the key store, including the key store file path, and key store password:

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getKeyStoreConfiguration();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.sync.sslkeystore", "<key store file path>");
properties.put("sup.sync.sslkeystore_password", "<key store
password>");
scvo_new.setProperties(properties);
supServerConf.updateKeyStoreConfiguration(scvo_new);
supServerConf.commit();
```

### Trust Store Configuration Retrieval

This method has been deprecated. Retrieves the properties of the trust store configuration.

### Syntax

```
ServerComponentVO getTrustStoreConfiguration() throws
SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieval**

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getTrustStoreConfiguration();
System.out.println(scvo.getID());
System.out.println(scvo.getType());
System.out.println(scvo.getProperties());
```

### Trust Store Configuration Update

This method has been deprecated. Updates the configuration of the trust store.

## Syntax

```
void updateTrustStoreConfiguration(ServerComponentVO
serverComponent) throws SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Update** – updates the configuration of the trust store, including the trust store file path and trust store password:

```
supServerConf.refresh();
ServerComponentVO scvo =
supServerConf.getTrustStoreConfiguration();
ServerComponentVO scvo_new = new ServerComponentVO(scvo.getID(),
scvo.getType());
Map<String, String> properties = scvo.getProperties();
properties.put("sup.sync.ssltruststore", "<trust store file
path>");
properties.put("sup.sync.ssltruststore_password", "<trust store
password>");
scvo_new.setProperties(properties);
supServerConf.updateTrustStoreConfiguration(scvo_new);
supServerConf.commit();
```

## Retrieval of Apple Push Notification Configurations

This method has been deprecated. Retrieves Apple Push Notification configurations.

## Syntax

```
List<APNSApplicationSettingsVO>
getApplePushNotificationConfigurations(boolean getPendingConfig)
throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Retrieval: getPendingConfig is true** – retrieves Apple Push Notification application settings that are applied to the Unwired Server the next time the Unwired Server starts:

```
// List Apple push configuration
List<APNSAppSettingsVO> list =
supServerConf.getApplePushNotificationConfigurations(true);
```

- **Retrieval: getPendingConfig is false** – retrieves current Apple Push Notification application settings:

```
// List Apple push configuration
List<APNSAppSettingsVO> list =
supServerConf.getApplePushNotificationConfigurations(false);
```

### **Addition of an Apple Push Notification Configuration**

This method has been deprecated. Adds a configuration for Apple Push Notification.

#### **Syntax**

```
void
addApplePushNotificationConfiguration(APNSApplicationSettingsVO
settings, byte[] p12Certificate, boolean overwrite, boolean restart)
throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful (for example, if a certificate of the same name exists and `overwrite` is false), returns `SUPAdminException`.

#### **Examples**

- **Add configuration**

```
// Add Apple push configuration
APNSAppSettingsVO settings = buildAPNSSettings();
byte[] certificate = getCertificate();
supServerConf.addApplePushNotificationConfiguration(settings,
certificate, false, false);
```

### **Deletion of an Apple Push Notification Configuration**

This method has been deprecated. Deletes an Apple Push Notification configuration.

#### **Syntax**

```
Boolean deleteApplePushNotificationConfiguration(String
apnsConfigName, boolean restart) throws SUPAdminException;
```

#### **Returns**

If successful, returns true if the specified APNS configuration has been removed, or false if the specified APNS configuration does not exist. If unsuccessful, returns `SUPAdminException`.



## Examples

- **Removal**

```
// Delete Apple push configuration
supServerConf.deleteApplePushNotificationConfiguration("smithj_APNS_configuration1", false);
```

## Update of an Apple Push Notification Configuration

This method has been deprecated. Updates an Apple Push Notification configuration.

## Syntax

```
void
updateApplePushNotificationConfiguration(APNSApplicationSettingsVO
settings, byte[] p12Certificate, boolean overwrite, boolean restart)
throws SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Update** – updates an Apple Push Notification configuration including the feedback gateway and the Apple Push Notification settings:

```
// Update Apple push configuration
APNSAppSettingsVO settings = buildAPNSSettings();
settings.setFeedbackGateway("testfeedback.push2.example.com ");
byte[] certificate = getCertificate();
supServerConf.updateApplePushNotificationConfiguration(settings,
certificate, true, false);
```

## Retrieval of Certificate Names

This method is deprecated. Retrieves a list of file names for the .p12 certificates on the SAP Mobile Server.

## Syntax

```
List<String> getApplePushNotificationCertificateNames() throws
SUPAdminException;
```

## Returns

If successful, returns a list of objects of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieval**

```
// List APNS certificate names
List<String> list =
supServerConf.getApplePushNotificationCertificateNames();
```

### Set Apple Notification Values

Constructs a value object, `APNSAppSettingsVO`, which sets values for Apple Push Notification Service settings used for iPhone push notifications.

### Syntax

```
public java.lang.String getCertificateFileName()
public void setCertificateFileName(java.lang.String value)
public java.lang.String getCertificatePassword()
public void setCertificatePassword(java.lang.String value)
public java.lang.String getPushGateway()
public void setPushGateway(java.lang.String value)
public int getPushGatewayPort()
public void setPushGatewayPort(int value)
public int getNumberOfChannels()
public void setNumberOfChannels(int value)
public java.lang.String getFeedbackGateway()
public void setFeedbackGateway(java.lang.String value)
public int getFeedbackGatewayPort()
public void setFeedbackGatewayPort(int value)
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update**

```
// construct an APNSAppSettingsVO
private APNSAppSettingsVO buildAPNSSettings() {
 APNSAppSettingsVO settings = new APNSAppSettingsVO();
 settings.setCertificateFileName("C:/
PushDevCertificate_smithj.p12");
 settings.setCertificatePassword("iM0;APNS");
 settings.setFeedbackGateway("testfeedback.push.example.com");
 settings.setFeedbackGatewayPort(123);
 settings.setName("smithj_APNS_configuration1");
 settings.setNumberOfChannels(3);
 settings.setPushGateway("testgateway.push.example.com ");
 settings.setPushGatewayPort(456);
 return settings;
}
```

## **Start or Stop Outbound Enablers**

Starts or stops Relay Server outbound enablers.

### **Syntax**

```
void startOutboundEnablers (List<OutboundEnablerVO> outBoundEnabler)
throws SUPAdminException;
void stopOutboundEnablers (List<OutboundEnablerVO> outBoundEnabler)
throws SUPAdminException;
```

### **Parameters**

- **outBoundEnabler** – The list of outbound enablers to start or stop.

### **Returns**

If successful, starts or stops all the outbound enablers of the server. If unsuccessful, throws SUPAdminException.

### **Examples**

- **Start all outbound enablers** – Start and stop all outbound enablers.

```
@Test
public void testStartAndStopOutboundEnabler() throws Exception {
 AgentContext agentContext = new
 AgentContext("mySCC.sybase.com", 9999,
 "supAdmin", "s3pAdmin");
 ServerContext context = new ServerContext("mySUP.sybase.com",
 2000,
 "supAdmin", "s3pAdmin", false, agentContext);
 SUPServerConfiguration serverConfig = SUPObjectFactory
 .getSUPServerConfiguration(context);
 // Get all Outbound Enablers of this Mobile Server node.
 List<OutboundEnablerVO> outBoundEnablers = serverConfig
 .getOutboundEnablers();
 // Start all the Outbound Enablers
 serverConfig.startOutboundEnablers(outBoundEnablers);
 // Stop all the Outbound Enablers
 serverConfig.stopOutboundEnablers(outBoundEnablers);
}
```

## **Retrieval of Relay Server Outbound Enablers**

Retrieves one or more Relay Server outbound enabler configurations for the SAP Mobile Server.

### **Syntax**

```
List<OutboundEnablerVO> getOutboundEnablers() throws
SUPAdminException;
```

```
OutboundEnablerVO getOutboundEnabler() throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Retrieval of all outbound enablers**

```
// get all Outbound Enablers of this server
List<OutboundEnablerVO> outboundEnablers = serverConfiguration
 .getOutboundEnablers();
```

- **Retrieval of a specific outbound enabler**

```
// get Outbound Enabler by specified ID.
OutboundEnablerVO outboundEnabler = serverConfiguration
 .getOutboundEnabler(3);
```

### Creation or Update of Relay Server Outbound Enabler

Creates a new outbound enabler configuration if it does not exist; otherwise updates the existing one.

### Syntax

```
void saveOutboundEnabler(OutboundEnablerVO outboundEnabler) throws
com.sybase.sup.admin.exception.SUPAdminException
```

### Parameters

- **outboundEnabler** – the outbound enabler configuration to be created or updated.

### Returns

If successful, creates or updates the specified outbound enabler configuration. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Update an outbound enabler configuration**

```
Iterator<OutboundEnablerVO> iter = supServerConf
 .getOutboundEnablers().iterator();
if (!iter.hasNext())
 return;

//update a outbound enabler configuration
OutboundEnablerVO oel = iter.next();
oel.setUnwiredServerHost("<new host>");
supServerConf.saveOutboundEnabler(oel);
```

- **Create an outbound enabler configuration**

```
//create a outbound enabler configuration
OutboundEnablerVO oe2 = new OutboundEnablerVO();
OutboundEnablerVOBuilder bld = new OutboundEnablerVOBuilder(oe2);
//copy setting from a existing one
bld.copy(oe1);
oe2.getServerNode().
 setName("<a different host name from the one copied>");
supServerConf.saveOutboundEnabler(oe2);
```

### **Update of Relay Server Outbound Enabler**

Updates an existing Relay Server outbound enabler configuration.

#### **Syntax**

```
void updateOutboundEnabler(int outboundEnablerId, OutboundEnablerVO
outboundEnabler) throws
com.sybase.sup.admin.exception.SUPAdminException
```

#### **Parameters**

- **outboundEnablerId** – the ID of the outbound enabler configuration to be updated.
- **outboundEnabler** – the outbound enabler configuration containing the property values to be updated.

#### **Returns**

If successful, updates the specified outbound enabler configuration(s). If unsuccessful, returns `SUPAdminException`.

#### **Examples**

- **Update an outbound enabler configuration**

```
Iterator<OutboundEnablerVO> iter = supServerConf
 .getOutboundEnablers().iterator();
if (!iter.hasNext())
 return;

OutboundEnablerVO oe1 = iter.next();
oe1.setCertificateFile("4321");

supServerConf.updateOutboundEnabler(oe1.getID(), oe1);
```

### **Deletion of Relay Server Outbound Enabler**

Deletes one or more Relay Server outbound enabler configurations.

You can delete the following:

## Management API

- a group of outbound enabler configurations based on a list of IDs
- a specific outbound enabler configuration based on an ID

### Syntax

```
void deleteOutboundEnablers(java.util.Set<java.lang.Integer>
outboundEnablerIds) throws
com.sybase.sup.admin.exception.SUPAdminException
```

```
void deleteOutboundEnabler(java.lang.Integer outboundEnablerId)
throws com.sybase.sup.admin.exception.SUPAdminException
```

### Parameters

- **outboundEnablerIds** – The list of outbound enabler configuration IDs that you want to delete.
- **outboundEnablerId** – The ID of the outbound enabler configuration you want to delete.

### Returns

If successful, deletes a list of outbound enabler configurations, or a specific outbound enabler configuration. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Deletion of a list of outbound enabler configurations by ID** – Deletes all outbound enabler configurations with the given IDs.

```
serverConfiguration.deleteOutboundEnabler(1);
Set<Integer> outboundEnablerIds=new HashSet<Integer>();
outboundEnablerIds.add(2);
outboundEnablerIds.add(3);
serverConfiguration.deleteOutboundEnablers(outboundEnablerIds);
```

### Addition of Relay Server Outbound Enabler Certificate Files

Adds a new certificate file to be used by Relay Server outbound enablers.

### Syntax

```
void addOutboundEnablerCertificateFile(java.lang.String fileName,
byte[] certificateBlob, java.lang.Boolean overwrite) throws
com.sybase.sup.admin.exception.SUPAdminException
```

### Parameters

- **fileName** – the file name to be used to save the certificate blob.
- **certificateBlob** – the certificate blob.

- **overwrite** – TRUE to overwrite existing file with the same name, FALSE to preserve the old one.

### **Returns**

If successful, creates the specified certificate file. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Addition of an outbound enabler certificate file**

```
File file = new File("D:\\rsoe.cer");
byte[] blob = new byte[(int)file.length()];
new DataInputStream(new FileInputStream(file)).readFully(blob);
supServerConf.addOutboundEnablerCertificateFile(file.getName(),
blob, true);
```

### **Retrieval of Relay Server Outbound Enabler Certificate Files**

Retrieve a list of Relay Server outbound enabler certificate files.

### **Syntax**

```
List<java.lang.String> getOutboundEnablerCertificateFiles() throws
com.sybase.sup.admin.exception.SUPAdminException
```

### **Returns**

If successful, returns a list of objects of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Retrieval of outbound enabler certificate files**

```
List<String>
fileNames=serverConfiguration.getOutboundEnablerCertificateFiles(
);
```

### **Deletion of Relay Server Outbound Enabler Certificate Files**

Deletes one or more Relay Server outbound enabler certificate files.

You can delete the following:

- a group of outbound enabler certificate files based on a list of file names.
- a specific outbound enabler certificate file based on a file name.

### **Syntax**

```
void
deleteOutboundEnablerCertificateFiles(java.util.Set<java.lang.Strin
```

## Management API

```
g> fileNames) throws
com.sybase.sup.admin.exception.SUPAdminException
```

```
void deleteOutboundEnablerCertificateFile(java.lang.String fileName)
throws com.sybase.sup.admin.exception.SUPAdminException
```

### Parameters

- **fileNames** – The list of outbound enabler certificate file names that you want to delete.
- **fileName** – The file name of the outbound enabler certificate file you want to delete.

### Returns

If successful, delete the specified file(s). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Deletion of a List of Outbound Enabler Certificate Files** – Deletes all outbound enabler certificates with the given file names.

```
serverConfiguration.deleteOutboundEnablerCertificateFile("rsoe.cer");
Set<String> certFileNames = new HashSet<String>();
certFileNames.add("rsoe2.cer");
certFileNames.add("rsoe3.cer");
serverConfiguration.deleteOutboundEnablerCertificateFiles(certFileNames);
```

## Configuring Security Configurations

The SAP Mobile Platform security configuration is a metadata-based configuration that includes several components.

- Authentication provider
- Authorization provider
- Audit provider
- Attribution provider (only if one has been developed with the CSI SDK)

Each of these components is a security provider, and is represented by `SecurityProviderVO`. The properties of `SecurityProviderVO` differentiate the components. See *Developer Guide: SAP Mobile Server Runtime > Management API > Client Metadata*.

Manage the SAP Mobile Platform security configuration using the `SUPSecurityConfiguration` interface. This interface provides different methods for the components. The changes made through these methods are cached locally unless the `commit()` method is called to send the cached configuration of all the components to the SAP Mobile Server.



## **Start Security Configuration Management**

Starts the management of an SAP Mobile Server security configuration.

### **Syntax**

```
public static SUPSecurityConfiguration
getSUPSecurityConfiguration(SecurityContext securityContext) throws
SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Security configuration**

```
//Retrieve a list of security configuration names currently
defined
Collection<String> securityConfigurations=
supCluster.getSecurityConfigurations();

//Start administration on one of the security configurations
securityContext = serverContext.getSecurityContext("<security
configuration name>");
SUPSecurityConfiguration supSecConf =
SUPObjectFactory.getSUPSecurityConfiguration(securityContext);
```

### **Usage**

When an instance of SUPSecurityConfiguration is returned from the SUPObjectFactory, call its method.

### **SecurityProviderVO**

The ServerProviderVO class has a read-only property that you must initialize at construction time.

The type property specifies the provider type, as described in *Developer Guide: SAP Mobile Server Runtime > Management API > Client Metadata > Security Configuration*.

### **Populate Security Configuration**

Populates an SAP Mobile Server security configuration with the currently effective configuration. The returned ConfigurationValidationStatus contains the validation status of the security configuration on the SAP Mobile Server.

### **Syntax**

```
ConfigurationValidationStatus refresh() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Populate security configuration**

```
supSecConf.refresh();
```

### **Usage**

Each call to `commit()` and `refresh()` expires all the previous `ServerProviderVO`, because all the IDs are regenerated.

`supSecConf.refresh()` retrieves from the SAP Mobile Server the current configuration, which does not include any committed changes that are pending a server restart, and caches it locally.

### **Commit Local Changes to SAP Mobile Server**

Commits local changes to the SAP Mobile Server. The returned `ConfigurationValidationStatus` contains the validation status of the security configuration on the SAP Mobile Server.

### **Syntax**

```
ConfigurationValidationStatus commit() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Commit local changes**

```
ConfigurationValidationStatus cvs = supServerConf.commit();
if(cvs.isValid()){
 //succeed.
}
else{
 //fail.
}
```

### **Active Security Providers**

Active security providers are those that are currently effective on the SAP Mobile Server. Each active security provider has a location in the respective active security provider stack. These

locations are reflected in the sequence when iterating through the returned collection. You can retrieve, update, add, or delete active security providers.

### Retrieval of Active Security Providers

Retrieves one or more active security providers.

### Syntax

```
public SecurityProviderVO getActiveAuditProvider(String
auditProviderID) throws SUPAdminException;

public SecurityProviderVO getActiveAuthenticationProvider(String
authenticationProviderID) throws SUPAdminException;

public SecurityProviderVO getActiveAuthorizationProvider(String
authorizationProviderID) throws SUPAdminException;

public SecurityProviderVO getActiveAttributionProvider(String
attributionProviderID) throws SUPAdminException;

java.util.Collection<SecurityProviderVO>
getActiveAttributionProviders()

java.util.Collection<SecurityProviderVO> getActiveAuditProviders()

java.util.Collection<SecurityProviderVO>
getActiveAuthenticationProviders()

java.util.Collection<SecurityProviderVO>
getActiveAuthorizationProviders()
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### Examples

- **Retrieval**

```
supSecConf.refresh();

Collection<SecurityProviderVO> spvos_audit =
supSecConf.getActiveAuditProviders();
SecurityProviderVO spvo_audit =
supSecConf.getActiveAuditProvider("<security provider id>");

Collection<SecurityProviderVO> spvos_authentication =
supSecConf.getActiveAuthenticationProviders();
SecurityProviderVO spvo_authentication =
supSecConf.getActiveAuthenticationProvider("<security provider
id>");
```

```
Collection<SecurityProviderVO> spvos_authorization =
supSecConf.getActiveAuthorizationProviders();
SecurityProviderVO spvo_authorization =
supSecConf.getActiveAuthorizationProvider("<security provider
id>");

Collection<SecurityProviderVO> spvos_attribution =
supSecConf.getActiveAttributionProviders();
SecurityProviderVO spvo_attribution =
supSecConf.getActiveAttributionProvider("security provider id");
```

### Update of Active Security Providers

Updates the active security providers, including the active attribution provider, audit provider, authentication provider, or authorization provider.

### Syntax

```
public void updateActiveAuditProvider(String auditProviderID,
SecurityProviderVO securityProvider) throws SUPAdminException;

public void updateActiveAuthenticationProvider(String
authenticationProviderID, SecurityProviderVO securityProvider)
throws SUPAdminException;

public void updateActiveAuthorizationProvider(String
authorizationProviderID, SecurityProviderVO securityProvider) throws
SUPAdminException;

void updateActiveAttributionProvider(java.lang.String
attributionProviderID, SecurityProviderVO securityProvider) throws
SUPAdminException
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Update audit providers**

```
supSecConf.refresh();
SecurityProviderVO spvo_audit =
supSecConf.getActiveAuditProviders()
.iterator().next();
SecurityProviderVO spvo_authentication = supSecConf
.getActiveAuthenticationProviders().iterator().next();
SecurityProviderVO spvo_authorization = supSecConf
.getActiveAuthorizationProviders().iterator().next();
supSecConf.updateActiveAuditProvider("<security provider id>",
spvo_audit);
supSecConf.updateActiveAuthenticationProvider("<security provider
id>", spvo_authentication);
```

```
supSecConf.updateActiveAuthorizationProvider("<security provider
id>", spvo_authorization);
supSecConf.commit();
```

- **Update attribution providers**

```
SecurityProviderVO spvo_attribution =
supSecConf.getActiveAttributionProviders().iterator().next();
supSecConf.updateActiveAttributionProvider("<security provider
id>", spvo_attribution);
```

### Addition of an Active Authentication Provider

Adds an active authentication provider.

#### **Syntax**

```
public void addActiveAuthenticationProvider(SecurityProviderVO
securityProvider) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Add active authentication provider**

```
supSecConf.refresh();

SecurityProviderVO spvo = new SecurityProviderVO(
 "com.sybase.security.ldap.LDAPLoginModule");
Map<String, String> properties = new HashMap<String, String>();
spvo.setProperties(properties);
//Mandatory properties.
properties.put("implementationClass",
 "com.sybase.security.ldap.LDAPLoginModule");
properties.put("providerType", "LoginModule");
properties.put("ProviderURL", "ldap://localhost:389");
properties.put("controlFlag", "optional");
//Optional properties.
properties.put("ServerType", "sunone5");

spvo.setProperties(properties);
supSecConf.addActiveAuthenticationProvider(spvo);
supSecConf.commit();
```

### Addition of an Active Authorization Provider

Adds an active authorization provider.

#### **Syntax**

```
public void addActiveAuthorizationProvider(SecurityProviderVO
securityProvider) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Add active authorization provider**

```
supSecConf.refresh();

SecurityProviderVO spvo = new SecurityProviderVO(
 "com.sybase.security.ldap.LDAPAuthorizer");
Map<String, String> properties = new HashMap<String, String>();
spvo.setProperties(properties);
// Mandatory properties.
properties.put("implementationClass",
 "com.sybase.security.ldap.LDAPAuthorizer");
properties.put("providerType", "Authorizer");
// Optional properties.
properties.put("ProviderURL", "ldap://localhost:389");
properties.put("ServerType", "sunone5");

spvo.setProperties(properties);
supSecConf.addActiveAuthorizationProvider(spvo);
supSecConf.commit();
```

### **Addition of an Active Attribution Provider**

Adds an active attribution provider.

### **Syntax**

```
public void addActiveAttributionProvider(SecurityProviderVO
securityProvider) throws SUPAdminException
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Add active attribution provider**

```
supSecConf.refresh();
SecurityProviderVO spvo = new SecurityProviderVO(
 "com.sybase.security.core.NoSecAttributer");
Map<String, String> props = MapUtil.newMap(new String[][] {
 { "implementationClass",
 "com.sybase.security.core.NoSecAttributer" },
 { "providerType", "Attributer" } });
spvo.setProperties(props);
supSecConf.addActiveAttributionProvider(spvo);
supSecConf.commit();
```

### Addition of an Active Audit Provider

Adds an active audit provider.

#### Syntax

```
public void addActiveAuditProvider(SecurityProviderVO
securityProvider) throws SUPAdminException;
```

#### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### Examples

- **Add active audit provider**

```
supSecConf.refresh();

SecurityProviderVO spvo = new SecurityProviderVO("auditor");

SecurityProviderVO spvo_dest = new SecurityProviderVO(
 "auditDestination");
SecurityProviderVO spvo_filter = new
SecurityProviderVO("auditFilter");
SecurityProviderVO spvo_formatter = new SecurityProviderVO(
 "auditFormatter");

Map<String, String> properties_dest = new HashMap<String,
String>();
Map<String, String> properties_filter = new HashMap<String,
String>();
Map<String, String> properties_formatter = new HashMap<String,
String>();

properties_dest.put("controlFlag", "optional");
properties_dest.put("implementationClass", "");
properties_dest.put("providerType", "AuditDestination");

properties_filter.put("implementationClass", "");
properties_filter.put("providerType", "AuditFilter");

properties_formatter.put("implementationClass", "");
properties_formatter.put("providerType", "AuditFormatter");

spvo_dest.setProperties(properties_dest);
spvo_filter.setProperties(properties_filter);
spvo_formatter.setProperties(properties_formatter);

spvo.setChildren(Arrays.asList(new SecurityProviderVO[]
{ spvo_dest, spvo_filter, spvo_formatter }));
```

```
supSecConf.addActiveAuditProvider(spvo);
supSecConf.commit();
```

### Deletion of an Active Security Provider

Deletes an active security provider.

### Syntax

```
public void deleteActiveAuditProvider(String auditProviderID) throws
SUPAdminException;
```

```
public void deleteActiveAuthenticationProvider(String
authenticationProviderID) throws SUPAdminException;
```

```
public void deleteActiveAuthorizationProvider(String
authorizationProviderID) throws SUPAdminException;
```

```
void deleteActiveAttributionProvider(java.lang.String
attributionProviderID)
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### Examples

- **Delete**

```
supSecConf.refresh();

supSecConf.deleteActiveAuditProvider("<security provider id>");
supSecConf.deleteActiveAuthenticationProvider("<security provider
id>");
supSecConf.deleteActiveAuthorizationProvider("<security provider
id>");
supSecConf.deleteActiveAttributionProvider("<security provider
id>");

supSecConf.commit();
```

### Security Configuration Validation

Delivers modified SAP Mobile Platform security configuration to the SAP Mobile Server for validation. The current SAP Mobile Server security configuration is not affected.

### Syntax

```
ConfigurationValidationStatus validate() throws SUPAdminException;
```



## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## **Examples**

- **Validation**

```
ConfigurationValidationStatus cvs = supSecConf.validate();
if(cvs.isValid()){
 //valid.
}
else{
 //invalid.
}
```

## **Adjustment of the Sequence of Active Security Providers**

Security provider instances are grouped together by their provider types (attribution provider, audit provider, authentication provider, and authorization provider) and ordered in a sequence.

The following methods adjust the sequence of security providers in each group.

## **Syntax**

```
public void moveDownActiveAuditProvider(String auditProviderID)
throws SUPAdminException;
```

```
public void moveDownActiveAuthenticationProvider(String
authenticationProviderID) throws SUPAdminException;
```

```
public void moveDownActiveAuthorizationProvider(String
authorizationProviderID) throws SUPAdminException;
```

```
public void moveUpActiveAuditProvider(String auditProviderID) throws
SUPAdminException;
```

```
public void moveUpActiveAuthenticationProvider(String
authenticationProviderID) throws SUPAdminException;
```

```
public void moveUpActiveAuthorizationProvider(String
authorizationProviderID) throws SUPAdminException;
```

## **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Adjust sequence**

```
supSecConf.refresh();

supSecConf.moveDownActiveAuditProvider("<security provider id>");
supSecConf.moveDownActiveAuthenticationProvider("<security provider id>");
supSecConf.moveDownActiveAuthorizationProvider("<security provider id>");
supSecConf.commit();

supSecConf.moveUpActiveAuditProvider("<security provider id>");
supSecConf.moveUpActiveAuthenticationProvider("<security provider id>");
supSecConf.moveUpActiveAuthorizationProvider("<security provider id>");
supSecConf.commit();

supSecConf.moveDownActiveAttributionProvider("<security provider id>");
supSecConf.moveUpActiveAttributionProvider("<security provider id>")
```

### **Retrieval of Installed Security Providers**

Retrieves a list of the security providers installed in the SAP Mobile Server.

### **Syntax**

```
public Collection<String> getInstalledAuditDestinationProviders()
throws SUPAdminException;

public Collection<String> getInstalledAuditFilterProviders() throws
SUPAdminException;

public Collection<String> getInstalledAuditFormatterProviders()
throws SUPAdminException;

public Collection<String> getInstalledAuthenticationProviders()
throws SUPAdminException;

public Collection<String> getInstalledAuthorizationProviders()
throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieval of installed security providers**

```
supSecConf.refresh();

Collection<String> spvos_audit_dest = supSecConf
 .getInstalledAuditDestinationProviders();
Collection<String> spvos_audit_filter = supSecConf
 .getInstalledAuditFilterProviders();
Collection<String> spvos_audit_formatter = supSecConf
 .getInstalledAuditFormatterProviders();
Collection<String> spvos_authentication = supSecConf
 .getInstalledAuthenticationProviders();
Collection<String> spvos_authorization = supSecConf
 .getInstalledAuthorizationProviders();
```

- **Retrieval of attribution provider**

```
java.util.Collection<java.lang.String>
getInstalledAttributionProviders()
Collection<String> spvos_attribution = supSecConf
 .getInstalledAttributionProviders();
```

## Create Logical Role for Security Configuration

Creates logical role and sets the role mappings for this logical role.

### Syntax

```
public void createLogicalRole(String logicalRole, ROLE_MAPPING_TYPE
type,
 List<String> physicalRoles) throws SUPAdminException;
```

### Parameters

- **logicalRole** – The logical role name.
- **type** – One of the role mapping types: `ROLE_MAPPING_TYPE.AUTO`, `ROLE_MAPPING_TYPE.MAPPED`, or `ROLE_MAPPING_TYPE.NONE`.
- **physicalRoles** – The physical roles list if the type parameter is `ROLE_MAPPING_TYPE.MAPPED`, otherwise this parameter will be ignored.

### Returns

If successful, returns silently. If unsuccessful, throws `SUPAdminException`.

## Delete logical role for security configuration

Deletes logical role.

### Syntax

```
public void deleteLogicalRole(String logicalRole) throws
SUPAdminException;
```

### Parameters

- **logicalRole** – The logical role name to delete.

### Returns

If successful, returns silently. If the logical role is already referred to by the application connection templates, throws SUPAdminException.

### Retrieve role mappings for security configuration logical roles

Returns cluster level role mappings.

### Syntax

```
public Collection<RoleMappingVO> getRoleMappings() throws
SUPAdminException;
```

### Returns

If successful, returns the list of RoleMappingVO objects. If unsuccessful, returns SUPAdminException.

### Update Role Mapping for Security Configuration Logical Role

Updates role mappings for security configuration.

### Syntax

```
public void updateLogicalRole(String logicalRole, ROLE_MAPPING_TYPE
type,
List<String> physicalRoles) throws SUPAdminException;
```

### Parameters

- **logicalRole** – The logical role name.
- **type** – One of the role mapping types: ROLE\_MAPPING\_TYPE.AUTO, ROLE\_MAPPING\_TYPE.MAPPED, or ROLE\_MAPPING\_TYPE.NONE.
- **physicalRoles** – The physical roles list if the type parameter is ROLE\_MAPPING\_TYPE.MAPPED, otherwise this parameter will be ignored.

### Returns

If successful, returns silently. If unsuccessful, throws SUPAdminException.

## Managing Mobile Workflows

This interface has been deprecated. Mobile workflow packages, typically created through the Mobile Workflow Application Designer, allow a developer to design mobile workflow

screens that can call on the create, update, and delete operations, as well as object queries, of a mobile business object.

You can manage mobile workflow packages through the `SUPMobileWorkflow` interface. Operations you can perform with this interface include:

- Starting administration of mobile workflow packages
- Package management and installation: listing packages, installing packages, and deleting packages
- Retrieving matching rules, context variables, error lists, and queue items
- Updating properties, matching rules, and context variables
- Managing mobile workflow device assignment
- Managing e-mail settings

### **Start Management of Mobile Workflow Packages**

This method has been deprecated. Starts the management of mobile workflow packages.

### **Syntax**

```
public static SUPMobileWorkflow getSUPMobileWorkflow(ClusterContext
clusterContext) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Start mobile workflow package management**

```
...
private SUPMobileWorkflow workflow;
...
ServerContext serverContext = new ServerContext("wangf-dell",
2000, "supAdmin", "supPwd", false);
clusterContext = serverContext.getClusterContext("wangf's
cluster");
workflow = SUPObjectFactory.getSUPMobileWorkflow(clusterContext);
```

### **Usage**

To manage SAP Mobile Server mobile workflow packages, you must create an instance of `ServerContext` with the correct information, and pass it to `SUPObjectFactory.getSUPMobileWorkflow()`. When an instance of `SUPMobileWorkflow` is returned, you can call its method as a typical Java method call.

### **Mobile Workflow Package Retrieval**

This method has been deprecated. Retrieves a list of mobile workflow packages.

#### **Syntax**

```
List<MobileWorkflowVO> getMobileWorkflowList() throws
SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Mobile workflow package retrieval**

```
// List workflows
List<WorkflowVO> workflows = workflow.getMobileWorkflowList();
```

### **Installation of a Mobile Workflow Package**

This method has been deprecated. Installs a mobile workflow package.

#### **Syntax**

```
MobileWorkflowIDVO installMobileWorkflow(byte[]
zippedWorkflowPackage) throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Mobile workflow package installation** – This code fragment installs a mobile workflow package named `ActivitiesPackage.zip`, and returns the package name once it is successfully installed:

```
// Install workflow
byte[] workflowBytes= getWorkflowBytes();
MobileWorkflowIDVO workflowID = workflow
 .installMobileWorkflow(zippedWorkflowPackage);

private byte[] getWorkflowBytes() throws URISyntaxException,
IOException {
 String ZIP_NAME = "C:/ActivitiesPackage.zip";
 File zipFile = new File(ZIP_NAME);
 byte[] zippedWorkflowPackage = new byte[(int)
zipFile.length()];
```

```

 DataInputStream inputStream = new DataInputStream(new
FileInputStream(
 zipFile));
 inputStream.readFully(zippedWorkflowPackage);
 return zippedWorkflowPackage;
}

```

### **Deletion of a Mobile Workflow Package**

This method has been deprecated. Deletes the specified mobile workflow package.

### **Syntax**

```

void deleteMobileWorkflow(MobileWorkflowIDVO workflowID) throws
SUPAdminException;

```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Mobile workflow package deletion** – This code fragment deletes a mobile workflow package with the specified workflow ID:

```

// delete workflow
workflow.deleteMobileWorkflow(workflowID);

```

### **Retrieval of Matching Rules**

This method has been deprecated. Retrieves matching rules for the specified mobile workflow package.

Matching rules are used by the email listener to identify e-mails that match the rules specified by the administrator. When an e-mail message matches the rule, Unwired Server sends the e-mail message as a workflow to the device that matches the rule.

### **Syntax**

```

MobileWorkflowMatchingRulesVO
getMobileWorkflowMatchingRule(MobileWorkflowIDVO workflowID) throws
SUPAdminException;

```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Mobile workflow matching rules**

```

// Get workflow Matching rule
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();

```

```
workflowID.setVersion(1);
workflowID.setWID(6);
MobileWorkflowMatchingRulesVO vo =
workflow.getMobileWorkflowMatchingRule(workflowID);
```

### **Retrieval of Context Variables**

This method has been deprecated. Retrieves context variables for the specified mobile workflow package.

Context variables customize how data is loaded into the SAP Mobile Server cache. You can use context variables to create a smaller, more focused data set that may yield improved performance.

### **Syntax**

```
List<MobileWorkflowContextVariableVO>
getMobileWorkflowContextVariables(MobileWorkflowIDVO workflowID)
throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Mobile workflow context variables** – This code fragment retrieves context variables for the specified mobile workflow package:

```
// Get workflow context variables
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(1);
workflowID.setWID(6);
List<WorkflowContextVariableVO> list = workflow
 .getMobileWorkflowContextVariables(workflowID);
```

### **Retrieval of an Error List**

This method has been deprecated. Retrieves an error list for the specified mobile workflow package for the specified time period, and paginates the results.

### **Syntax**

```
PaginationResult<MobileWorkflowErrorVO>
getMobileWorkflowErrorList(int startIndex, int maxRecordsToReturn,
MobileWorkflowIDVO id, String userName, Calendar startDate, Calendar
endDate, String orderByField, boolean bAscending) throws
SUPAdminException;
```



## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Mobile workflow error list** – retrieves an error list for the mobile workflow package starting from the date September 30, 2009:

```
// Get workflow error list
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(1);
workflowID.setWID(7);
Calendar startDate = Calendar.getInstance();
startDate.set(2009, 9, 30);
Calendar endDate = Calendar.getInstance();
PaginationResult<WorkflowErrorVO> list = workflow
 .getMobileWorkflowErrorList(0, 1, workflowID,
 "TEST4", startDate,
 endDate, null, true);
```

## Retrieval and Management of Queue Items

This method has been deprecated. Retrieves a list of queue items for the specified Mobile Workflow package, and deletes the specified queue items.

## Syntax

```
PaginationResult<MobileWorkflowQueueItemVO>
getMobileWorkflowQueueItems(int startIndex, int maxRecordsToReturn,
MobileWorkflowIDVO id, List<Integer> deviceIDs, List<String>
userNames, String orderByField, boolean ascending) throws
SUPAdminException;

void deleteMobileWorkflowQueueItem(Integer queueItemID, Boolean
forTransformQueue) throws SUPAdminException;
```

## Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

## Examples

- **Mobile workflow queue items**

```
// Get workflow queue items
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(1);
workflowID.setWID(1);
List<Integer> deviceIds = new ArrayList<Integer>();
deviceIds.add(4);
PaginationResult<MobileWorkflowQueueItemVO> list = workflow
```

```
.fetchWorkflowQueueItems(0, 2, workflowID, null, null, null, false);

//Delete MobileWorkflow queue items.
workflow.deleteMobileWorkflowQueueItem(1, true);
```

### **Update of Properties**

This method has been deprecated. Updates the properties for the specified Mobile Workflow package.

### **Syntax**

```
void updateMobileWorkflowDisplayName(MobileWorkflowIDVO workflowID,
String displayName) throws SUPAdminException;

void updateMobileWorkflowIconIndex(MobileWorkflowIDVO workflowID,
int iconIndex) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Mobile workflow properties** – updates the display name and icon index for the specified Mobile Workflow package:

```
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(1);
workflowID.setWID(6);

// Update workflow display name
workflow.updateMobileWorkflowDisplayName(workflowID, ":");

// Update workflow icon index
workflow.updateMobileWorkflowIconIndex(workflowID, 100);
```

### **Update of Matching Rules**

This method has been deprecated. Updates a matching rule for the specified Mobile Workflow package.

### **Syntax**

```
void updateMobileWorkflowMatchingRule(MobileWorkflowIDVO workflowID,
MobileWorkflowMatchingRulesVO matchRule) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Mobile workflow matching rules**

```
// Update workflow matching rule
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(1);
workflowID.setWID(6);
MobileWorkflowMatchingRulesVO matchRule = workflow
 .getWorkflowMatchingRule(workflowID);
matchRule.setBODYExpressionType(MobileWorkflowMatchingRulesVO.EXP
 RESSION_TYPE_REGULAREXPRESSION);
matchRule.setBODYExpression(".*wang.*");
workflow.updateMobileWorkflowMatchingRule(workflowID, matchRule);
```

## Update of Context Variables

This methods has been deprecated. Updates context variables for the specified Mobile Workflow package.

## Syntax

```
void updateMobileWorkflowContextVariables(MobileWorkflowIDVO
workflowID, List<MobileWorkflowContextVariableVO> contextVariables)
throws SUPAdminException;
```

## Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Mobile workflow context variables** – updates context variables for an existing mobile workflow package with workflow ID 2:

```
// Update MobileWorkflow context variables
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
// ID 2 version 1 is a existing Mobile Workflow on the server
workflowID.setVersion(1);
workflowID.setWID(2);
List<MobileWorkflowContextVariableVO> contextVariables = workflow
 .getMobileWorkflowContextVariables(workflowID);
contextVariables.get(0).setValue("string value updated");
workflow.updateMobileWorkflowContextVariables(workflowID, contextV
 ariables);
```

## Usage

For mobile workflow packages that do not support certificate-based authentication, use the following context variables to specify credentials:

- SupUser
- SupPassword

For mobile workflow packages that support certificate-based authentication, use the above variables and the following additional context variables:

- SupCertificateIssuer
- SupCertificateSubject
- SupCertificateNotAfter
- SupCertificateNotBefore

---

**Note:** In this case, all the context variables are read-only.

---

### **Retrieval of Mobile Workflow Device Status**

This method has been deprecated. Retrieves mobile workflow status for a device from the value object DeviceMobileWorkflowStatusVO.

### **Syntax**

```
List<DeviceMobileWorkflowStatusVO>
getDeviceMobileWorkflowStatus (MobileWorkflowIDVO workflowID) throws
SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Mobile workflow device assignments**

```
// get MobileWorkflow assignment info
List<DeviceMobileWorkflowStatusVO> list = workflow
 .getDeviceMobileWorkflowStatus (workflowID);
```

### **Assignment of a Workflow Package**

This method has been deprecated. Defines a mobile workflow package and devices, and assigns the package to the devices.

### **Syntax**

```
void assignMobileWorkflowToDevices (MobileWorkflowIDVO workflowID,
List<Integer> deviceIDs) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Package assignment**

```
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(1);
workflowID.setWID(2);
List<Integer> deviceIDs = new ArrayList<Integer>();
deviceIDs.add(64);
// assign MobileWorkflow to devices
workflow.assignMobileWorkflowToDevices(workflowID, deviceIDs);
```

## Unassignment of a Workflow Package

This method has been deprecated. Unassigns a Mobile Workflow package from devices.

### Syntax

```
void unassignMobileWorkflowFromDevices(MobileWorkflowIDVO
workflowID, List<Integer> deviceIDs) throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Package unassignment**

```
// unassign MobileWorkflow to devices
workflow.unassignMobileWorkflowFromDevices(workflowID,
deviceIDs);
```

## Retrieval of Device Workflow Assignments

This method has been deprecated. Retrieves all mobile workflow packages that are assigned to the specified device.

### Syntax

```
List<MobileWorkflowAssignmentVO>
getDeviceWorkflowAssignments(Integer
applicationConnectionNumericID) throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## Examples

- **Retrieve mobile workflow device assignments**

```
// get all MobileWorkflows that assign to the device. Where 3 is a
// existing application connection numeric ID.
```

```
List<MobileWorkflowAssignmentVO> assignments = workflow
 .getDeviceWorkflowAssignments(3);
```

### **E-mail Settings Configuration**

This method has been deprecated. Updates or retrieves the e-mail settings for a mobile workflow package.

E-mail settings allow the administrator to configure a listener to scan all incoming e-mail messages delivered to an inbox that the administrator indicates during configuration.

### **Syntax**

```
Boolean testEmailConnection(String configXml) throws
 SUPAdminException;

void configureEmail(String configurationXML) throws
 SUPAdminException;

void enableEmail(boolean enable) throws SUPAdminException;

String getEmailConfiguration() throws SUPAdminException;

Boolean isEmailEnabled() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Mobile workflow e-mail settings**

```
String configXmlString = readEmailConfig();

// Test Email Multicast connection
Boolean test = workflow.testEmailConnection(config);

// Config Email Multicast
workflow.configureEmail(config);

// Enable Email Multicast
workflow.enableEmail(true);

// Get Email Multicast configuration
String config = workflow.getEmailConfiguration();

// Check if Email Multicast enabled
boolean enable = workflow.isEmailEnabled();

// Read Email Multicast config XML string from file
private String readEmailConfig() throws IOException {
 StringBuffer sb = new StringBuffer();
 InputStream in = getClass().getResourceAsStream(
```

```

 "/com/sybase/sup/example/email/EmailMulticastConfig.xml");
 BufferedReader reader = new BufferedReader(new
InputStreamReader(in));
 String line;
 while ((line = reader.readLine()) != null) {
 sb.append(line);
 System.out.println(line);
 }
 reader.close();
 return sb.toString();
}

```

### **Unblock Mobile Workflow Queue**

This method has been deprecated. Unblocks the mobile workflow queue for the selected workflows and devices.

### **Syntax**

```

void unblockWorkflowQueueForDevices (MobileWorkflowIDVO workflowID,
List<Integer> applicationConnectionNumericIDs, Boolean
forTransformQueue) throws SUPAdminException;

```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Mobile workflow queue**

```

// prepare mobile workflow ID
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setVersion(100);
workflowID.setWID(2);
// prepare application connection numeric ids
List<Integer> applicationConnectionNumericIDs = new
ArrayList<Integer>();
applicationConnectionNumericIDs.add(1);
applicationConnectionNumericIDs.add(2);
// Unblock mobile workflow queue for application connections
workflow.unblockWorkflowQueueForDevices(workflowID,
applicationConnectionNumericIDs, true);

```

### **Replace Mobile Workflow Certificate**

This method has been deprecated. Replaces the certificate for a mobile workflow package.

### **Syntax**

```

void replaceMobileWorkflowCertificate(workflowID,
baos.toByteArray(), "password");

```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Replace certificate**

```
InputStream is = workflowRL.getResourceAsStream("sybase101.p12");
ByteArrayOutputStream baos = new ByteArrayOutputStream();
byte[] buf = new byte[512];
int count;
while ((count = is.read(buf)) != -1) {
 baos.write(buf, 0, count);
}
is.close();
baos.flush();
baos.close();
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setWID(4);
workflowID.setVersion(1);

workflow.replaceMobileWorkflowCertificate(workflowID,
 baos.toByteArray(), "password");
```

## **Managing Hybrid Apps**

Hybrid App packages, typically created through the Hybrid App Designer, allow a developer to design Hybrid App screens that can call on the create, update, and delete operations, as well as object queries, of a mobile business object.

You can manage Hybrid App packages through the `SUPMobileHybridApp` interface. Operations you can perform with this interface include:

- Starting administration of Hybrid App packages
- Package management and installation: listing packages, installing packages, and deleting packages
- Retrieving matching rules, context variables, error lists, and queue items
- Updating properties, matching rules, and context variables
- Managing Hybrid App device assignment
- Managing e-mail settings

### **Start Management of Hybrid App Packages**

Starts the management of Hybrid App packages.

### **Syntax**

```
public static SUPMobileHybridApp
getSUPMobileHybridApp(ClusterContext clusterContext) throws
SUPAdminException;
```



## Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

## Examples

- **Start Hybrid App package management**

```

...
private SUPMobileHybridApp mobileHybridApp;
...
ServerContext serverContext = new ServerContext("wangf-dell",
2000, "supAdmin", "supPwd", false);
clusterContext = serverContext.getClusterContext("my cluster");
mobileHybridApp =
SUObjectFactory.getSUPMobileHybridApp(clusterContext);

```

## Usage

To manage SAP Mobile Server Hybrid App packages, you must create an instance of `ServerContext` with the correct information, and pass it to `SUObjectFactory.getSUPMobileHybridApp()`. When an instance of `SUPMobileHybridApp` is returned, you can call its method as a typical Java method call.

## Hybrid App Package Retrieval

Retrieves a list of Hybrid App packages.

## Syntax

```
List<MobileHybridAppVO> getMobileHybridAppList() throws
SUPAdminException;
```

## Returns

If successful, returns a list of objects of the specified type (can be null).

## Examples

- **Hybrid App package retrieval**

```
List<MobileHybridAppVO>
hybridApps=mobileHybridApp.getMobileHybridAppList();
```

## Installation of a Hybrid App Package

Installs a Hybrid App package.

## Syntax

```
MobileHybridAppIDVO installMobileHybridApp(byte[]
zippedWorkflowPackage) throws SUPAdminException;
```

### Returns

If successful, returns an object of the specified type (can be null). If unsuccessful, returns `SUPAdminException`.

### Examples

- **Hybrid App package installation** – This code fragment installs a Hybrid App package named `ActivitiesPackage.zip`, and returns the package name once it is successfully installed:

```
// Install workflow
byte[] hybridAppBytes= getHybridAppBytes();
MobileHybridAppIDVO hybridAppID = mobileHybridApp
 .installMobileHybridApp(hybridAppBytes);

private byte[] getHybridAppBytes() throws URISyntaxException,
IOException {
 String ZIP_NAME = "C:/ActivitiesPackage.zip";
 File zipFile = new File(ZIP_NAME);
 byte[] zippedWorkflowPackage = new byte[(int)
zipFile.length()];
 DataInputStream inputStream = new DataInputStream(new
FileInputStream(
 zipFile));
 inputStream.readFully(zippedWorkflowPackage);
 return zippedWorkflowPackage;
}
}
```

### Deletion of a Hybrid App Package

Deletes the specified Hybrid App package.

### Syntax

```
void deleteMobileHybridApp(MobileHybridAppIDVO hybridAppID) throws
SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### Examples

- **Hybrid App package deletion** – This code fragment deletes a Hybrid App package with the specified Hybrid App ID:

```
mobileHybridApp.deleteMobileHybridApp(hybridAppID);
```

### **Retrieval of Hybrid App Matching Rules**

Retrieves matching rules for the specified Hybrid App package.

Matching rules are used by the e-mail listener to identify e-mails that match the rules specified by the administrator. When an e-mail message matches the rule, SAP Mobile Server sends the e-mail message as a workflow to the device that matches the rule.

#### **Syntax**

```
MobileHybridAppMatchingRulesVO
getMobileHybridAppMatchingRule (MobileHybridAppIDVO hybridAppID)
throws SUPAdminException;
```

#### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Mobile Hybrid App matching rules**

```
// Get HybridApp Matching rule
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(6);
MobileHybridAppMatchingRulesVO vo =
mobileHybridApp.getMobileHybridAppMatchingRule (hybridAppID);
```

### **Retrieval of Hybrid App Context Variables**

Retrieves context variables for the specified Hybrid App package.

Context variables customize how data is loaded into the SAP Mobile Server cache. You can use context variables to create a smaller, more focused data set that may yield improved performance.

#### **Syntax**

```
List<MobileHybridAppContextVariableVO>
getMobileHybridAppContextVariables (MobileHybridAppIDVO hybridAppID)
throws SUPAdminException;
```

#### **Returns**

If successful, returns a list of objects of the specified type (can be null).

### Examples

- **Hybrid App context variables** – This code fragment retrieves context variables for the specified Hybrid App package:

```
// Get HybridApp context variables
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(6);
List<MobileHybridAppContextVariableVO> list = mobileHybridApp
 .getMobileHybridAppContextVariables(hybridAppID);
```

### Retrieval of a Hybrid App Error List

Retrieves an error list for the specified Hybrid App package for the specified time period, and paginates the results.

### Syntax

```
PaginationResult<MobileHybridAppErrorVO>
getMobileHybridAppErrorList(int startIndex, int maxRecordsToReturn,
MobileHybridAppIDVO id, String userName, Calendar startDate,
Calendar endDate, String orderByField, boolean bAscending) throws
SUPAdminException;
```

### Returns

If successful, returns a list of objects of the specified type (can be null).

### Examples

- **Hybrid App error list** – retrieves an error list for the Hybrid App package starting from the date September 30, 2009:

```
// Get HybridApp error list
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(6);
Calendar startDate = Calendar.getInstance();
startDate.set(2009, 9, 30);
Calendar endDate = Calendar.getInstance();
PaginationResult<MobileHybridAppErrorVO> result = mobileHybridApp
 .getMobileHybridAppErrorList(0, 1, hybridAppID,
"TEST4", startDate,
 endDate, null, true);
```

### Retrieval and Management of Hybrid App Queue Items

Retrieves a list of queue items for the specified Hybrid App package, and deletes the specified queue items.

### Syntax

```
PaginationResult<MobileHybridAppQueueItemVO>
getMobileHybridAppQueueItems(int startIndex, int maxRecordsToReturn,
```

```
MobileHybridAppIDVO id, List<Integer> deviceIDs, List<String>
userNames, String orderByField, boolean ascending) throws
SUPAdminException;
 void deleteMobileHybridAppQueueItem(Integer queueItemID, Boolean
forTransformQueue) throws SUPAdminException;
```

### **Returns**

If successful, returns a list of objects of the specified type (can be null).

### **Examples**

- **Hybrid App queue items**

```
// Get HybridApp queue items
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(1);
List<Integer> deviceIds = new ArrayList<Integer>();
deviceIds.add(4);
PaginationResult<MobileHybridAppQueueItemVO> list =
mobileHybridApp
 .getMobileHybridAppQueueItems(0, 2, hybridAppID, null,
null, null, false);

//Delete HybridApp queue items.
mobileHybridApp.deleteMobileHybridAppQueueItem(1, true);
```

### **Update of Hybrid App Properties**

Updates the properties for the specified Hybrid App package.

### **Syntax**

```
void updateMobileHybridAppDisplayName (MobileHybridAppIDVO
hybridAppID, String displayName) throws SUPAdminException;
void updateMobileHybridAppIconIndex (MobileHybridAppIDVO hybridAppID,
int iconIndex) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

### **Examples**

- **Hybrid App properties** – updates the display name and icon index for the specified Hybrid App package:

```
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(6);

// Update workflow display name
mobileHybridApp.updateMobileHybridAppDisplayName (hybridAppID, "My
```

## Management API

```
Hybrid App");

// Update workflow icon index
mobileHybridApp.updateMobileHybridAppIconIndex (hybridAppID, 100);
```

### **Update of Hybrid App Matching Rules**

Updates a matching rule for the specified Hybrid App package.

#### **Syntax**

```
void updateMobileHybridAppMatchingRule (MobileHybridAppIDVO
hybridAppID, MobileHybridAppMatchingRulesVO matchRule) throws
SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Hybrid App matching rules**

```
// Update HybridApp matching rule
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(6);
MobileHybridAppMatchingRulesVO matchRule = mobileHybridApp
.getMobileHybridAppMatchingRule (hybridAppID);
matchRule.setBODYExpressionType (MobileHybridAppMatchingRulesVO.EX
PRESSION_TYPE_REGULAREXPRESSION);
matchRule.setBODYExpression (".*wang.*");
mobileHybridApp.updateMobileHybridAppMatchingRule (hybridAppID,
matchRule);
```

### **Update of Hybrid App Context Variables**

Updates context variables for the specified Hybrid App package.

#### **Syntax**

```
void updateMobileHybridAppContextVariables (MobileHybridAppIDVO
hybridAppID, List<MobileHybridAppContextVariableVO>
contextVariables) throws SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Hybrid App context variables** – updates context variables for an existing Hybrid App package with workflow ID 2:

```
// Update HybridApp context variables
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
// ID 2 version 1 is a existing HybridApp on the server
hybridAppID.setVersion(1);
hybridAppID.setWID(2);
List<MobileHybridAppContextVariableVO> contextVariables =
mobileHybridApp
 .getMobileHybridAppContextVariables(hybridAppID);
contextVariables.get(0).setValue("string value updated");
mobileHybridApp.updateMobileHybridAppContextVariables(hybridAppID
, contextVariables);
```

## **Usage**

For Hybrid App packages that do not support certificate-based authentication, use the following context variables to specify credentials:

- SupUser
- SupPassword

For Hybrid App packages that support certificate-based authentication, use the above variables and the following additional context variables:

- SupCertificateIssuer
- SupCertificateSubject
- SupCertificateNotAfter
- SupCertificateNotBefore

---

**Note:** In this case, all the context variables are read-only.

---

## **Retrieval of Hybrid App Device Status**

Retrieves Hybrid App status for a device from the value object

DeviceMobileHybridAppStatusVO.

## **Syntax**

```
List<MobileHybridAppAssignmentVO>
getDeviceHybridAppAssignments(Integer
applicationConnectionNumericID) throws SUPAdminException;
```

## **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

## **Examples**

- **Hybrid App device assignments**

```
// get all HybridApp that assign to the device. Where 3 is a
// existing application connection numeric ID.
```

```
List<MobileHybridAppAssignmentVO> assignments = mobileHybridApp
.getDeviceHybridAppAssignments(3);
```

### **Assignment of a Hybrid App Package**

Defines a Hybrid App package and devices, and assigns the package to the devices.

#### **Syntax**

```
void assignMobileHybridAppToDevices (MobileHybridAppIDVO
hybridAppID, List<Integer> applicationConnectionNumericIDs) throws
SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Package assignment**

```
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(2);
List<Integer> deviceIDs = new ArrayList<Integer>();
deviceIDs.add(64);
// assign Hybrid App to devices
hybridApp.assignMobileHybridAppToDevices(hybridAppID, deviceIDs);
```

### **Unassignment of a Hybrid App Package**

Unassigns a Hybrid App package from devices.

#### **Syntax**

```
void unassignMobileHybridAppFromDevices (MobileHybridAppIDVO
hybridAppID, List<Integer> applicationConnectionNumericIDs) throws
SUPAdminException;
```

#### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

#### **Examples**

- **Package unassignment**

```
// unassign Hybrid App to devices
hybridApp.unassignMobileHybridAppFromDevices (HybridAppID,
deviceIDs);
```



## **Retrieval of Device Hybrid App Assignments**

Retrieves all Hybrid App packages that are assigned to the specified device.

### **Syntax**

```
List<MobileHybridAppAssignmentVO>
getDeviceHybridAppAssignments(Integer
applicationConnectionNumericID) throws SUPAdminException;
```

### **Returns**

If successful, returns a list of objects of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Retrieve mobile workflow device assignments**

```
// get all Mobile Hybrid App that assign to the device. Where 3 is
a
// existing application connection numeric ID.
List<MobileHybridAppAssignmentVO> assignments =
hybridApp.getDeviceHybridAppAssignments(3);
```

## **E-mail Settings Configuration for Hybrid Apps**

Updates or retrieves the e-mail settings for a Hybrid App package.

E-mail settings allow the administrator to configure a listener to scan all incoming e-mail messages delivered to an inbox that the administrator indicates during configuration.

### **Syntax**

```
Boolean testEmailConnection(String configXml) throws
SUPAdminException;

void configureEmail(String configurationXML) throws
SUPAdminException;

void enableEmail(boolean enable) throws SUPAdminException;

String getEmailConfiguration() throws SUPAdminException;

Boolean isEmailEnabled() throws SUPAdminException;
```

### **Returns**

If successful, returns an object of the specified type (can be null). If unsuccessful, returns SUPAdminException.

### **Examples**

- **Hybrid App e-mail settings**

```
String configXmlString = readEmailConfig();

// Test Email Multicast connection
Boolean test = mobileHybridApp.testEmailConnection(config);

// Config Email Multicast
mobileHybridApp.configureEmail(config);

// Enable Email Multicast
mobileHybridApp.enableEmail(true);

// Get Email Multicast configuration
String config = mobileHybridApp.getEmailConfiguration();

// Check if Email Multicast enabled
boolean enable = mobileHybridApp.isEmailEnabled();

// Read Email Multicast config XML string from file
private String readEmailConfig() throws IOException {
StringBuffer sb = new StringBuffer();
InputStream in = getClass().getResourceAsStream(
 "/com/sybase/sup/example/email/EmailMulticastConfig.xml");
BufferedReader reader = new BufferedReader(new
InputStreamReader(in));
String line;
while ((line = reader.readLine()) != null) {
 sb.append(line);
 System.out.println(line);
}
reader.close();
return sb.toString();
}
```

### **Unblock Hybrid App Queue**

Unblocks the Hybrid App queue for the selected Hybrid Apps and devices.

### **Syntax**

```
void unblockHybridAppQueueForDevices(MobileHybridAppIDVO
hybridAppID, List<Integer> applicationConnectionNumericIDs, Boolean
forTransformQueue) throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Hybrid App queue**

```
// prepare mobile HybridApp ID
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setVersion(1);
hybridAppID.setWID(2);
// prepare application connection numeric ids
List<Integer> applicationConnectionNumericIDs = new
ArrayList<Integer>();
applicationConnectionNumericIDs.add(1);
applicationConnectionNumericIDs.add(2);
// Unblock mobile HybridApp queue for application connections
mobileHybridApp.unblockHybridAppQueueForDevices(hybridAppID,
applicationConnectionNumericIDs, true);
```

## Replace Hybrid App Certificate

Replaces the certificate for a Hybrid App package.

### Syntax

```
void replaceMobileHybridAppCertificate(MobileHybridAppIDVO
hybridAppID, byte[] certificateBlob, String certificatePassword)
throws SUPAdminException;
```

### Returns

If successful, returns silently. If unsuccessful, returns SUPAdminException.

## Examples

- **Replace certificate**

```
InputStream is = getClass().getResourceAsStream("sybase101.p12");
ByteArrayOutputStream baos = new ByteArrayOutputStream();
byte[] buf = new byte[512];
int count;
while ((count = is.read(buf)) != -1) {
 baos.write(buf, 0, count);
}
is.close();
baos.flush();
baos.close();
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setWID(4);
hybridAppID.setVersion(1);

mobileHybridApp.replaceMobileHybridAppCertificate(hybridAppID,
baos.toByteArray(), "password");
```

## **Management Client Application Shutdown**

Releases resources currently held by the API. This method only needs to be called on the termination of the management client application.

### **Syntax**

```
public static void shutdown() throws SUPAdminException;
```

### **Returns**

If successful, returns silently. If unsuccessful, returns `SUPAdminException`.

### **Examples**

- **Shutdown**

```
SUPObjectFactory.shutdown();
```

## **Client Metadata**

---

Use metadata to add values the administrator can use to configure SAP Mobile Platform properties.

### **See also**

- *Administration Interfaces* on page 343
- *Metadata* on page 345

## **Security Configuration**

The security configuration for SAP Mobile Platform consists of the several types of security providers.

- Audit provider
- Authentication provider
- Authorization provider
- Attribution provider

Each of these provider types can have multiple instances in the security configuration. For example, a security configuration could have two audit providers, four authentication providers, and five authorization providers. Each security provider instance has a unique ID.

Security provider instances are grouped together by type; the instance stack sequence in each group can be adjusted.

**Audit Provider**

An auditor consists of one destination, one filter, and one formatter:

- The supported value for destination is `com.sybase.security.core.FileAuditDestination`. Optionally, you can develop a custom provider and configure it as the audit destination, formatter, and filter.
- The only supported value for the filter is `com.sybase.security.core.DefaultAuditFilter`.
- The only supported value for the formatter is `com.sybase.security.core.XmlAuditFormatter`.

***com.sybase.security.core.FileAuditDestination***

The `com.sybase.security.core.FileAuditDestination` class contains the following configurable properties:

Marking an audit destination required or requisite means all operations being audited (authorization, authentication, role check, and so on) will fail if the event cannot successfully be logged to that audit destination.

**Table 16. controlFlag**

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                                                         |
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 17. implementationClass**

|                          |                                                            |
|--------------------------|------------------------------------------------------------|
| Datatype                 | String                                                     |
| Default                  | <code>com.sybase.security.core.FileAuditDestination</code> |
| Required?                | Yes                                                        |
| Requires server restart? | No                                                         |
| Read-only?               | No                                                         |

**Table 18. providerType**

|                          |                  |
|--------------------------|------------------|
| Datatype                 | String           |
| Default                  | AuditDestination |
| Required?                | Yes              |
| Requires server restart? | No               |
| Read-only?               | Yes              |

**Table 19. auditFile**

|                          |                                                                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String                                                                                                                                                                                  |
| Default                  | `\${djc.home}/logs/{security configuration name}-audit.log, where {security configuration name} is the name of the security configuration in which the audit destination is configured. |
| Required?                | Yes                                                                                                                                                                                     |
| Requires server restart? | No                                                                                                                                                                                      |
| Read-only?               | Yes                                                                                                                                                                                     |

**Table 20. compressionThreshold**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 21. deleteThreshold**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 22. encoding**

|          |        |
|----------|--------|
| Datatype | String |
| Default  | utf-8  |

|                          |    |
|--------------------------|----|
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 23. errorThreshold**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 24. logSize**

|                          |      |
|--------------------------|------|
| Datatype                 | long |
| Required?                | No   |
| Requires server restart? | No   |
| Read-only?               | No   |

*com.sybase.security.core.DefaultAuditFilter*

The `com.sybase.security.core.DefaultAuditFilter` class contains the following configurable properties:

**Table 25. implementationClass**

|                          |                                                          |
|--------------------------|----------------------------------------------------------|
| Datatype                 | String                                                   |
| Default                  | <code>com.sybase.security.core.DefaultAuditFilter</code> |
| Required?                | Yes                                                      |
| Requires server restart? | No                                                       |
| Read-only?               | No                                                       |

**Table 26. providerType**

|                          |                          |
|--------------------------|--------------------------|
| Datatype                 | String                   |
| Default                  | <code>AuditFilter</code> |
| Required?                | Yes                      |
| Requires server restart? | No                       |

|            |     |
|------------|-----|
| Read-only? | Yes |
|------------|-----|

**Table 27. caseSensitiveFiltering**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 28. filter**

|                          |                                                                                                                                                                                                                                                                                                        |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String                                                                                                                                                                                                                                                                                                 |
| Default                  | (ResourceClass=core.subject, Action=authorization.role) (ResourceClass=core.subject, Action=authorization.resource) (ResourceClass=core.subject, Action=authentication) (ResourceClass=core.subject, Action=logout) (ResourceClass=core.profile) (ResourceClass=providers.*) (ResourceClass=clients.*) |
| Required?                | No                                                                                                                                                                                                                                                                                                     |
| Requires server restart? | No                                                                                                                                                                                                                                                                                                     |
| Read-only?               | No                                                                                                                                                                                                                                                                                                     |

*com.sybase.security.core.XmlAuditFormatter*

The com.sybase.security.core.XmlAuditFormatter class contains the following configurable properties:

**Table 29. implementationClass**

|                          |                                            |
|--------------------------|--------------------------------------------|
| Datatype                 | String                                     |
| Default                  | com.sybase.security.core.XmlAuditFormatter |
| Required?                | Yes                                        |
| Requires server restart? | No                                         |
| Read-only?               | No                                         |

**Table 30. providerType**

|          |        |
|----------|--------|
| Datatype | String |
|----------|--------|



|                          |                |
|--------------------------|----------------|
| Default                  | AuditFormatter |
| Required?                | Yes            |
| Requires server restart? | No             |
| Read-only?               | Yes            |

### **Authentication Provider**

Supported authenticators.

- com.sybase.security.core.NoSecLoginModule
- com.sybase.security.core.CertificateValidationLoginModule
- com.sybase.security ldap.LDAPLoginModule
- com.sybase.security.os.NTPProxyLoginModule
- com.sybase.security.sap.SAPSSOTokenLoginModule
- com.sybase.security.core.CertificateAuthenticationLoginModule
- com.sybase.security.core.PreConfiguredUserLoginModule
- com.sybase.security.http.HttpAuthenticationLoginModule
- com.sybase.security.core.ClientValuePropagatingLoginModule
- com.sybase.security.radius.RadiusLoginModule

### **com.sybase.security.core.NoSecLoginModule**

The com.sybase.security.core.NoSecLoginModule package includes the following configurable properties:

**Table 31. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 32. controlFlag**

|          |                     |
|----------|---------------------|
| Datatype | String (enumerated) |
|----------|---------------------|

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 33. identity**

|                          |                |
|--------------------------|----------------|
| Datatype                 | String         |
| Default                  | nosec_identity |
| Required?                | No             |
| Requires server restart? | No             |
| Read-only?               | No             |

**Table 34. implementationClass**

|                          |                                           |
|--------------------------|-------------------------------------------|
| Datatype                 | String                                    |
| Default                  | com.sybase.security.core.NoSecLoginModule |
| Required?                | Yes                                       |
| Requires server restart? | No                                        |
| Read-only?               | No                                        |

**Table 35. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 36. storePass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 37. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 38. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 39. useUsernameAsIdentity**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | TRUE    |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

*com.sybase.security.core.CertificateValidationLoginModule*

The com.sybase.security.core.CertificateValidationLoginModule package contains the following configurable properties:

**Table 40. controlFlag**

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                                                         |
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 41. implementationClass**

|                          |                                                           |
|--------------------------|-----------------------------------------------------------|
| Datatype                 | String                                                    |
| Default                  | com.sybase.security.core.CertificateValidationLoginModule |
| Required?                | Yes                                                       |
| Requires server restart? | No                                                        |
| Read-only?               | No                                                        |

**Table 42. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 43. validatedCertificateIsIdentity**

|          |         |
|----------|---------|
| Datatype | boolean |
|----------|---------|

|                          |       |
|--------------------------|-------|
| Default                  | FALSE |
| Required?                | No    |
| Requires server restart? | No    |
| Read-only?               | No    |

**Table 44. enableRevocationChecking**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 45. trustedCertStore**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 46. trustedCertStorePassword**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 47. trustedCertStoreProvider**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 48. trustedCertStoreType**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 49. validateCertPath**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

*com.sybase.security.ldap.LDAPLoginModule*

The com.sybase.security.ldap.LDAPLoginModule package contains the following configurable properties:

**Table 50. AuthenticationFilter**

|                          |         |
|--------------------------|---------|
| Datatype                 | String. |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 51. AuthenticationMethod**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | simple |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 52. AuthenticationScope**

|          |                     |
|----------|---------------------|
| Datatype | String (enumerated) |
|----------|---------------------|

|                          |                                                                                 |
|--------------------------|---------------------------------------------------------------------------------|
| Allowable values         | <ul style="list-style-type: none"> <li>• onelevel</li> <li>• subtree</li> </ul> |
| Default                  | onelevel                                                                        |
| Required?                | No                                                                              |
| Requires server restart? | No                                                                              |
| Read-only?               | No                                                                              |

**Table 53. AuthenticationSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 54. BindDN**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 55. BindPassword**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |
| Encrypted?               | Yes    |

**Table 56. CertificateAuthenticationFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 57. ConnectTimeout**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Default                  | 0   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 58. DefaultSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 59. DigestMD5AuthenticationFormat**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 60. EnableLDAPConnectionTrace**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 61. InitialContextFactory**

|          |                                  |
|----------|----------------------------------|
| Datatype | String                           |
| Default  | com.sun.jndi.ldap.LdapCtxFactory |



|                          |    |
|--------------------------|----|
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 62. LDAPPoolMaxActive**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Default                  | 8   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 63. ProviderURL**

|                          |                      |
|--------------------------|----------------------|
| Datatype                 | String               |
| Default                  | ldap://localhost:389 |
| Required?                | Yes                  |
| Requires server restart? | No                   |
| Read-only?               | No                   |

**Table 64. ReadTimeout**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 65. Referral**

|                  |                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------|
| Datatype         | String (enumerated)                                                                           |
| Allowable values | <ul style="list-style-type: none"> <li>• ignore</li> <li>• follow</li> <li>• throw</li> </ul> |
| Default          | ignore                                                                                        |
| Required?        | No                                                                                            |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 66. RoleFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 67. RoleMemberAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 68. RoleNameAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | cn     |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 69. RoleScope**

|                          |                                                                                 |
|--------------------------|---------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                             |
| Allowable values         | <ul style="list-style-type: none"> <li>• onelevel</li> <li>• subtree</li> </ul> |
| Required?                | No                                                                              |
| Requires server restart? | No                                                                              |
| Read-only?               | No                                                                              |

**Table 70. RoleSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 71. SecurityProtocol**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 72. SelfRegistrationSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 73. SerializationKey**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 74. ServerType**

|                  |                                                                                                                    |
|------------------|--------------------------------------------------------------------------------------------------------------------|
| Datatype         | String (enumerated)                                                                                                |
| Allowable values | <ul style="list-style-type: none"> <li>• sunone5</li> <li>• msad2k</li> <li>• nsds4</li> <li>• openldap</li> </ul> |
| Required?        | No                                                                                                                 |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 75. SkipRoleLookup**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 76. UnmappedAttributePrefix**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | LDAP   |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 77. UseUserAccountControlAttribute**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 78. UserFreeformRoleMembershipAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 79. UserRoleMembershipAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 80. certificateAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 81. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 82. controlFlag**

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                                                         |
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 83. enableCertificateAuthentication**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 84. implementationClass**

|                          |                                          |
|--------------------------|------------------------------------------|
| Datatype                 | String                                   |
| Default                  | com.sybase.security.ldap.LDAPLoginModule |
| Required?                | Yes                                      |
| Requires server restart? | No                                       |
| Read-only?               | No                                       |

**Table 85. ldapAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 86. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 87. storePass**

|           |         |
|-----------|---------|
| Datatype  | boolean |
| Default   | FALSE   |
| Required? | No      |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 88. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 89. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

*com.sybase.security.os.NTProxyLoginModule*

The `com.sybase.security.os.NTProxyLoginModule` package contains the following configurable properties:

**Table 90. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 91. controlFlag**

|          |                     |
|----------|---------------------|
| Datatype | String (enumerated) |
|----------|---------------------|

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 92. defaultAuthenticationServer**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 93. defaultDomain**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 94. extractDomainFromUsername**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | TRUE    |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 95. failAuthenticationIfNoRoles**

|          |         |
|----------|---------|
| Datatype | boolean |
| Default  | FALSE   |



|                          |    |
|--------------------------|----|
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 96. implementationClass**

|                          |                                           |
|--------------------------|-------------------------------------------|
| Datatype                 | String                                    |
| Default                  | com.sybase.security.os.NTProxyLoginModule |
| Required?                | Yes                                       |
| Requires server restart? | No                                        |
| Read-only?               | No                                        |

**Table 97. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 98. storePass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 99. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 100. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

*com.sybase.security.sap.SAPSSOTokenLoginModule*

The SAPSSOTokenLoginModule has been deprecated, Use the HttpAuthenticationLoginModule when SAP SSO2 token authentication is required. This authentication module will be removed in a future release.

The com.sybase.security.sap.SAPSSOTokenLoginModule package contains the following configurable properties:

**Table 101. DisableServerCertificateValidation**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 102. SapSSOTokenPersistenceDataStore**

|                          |              |
|--------------------------|--------------|
| Datatype                 | String       |
| Default                  | jdbc/default |
| Required?                | No           |
| Requires server restart? | No           |
| Read-only?               | Yes          |

**Table 103. SapServerCertificate**

|           |        |
|-----------|--------|
| Datatype  | String |
| Required? | No     |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 104. SapServerCertificatePassword**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 105. SapServerURL**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 106. TokenExpirationInterval**

|                          |      |
|--------------------------|------|
| Datatype                 | long |
| Default                  | 120  |
| Required?                | No   |
| Requires server restart? | No   |
| Read-only?               | No   |

**Table 107. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 108. controlFlag**

|          |                     |
|----------|---------------------|
| Datatype | String (enumerated) |
|----------|---------------------|

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 109. implementationClass**

|                          |                                                |
|--------------------------|------------------------------------------------|
| Datatype                 | String (enumerated)                            |
| Default                  | com.sybase.security.sap.SAPSSOTokenLoginModule |
| Required?                | Yes                                            |
| Requires server restart? | No                                             |
| Read-only?               | No                                             |

**Table 110. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 111. storePass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 112. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 113. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

*com.sybase.security.core.CertificateAuthenticationLoginModule*

The `com.sybase.security.core.CertificateAuthenticationLoginModule` package contains the following configurable properties:

**Table 114. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 115. controlFlag**

|                  |                                                                                                                             |
|------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Datatype         | String (enumerated)                                                                                                         |
| Allowable values | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default          | optional                                                                                                                    |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 116. enableRevocationChecking**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 117. implementationClass**

|                          |                                                               |
|--------------------------|---------------------------------------------------------------|
| Datatype                 | String (enumerated)                                           |
| Default                  | com.sybase.security.core.CertificateAuthenticationLoginModule |
| Required?                | Yes                                                           |
| Requires server restart? | No                                                            |
| Read-only?               | No                                                            |

**Table 118. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 119. storePass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 120. trustedCertStore**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 121. trustedCertStorePassword**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 122. trustedCertStoreProvider**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 123. trustedCertStoreType**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 124. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 125. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 126. validateCertPath**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

*com.sybase.security.core.PreConfiguredUserLoginModule*

The com.sybase.security.core.PreConfiguredUserLoginModule package contains the following configurable properties:

**Table 127. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 128. controlFlag**

|          |                     |
|----------|---------------------|
| Datatype | String (enumerated) |
|----------|---------------------|



|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 129. implementationClass**

|                          |                                                       |
|--------------------------|-------------------------------------------------------|
| Datatype                 | String (enumerated)                                   |
| Default                  | com.sybase.security.core.PreConfiguredUserLoginModule |
| Required?                | Yes                                                   |
| Requires server restart? | No                                                    |
| Read-only?               | No                                                    |

**Table 130. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 131. storePass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 132. trustedCertStore**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 133. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 134. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 135. username**

|                          |                                         |
|--------------------------|-----------------------------------------|
| Datatype                 | String. Cannot contain , = : ' " * ? &. |
| Default                  | supAdmin                                |
| Required?                | Yes                                     |
| Requires server restart? | No                                      |
| Read-only?               | No                                      |

**Table 136. Password**

|           |        |
|-----------|--------|
| Datatype  | String |
| Default   | ""     |
| Required? | Yes    |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 137. roles**

|                          |                   |
|--------------------------|-------------------|
| Datatype                 | String            |
| Default                  | SUP Administrator |
| Required?                | No                |
| Requires server restart? | No                |
| Read-only?               | No                |

*com.sybase.security.http.HttpAuthenticationLoginModule*

The `com.sybase.security.http.HttpAuthenticationLoginModule` package contains the following configurable properties:

**Table 138. implementationClass**

|                          |                                                                     |
|--------------------------|---------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                 |
| Default                  | <code>com.sybase.security.http.HttpAuthenticationLoginModule</code> |
| Required?                | Yes                                                                 |
| Requires server restart? | No                                                                  |
| Read-only?               | No                                                                  |

**Table 139. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |

**Table 140. controlFlag**

|          |                     |
|----------|---------------------|
| Datatype | String (enumerated) |
|----------|---------------------|

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 141. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 142. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 143. storePass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 144. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 145. URL**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 146. DisableServerCertificateValidation**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 147. RolesHTTPHeader**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 148. SSOCookieNames**

|          |        |
|----------|--------|
| Datatype | String |
| Default  | None   |

|                          |    |
|--------------------------|----|
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 149. SuccessfulConnectionStatusCode**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 200 |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 150. CredentialName**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 151. SendPasswordAsCookie**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 152. ClientHttpValuesToSend**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 153. SendClientHttpValueAs**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 154. UsernameHttpHeader**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 155. TokenExpirationTimeHttpHeader**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 156. RegexForUsernameMatch**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 157. TryBasicAuthIfTokenAuthFails**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 158. HttpConnectionTimeoutInterval**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | 60000 |
| Required?                | No    |
| Requires server restart? | No    |
| Read-only?               | No    |

**Table 159. TokenExpirationallInterval**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 0   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

*com.sybase.security.radius.RadiusLoginModule*

The `com.sybase.security.radius.RadiusLoginModule` package contains the following configurable properties:

**Table 160. implementationClass**

|                          |                                                           |
|--------------------------|-----------------------------------------------------------|
| Datatype                 | String                                                    |
| Default                  | <code>com.sybase.security.radius.RadiusLoginModule</code> |
| Required?                | Yes                                                       |
| Requires server restart? | No                                                        |
| Read-only?               | No                                                        |



|            |    |
|------------|----|
| Encrypted? | No |
|------------|----|

**Table 161. providerType**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | LoginModule |
| Required?                | Yes         |
| Requires server restart? | No          |
| Read-only?               | Yes         |
| Encrypted?               | No          |

**Table 162. controlFlag**

|                          |                                                                                                                             |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                                                         |
| Allowable values         | <ul style="list-style-type: none"> <li>• optional</li> <li>• sufficient</li> <li>• required</li> <li>• requisite</li> </ul> |
| Default                  | optional                                                                                                                    |
| Required?                | Yes                                                                                                                         |
| Requires server restart? | No                                                                                                                          |
| Read-only?               | No                                                                                                                          |

**Table 163. AuthenticationMethod**

|                          |                                                                         |
|--------------------------|-------------------------------------------------------------------------|
| Datatype                 | String                                                                  |
| Allowable values         | <ul style="list-style-type: none"> <li>• PAP</li> <li>• CHAP</li> </ul> |
| Default                  | PAP                                                                     |
| Required?                | No                                                                      |
| Requires server restart? | No                                                                      |
| Read-only?               | No                                                                      |
| Encrypted?               | No                                                                      |

**Table 164. SharedSecret**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |
| Encrypted?               | Yes    |

**Table 165. RadiusServerHostName**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | Yes    |
| Requires server restart? | No     |
| Read-only?               | No     |
| Encrypted?               | No     |

**Table 166. RadiusServerAuthPort**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | 1812 |
| Required?                | Yes  |
| Requires server restart? | No   |
| Read-only?               | No   |
| Encrypted?               | No   |

**Table 167. RadiusServerAuthPort**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 3   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

|            |    |
|------------|----|
| Encrypted? | No |
|------------|----|

**Table 168. caseSensitiveMatching**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |
| Encrypted?               | No      |

**Table 169. useFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |
| Encrypted?               | No      |

**Table 170. tryFirstPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |
| Encrypted?               | No      |

**Table 171. storePass**

|           |         |
|-----------|---------|
| Datatype  | boolean |
| Default   | FALSE   |
| Required? | No      |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |
| Encrypted?               | No |

**Table 172. clearPass**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |
| Encrypted?               | No      |

**Authorization Provider**

Supported authorizers.

- com.sybase.security.core.NoSecAuthorizer
- com.sybase.security.ldap.LDAPAuthorizer

**com.sybase.security.core.NoSecAuthorizer**

The com.sybase.security.core.NoSecAuthorizer package contains the following configurable properties:

**Table 173. implementationClass**

|                          |                                          |
|--------------------------|------------------------------------------|
| Datatype                 | String                                   |
| Default                  | com.sybase.security.core.NoSecAuthorizer |
| Required?                | Yes                                      |
| Requires server restart? | No                                       |
| Read-only?               | No                                       |

**Table 174. providerType**

|           |            |
|-----------|------------|
| Datatype  | String     |
| Default   | Authorizer |
| Required? | Yes        |

|                          |     |
|--------------------------|-----|
| Requires server restart? | No  |
| Read-only?               | Yes |

*com.sybase.security.ldap.LDAPAuthorizer*

The `com.sybase.security.ldap.LDAPAuthorizer` package contains the following configurable properties:

**Table 175. AuthenticationFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 176. AuthenticationMethod**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | simple |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 177. AuthenticationScope**

|                          |                                                                                 |
|--------------------------|---------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                             |
| Allowable values         | <ul style="list-style-type: none"> <li>• onelevel</li> <li>• subtree</li> </ul> |
| Default                  | onelevel                                                                        |
| Required?                | No                                                                              |
| Requires server restart? | No                                                                              |
| Read-only?               | No                                                                              |

**Table 178. AuthenticationSearchBase**

|           |        |
|-----------|--------|
| Datatype  | String |
| Required? | No     |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 179. BindDN**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 180. BindPassword**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 181. CertificateAuthenticationFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 182. ConnectTimeout**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Default                  | 0   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 183. DefaultSearchBase**

|           |                                         |
|-----------|-----------------------------------------|
| Datatype  | String                                  |
| Required? | Yes (if RoleSearchBase is not provided) |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 184. DigestMD5AuthenticationFormat**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 185. EnableLDAPConnectionTrace**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 186. InitialContextFactory**

|                          |                                  |
|--------------------------|----------------------------------|
| Datatype                 | String                           |
| Default                  | com.sun.jndi.ldap.LdapCtxFactory |
| Required?                | No                               |
| Requires server restart? | No                               |
| Read-only?               | No                               |

**Table 187. LDAPPoolMaxActive**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Default                  | 8   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 188. ProviderURL**

|                          |                      |
|--------------------------|----------------------|
| Datatype                 | String               |
| Default                  | ldap://localhost:389 |
| Required?                | Yes                  |
| Requires server restart? | No                   |
| Read-only?               | No                   |

**Table 189. ReadTimeout**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 190. Referral**

|                          |                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                           |
| Allowable values         | <ul style="list-style-type: none"> <li>• ignore</li> <li>• follow</li> <li>• throw</li> </ul> |
| Default                  | ignore                                                                                        |
| Required?                | No                                                                                            |
| Requires server restart? | No                                                                                            |
| Read-only?               | No                                                                                            |

**Table 191. RoleFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 192. RoleMemberAttributes**

|          |        |
|----------|--------|
| Datatype | String |
|----------|--------|



|                          |    |
|--------------------------|----|
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 193. RoleNameAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | cn     |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 194. RoleScope**

|                          |                                                                                 |
|--------------------------|---------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                             |
| Allowable values         | <ul style="list-style-type: none"> <li>• onelevel</li> <li>• subtree</li> </ul> |
| Required?                | No                                                                              |
| Requires server restart? | No                                                                              |
| Read-only?               | No                                                                              |

**Table 195. RoleSearchBase**

|                          |                                            |
|--------------------------|--------------------------------------------|
| Datatype                 | String                                     |
| Required?                | Yes (if DefaultSearchBase is not provided) |
| Requires server restart? | No                                         |
| Read-only?               | No                                         |

**Table 196. SecurityProtocol**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 197. SelfRegistrationSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 198. ServerType**

|                          |                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                                                |
| Allowable values         | <ul style="list-style-type: none"> <li>• sunone5</li> <li>• msad2k</li> <li>• nsds4</li> <li>• openldap</li> </ul> |
| Required?                | No                                                                                                                 |
| Requires server restart? | No                                                                                                                 |
| Read-only?               | No                                                                                                                 |

**Table 199. SkipRoleLookup**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 200. UnmappedAttributePrefix**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | LDAP   |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 201. UseUserAccountControlAttribute**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 202. UserFreeformRoleMembershipAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 203. UserRoleMembershipAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 204. certificateAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 205. enableCertificateAuthentication**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 206. implementationClass**

|                          |                                         |
|--------------------------|-----------------------------------------|
| Datatype                 | String                                  |
| Default                  | com.sybase.security.ldap.LDAPAuthorizer |
| Required?                | Yes                                     |
| Requires server restart? | No                                      |
| Read-only?               | No                                      |

**Table 207. ldapAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 208. providerType**

|                          |            |
|--------------------------|------------|
| Datatype                 | String     |
| Default                  | Authorizer |
| Required?                | Yes        |
| Requires server restart? | No         |
| Read-only?               | Yes        |

**Attribution Provider**

Supported attributions.

- com.sybase.security.core.NoSecAttributer
- com.sybase.security.ldap.LDAPAttributer

com.sybase.security.core.NoSecAttributer

The com.sybase.security.core.NoSecAttributer package contains the following configurable properties:

**Table 209. implementationClass**

|          |                                          |
|----------|------------------------------------------|
| Datatype | String                                   |
| Default  | com.sybase.security.core.NoSecAttributer |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 210. providerType**

|                          |            |
|--------------------------|------------|
| Datatype                 | String     |
| Default                  | Attributer |
| Required?                | Yes        |
| Requires server restart? | No         |
| Read-only?               | Yes        |

*com.sybase.security.ldap.LDAPAttributer*

The com.sybase.security.ldap.LDAPAttributer package contains the following configurable properties:

**Table 211. AuthenticationFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 212. AuthenticationMethod**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | simple |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 213. AuthenticationScope**

|                  |                                                                                 |
|------------------|---------------------------------------------------------------------------------|
| Datatype         | String (enumerated)                                                             |
| Allowable values | <ul style="list-style-type: none"> <li>• onelevel</li> <li>• subtree</li> </ul> |

|                          |          |
|--------------------------|----------|
| Default                  | onelevel |
| Required?                | No       |
| Requires server restart? | No       |
| Read-only?               | No       |

**Table 214. AuthenticationSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 215. BindDN**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 216. BindPassword**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 217. CertificateAuthenticationFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 218. ConnectTimeout**

|          |     |
|----------|-----|
| Datatype | Int |
|----------|-----|

|                          |    |
|--------------------------|----|
| Default                  | 0  |
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 219. DefaultSearchBase**

|                          |                                         |
|--------------------------|-----------------------------------------|
| Datatype                 | String                                  |
| Required?                | Yes (if RoleSearchBase is not provided) |
| Requires server restart? | No                                      |
| Read-only?               | No                                      |

**Table 220. DigestMD5AuthenticationFormat**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 221. EnableLDAPConnectionTrace**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 222. InitialContextFactory**

|                          |                                  |
|--------------------------|----------------------------------|
| Datatype                 | String                           |
| Default                  | com.sun.jndi.ldap.LdapCtxFactory |
| Required?                | No                               |
| Requires server restart? | No                               |
| Read-only?               | No                               |

**Table 223. LDAPPoolMaxActive**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Default                  | 8   |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 224. ProviderURL**

|                          |                      |
|--------------------------|----------------------|
| Datatype                 | String               |
| Default                  | ldap://localhost:389 |
| Required?                | Yes                  |
| Requires server restart? | No                   |
| Read-only?               | No                   |

**Table 225. ReadTimeout**

|                          |     |
|--------------------------|-----|
| Datatype                 | Int |
| Required?                | No  |
| Requires server restart? | No  |
| Read-only?               | No  |

**Table 226. Referral**

|                          |                                                                                               |
|--------------------------|-----------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                           |
| Allowable values         | <ul style="list-style-type: none"> <li>• ignore</li> <li>• follow</li> <li>• throw</li> </ul> |
| Default                  | ignore                                                                                        |
| Required?                | No                                                                                            |
| Requires server restart? | No                                                                                            |
| Read-only?               | No                                                                                            |



**Table 227. RoleFilter**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 228. RoleMemberAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 229. RoleNameAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | cn     |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 230. RoleScope**

|                          |                                                                                 |
|--------------------------|---------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                             |
| Allowable values         | <ul style="list-style-type: none"> <li>• onelevel</li> <li>• subtree</li> </ul> |
| Required?                | No                                                                              |
| Requires server restart? | No                                                                              |
| Read-only?               | No                                                                              |

**Table 231. RoleSearchBase**

|           |                                            |
|-----------|--------------------------------------------|
| Datatype  | String                                     |
| Required? | Yes (if DefaultSearchBase is not provided) |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 232. SecurityProtocol**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 233. SelfRegistrationSearchBase**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 234. ServerType**

|                          |                                                                                                                    |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String (enumerated)                                                                                                |
| Allowable values         | <ul style="list-style-type: none"> <li>• sunone5</li> <li>• msad2k</li> <li>• nsds4</li> <li>• openldap</li> </ul> |
| Required?                | No                                                                                                                 |
| Requires server restart? | No                                                                                                                 |
| Read-only?               | No                                                                                                                 |

**Table 235. SkipRoleLookup**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | False   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 236. UnmappedAttributePrefix**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | LDAP   |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 237. UseUserAccountControlAttribute**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 238. UserFreeformRoleMembershipAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 239. UserRoleMembershipAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 240. certificateAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 241. enableCertificateAuthentication**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | FALSE   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 242. implementationClass**

|                          |                                         |
|--------------------------|-----------------------------------------|
| Datatype                 | String                                  |
| Default                  | com.sybase.security.ldap.LDAPAttributer |
| Required?                | Yes                                     |
| Requires server restart? | No                                      |
| Read-only?               | No                                      |

**Table 243. IdapAttributes**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 244. providerType**

|                          |            |
|--------------------------|------------|
| Datatype                 | String     |
| Default                  | Attributer |
| Required?                | Yes        |
| Requires server restart? | No         |
| Read-only?               | Yes        |

## **Cluster Configuration**

You can configure the following cluster components through metadata.

- ReplicationSyncserver
- MessagingSyncserver

- AdministrationListener
- SecureAdministrationListener
- HTTPListener
- SecureHTTPListener
- SSLSecurityProfile
- KeyStore
- TrustStore
- OCSP
- SolutionManager
- DCN Performance
- WebContainer
- ConfigurationCache

### **ReplicationSyncServer**

The `ReplicationSyncServer` component contains the following configurable properties:

**Table 245. ml.cachesize**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | 50M    |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 246. ml.threadcount**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 5   |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 247. sup.sync.certificate**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | Yes    |
| Requires server restart? | Yes    |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 248. sup.sync.certificate\_password**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 249. sup.sync.httpsport**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | 2481 |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 250. sup.sync.port**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | 2480 |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 251. sup.sync.protocol**

|                          |                                                                           |
|--------------------------|---------------------------------------------------------------------------|
| Datatype                 | String                                                                    |
| Allowable values         | <ul style="list-style-type: none"> <li>• http</li> <li>• https</li> </ul> |
| Default                  | http                                                                      |
| Required?                | Yes                                                                       |
| Requires server restart? | Yes                                                                       |
| Read-only?               | No                                                                        |

**Table 252. sup.user.options**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 253. sup.sync.e2ee\_type**

|                          |                                                                                  |
|--------------------------|----------------------------------------------------------------------------------|
| Datatype                 | String                                                                           |
| Allowable values         | <ul style="list-style-type: none"> <li>• rsa</li> <li>• &lt;empty&gt;</li> </ul> |
| Required?                | No                                                                               |
| Requires server restart? | Yes                                                                              |
| Read-only?               | No                                                                               |

**Table 254. sup.sync.e2ee\_private\_key**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 255. sup.sync.e2ee\_private\_key\_password**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**MessagingSyncServer**

The `MessagingSyncServer` component contains the following configurable properties.

**Table 256. msg.http.server.proxy.ports**

|          |        |
|----------|--------|
| Datatype | String |
| Default  | 5001   |

|                          |     |
|--------------------------|-----|
| Required?                | No  |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 257. sup.msg.outbound\_count**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 5   |
| Required?                | No  |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 258. sup.msg.outbound\_queue\_prefix**

|                          |               |
|--------------------------|---------------|
| Datatype                 | String        |
| Default                  | sup.mbs.moca. |
| Required?                | No            |
| Requires server restart? | Yes           |
| Read-only?               | No            |

### **AdministrationListener**

The `AdministrationListener` component contains the following configurable properties:

**Table 259. sup.admin.port**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | null |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |



**SecureAdministrationListener**

The `SecureAdministrationListener` component contains the following configurable properties:

**Table 260. sup.admin.iiopsport**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | null |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 261. sup.admin.iiops.profile**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | iiop   |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**HTTPListener**

The `HTTPListener` component contains the following configurable properties:

**Table 262. sup.socket.listener.enabled**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | TRUE    |
| Required?                | Yes     |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 263. sup.socket.listener.port**

|          |      |
|----------|------|
| Datatype | int  |
| Default  | null |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 264. sup.socket.listener.protocol**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | iiop   |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 265. sup.socket.listener.maxthreads**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 30  |
| Required?                | No  |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 266. sup.socket.listener.maxIdleTime**

|                          |         |
|--------------------------|---------|
| Datatype                 | int     |
| Default                  | 3600000 |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 267. sup.socket.listener.numberOfAcceptors**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 3   |
| Required?                | No  |
| Requires server restart? | Yes |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 268. sup.socket.listener.lowResourcesConnections**

|                          |         |
|--------------------------|---------|
| Datatype                 | int     |
| Default                  | 3600000 |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 269. sup.socket.listener.includeServerInfoInSession**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | FALSE |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**Table 270. sup.socket.listener.headerBufferSize**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | 8092    |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 271. sup.socket.listener.soLingerTime**

|                          |      |
|--------------------------|------|
| Datatype                 | Int  |
| Default                  | 1000 |
| Required?                | No   |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 272. sup.socket.listener.acceptorPriority**

|                          |        |
|--------------------------|--------|
| Datatype                 | string |
| Default                  | 0      |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 273. sup.socket.listener.securityProfile**

|                          |        |
|--------------------------|--------|
| Datatype                 | string |
| Default                  | null   |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 274. sup.socket.listener.responseBufferSize**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | 65536 |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**Table 275. sup.socket.listener.statsOn**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | FALSE |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**Table 276. sup.socket.listener.listenBacklog**

|          |         |
|----------|---------|
| Datatype | boolean |
| Default  | 50      |

|                          |     |
|--------------------------|-----|
| Required?                | No  |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 277. sup.socket.listener.requestBufferSize**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | 32768 |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**SecureHTTPListener**

The `SecureHTTPListener` component contains the following configurable properties:

**Table 278. sup.socket.listener.enabled**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | TRUE    |
| Required?                | Yes     |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 279. sup.socket.listener.port**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | null |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 280. sup.socket.listener.protocol**

|          |        |
|----------|--------|
| Datatype | String |
| Default  | iiop   |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 281. sup.socket.listener.security.profile**

|                          |         |
|--------------------------|---------|
| Datatype                 | String  |
| Default                  | default |
| Required?                | Yes     |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 282. sup.socket.listener.maxthreads**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 30  |
| Required?                | No  |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 283. sup.socket.listener.maxIdleTime**

|                          |         |
|--------------------------|---------|
| Datatype                 | int     |
| Default                  | 3600000 |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 284. sup.socket.listener.numberOfAcceptors**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 3   |
| Required?                | No  |
| Requires server restart? | Yes |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 285. sup.socket.listener.lowResourcesConnections**

|                          |         |
|--------------------------|---------|
| Datatype                 | int     |
| Default                  | 3600000 |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 286. sup.socket.listener.includeServerInfoInSession**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | FALSE |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**Table 287. sup.socket.listener.headerBufferSize**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | 8092    |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 288. sup.socket.listener.soLingerTime**

|                          |      |
|--------------------------|------|
| Datatype                 | Int  |
| Default                  | 1000 |
| Required?                | No   |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 289. sup.socket.listener.acceptorPriority**

|                          |        |
|--------------------------|--------|
| Datatype                 | string |
| Default                  | 0      |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 290. sup.socket.listener.securityProfile**

|                          |        |
|--------------------------|--------|
| Datatype                 | string |
| Default                  | null   |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 291. sup.socket.listener.responseBufferSize**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | 65536 |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**Table 292. sup.socket.listener.statsOn**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | FALSE |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**Table 293. sup.socket.listener.listenBacklog**

|          |         |
|----------|---------|
| Datatype | boolean |
| Default  | 50      |



|                          |     |
|--------------------------|-----|
| Required?                | No  |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 294. sup.socket.listener.requestBufferSize**

|                          |       |
|--------------------------|-------|
| Datatype                 | int   |
| Default                  | 32768 |
| Required?                | No    |
| Requires server restart? | Yes   |
| Read-only?               | No    |

**SSLSecurityProfile**

The `SSLSecurityProfile` component contains the following configurable properties:

**Table 295. securityCharacteristics**

|                          |                                                                                                                                                                           |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Datatype                 | String                                                                                                                                                                    |
| Allowable values         | <ul style="list-style-type: none"> <li>• intl</li> <li>• intl_mutual</li> <li>• strong</li> <li>• strong_mutual</li> <li>• domestic</li> <li>• domestic_mutual</li> </ul> |
| Default                  | intl                                                                                                                                                                      |
| Required?                | Yes                                                                                                                                                                       |
| Requires server restart? | Yes                                                                                                                                                                       |
| Read-only?               | No                                                                                                                                                                        |

**Table 296. certificateLabel**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | null   |
| Required?                | Yes    |
| Requires server restart? | Yes    |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 297. sup.security.profile.name**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | null   |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | Yes    |

**KeyStore**

The `KeyStore` component contains the following configurable properties:

**Table 298. sup.sync.sslkeystore**

|                          |                                  |
|--------------------------|----------------------------------|
| Datatype                 | String                           |
| Default                  | Repository/Security/keystore.jks |
| Required?                | Yes                              |
| Requires server restart? | Yes                              |
| Read-only?               | Yes                              |

**Table 299. sup.sync.sslkeystore\_password**

|                          |          |
|--------------------------|----------|
| Datatype                 | String   |
| Default                  | changeit |
| Required?                | Yes      |
| Requires server restart? | Yes      |
| Read-only?               | Yes      |

**TrustStore**

The `TrustStore` component contains the following configurable properties:

**Table 300. sup.sync.ssltruststore**

|           |                                    |
|-----------|------------------------------------|
| Datatype  | String                             |
| Default   | Repository/Security/truststore.jks |
| Required? | Yes                                |

|                          |     |
|--------------------------|-----|
| Requires server restart? | Yes |
| Read-only?               | Yes |

**Table 301. sup.sync.ssltruststore\_password**

|                          |          |
|--------------------------|----------|
| Datatype                 | String   |
| Default                  | changeit |
| Required?                | Yes      |
| Requires server restart? | Yes      |
| Read-only?               | Yes      |

**OCSP**

The OCSP (Online Certificate Status Protocol) component contains the following configurable properties:

**Table 302. ocsf.enable**

|                          |         |
|--------------------------|---------|
| Datatype                 | Boolean |
| Default                  | False   |
| Required?                | No      |
| Requires server restart? | No      |
| Read-only?               | No      |

**Table 303. ocsf.responderURL**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 304. ocsf.responderCertIssuerName**

|           |        |
|-----------|--------|
| Datatype  | String |
| Default   | None   |
| Required? | No     |

|                          |    |
|--------------------------|----|
| Requires server restart? | No |
| Read-only?               | No |

**Table 305. ocsf.responderCertSerialNumber**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**Table 306. ocsf.responderCertSubjectName**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | None   |
| Required?                | No     |
| Requires server restart? | No     |
| Read-only?               | No     |

**SolutionManager**

The `SolutionManager` component contains the following configurable properties.

**Table 307. com.sap.solutionmanager.url**

|                          |        |
|--------------------------|--------|
| Datatype                 | string |
| Default                  |        |
| Required?                | Yes    |
| Requires Server Restart? | Yes    |
| Read-only?               | No     |

**DCN**

The `DCN` component contains the following configurable properties.

**Table 308. sup.dcn.http.get.enabled**

|          |         |
|----------|---------|
| Datatype | boolean |
| Default  | False   |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

### **Performance**

The `Performance` component contains the following configurable properties.

**Table 309. sup.msg.inbound\_count**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 25  |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 310. sup.msg.outbound\_count**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 5   |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 311. sup.msg.subscribe.receiver.count**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 5   |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 312. sup.msg.proxy.connections.pool.max**

|          |      |
|----------|------|
| Datatype | int  |
| Default  | 1000 |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 313. ml.threadcount**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 10  |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 314. ml.cachesize**

|                          |        |
|--------------------------|--------|
| Datatype                 | string |
| Default                  | 70p    |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only                | No     |

**Table 315. webservices.connections.max.total**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 250 |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 316. webservices.connections.max.per.host**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 50  |
| Required?                | Yes |
| Requires server restart? | Yes |

|            |    |
|------------|----|
| Read-only? | No |
|------------|----|

**Table 317. webservices.connection.manager.timeout**

|                          |        |
|--------------------------|--------|
| Datatype                 | long   |
| Default                  | 180000 |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**WebContainer**

The `WebContainer` component contains the following configurable properties.

**Table 318. gzipFilter**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | False   |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 319. maxFormContentSize**

|                          |          |
|--------------------------|----------|
| Datatype                 | string   |
| Default                  | 10000000 |
| Required?                | No       |
| Requires server restart? | Yes      |
| Read-only?               | No       |

**ConfigurationCache**

The `ConfigurationCache` component contains the following configurable properties.

**Table 320. cache.core.pool.size**

|           |     |
|-----------|-----|
| Datatype  | int |
| Default   | 40  |
| Required? | Yes |

|                          |     |
|--------------------------|-----|
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 321. cache.max.pool.size**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 40  |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 322. cache.keep.alive.time**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 300 |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 323. cache.network.encrypt**

|                          |         |
|--------------------------|---------|
| Datatype                 | boolean |
| Default                  | False   |
| Required?                | Yes     |
| Requires server restart? | Yes     |
| Read-only?               | No      |

**Table 324. cache.port.autoinc**

|                          |         |
|--------------------------|---------|
| Datatypes                | boolean |
| Default                  | False   |
| Required?                | Yes     |
| Requires server restart? | Yes     |
| Read-only?               | No      |



**Table 325. cache.port**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | 5701 |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |

## **Server Configuration**

You can configure the following components through metadata:

- ConsolidatedDB
- JVM

---

**Note:** Properties you configure for an SAP Mobile Server are cluster-affecting. Therefore, to make sure they are propagated correctly, SAP recommends that you set them only on a primary cluster server.

---

### **ConsolidatedDB**

The ConsolidatedDB component contains the following configurable properties:

**Table 326. cdb.asa.mode**

|                          |         |
|--------------------------|---------|
| Datatype                 | String  |
| Default                  | primary |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | Yes     |

**Table 327. cdb.databasesname**

|                          |         |
|--------------------------|---------|
| Datatype                 | String  |
| Default                  | default |
| Required?                | Yes     |
| Requires server restart? | Yes     |
| Read-only?               | Yes     |

**Table 328. cdb.dnsname**

|                          |             |
|--------------------------|-------------|
| Datatype                 | String      |
| Default                  | default-cdb |
| Required?                | Yes         |
| Requires server restart? | Yes         |
| Read-only?               | Yes         |

**Table 329. cdb.install\_type**

|                          |         |
|--------------------------|---------|
| Datatype                 | String  |
| Default                  | default |
| Required?                | No      |
| Requires server restart? | Yes     |
| Read-only?               | Yes     |

**Table 330. cdb.password**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | sql    |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 331. cdb.serverhost**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | gma    |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | Yes    |

**Table 332. cdb.servername**

|          |             |
|----------|-------------|
| Datatype | String      |
| Default  | gma_primary |

|                          |     |
|--------------------------|-----|
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | Yes |

**Table 333. cdb.serverport**

|                          |      |
|--------------------------|------|
| Datatype                 | int  |
| Default                  | 5200 |
| Required?                | Yes  |
| Requires server restart? | Yes  |
| Read-only?               | No   |

**Table 334. cdb.threadcount**

|                          |     |
|--------------------------|-----|
| Datatype                 | int |
| Default                  | 20  |
| Required?                | Yes |
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 335. cdb.type**

|                          |                                                                                      |
|--------------------------|--------------------------------------------------------------------------------------|
| Datatype                 | String                                                                               |
| Allowable values         | <ul style="list-style-type: none"> <li>• Sybase_ASA</li> <li>• Sybase_ASE</li> </ul> |
| Default                  | Sybase_ASA                                                                           |
| Required?                | Yes                                                                                  |
| Requires server restart? | Yes                                                                                  |
| Read-only?               | Yes                                                                                  |

**Table 336. cdb.user.options**

|           |        |
|-----------|--------|
| Datatype  | String |
| Required? | Yes    |

|                          |     |
|--------------------------|-----|
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 337. cdb.username**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | dba    |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**JVM**

The JVM component contains the following configurable properties:

**Table 338. DJC\_JVM\_MINHEAP**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | 64M    |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 339. DJC\_JVM\_MAXHEAP**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | 256M   |
| Required?                | Yes    |
| Requires server restart? | Yes    |
| Read-only?               | No     |

**Table 340. DJC\_JVM\_STACKSIZE**

|           |        |
|-----------|--------|
| Datatype  | String |
| Default   | 400K   |
| Required? | Yes    |

|                          |     |
|--------------------------|-----|
| Requires server restart? | Yes |
| Read-only?               | No  |

**Table 341. DJC\_JVM\_USEROPTIONS**

|                          |        |
|--------------------------|--------|
| Datatype                 | String |
| Default                  | ""     |
| Required?                | No     |
| Requires server restart? | Yes    |
| Read-only?               | No     |

## **Server Log Configuration**

You can perform log configuration through the `LocalFileAppender` log appenders. The log appender can contain one or more of the following log buckets:

- MSG
- Trace
- MMS
- Security
- Mobilink
- DataServices
- Proxy
- Other

### **LocalFileAppender**

The `LocalFileAppender` log appender contains the following configurable properties:

**Table 342. LogLevel**

|                  |                                                                                                                                            |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Datatype         | String                                                                                                                                     |
| Allowable values | <ul style="list-style-type: none"> <li>• TRACE</li> <li>• DEBUG</li> <li>• INFO</li> <li>• WARN</li> <li>• ERROR</li> <li>• OFF</li> </ul> |
| Default          | WARN                                                                                                                                       |

|                          |    |
|--------------------------|----|
| Required?                | No |
| Requires server restart? | No |
| Read-only?               | No |

**Table 343. async**

|                           |         |
|---------------------------|---------|
| Datatype                  | boolean |
| Default                   | FALSE   |
| Required?                 | No      |
| Requires Server Re-start? | No      |
| Read Only?                | No      |

**Table 344. dateRollover**

|                           |                                                                                                                                                    |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| Datatype                  | String                                                                                                                                             |
| Allowable Values          | <ul style="list-style-type: none"> <li>• NONE</li> <li>• HOURLY</li> <li>• DAILY</li> <li>• WEEKLY</li> <li>• MONTHLY</li> <li>• YEARLY</li> </ul> |
| Default                   | NONE                                                                                                                                               |
| Required?                 | No                                                                                                                                                 |
| Requires Server Re-start? | No                                                                                                                                                 |
| Read Only?                | No                                                                                                                                                 |

**Table 345. filename**

|                           |        |
|---------------------------|--------|
| Datatype                  | String |
| Default                   | null   |
| Required?                 | Yes    |
| Requires Server Re-start? | No     |

|            |    |
|------------|----|
| Read Only? | No |
|------------|----|

**Table 346. maximumRolloverFiles**

|                           |     |
|---------------------------|-----|
| Datatype                  | int |
| Default                   | 1   |
| Required?                 | No  |
| Requires Server Re-start? | No  |
| Read Only?                | No  |

**Table 347. sizeRollover**

|                           |        |
|---------------------------|--------|
| Datatype                  | String |
| Default                   | 10mb   |
| Required?                 | No     |
| Requires Server Re-start? | No     |
| Read Only?                | No     |

## Property Reference

---

Review properties of the Management API.

### Application Connection Properties

Application Connection properties fall into various categories.

- Apple Push Notifications
- Application Settings
- BlackBerry Push Notifications
- Connection
- Custom Settings
- Device Advanced
- Device Info
- Proxy
- Security Settings
- User registration

**Apple Push Notification Properties**

Apple Push Notification properties allow iPhone users to install messaging client software on their devices. This requires you to create different e-mail activation messages using the appropriate push notification properties.

| <b>ID: property name (type)</b>  | <b>Description</b>                                                                                                                                                                                                                                                                                                            | <b>Default</b>      |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| 2600: Enable (boolean)           | Enables if push notification using APNs is enabled or not.                                                                                                                                                                                                                                                                    | True                |
| 2601: Alert (boolean)            | Use the iOS standard alert.                                                                                                                                                                                                                                                                                                   | True                |
| 2602: Badges (boolean)           | Use the badge of the application icon.                                                                                                                                                                                                                                                                                        | True                |
| 2603: Sounds (boolean)           | Use a if a sound is a made when a notification is received. The sound files must reside in the main bundle of the client application. Because custom alert sounds are played by the iOS system-sound facility, they must be in one of the supported audio data formats. See the iOS developer documentation.                  | True                |
| 2605: Alert Message (string)     | The message that appears on the client device when alerts are enabled.                                                                                                                                                                                                                                                        | New items available |
| 2606: APNS Device Token (string) | The Apple push notification service token. An application must register with Apple push notification service for the iOS to receive remote notifications sent by the application's provider. After the device is registered for push properly, this should contain a valid device token. See the iOS developer documentation. | n/a                 |

**Application Settings Properties**

Application settings display details that identify the Application Identifier, Domain, Security Configuration of an application connection template.

| <b>ID: property name (type)</b> | <b>Description</b>                                              | <b>Default</b> |
|---------------------------------|-----------------------------------------------------------------|----------------|
| Domain                          | The domain selected for the connection template.                |                |
| Security Configuration          | The security configuration defined for the connection template. |                |



| ID: property name (type)       | Description                                                                                                | Default |
|--------------------------------|------------------------------------------------------------------------------------------------------------|---------|
| Automatic Registration Enabled | The value is set to <b>True</b> when the application connection registration is carried out automatically. |         |
| Application Identifier         | The application identifier registered on SAP Control Center.                                               |         |
| Customization Resource         | The application configuration (customization resource bundles) associated with the application.            |         |

### **BlackBerry Push Notification Properties**

BlackBerry push notification properties enable the server to send notifications to BlackBerry devices using Blackberry Enterprise Server (BES).

| Property               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Enabled                | Enables notifications to the device if the device is offline. This feature sends a push notification over an IP connection only long enough to complete the Send/Receive data exchange. BlackBerry Push notifications overcome issues with always-on connectivity and battery life consumption over wireless networks. Acceptable values: true (enabled) and false (disabled). If this setting is false, all other related settings are ignored. Default: true |
| BES Push Listener Port | The listener port for BES notifications. The port is discovered and set by the client, and is read-only on the server.                                                                                                                                                                                                                                                                                                                                         |
| Device PIN             | Every Blackberry device has a unique permanent PIN. During initial connection and settings exchange, the device sends this information to the server. SAP Mobile Server uses this PIN to address the device when sending notifications, by sending messages through the BES/MDS using an address such as: Device="Device PIN" + Port="Push Listener port". Default: 0                                                                                          |

### **Connection Properties**

Connection properties define the connection information for a client application so it can locate the appropriate SAP Mobile Server synchronization service.

| ID: property name (type) | Description                                                                                                                                                                                       | Default |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1: Server Name (string)  | The DNS name or IP address of the SAP Mobile Server, such as "myserver.mycompany.com". If using Relay Server, the server name is the IP address or fully qualified name of the Relay Server host. | n/a     |

| ID: property name (type)    | Description                                                                                                                                                                                          | Default |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 2: Server Port (integer)    | The port used for messaging connections between the device and SAP Mobile Server. If using Relay Server, this is the Relay Server port.                                                              | 5011    |
| 3: Farm ID (string)         | The string associated with the Relay Server farm ID. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both.                                               | 0       |
| 6: Activation Code (string) | The original code sent to the user in the activation e-mail. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Acceptable range: 1 to 10 characters. | n/a     |

### **Custom Settings Properties**

Define one of four available custom strings that are retained during reregistration and cloning.

Change the property name and value according to the custom setting you require. The custom settings can be of variable length, with no practical limit imposed on the values. You can use these properties to either manually control or automate how Hybrid App-related messages are processed:

- Manual control – an administrator can store an employee title in one of the custom fields. This allows employees of a specific title to respond to a particular message.
- Automated – a developer stores the primary key of a back-end database using a custom setting. This key allows the database to process messages based on messaging device ID.

| ID: property name (type) | Description                                                          | Default |
|--------------------------|----------------------------------------------------------------------|---------|
| 2300: Custom 1(string)   | A custom string which is retained during reregistration and cloning. | n/a     |
| 2301: Custom 2(string)   | A custom string which is retained during reregistration and cloning. | n/a     |
| 2302: Custom 3(string)   | A custom string which is retained during reregistration and cloning. | n/a     |
| 2303: Custom 4(string)   | A custom string which is retained during reregistration and cloning. | n/a     |

### **Device Information Properties**

Information properties display details that identify the mobile device, including International Mobile Subscriber identity (IMSI), phone number, device subtype, and device model.

| <b>ID: property name (type)</b> | <b>Description</b>                                                                                                                                                                                                                                                                         | <b>Default</b> |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 1200: Model (string)            | The manufacturer of the registered mobile device.                                                                                                                                                                                                                                          | n/a            |
| 1201: Device Sub-type (string)  | The device subtype of the messaging device. For example, if the device model is a BlackBerry, the subtype is the form factor (for example, BlackBerry Bold).                                                                                                                               | n/a            |
| 1202: Phone Number (string)     | The phone number associated with the registered mobile device.                                                                                                                                                                                                                             | n/a            |
| 1203: IMSI (string)             | The International Mobile Subscriber identity, which is a unique number associated with all Global System for Mobile communication (GSM) and Universal Mobile Telecommunications System (UMTS) network mobile phone users. To locate the IMSI, check the value on the SIM inside the phone. | n/a            |

### **Advanced Device Properties**

Advanced properties set specific behavior for messaging devices.

| <b>ID: property name (type)</b>   | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | <b>Default</b> |
|-----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 1300: Keep Alive (sec) (integer)  | The Keep Alive frequency used to maintain the wireless connection, in seconds. Acceptable values: 30 to 1800.                                                                                                                                                                                                                                                                                                                                                                                             | 240            |
| 1301: Device Log Items(integer)   | The number of items persisted in the device status log. Acceptable values: 5 to 100.                                                                                                                                                                                                                                                                                                                                                                                                                      | 50             |
| 1302: Debug Trace Level (integer) | <p>The amount of detail to record to the device log. Acceptable values: 1 to 5, where 5 has the most level of detail and 1 the least.</p> <ul style="list-style-type: none"> <li>• 1: Basic information, including errors</li> <li>• 2: Some additional details beyond basic</li> <li>• 3: Medium amount of information logged</li> <li>• 4: Maximum tracing of debugging and timing information</li> <li>• 5: Maximum tracing of debugging and timing information (currently same as level 4)</li> </ul> | 1              |

| ID: property name (type)               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Default |
|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1303: Debug Trace Size (KB) (integer)  | The size of the trace log on the device (in KB). Acceptable values: 50 to 10,000.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 50      |
| 1304: Allow Roaming (boolean)          | Use ifdevice is allowed to connect to server while roaming. Acceptable values: true and false.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | True    |
| 1305: Relay Server URL Prefix (string) | <p>The URL prefix to be used when the device client is connecting through Relay Server. The prefix you set depends on whether Relay Server is installed on IIS or Apache. For IIS, this path is relative. Acceptable values include:</p> <ul style="list-style-type: none"> <li>For IIS – use /ias_relay_server/client/rs_client.dll.</li> <li>For Apache – use /cli/iasrelayserver.</li> </ul> <p><b>Note:</b> The value used in the client application connection for the URL suffix must match what the administrator configures in the URL suffix. Otherwise, the connection fails. Use the Diagnostic Tool command line utility to test these values. See <i>Diagnostic Tool Command Line Utility (diagtool.exe) Reference</i> in <i>System Administration</i>.</p> | n/a     |

### **Proxy Properties**

Proxy properties define parameters to connect Relay Server Outbound Enabler to a Relay Server through a proxy server.

| ID: property name (type) | Description                    | Default                                             |
|--------------------------|--------------------------------|-----------------------------------------------------|
| Application Endpoint     | The application endpoint.      | n/a                                                 |
| Push Endpoint            | The URL for the push endpoint. | http://<server_host-name>:8000/GWC/SUP-Notification |

### **Security Settings Properties**

Security settings display the device security configuration.

| <b>ID: property name (type)</b> | <b>Description</b>                                                                                                                                    | <b>Default</b> |
|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| E2E Encryption Enabled          | Allows you to indicate whether end-to-end encryption is enabled or not: true indicates encryption is enabled; false indicates encryption is disabled. |                |
| E2E Encryption Type             | Allows you to use RSA as the asymmetric cipher used for key exchange for end-to-end encryption.                                                       |                |
| TLS Type                        | Allows you to use RSA as the TLS type for device to SAP Mobile Server communication.                                                                  |                |

### **User Registration Properties**

Device user registration properties allow you to customize the registration request that is delivered to the device.

| <b>ID: property name (type)</b>                   | <b>Description</b>                                                                                                                                   | <b>Default</b> |
|---------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 900: Activation code length (integer)             | The number of characters to be contained in the activation code. Acceptable values: 1 to 10.                                                         | 3              |
| 901: Activation code expiration (hours) (integer) | Defines how long a user has to activate their account, in hours, before the account activation period expires. Acceptable values: 1 to 10,000 hours. | 72             |

## **EIS Data Source Connection Properties Reference**

Name and configure connection properties when you create connection pools in SAP Control Center to enterprise information systems (EIS) .

### **JDBC Properties**

Configure Java Database Connectivity (JDBC) connection properties.

This list of properties can be used by all datasource types. SAP does not document native properties used only by a single driver. However, you can also use native driver properties, naming them using this syntax:

```
jdbc:<NativeConnPropName>=<Value>
```

**Note:** If SAP Mobile Server is connecting to a database with a JDBC driver, ensure you have copied required JAR files to correct locations. See *Installation Guide for Runtime*.

| Name               | Description                                                                                                                                                                                                                                                           | Supported values                                                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| afterInsert        | Changes the value to <code>into</code> if a database requires <code>insert into</code> rather than the abbreviated <code>into</code> .                                                                                                                                | <code>into</code>                                                                                                                                              |
| batchDelimiter     | Sets a delimiter, for example, a semicolon, that can be used to separate multiple SQL statements within a statement batch.                                                                                                                                            | <code>&lt;delimiter&gt;</code>                                                                                                                                 |
| blobUpdater        | Specifies the name of a class that can be used to update database BLOB (long binary) objects when the BLOB size is greater than <code>psMaximumBlobLength</code> .                                                                                                    | <code>&lt;class name&gt;</code><br><br>The class must implement the <code>com.sybase.djc.sql.BlobUpdater</code> interface.                                     |
| compactColumnAlias | An expression that uses the nested variables “ <code>_\${index}</code> ” and “ <code>_\${column}</code> ” for shortening column names in result sets. This can reduce the data transmitted between the database server and the application server.                    | An expression. For example: <code>_\${index}=\${column} \${column} AS _\${index}</code>                                                                        |
| clobUpdater        | Specifies the name of a class that can be used to update database CLOB (long string) objects when the CLOB size is greater than <code>psMaximumClobLength</code> .                                                                                                    | <code>&lt;class name&gt;</code><br><br>The class must implement the <code>com.sybase.djc.sql.ClobUpdater</code> interface.                                     |
| codeSet            | Specifies how to represent a repertoire of characters by setting the value of <code>CS_SYB_CHARSET</code> for this datasource. Used when the data in the datasource is localized. If you do not specify the correct code set, characters may be rendered incorrectly. | <code>[server]</code><br><br>If the value is <code>server</code> , the value of the current application server’s <code>defaultCodeSet</code> property is used. |

| Name           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Supported values                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| commitProtocol | <p>Specifies how SAP Mobile Server handles connections for a data-source at commit time, specifically when a single transaction requires data from multiple endpoints.</p> <p>If you use XA, the recovery log is stored in the tx_manager data-source, and its commit protocol must be optimistic. If tx_manager is aliased to another datasource (that is, one that is defined with the alias-For property), the commit protocol for that datasource must be optimistic. A last-resource optimization ensures full conformance with the XA specification. The commit protocol for all other datasources should be XA_2PC. Alternately, a transaction that accesses multiple datasources for which the commit protocols are optimistic is permitted.</p> | <p>[optimistic   pessimistic   XA_2PC]</p> <p>Choose only one of these protocols:</p> <ul style="list-style-type: none"> <li>• Optimistic – enables connections to be committed without regard for other connections enlisted in the transaction, assuming that the transaction is not marked for rollback and will successfully commit on all resources. Note: if a transaction accesses multiple data sources with commit protocol of "optimistic", atomicity is not guaranteed.</li> <li>• Pessimistic – specifies that you do not expect any multi-resource transactions. An exception will be thrown (and transaction rolled back) if any attempt is made to use more than one "pessimistic" data source in the same transaction.</li> <li>• XA_2PC – specifies use of the XA two phase commit protocol. If you are using two phase commit, then the recovery log is stored in the "tx_manager" data source, and that data source (or the one it is aliased to) must have the commit protocol of "optimistic" or "pessimistic". All other data sources for which atomicity must be ensured should have the "XA_2PC" commit protocol.</li> </ul> |

| Name                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Supported values                                                                                                                                                                                                   |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dataSourceClass     | <p>Sets the class that implements the JDBC datasource.</p> <p>Use this property (along with the driverClass property) only if you do not have a predefined database-type entry in SAP Mobile Server for the kind of SQL database you are connecting to. For example, you must use this property for MySQL database connections.</p> <p>You can implement a datasource class to work with a distributed transaction environment. Because SAP Mobile Server supports distributed transactions, some datasources may require that a datasource class be implemented for SAP Mobile Server to interact with it.</p> <p>For two-phase transactions, use the xaDataSourceClass connection property instead.</p> | <p>&lt;com.mydata-source.jdbc.Driver&gt;</p>                                                                                                                                                                       |
| databaseCommandEcho | <p>Echoes a database command to both the console window and the server log file.</p> <p>Use this property to immediately see and record the status or outcome of database commands.</p> <p>When you enable this property, SAP Mobile Server echoes every SQL query to ml.log, which may help you debug your application.</p>                                                                                                                                                                                                                                                                                                                                                                              | <p>[true false]</p> <p>Set a value of 1 to echo the database commands like databaseStartCommand, and databaseStopCommand.</p> <p>Otherwise, do not set this property, or use a value of 0 to disable the echo.</p> |



| Name                  | Description                                                                                                                                                                                                                                                                                                                                                      | Supported values                                                                                                                         |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| databaseCreateCommand | <p>Specifies the operating system command used to create the database for this datasource. If this command is defined and the file referenced by <code>\${databaseFile}</code> does not exist, the command is run to create the database when an application component attempts to obtain the first connection from the connection pool for this datasource.</p> | <p>&lt;command&gt;</p> <p>Example: SMP_HOME\Servers\SQLAny-where11\BIN32\dbinit -q <code>\${databaseFile}</code></p>                     |
| databaseFile          | <p>Indicates the database file to load when connecting to a datasource.</p> <p>Use this property when the path to the database file differs from the one normally used by the database server.</p> <p>If the database you want to connect to is already running, use the <code>databaseName</code> connection parameter.</p>                                     | <p>&lt;string&gt;</p> <p>Supply a complete path and file name. The database file you specify must be on the same host as the server.</p> |

| Name                 | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Supported values                                                                                                                                                                                                                  |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| databaseName         | <p>Identifies a loaded database with which to establish a connection, when connecting to a datasource.</p> <p>Set a database name, so you can refer to the database by name in other property definitions for a datasource.</p> <p>If the database to connect to is not already running, use the database-File connection parameter so the database can be started.</p> <hr/> <p><b>Note:</b> For SAP Mobile Server, you typically do not need to use this property. Usually, when you start a database on a server, the database is assigned a name. The mechanism by which this occurs varies. An administrator can use the DBN option to set a unique name, or the server may use the base of the file name with the extension and path removed.</p> | <p>[DBN   default]</p> <p>If you set this property to default, the name is obtained from the DBN option set by the database administrator.</p> <p>If no value is used, the database name is inherited from the database type.</p> |
| databaseStartCommand | <p>Specifies the operating system command used to start the database for this datasource. If this command is defined and the database is not running, the command is run to start the database when the datasource is activated.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | <p>&lt;command&gt;</p> <p>Example: SMP_HOME\Servers\SQLAny-where11\BIN32\dbsrvXX.exe</p>                                                                                                                                          |
| databaseStopCommand  | <p>Specifies the operating system command used to stop the database for this datasource. If this property is defined and the database is running, this command executes during shutdown.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <p>&lt;command&gt;</p> <p>For a SQL Anywhere® database, where the user name and password are the defaults (dba and sql), enter:</p> <p>SMP_HOME\Servers\SQLAny-where11\BIN32\dbsrvXX.exe</p>                                      |
| databaseType         | <p>Specifies the database type.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | <p>&lt;database type&gt;</p>                                                                                                                                                                                                      |

| Name              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                 | Supported values                                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| databaseURL       | <p>Sets the JDBC URL for connecting to the database if the datasource requires an Internet connection.</p> <p>Typically, the server attempts to construct the database URL from the various connection properties you specify (for example, portNumber, databaseName). However, because some drivers require a special or unique URL syntax, this property allows you to override the server defaults and instead provide explicit values for this URL.</p> | <p>&lt;JDBCurl&gt;</p> <p>The database URL is JDBC driver vendor-specific. For details, refer to the driver vendor's JDBC documentation.</p> |
| disableAutoCommit | <p>Enables or disables calling auto-commit mode. Auto-commit means that every update to the database is immediately made permanent.</p>                                                                                                                                                                                                                                                                                                                     | <p>[true false]</p> <p>The default is false.</p>                                                                                             |
| disablePrefetch   | <p>Enables or disables prefetch. Prefetch optimizes container-managed persistence by batching queries from a parent to its children (for example, from a customer to orders), to reduce the calls from the application server to the database.</p>                                                                                                                                                                                                          | <p>[true false]</p> <p>The default is true.</p>                                                                                              |
| disableTriggers   | <p>Select to deactivate database triggers, on a per-connection basis, when the application server accesses the database. If selected, the database must support both the <code>set triggers on</code> and <code>set triggers off</code> commands.</p>                                                                                                                                                                                                       | <p>[true false]</p> <p>The default is false.</p>                                                                                             |

| Name                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Supported values                                                                                                                                              |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| driverClass         | <p>Sets the name of the class that implements the JDBC driver.</p> <p>Use this property (along with the dataSourceClass property) only if you do not have a predefined database-type entry in SAP Mobile Server for the kind of SQL database you are connecting to. For example, MySQL database connections require you to use this connection property.</p> <p>To create a connection to a database system, you must use the compatible JDBC driver classes. SAP does not provide these classes; you must obtain them from the database manufacturer.</p> | <p><code>&lt;Class.forName("foo.bar.Driver")&gt;</code></p> <p>Replace <code>&lt;Class.forName("foo.bar.Driver")&gt;</code> with the name of your driver.</p> |
| driverDebug         | Enables debugging for the driver.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | <p>[true false]</p> <p>Set to true to enable debugging, or false to disable.</p>                                                                              |
| driverDebugSettings | Configures debug settings for the driver debugger.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <p>[default &lt;setting&gt;]</p> <p>The default is STATIC:ALL.</p>                                                                                            |
| endpointName        | The JDBC datasource name.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | JDBC datasource name.                                                                                                                                         |
| getDateAndTime      | A SQL query to get the date and time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | <p>A valid SQL query.</p> <p>The default is <code>select get-date()</code>.</p>                                                                               |

| Name                                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Supported values                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| InitialPoolSize                         | <p>Sets the initial number of connections in the pool for a datasource.</p> <p>In general, holding a connection causes a less dramatic performance impact than creating a new connection. Keep your pool size large enough for the number of concurrent requests you have; ideally, your connection pool size should ensure that you never run out of available connections.</p> <p>The initialPoolSize value is applied to the next time you start SAP Mobile Server.</p> | <p>&lt;int&gt;</p> <p>Replace &lt;int&gt; with an integer to preallocate and open the specified number of connections at start-up. The default is 0.</p> <p>SAP suggests that you start with 0, and create additional connections as necessary. The value you choose allows you to create additional connections before client synchronization requires the server to create them.</p> |
| isDownloadZipped                        | <p>Specifies whether the driver file downloaded from jdbcDriverDownloadURL is in .ZIP format.</p> <p>This property is ignored if the value of jdbcDriverDownloadURL connection is an empty string.</p>                                                                                                                                                                                                                                                                     | <p>[True False]</p> <p>The default is false. The file is copied, but not zipped to <i>SMP_HOME\lib\jdbc</i>.</p> <p>Set isDownloadZipped to true to save the file to <i>SMP_HOME\lib\jdbc</i> and unzip the archived copy.</p>                                                                                                                                                         |
| jdbc:DISABLE_UNPROCESSED_PARAM_WARNINGS | <p>All properties starting with "jdbc:" are used to pass the suffix (such as DISABLE_UNPROCESSED_PARAM_WARNINGS ) to the JDBC driver while getting a connection. This property is used for the jConnect driver. Set this property to true can disable the warning of "An output parameter was received and ignored".</p>                                                                                                                                                   | <p>[True False]</p> <p>The default is false.</p> <p>This property is for SAP ASA or SAP ASE databases only.</p>                                                                                                                                                                                                                                                                        |

| Name                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Supported values                                                                                    |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| jdbc:IS_CLOSED_TEST    | <p>As above, this property is used for the jConnect driver. You can force jConnect to follow the standard JDBC behavior for <code>isClosed()</code> by setting the IS_CLOSED_TEST connection property to the special value 'INTERNAL'. The INTERNAL setting means that jConnect returns true for <code>isClosed()</code> only when <code>Connection.close()</code> has been called, or when jConnect has detected an <code>IOException</code> that has disabled the Connection.</p> <p>You can specify a query other than <code>sp_mda</code> to use when <code>isClosed()</code> is called. For example, if you want jConnect to try <code>select 1</code> when <code>isClosed()</code> is called, you can set the IS_CLOSED_TEST connection property to <code>select 1</code>.</p> | The default is INTERNAL.                                                                            |
| jdbc:DriverType        | The driverType property to be passed to the JDBC driver class. For example, for Oracle, you can set this property to "thin".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | The driverType property.<br>For an Oracle database type, use "thin".                                |
| jdbc:DriverDownloadURL | <p>Specifies the URL from which you can download a database driver.</p> <p>Use this property with <code>isDownloadZipped</code> to put the driver in an archive file before the download starts.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <p>&lt;URL&gt;</p> <p>Replace &lt;URL&gt; with the URL from which the driver can be downloaded.</p> |
| jit:imageParameterType | Defines the SQL type of the image parameter. All properties that start with "jit:" are used for the SAP JIT DataSource only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | A varbinary (16384) value.<br>For example, <code>varbinary(255)</code> .                            |
| jit:textParameterType  | Defines the SQL type of the text parameter. Used for the SAP JIT DataSource only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | A varchar (16384) value.                                                                            |

| Name                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Supported values                                                                                                                 |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| jit:unitextParameterType | Defines the SQL type of the unicode text parameter. Used for the SAP JIT DataSource only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | A univarchar (16384) value.                                                                                                      |
| language                 | <p>For those interfaces that support localization, this property specifies the language to use when connecting to your target database. When you specify a value for this property, SAP Mobile Server:</p> <ul style="list-style-type: none"> <li>• Allocates a CS_LOCALE structure for this connection</li> <li>• Sets the CS_SYB_LANG value to the language you specify</li> <li>• Sets the Microsoft SQL Server CS_LOC_PROP connection property with the new locale information</li> </ul> <p>SAP Mobile Server can access Unicode data in an Adaptive Server® 12.5 or later, or in Unicode columns in Adaptive Server 12.5 or later. SAP Mobile Server automatically converts between double-byte character set (DBCS) data and Unicode, provided that the Language and CodeSet parameters are set with DBCS values.</p> | <p>&lt;language&gt;</p> <p>Replace &lt;language&gt; with the language being used.</p>                                            |
| maxIdleTime              | Specifies the number of seconds an idle connection remains in the pool before it is dropped.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | <p>&lt;int&gt;</p> <p>If the value is 0, idle connections remain in the pool until the server shuts down. The default is 60.</p> |

| Name          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Supported values                                                                                             |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| maxPoolSize   | <p>Sets the maximum number of connections allocated to the pool for this datasource.</p> <p>Increase the maxPoolSize property value when you have a large user base. To determine whether a value is high enough, look for ResourceMonitorTimeoutException exceptions in <code>&lt;hostname&gt;-server.log</code>. Continue increasing the value, until this exception no longer occurs.</p> <p>To further reduce the likelihood of deadlocks, configure a higher value for maxWaitTime.</p> <p>To control the range of the pool size, use this property with minPoolSize.</p> | <p>&lt;int&gt;</p> <p>A value of 0 sets no limit to the maximum connection pool size. The default is 10.</p> |
| maxWaitTime   | <p>Sets the maximum number of seconds to wait for a connection before the request is cancelled.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | <p>&lt;int&gt;</p> <p>The default is 60.</p>                                                                 |
| maxStatements | <p>Specifies the maximum number of JDBC prepared statements that can be cached for each connection by the JDBC driver. The value of this property is specific to each JDBC driver.</p>                                                                                                                                                                                                                                                                                                                                                                                         | <p>&lt;int&gt;</p> <p>A value of 0 (default) sets no limit to the maximum statements.</p>                    |
| minPoolSize   | <p>Sets the minimum number of connections allocated to the pool for this datasource.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <p>&lt;int&gt;</p> <p>A value of 0 (default) sets no limit to the minimum connection pool size.</p>          |



| Name                  | Description                                                                                                                                                                                                                                                                                                                                                                                | Supported values                                                                                                                               |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| networkProtocol       | <p>Sets the protocol used for network communication with the datasource.</p> <p>Use this property (along with the driverClass, and dataSourceClass properties) only if you do not have a predefined database-type entry in SAP Mobile Server for the kind of SQL database you are connecting to. For example, you may be required to use this property for MySQL database connections.</p> | <p>The network protocol is JDBC driver vendor-specific. There are no predefined values.</p> <p>See the driver vendor's JDBC documentation.</p> |
| ownerPrefix           | <p>The owner prefix for stored procedures and table names in this datasource. A prefix is used by the EJB persistence manager and JIT driver wrappers to qualify database identifiers for stored procedures and tables.</p>                                                                                                                                                                | <p>An owner prefix.</p>                                                                                                                        |
| password              | <p>Specifies the password for connecting to the database.</p>                                                                                                                                                                                                                                                                                                                              | <p>[default   &lt;password&gt;]</p>                                                                                                            |
| pingAndSetSessionAuth | <p>Runs the ping and session-authorization commands in a single command batch; may improve performance. You can only enable the Ping and Set Session Auth property if you have enabled the Set Session Auth property so database work runs under the effective user ID of the client.</p>                                                                                                  | <p>[True   False]</p> <p>Set to true to enable, or false to disable.</p>                                                                       |
| pingConnections       | <p>Pings connections before attempting to reuse them from the connection pool.</p>                                                                                                                                                                                                                                                                                                         | <p>[True   False]</p> <p>Set to true to enable ping connections, or false to disable.</p>                                                      |
| pingSQL               | <p>Specify the SQL statement to use when testing the database connection with ping.</p>                                                                                                                                                                                                                                                                                                    | <p>[default   &lt;statement&gt;]</p> <p>Replace &lt;statement&gt; with the SQL statement identifier. The default is "select 1".</p>            |

| Name                 | Description                                                                                                                                                                                                                                    | Supported values                                                                                                                                                                                                                                                                |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| portNumber           | Sets the server port number where the database server listens for connection requests.                                                                                                                                                         | <p>[ default   &lt;port&gt; ]</p> <p>Replace &lt;port&gt; with the TCP/IP port number to use (that is, 1 – 65535).</p> <p>If you set the value as default, the default protocol of the datasource is used.</p>                                                                  |
| psMaximumBlobLength  | Indicates the maximum number of bytes allowed when updating a BLOB datatype using Prepared-Statement.setBytes.                                                                                                                                 | <p>[ default   &lt;int&gt; ]</p> <p>Replace &lt;int&gt; with the number of bytes allowed during an update. The default is 16384.</p>                                                                                                                                            |
| psMaximumClobLength  | Indicates the maximum number of characters allowed when updating a CLOB datatype using Prepared-Statement.setString.                                                                                                                           | <p>[ default   &lt;int&gt; ]</p> <p>Replace &lt;int&gt; with the number of bytes allowed during an update. The default is 16384.</p>                                                                                                                                            |
| roleName             | Sets the database role that the user must have to log in to the database.                                                                                                                                                                      | <p>[ default   &lt;name&gt; ]</p> <p>If you set this value to default, the default database role name of the data-source is used.</p>                                                                                                                                           |
| selectWithSharedLock | A template SQL statement for selecting rows and acquiring a shared lock. If your database server does not support shared locks, specify a template for acquiring exclusive locks.                                                              | <p>A template SQL statement.</p> <p>For example, for a SAP ASA database type:</p> <pre> \${selectList}\${into- Clause}\${fromClause} holdlock\${whereClause}                     </pre>                                                                                         |
| selectWithUpdateLock | A template SQL statement for selecting rows and acquiring an exclusive lock. The configuration property name is selectWithUpdateLock. If your database server does not support exclusive locks, specify a template for acquiring shared locks. | <p>A template SQL statement.</p> <p>For example, for a SAP ASA database type:</p> <pre> update \${mainTable} set \${touchColumn} = 1 -\${ touchColumn}\${from- Clause}\${whereClause};; \${selectList}\${into- Clause}\${fromClause}\${ whereClause}                     </pre> |

| Name                   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Supported values                                                                                                                                                                      |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| serializableSelect     | A template SQL statement for selecting rows and acquiring a lock that ensures strict serializability, in terms of equivalence with serial schedules.                                                                                                                                                                                                                                                                                                                                                  | A template SQL statement.<br><br>For example, for a SAP database type:<br><code>`\${selectList}``\${intoClause}``\${fromClause}``<br/><code>holdlock`\${whereClause}``</code></code>  |
| serverName             | Defines the host where the database server is running.                                                                                                                                                                                                                                                                                                                                                                                                                                                | <name><br><br>Replace <name> with an appropriate name for the server.                                                                                                                 |
| serviceName            | Defines the service name for the datasource.<br><br>For SQL Anywhere servers, use this property to specify the database you are attaching to.                                                                                                                                                                                                                                                                                                                                                         | <name><br><br>Replace <name> with an appropriate name for the service.                                                                                                                |
| setSessionAuth         | Establishes an effective database identity that matches the current mobile application user.<br><br>If you use this property, you must also use <code>setSessionAuthSystemID</code> to set the session ID.<br><br>Alternately you can <code>pingAndSetSessionAuth</code> if you are using this property with <code>pingConnection</code> . The <code>pingAndSetSessionAuth</code> property runs the ping and session-authorization commands in a single command batch, which may improve performance. | [true false]<br><br>Choose a value of 1 to use an ANSI SQL set session authorization command at the start of each database transaction. Set to 0 to use session-based authorizations. |
| setSessionAuthSystemID | If Set Session Authorization is enabled, specifies the database identity to use when the application server accesses the database from a transaction that runs with "system" identity.                                                                                                                                                                                                                                                                                                                | <database identity><br><br>Replace <database identity> with the database identifier.                                                                                                  |

| Name                 | Description                                                                                                                                                                                                                                                                   | Supported values                                                                                                                                                                                                                                                                      |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| startWait            | <p>Sets the wait time (in seconds) before a connection problem is reported. If the start command completes successfully within this time period, no exceptions are reported in the server log.</p> <p>startWait time is used only with the databaseStartCommand property.</p> | <p>&lt;int&gt;</p> <p>Replace &lt;int&gt; with the number of seconds SAP Mobile Server waits before reporting an error.</p>                                                                                                                                                           |
| truncateNanos        | <p>Sets a divisor/multiplier that is used to round the nanoseconds value in a java.sql.Timestamp to a granularity that the DBMS supports.</p>                                                                                                                                 | <p>[default   &lt;int&gt;]</p> <p>The default is 10 000 000.</p>                                                                                                                                                                                                                      |
| useQuotedIdentifiers | <p>Specifies whether or not SQL identifiers are quoted.</p>                                                                                                                                                                                                                   | <p>[True   False]</p> <p>Set to true to enable use of quoted identifiers, or false to disable.</p>                                                                                                                                                                                    |
| useTransactionalPing | <p>Enables or disables the attempt to ping a connection from within a new transaction.</p>                                                                                                                                                                                    | <p>[True   False]</p> <p>The default is true.</p>                                                                                                                                                                                                                                     |
| user/User            | <p>Identifies the user who is connecting to the database.</p>                                                                                                                                                                                                                 | <p>[default   &lt;user name&gt;]</p> <p>Replace &lt;user name&gt; with the database user name.</p> <p>For DB2 and SQL Server databases, this property is user. For Informix, Oracle, and SQL Anywhere databases, this property is User.</p>                                           |
| xaDataSourceClass    | <p>Specifies the class name or library name used to support two-phase commit transactions, and the name of the XA resource library.</p>                                                                                                                                       | <p>&lt;class name&gt;</p> <p>Replace &lt;class name&gt; with the class or library name.</p> <ul style="list-style-type: none"> <li>• SQL Anywhere database:<br/>com.sybase.jdbc3.jdbc.SybXADataSource</li> <li>• Oracle database: oracle.jdbc.xa.client.OracleXADataSource</li> </ul> |

### **SAP Java Connector Properties**

Configure SAP Java Connector (JCo) connection properties.

For a comprehensive list of SAP JCo properties you can use to create an instance of a client connection to a remote SAP system, see [http://help.sap.com/javadocs/NW04/current/jc/com/sap/mw/jco/JCO.html#createClient\(java.util.Properties\)](http://help.sap.com/javadocs/NW04/current/jc/com/sap/mw/jco/JCO.html#createClient(java.util.Properties)).

This list of properties can be used by all datasource types. SAP does not document all native endpoint properties. However, you can add native endpoint properties, naming them using this syntax:

```
<NativeConnPropName>=<SupportedValue>
```

**Table 348. General Connection Parameters**

| Name                  | Description                                                                                                                                                                                                                                                                                                                                                                                | Supported Values                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| jco.client.abap_debug | <p>Enables or disables ABAP debugging. If enabled, the connection is opened in debug mode and you can step through the invoked function module in the debugger.</p> <p>For debugging, an SAP graphical user interface (SAPGUI) must be installed on the same machine the client program is running on. This can be either a normal Windows SAPGUI or a Java GUI on Linux/UNIX systems.</p> | <p>Not supported.</p> <p>Do not set this parameter or leave it set to 0.</p>                  |
| jco.client.use_sapgui | <p>Specifies whether a remote SAP graphical user interface (SAPGUI) should be attached to the connection. Some older BAPIs need an SAPGUI because they try to send screen output to the client while executing.</p>                                                                                                                                                                        | <p>Not supported.</p> <p>Do not set this parameter or leave it set to 0.</p>                  |
| jco.client.getsso2    | <p>Generates an SSO2 ticket for the user after login to allow single sign-on. If RfcOpenConnection() succeeds, you can retrieve the ticket with RfcGetPartnerSSOTicket() and use it for additional logins to systems supporting the same user base.</p>                                                                                                                                    | <p>Not accessible by the customer.</p> <p>Do not set this parameter or leave it set to 0.</p> |

| Name                     | Description                                                                                                                                   | Supported Values                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jco.client.x509cert      | SAP Mobile Platform sets this property when a client uses an X509 certificate as the login credential.                                        | <p>If an EIS RFC operation is flagged for SSO (user name and password personalization keys selected in the authentication parameters) then SAP Mobile Platform automatically sets the appropriate properties to use X.509, SSO2, or user name and password SSO credentials.</p> <p>The corresponding properties should not be set by the administrator on the SAP endpoint.</p> |
| jco.client.grt_data      | Provides additional data for graphical user interface (GUI) to specify the SAProuter connection data for the SAPGUI when it is used with RFC. | Not supported.                                                                                                                                                                                                                                                                                                                                                                  |
| jco.client.use_guihost   | Identifies which host to redirect the remote graphical user interface to.                                                                     | Not supported.                                                                                                                                                                                                                                                                                                                                                                  |
| jco.client.use_guiserv   | Identifies which service to redirect the remote graphical user interface to.                                                                  | Not supported.                                                                                                                                                                                                                                                                                                                                                                  |
| jco.client.use_guiprogid | Indicates the program ID of the server that starts the remote graphical user interface.                                                       | Not supported.                                                                                                                                                                                                                                                                                                                                                                  |

### **SAP DOE-C Properties**

Configure SAP® Data Orchestration Engine Connector (DOE-C) properties. This type of connection is available in the list of connection templates only when you deploy a SAP® Data Orchestration Engine Connector package. No template exists for these types of connections.

---

**Note:** If you change the username or password property of a DOE-C connection, you must reopen the same dialog and click `Test Connection` after saving. Otherwise the error state of this DOE-C package is not set properly, and an error message is displayed. This will not work if you click `Test Connection` before saving the properties.

---

| Name              | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Supported values                                                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| techuser-name     | <p>Specifies the SAP user account ID. The SAP user account is used during interaction between the connected SAP system and client for certain administrative activities, such as sending acknowledgment messages during day-to-day operations or "unsubscribe" messages if a subscription for this connection is removed.</p> <p>This account is not used for messages containing business data; those types of messages are always sent within the context of a session authenticated with credentials provided by the mobile client.</p> <p>The technical user name and password or certificateAlias must be set to perform actions on subscriptions. The certificateAlias is mutually exclusive with and overrides the technical user name and password fields if set. The technical user name and password fields can be empty, but only if certificateAlias is set.</p> | Valid SAP login name for the DOE host system.                   |
| techuser-password | Specifies the password for the SAP user account.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Valid password.                                                 |
| doe-soap-timeout  | Specifies a timeout window during which unresponsive DOE requests are aborted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Positive value (in seconds).<br>The default is 420 (7 minutes). |

| Name                | Description                                                                                                                                                                                                                                                                                                                                                                                 | Supported values                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| doe-extract-window  | Specifies the number of messages allowed in the DOE extract window.                                                                                                                                                                                                                                                                                                                         | <p>Positive value (in messages).</p> <p>The minimum value is 10. The maximum value is 2000. The default is 50.</p> <p>When the number of messages in the DOE extract window reaches 50% of this value, DOE-C sends a <code>StatusReqFromClient</code> message, to advise the SAP DOE system of the client's messaging status and acknowledge the server's state.</p>                                                                                                            |
| doe-packetDrop-size | <p>Specifies the size, in bytes, of the largest JavaScript Object Notation (JSON) message that the DOE connector processes on behalf of a JSON client.</p> <p>The packet drop threshold size should be carefully chosen, so that it is larger than the largest message sent from the DOE to the client, but smaller than the maximum message size which may be processed by the client.</p> | <p>Positive value (in bytes).</p> <p>The default is 1048576 bytes (1MB).</p> <p>Do not set lower than 4096 bytes; there is no maximum limitation.</p>                                                                                                                                                                                                                                                                                                                           |
| service-address     | Specifies the DOE URL.                                                                                                                                                                                                                                                                                                                                                                      | <p>Valid DOE URL.</p> <p>If you are using DOE-C with SSL:</p> <ul style="list-style-type: none"> <li>• Modify the port from the standard <code>http://host:8000</code> to <code>https://host:8001/</code>.</li> <li>• Add the certificate being used as the technical user and DOE-C endpoint security profile certificate to the SAP DOE system's SSL Server certificate list by using the <code>STRUST</code> transaction. See your SAP documentation for details.</li> </ul> |
| listener-url        | Specifies the DOE-C server listener URL.                                                                                                                                                                                                                                                                                                                                                    | Valid DOE-C listener URL, for example <code>http://&lt;sup_host-name&gt;:8000/dae/publish</code> .                                                                                                                                                                                                                                                                                                                                                                              |



| Name             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Supported values                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------|
| certificateAlias | <p>Sets the alias for the SAP Mobile Platform keystore entry that contains the X.509 certificate for SAP Mobile Server's SSL peer identity.</p> <p>If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service.</p> <p>If you are using DOE-C with SSO use the "SAP Technical User Certificate Alias" only for configurations which require the technical user to identify itself using an X.509 certificate; it specifies the Certificate Alias to be used as the technical user. This overrides the "Username" and "Password" settings normally used.</p> | Valid certificate alias.                   |
| login-required   | <p>Indicates whether authentication credentials are required to login. The default value is true.</p> <p>For upgraded packages, "login-required=false" gets converted to "login-required=true" and a No-Auth security configuration "DOECNoAuth" is assigned to the upgraded package.</p>                                                                                                                                                                                                                                                                                                               | A read-only property with a value of true. |

### **Web Services Properties**

Configure connection properties for the Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) architectures.

| Name     | Description                                                                                    | Supported Values                                                                                                                      |
|----------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| password | Specifies the password for HTTP basic authentication, if applicable.                           | Password                                                                                                                              |
| address  | Specifies a different URL than the port address indicated in the WSDL document at design time. | HTTP URL address of the Web service. A backslash is appended to the address URL, if it does not already exist in the URL you specify. |
| user     | Specifies the user name for HTTP basic authentication, if applicable.                          | User name                                                                                                                             |

| Name                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Supported Values                                                                                                |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| certificateAlias          | <p>Sets the alias for the SAP Mobile Platform keystore entry that contains the X.509 certificate for SAP Mobile Server's SSL peer identity.</p> <p>If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                               | <p>Use the alias of a certificate stored in the SAP Mobile Server certificate keystore.</p>                     |
| authentication-Preemptive | <p>When credentials are available and this property is set to the default of false, it allows SAP Mobile Server to send the authentication credentials only in response to the receipt of a server message in which the HTTP status is 401 (UNAUTHORIZED) and the WWW-Authenticate header is set. In this case, the message exchange pattern is: request, UNAUTHORIZED response, request with credentials, service response.</p> <p>When set to true and basic credentials are available, this property allows SAP Mobile Server to send the authentication credentials in the original SOAP or REST HTTP request message. The message exchange pattern is: request with credentials, a service response.</p> | <p>False (default)<br/>True</p>                                                                                 |
| Socket Timeout            | <p>The socket timeout value controls the maximum time, in milliseconds, after a Web service operation (REST or SOAP) is allowed to wait for a response from the remote system; if the EIS does not respond in that time, the operation fails and the SMP thread is unblocked.</p>                                                                                                                                                                                                                                                                                                                                                                                                                             | <p>Time in milliseconds (default: 6000).<br/>Range of [0 – 2147483647], where 0 is interpreted as infinity.</p> |

| Name                   | Description                                                                                                                                                                                         | Supported Values                                                                                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| credential.<X>.name    | Defines the EIS connection definition to identify the NamedCredential. For more information on token-based SSO using NamedCredential, see <i>Single Sign-on Using NamedCredential in Security</i> . | <ul style="list-style-type: none"> <li>credential.&lt;X&gt;.name=credential name</li> <li>header:&lt;HTTP header name&gt;</li> <li>cookie: &lt;HTTP cookie name&gt;</li> </ul> |
| credential.<X>.mapping | Defines how the NamedCredential should be propagated to the EIS. For more information on token-based SSO using NamedCredential, see <i>Single Sign-on Using NamedCredential</i> .                   | credential.<X>.mapping=credential mapping to header/cookie                                                                                                                     |
| http.header.X.set      | Specifies the name of the static header to add to the Web service request (SOAP or REST).                                                                                                           | Header name                                                                                                                                                                    |
| http.header.X.value    | Specifies the content of the header to add to the Web service request (SOAP or REST).                                                                                                               | Header value                                                                                                                                                                   |
| http.cookie.X.set      | Specifies the name of the static cookie to add to the Web service request (SOAP or REST).                                                                                                           | Cookie name                                                                                                                                                                    |
| http.cookie.X.value    | Specifies the content of the cookie to add to the Web service request (SOAP or REST).                                                                                                               | Cookie value                                                                                                                                                                   |

### **Proxy Endpoint Properties**

Configure connection properties for the SAP Gateway proxy connection.

| Name     | Description                                | Supported values |
|----------|--------------------------------------------|------------------|
| password | Specifies the password for authentication. | Password         |
| address  | URL address of the Gateway Proxy endpoint. | URL              |
| user     | Specifies the username for authentication. | User name        |

| Name             | Description                                                                                                                                                                                                                                              | Supported values                                                                     |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| certificateAlias | Sets the alias for the SAP Mobile Platform keystore entry that contains the X.509 certificate for SAP Mobile Server's SSL peer identity.<br><br>If you do not set a value, mutual authentication for SSL is not used when connecting to the Web service. | Use the alias of a certificate stored in the SAP Mobile Server certificate keystore. |
| poolSize         | An integer value representing the number of connections that can be made in the connection pool.                                                                                                                                                         | An integer.                                                                          |

## Error Code Reference

Error codes are thrown with each `SUPAdminException`, to allow developers to diagnose what occurred when the exception is thrown. `#{error_sub}` and `#{reason_sub_x}` are placeholders for additional information which will be provided at runtime.

| Numeric Error Code | Message                                                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00001              | Failed to retrieve cluster properties ( <code>#{error_sub}</code> ).                                                                                                                                                                                    |
| 00002              | Failed to authenticate the user ( <code>#{error_sub}</code> ) as SUP administrator.                                                                                                                                                                     |
| 00003              | Failed to validate the security configuration ( <code>#{error_sub}</code> ).                                                                                                                                                                            |
| 00004              | Failed to create the security configuration ( <code>#{error_sub}</code> ).                                                                                                                                                                              |
| 00005              | Cannot create the security provider ( <code>#{error_sub}</code> ). The security configuration ( <code>#{reason_sub}</code> ) is no longer valid or viable.                                                                                              |
| 00006              | Failed to delete the security configuration ( <code>#{error_sub}</code> ).                                                                                                                                                                              |
| 00007              | Cannot delete the selected security provider ( <code>#{error_sub}</code> ). The security configuration ( <code>#{reason_sub}</code> ) is no longer valid or viable.                                                                                     |
| 00008              | Failed to retrieve the selected security configuration ( <code>#{error_sub}</code> ).                                                                                                                                                                   |
| 00009              | Failed to retrieve the security configuration ( <code>#{error_sub}</code> ) for the selected package. The package ( <code>#{reason_sub_1}</code> ) is of the wrong type and therefore this operation ( <code>#{reason_sub_2}</code> ) is not supported. |
| 00010              | Failed to update the selected security configuration ( <code>#{error_sub}</code> ).                                                                                                                                                                     |

| Numeric Error Code | Message                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| 00011              | Cannot delete the selected security provider ({error_sub}). The security configuration ({reason_sub}) is no longer valid or viable. |
| 00012              | Failed to create the authentication provider ({error_sub}). The authentication provider ({reason_sub}) cannot be located.           |
| 00013              | Failed to retrieve the authentication provider ({error_sub}). The selected provider ({reason_sub}) does not exist.                  |
| 00014              | Failed to retrieve the authentication provider ({error_sub}). The authentication provider ({reason_sub}) cannot be located.         |
| 00015              | Failed to create the authorization provider ({error_sub}). The authorization provider ({reason_sub}) cannot be located.             |
| 00016              | Failed to retrieve the authorization provider ({error_sub}). The selected provider ({reason_sub}) does not exist.                   |
| 00017              | Failed to retrieve the authorization provider ({error_sub}). The authorization provider ({reason_sub}) cannot be located.           |
| 00018              | Failed to create the attribution provider ({error_sub}). The attribution provider ({reason_sub}) cannot be located.                 |
| 00019              | Failed to retrieve the attribution provider ({error_sub}). The selected provider ({reason_sub}) does not exist.                     |
| 00020              | Failed to retrieve the attribution provider ({error_sub}). The attribution provider ({reason_sub}) cannot be located.               |
| 00021              | Failed to create the audit provider ({error_sub}). The audit provider ({reason_sub}) cannot be located.                             |
| 00022              | Failed to retrieve the audit provider ({error_sub}). The selected provider ({reason_sub}) does not exist.                           |
| 00023              | Failed to create the audit destination ({error_sub}). The audit provider ({reason_sub}) cannot be located.                          |
| 00024              | Failed to retrieve the audit destination ({error_sub}).                                                                             |
| 00025              | Failed to create the audit filter ({error_sub}). The audit provider ({reason_sub}) cannot be located.                               |
| 00026              | Failed to retrieve the audit filter ({error_sub}). The audit provider ({reason_sub}) cannot be located.                             |

| Numeric Error Code | Message                                                                                                                                                       |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00027              | Failed to create the audit formatter ({error_sub}). The audit provider ({reason_sub}) cannot be located.                                                      |
| 00028              | Failed to retrieve the audit formatter ({error_sub}). The audit provider ({reason_sub}) cannot be located.                                                    |
| 00029              | Cannot delete the selected security provider ({error_sub}). This security provider ({reason_sub}) does not exist.                                             |
| 00030              | Cannot update the selected security provider ({error_sub}). This security provider ({reason_sub}) does not exist.                                             |
| 00031              | Failed to enable the domain ({error_sub}).                                                                                                                    |
| 00032              | Failed to create the domain ({error_sub}).                                                                                                                    |
| 00033              | Failed to delete the domain ({error_sub}).                                                                                                                    |
| 00034              | Failed to retrieve the domain ({error_sub}).                                                                                                                  |
| 00035              | Failed to update the domain ({error_sub}) properties.                                                                                                         |
| 00036              | Failed to retrieve the domain log configuration ({error_sub}).                                                                                                |
| 00037              | Failed to retrieve the domain log ({error_sub}). A configuration property ({reason_sub}) is not supported.                                                    |
| 00038              | Cannot retrieve the domain log configuration ({error_sub}).                                                                                                   |
| 00039              | Failed to update the domain log configuration ({error_sub}).                                                                                                  |
| 00040              | Failed to retrieve the domain log purge time threshold value ({error_sub}).                                                                                   |
| 00041              | Failed to update the domain log purge time threshold value ({error_sub}).                                                                                     |
| 00042              | Package deployment failed ({error_sub}).                                                                                                                      |
| 00043              | Failed to deploy selected package ({error_sub}). You must select a security configuration.                                                                    |
| 00044              | Failed to deploy the selected package ({error_sub}). The package ({reason_sub}) is the wrong type and this operation is not supported.                        |
| 00045              | Failed to deploy package ({error_sub}). Either the deployment unit ({reason_sub_1}) does not exist, or the file ({reason_sub_2}) may be invalid or corrupted. |
| 00046              | Failed to deploy package ({error_sub}). The deployment unit may be invalid or corrupted.                                                                      |

| Numeric Error Code | Message                                                                                                                                                         |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00047              | Failed to deploy package ({error_sub}). The deployment descriptor may be invalid or corrupted.                                                                  |
| 00048              | Failed to deploy package ({error_sub}). A required property ({reason_sub}) has not been configured.                                                             |
| 00049              | Failed to deploy package ({error_sub}). A required property ({reason_sub}) has not been configured.                                                             |
| 00050              | Package export failed ({error_sub}).                                                                                                                            |
| 00051              | Failed to export the selected package ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.       |
| 00052              | Failed to export package ({error_sub}). The file ({reason_sub}) does not exist.                                                                                 |
| 00053              | Package import failed ({error_sub}).                                                                                                                            |
| 00054              | Failed to enable package ({error_sub}).                                                                                                                         |
| 00055              | Failed to enable the selected package ({error_sub}). The package ({reason_sub}) is the wrong type and this operation ({reason_sub}) is not supported.           |
| 00056              | Failed to delete the selected package ({error_sub}).                                                                                                            |
| 00057              | Failed to retrieve package(s).                                                                                                                                  |
| 00058              | Failed to retrieve cache group(s) ({error_sub}).                                                                                                                |
| 00059              | Failed to retrieve the cache group ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.          |
| 00060              | Failed to update cache group(s) ({error_sub}).                                                                                                                  |
| 00061              | Failed to update the cache group ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.            |
| 00062              | Failed to retrieve the cache group schedule ({error_sub}).                                                                                                      |
| 00063              | Failed to retrieve the cache group schedule ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported. |
| 00064              | Failed to update the cache group schedule ({error_sub}).                                                                                                        |
| 00065              | Failed to update the cache group schedule ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.   |
| 00066              | Failed to retrieve the personalization key ({error_sub}).                                                                                                       |

| Numeric Error Code | Message                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00067              | Failed to retrieve the personalization key ({error_sub}). The package ({reason_sub_2}) is the wrong type and this operation ({reason_sub_2}) is not supported.   |
| 00068              | Failed to retrieve the package role mapping ({error_sub}).                                                                                                       |
| 00069              | Failed to retrieve the package role mappings ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported. |
| 00070              | Failed to updated the package role mapping ({error_sub}).                                                                                                        |
| 00071              | Failed to update the package role mapping ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.    |
| 00072              | Failed to retrieve the synchronization group ({error_sub}).                                                                                                      |
| 00073              | Failed to retrieve the synchronization group ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported. |
| 00074              | Failed to update the synchronization group ({error_sub}).                                                                                                        |
| 00075              | Failed to update the synchronization group ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.   |
| 00076              | Failed to retrieve the MBO(s).                                                                                                                                   |
| 00077              | Failed to retrieve the MBO(s). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported.                              |
| 00078              | Failed to retrieve the configured MBO server connection ({error_sub}).                                                                                           |
| 00079              | Failed to delete error history of the MBO ({error_sub}).                                                                                                         |
| 00080              | Failed to retrieve the error history of the MBO ({error_sub}).                                                                                                   |
| 00081              | Failed to retrieve the last valid playback timestamp for the MBO ({error_sub}).                                                                                  |
| 00082              | Failed to retrieve the operation(s) ({error_sub}).                                                                                                               |
| 00083              | Failed to retrieve the configured server connection of the operation ({error_sub}).                                                                              |
| 00084              | Failed to delete the error history of the operation ({error_sub}).                                                                                               |
| 00085              | Failed to retrieve the error history of the operation ({error_sub}).                                                                                             |
| 00086              | Failed to retrieve the last valid playback timestamp for the operation ({error_sub}).                                                                            |
| 00087              | Failed to retrieve package log configuration ({error_sub}).                                                                                                      |
| 00088              | Failed to update package log configuration ({error_sub}).                                                                                                        |



| Numeric Error Code | Message                                                                                                                                                              |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00089              | Failed to enable package synchronization tracing ({error_sub}).                                                                                                      |
| 00090              | Failed to enable package synchronization tracing ({error_sub}). The package ({reason_sub_1}) is the wrong type and this operation ({reason_sub_2}) is not supported. |
| 00091              | Failed to retrieve the package log level ({error_sub}).                                                                                                              |
| 00092              | Failed to update the package log level ({error_sub}).                                                                                                                |
| 00093              | Failed to ping the replication package subscription ({error_sub}).                                                                                                   |
| 00094              | Failed to ping the replication package subscription ({error_sub}). The selected subscription ({reason_sub}) does not exist.                                          |
| 00095              | Failed to delete the replication package subscription ({error_sub}).                                                                                                 |
| 00096              | Failed to delete the replication package subscription(s) ({error_sub}). The selected subscription(s) ({reason_sub}) does(do) not exist.                              |
| 00097              | Failed to retrieve the replication package subscription(s) ({error_sub}).                                                                                            |
| 00098              | Failed to update the replication package subscription ({error_sub}).                                                                                                 |
| 00099              | Failed to create the replication package subscription template ({error_sub}).                                                                                        |
| 00100              | Failed to delete the replication package subscription template(s) ({error_sub}).                                                                                     |
| 00101              | Failed to retrieve the replication package subscription template(s) ({error_sub}).                                                                                   |
| 00102              | Failed to suspend the messaging package subscription(s) ({error_sub}).                                                                                               |
| 00103              | Failed to resume the messaging package subscription(s) ({error_sub}).                                                                                                |
| 00104              | Failed to delete the messaging package subscription(s) ({error_sub}).                                                                                                |
| 00105              | Failed to retrieve the messaging package subscription(s) ({error_sub}).                                                                                              |
| 00106              | Failed to reset the messaging package subscription(s) ({error_sub}).                                                                                                 |
| 00107              | Failed to re-synchronize the DOEC package subscription(s) ({error_sub}).                                                                                             |
| 00108              | Failed to reset the DOEC package subscription(s) ({error_sub}).                                                                                                      |
| 00109              | Failed to delete the DOEC package subscription(s) ({error_sub}).                                                                                                     |
| 00110              | Failed to retrieve the DOEC package subscription(s) ({error_sub}).                                                                                                   |
| 00111              | Failed to update the DOEC package subscription(s) ({error_sub}).                                                                                                     |

| Numeric Error Code | Message                                                                                                                                                                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00112              | Failed to reset the DOEC package subscription(s) ({error_sub}).                                                                                                                                    |
| 00113              | Failed to retrieve the log level for DOEC package subscription ({error_sub}). The package ({reason_sub_1}) is of the wrong type and therefore this operation ({reason_sub_2}) is not supported.    |
| 00114              | Failed to update the log level for DOEC package subscription(s) ({error_sub}).                                                                                                                     |
| 00115              | Failed to retrieve the log level for DOEC package subscription(s) ({error_sub}). The package ({reason_sub_1}) is of the wrong type and therefore this operation ({reason_sub_2}) is not supported. |
| 00116              | Failed to connect to the configured server connection ({error_sub}).                                                                                                                               |
| 00117              | Failed to create the server connection ({error_sub}).                                                                                                                                              |
| 00118              | Failed to create the server connection ({error_sub}). The server connection ({reason_sub_1}) is of the wrong type and therefore this operation ({reason_sub_2}) is not supported.                  |
| 00119              | Failed to delete the server connection ({error_sub}).                                                                                                                                              |
| 00120              | Failed to delete the server connection ({error_sub}). The server connection ({reason_sub_1}) is of the wrong type and therefore this operation ({reason_sub_2}) is not supported.                  |
| 00121              | Failed to retrieve the server connection(s) ({error_sub}).                                                                                                                                         |
| 00122              | Failed to update the server connection ({error_sub}).                                                                                                                                              |
| 00123              | Failed to retrieve the domain-level role mapping ({error_sub}).                                                                                                                                    |
| 00124              | Failed to update the domain-level role mapping ({error_sub}).                                                                                                                                      |
| 00125              | Failed to create the domain administrator ({error_sub}).                                                                                                                                           |
| 00126              | Failed to delete the domain administrator ({error_sub}).                                                                                                                                           |
| 00127              | Failed to retrieve the domain administrator(s) ({error_sub}).                                                                                                                                      |
| 00128              | Failed to update the domain administrator ({error_sub}).                                                                                                                                           |
| 00129              | Failed to authenticate the user as SUP domain administrator ({error_sub}).                                                                                                                         |
| 00130              | Failed to create the monitoring profile ({error_sub}).                                                                                                                                             |
| 00131              | Failed to delete the monitoring profile ({error_sub}).                                                                                                                                             |

| Numeric Error Code | Message                                                                                                   |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| 00132              | Failed to retrieve the monitoring profile(s) ({error_sub}).                                               |
| 00133              | Failed to update the monitoring profile ({error_sub}).                                                    |
| 00134              | Failed to update the monitoring profile ({error_sub}). A property ({reason_sub}) uses an incorrect value. |
| 00135              | Failed to export monitoring data ({error_sub}).                                                           |
| 00136              | Failed to delete monitoring data ({error_sub}).                                                           |
| 00137              | Failed to retrieve monitoring data ({error_sub}). A required parameter ({reason_sub}) is missing.         |
| 00138              | Failed to retrieve monitoring data ({error_sub}). A required parameter ({reason_sub}) is not expected.    |
| 00139              | Failed to retrieve monitoring data ({error_sub}). A required parameter ({reason_sub}) is empty            |
| 00140              | Failed to retrieve the replication package monitoring data ({error_sub}).                                 |
| 00141              | Failed to retrieve the replication package history ({error_sub}).                                         |
| 00142              | Failed to retrieve the replication package performance ({error_sub}).                                     |
| 00143              | Failed to retrieve the messaging package monitoring data ({error_sub}).                                   |
| 00144              | Failed to retrieve the messaging package history ({error_sub}).                                           |
| 00145              | Failed to retrieve the messaging package performance data ({error_sub}).                                  |
| 00146              | Failed to retrieve the messaging queue statistics ({error_sub}).                                          |
| 00147              | Failed to retrieve the data change notification history data ({error_sub}).                               |
| 00148              | Failed to retrieve the data change notification performance data ({error_sub}).                           |
| 00149              | Failed to retrieve the package statistics ({error_sub}).                                                  |
| 00150              | Failed to retrieve the operation statistics ({error_sub}).                                                |
| 00151              | Failed to retrieve user access history ({error_sub}).                                                     |
| 00152              | Failed to retrieve the package-level cache group performance data ({error_sub}).                          |
| 00153              | Failed to retrieve the package-level cache group statistics ({error_sub}).                                |
| 00154              | Failed to retrieve MBO-level cache group statistics ({error_sub}).                                        |

| Numeric Error Code | Message                                                                                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00155              | Failed to start SAP Mobile Server ({error_sub}). The path ({reason_sub}) to the server does not exist.                                                                            |
| 00156              | Failed to start SAP Mobile Server ({error_sub}). SAP Control Center is not installed on the same host computer, and this operation ({reason_sub}) cannot be performed remotely.   |
| 00157              | Failed to connect to SAP Mobile Server ({error_sub}).                                                                                                                             |
| 00158              | Failed to stop SAP Mobile Server ({error_sub}).                                                                                                                                   |
| 00159              | Failed to stop SAP Mobile Server ({error_sub}). The path ({reason_sub}) to the server does not exist.                                                                             |
| 00160              | Failed to stop SAP Mobile Server ({error_sub}). SAP Control Center is not installed on the same host computer, and this operation ({reason_sub}) cannot be performed remotely.    |
| 00161              | Failed to restart SAP Mobile Server ({error_sub}).                                                                                                                                |
| 00162              | Failed to restart SAP Mobile Server ({error_sub}). The path ({reason_sub}) to the server does not exist.                                                                          |
| 00163              | Failed to restart SAP Mobile Server ({error_sub}). SAP Control Center is not installed on the same host computer, and this operation ({reason_sub}) cannot be performed remotely. |
| 00164              | Failed to suspend SAP Mobile Server ({error_sub}).                                                                                                                                |
| 00165              | Failed to resume SAP Mobile Server ({error_sub}).                                                                                                                                 |
| 00166              | Failed to retrieve SAP Mobile Server properties ({error_sub}).                                                                                                                    |
| 00167              | Failed to create the SAP Mobile Server configuration ({error_sub}). The specified configuration type ({reason_sub}) does not exist.                                               |
| 00168              | Failed to create the SAP Mobile Server configuration ({error_sub}). A parameter ({reason_sub}) is not expected.                                                                   |
| 00169              | Failed to delete the SAP Mobile Server configuration ({error_sub}). The specified configuration ({reason_sub}) does not exist.                                                    |
| 00170              | Failed to update the SAP Mobile Server configuration ({error_sub}).                                                                                                               |
| 00171              | Failed to update the SAP Mobile Server configuration ({error_sub}). The selected SAP Mobile Server ({reason_sub}) does not exist.                                                 |

| Numeric Error Code | Message                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 00172              | Failed to update the SAP Mobile Server configuration ({error_sub}). A property value ({reason_sub}) in the configuration is not supported.          |
| 00173              | Failed to initialize the administration listener ({error_sub}). The listener ({reason_sub}) has not been configured.                                |
| 00174              | Failed to secure the administration listener ({error_sub}). The listener ({reason_sub}) has not been configured.                                    |
| 00175              | Failed to retrieve the key store configuration ({error_sub}). The key store ({reason_sub}) has not been configured.                                 |
| 00176              | Failed to retrieve the key store configuration ({error_sub}). The key store configuration ({reason_sub}) is corrupted.                              |
| 00177              | Failed to retrieve the trust store configuration ({error_sub}). The trust store ({reason_sub}) is not configured.                                   |
| 00178              | Failed to retrieve the trust store configuration ({error_sub}). The key store configuration ({reason_sub}) is corrupted.                            |
| 00179              | Failed to retrieve the cache database configuration ({error_sub}). The cache database ({reason_sub}) has not been configured.                       |
| 00180              | Failed to retrieve the replication synchronization server configuration ({error_sub}). The synchronization server ({reason_sub}) is not configured. |
| 00181              | Failed to retrieve the messaging synchronization server configuration ({error_sub}). The synchronization server ({reason_sub}) is not configured.   |
| 00182              | Failed to retrieve the replication push notification configuration ({error_sub}). The replication push component ({reason_sub}) is not configured.  |
| 00183              | Failed to retrieve the replication push notification gateway configuration ({error_sub}). The gateway ({reason_sub}) is not available.              |
| 00184              | Failed to validate the SAP Mobile Server log configuration ({error_sub}).                                                                           |
| 00185              | Failed to retrieve the SAP Mobile Server log configuration ({error_sub}).                                                                           |
| 00186              | Failed to update the SAP Mobile Server log configuration ({error_sub}).                                                                             |
| 00187              | Failed to create the SAP Mobile Server log appender ({error_sub}). The log appender type is not installed.                                          |

| Numeric Error Code | Message                                                                                                                 |
|--------------------|-------------------------------------------------------------------------------------------------------------------------|
| 00188              | Failed to create the SAP Mobile Server log appender ({error_sub}). The log bucket type ({reason_sub}) is not installed. |
| 00189              | Failed to delete the SAP Mobile Server log appender ({error_sub}). The log appender ({reason_sub}) does not exist.      |
| 00190              | Failed to update the SAP Mobile Server log appender ({error_sub}). The log appender ({reason_sub}) does not exist.      |
| 00191              | Failed to update the SAP Mobile Server log appender ({error_sub}). The log bucket type ({reason_sub}) is not installed. |
| 00192              | Failed to create the SAP Mobile Server log bucket ({error_sub}). The log appender ({reason_sub}) does not exist.        |
| 00193              | Failed to create the SAP Mobile Server log file ({error_sub}). The log appender type is not installed.                  |
| 00194              | Failed to delete the SAP Mobile Server log bucket ({error_sub}). The log bucket ({reason_sub}) does not exist.          |
| 00195              | Failed to update the SAP Mobile Server log bucket ({error_sub}). The log bucket ({reason_sub}) does not exist.          |
| 00196              | Failed to delete the SAP Mobile Server log ({error_sub}).                                                               |
| 00197              | Failed to retrieve the SAP Mobile Server log ({error_sub}).                                                             |
| 00198              | Failed to create the SAP Mobile Server log filter ({error_sub}).                                                        |
| 00199              | Failed to retrieve the SAP Mobile Server log filter ({error_sub}).                                                      |
| 00200              | Failed to connect to the SAP Control Center ({error_sub}).                                                              |
| 00201              | Failed to borrow a connection from the connection pool of SAP Control Center ({error_sub}).                             |
| 00202              | Failed to return a connection to the connection pool of SAP Control Center ({error_sub}).                               |
| 00203              | Cannot retrieve the SAP Mobile Server location ({error_sub}). The login has not authenticated.                          |
| 00204              | Cannot retrieve the SAP Mobile Server location ({error_sub}). The login is not authorized to access this server.        |

| Numeric Error Code | Message                                                                                                               |
|--------------------|-----------------------------------------------------------------------------------------------------------------------|
| 00205              | Cannot retrieve the SAP Mobile Server location ({error_sub}). The SAP Control Center connection has failed.           |
| 00206              | Failed to load the file ({error_sub}). The file does not exist.                                                       |
| 00207              | Failed to load the the administration interface ({error_sub}).                                                        |
| 00208              | Failed to initialize the administration interface ({error_sub}).                                                      |
| 00209              | Failed to retrieve data for administration interface ({error_sub}).                                                   |
| 00210              | Failed to validate the property ({error_sub}). A value other than NULL is required.                                   |
| 00211              | Failed to validate the property ({error_sub}). A monitored target is required.                                        |
| 00212              | The value entered exceeds the maximum value allowed ({error_sub}).                                                    |
| 00213              | The value entered exceeds the minimum value allowed ({error_sub}).                                                    |
| 00214              | Failed to invoke method ({error_sub}). The parameter ({reason_sub}) is either the wrong type or uses the wrong value. |
| 00215              | Failed to invoke method ({error_sub}). The parameter ({reason_sub}) requires a value.                                 |
| 00216              | Failed to invoke method ({error_sub}). The method ({reason_sub}) is not implemented.                                  |
| 00217              | Failed to invoke method ({error_sub}). Access to the method ({reason_sub}) is denied.                                 |
| 00218              | Monitoring data retrieve failed.                                                                                      |
| 00219              | Messaging-based synchronization server Hybrid App ({error_sub}) retrieve failed.                                      |
| 00220              | Messaging-based synchronization server Hybrid App error ({error_sub}) delete failed.                                  |
| 00221              | Java virtual machine start up options retrieve failed because java virtual machine start up options does not exist.   |
| 00222              | Object ({error_sub}) create failed because parameter ({reason_sub}) not supported.                                    |
| 00223              | Messaging-based synchronization server apple push notification certificate ({error_sub}) list failed.                 |
| 00224              | Device ({error_sub}) list failed.                                                                                     |
| 00225              | Messaging-based synchronization server email ({error_sub}) retrieve failed.                                           |

| Numeric Error Code | Message                                                                                                             |
|--------------------|---------------------------------------------------------------------------------------------------------------------|
| 00226              | Device ({error_sub}) retrieve failed.                                                                               |
| 00227              | SSL Security profile ({error_sub}) delete failed because parameter ({reason_sub}) null value not allowed.           |
| 00228              | Messaging-based synchronization server template ({error_sub}) list failed.                                          |
| 00229              | Secure administration listener ({error_sub}) update failed because parameter ({reason_sub}) null value not allowed. |
| 00230              | Messaging-based synchronization server Hybrid App context variable ({error_sub}) update failed.                     |
| 00231              | Messaging-based synchronization server template ({error_sub}) retrieve failed.                                      |
| 00232              | User ({error_sub}) delete failed.                                                                                   |
| 00233              | Replication notification gateway update failed because parameter ({reason_sub}) null value not allowed.             |
| 00234              | HTTP listener(s) ({error_sub}) delete failed because parameter ({reason_sub}) null value not allowed.               |
| 00235              | SSL Security profile ({error_sub}) update failed because parameter ({reason_sub}) null value not allowed.           |
| 00236              | Replication notification gateway enable failed because parameter ({reason_sub}) null value not allowed.             |
| 00237              | Messaging-based synchronization server Hybrid App error ({error_sub}) list failed.                                  |
| 00238              | Messaging-based synchronization server Hybrid App ({error_sub}) update failed.                                      |
| 00239              | Messaging-based synchronization server Hybrid App ({error_sub}) delete failed.                                      |
| 00240              | Java virtual machine start up options retrieve failed because java virtual machine start up options corrupted.      |
| 00241              | HTTP listener(s) ({error_sub}) update failed because parameter ({reason_sub}) null value not allowed.               |
| 00242              | Messaging-based synchronization server Hybrid App ({error_sub}) create failed.                                      |
| 00243              | Messaging-based synchronization server ({error_sub}) list failed.                                                   |
| 00244              | Messaging-based synchronization server apple push notification certificate ({error_sub}) update failed.             |



| Numeric Error Code | Message                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------|
| 00245              | Messaging-based synchronization server email ({error_sub}) update failed.                                    |
| 00246              | Messaging-based synchronization server apple push notification certificate ({error_sub}) delete failed.      |
| 00247              | Device ({error_sub}) update failed.                                                                          |
| 00248              | Device ({error_sub}) delete failed.                                                                          |
| 00249              | Secure HTTP listener(s) ({error_sub}) delete failed because parameter ({reason_sub}) null value not allowed. |
| 00250              | Messaging-based synchronization server Hybrid App display name ({error_sub}) update failed.                  |
| 00251              | Messaging-based synchronization server apple push notification certificate ({error_sub}) create failed.      |
| 00252              | Messaging-based synchronization server email ({error_sub}) create enabled.                                   |
| 00253              | Device ({error_sub}) create failed.                                                                          |
| 00254              | Monitored object ({error_sub}) update failed because parameter ({reason_sub}) invalid.                       |
| 00255              | License retrieve failed.                                                                                     |
| 00256              | Monitored object ({error_sub}) update failed because parameter ({reason_sub}) null value not allowed.        |
| 00257              | Object ({error_sub}) create failed because parameter ({reason_sub}) null value not allowed.                  |
| 00258              | Messaging-based synchronization server Hybrid App context variable ({error_sub}) list failed.                |
| 00259              | Messaging-based synchronization server template ({error_sub}) delete failed.                                 |
| 00260              | User ({error_sub}) list failed.                                                                              |
| 00261              | Secure HTTP listener(s) ({error_sub}) update failed because parameter ({reason_sub}) null value not allowed. |
| 00262              | Messaging-based synchronization server email ({error_sub}) validate failed.                                  |
| 00263              | Administration listener update failed because parameter ({reason_sub}) null value not allowed.               |

| Numeric Error Code | Message                                                                                                                                                     |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00264              | Replication notifier retrieve failed because parameter ({reason_sub}) null value not allowed.                                                               |
| 00265              | Messaging-based synchronization server Hybrid App ({error_sub}) list failed.                                                                                |
| 00266              | Failed to start SAP Mobile Server ({error_sub}). Server command and control ({reason_sub}) is in progress.                                                  |
| 00267              | Failed to stop SAP Mobile Server ({error_sub}). Server command and control ({reason_sub}) is in progress.                                                   |
| 00268              | Failed to restart SAP Mobile Server ({error_sub}). Server command and control ({reason_sub}) is in progress.                                                |
| 00269              | Cannot deploy the package as ({reason_sub}) type. When deployment unit contains an Online Cache Group, the package must be deployed as a MESSAGING package. |
| 00270              | Failed to delete the selected package ({error_sub}) because of invalid SAP credentials.                                                                     |
| 00271              | Failed to delete the Hybrid App ({error_sub}) because of invalid SAP credentials.                                                                           |
| 00272              | Failed to delete the DOEC package subscription(s) ({error_sub}) because of invalid SAP credentials.                                                         |
| 00273              | Failed to re-synchronize the DOEC package subscription(s) ({error_sub}) because of invalid SAP credentials.                                                 |
| 00274              | Failed to enable the scheduled purge task ({error_sub}).                                                                                                    |
| 00275              | Failed to retrieve the scheduled purge task ({error_sub}) enable status.                                                                                    |
| 00276              | Failed to purge the synchronization cache.                                                                                                                  |
| 00277              | Failed to purge the online cache.                                                                                                                           |
| 00278              | Failed to purge the client log.                                                                                                                             |
| 00279              | Failed to purge the error history.                                                                                                                          |
| 00280              | Failed to purge the subscription.                                                                                                                           |
| 00281              | Failed to retrieve the purge option ({error_sub}).                                                                                                          |
| 00282              | Failed to set the purge option ({error_sub}).                                                                                                               |
| 00283              | Failed to retrieve the purge task ({error_sub}) schedule.                                                                                                   |
| 00284              | Failed to update the purge task ({error_sub}) schedule.                                                                                                     |

| Numeric Error Code | Message                                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 00285              | Failed to purge the package's synchronization cache.                                                                                     |
| 00286              | Failed to purge the package's online cache.                                                                                              |
| 00287              | Failed to purge the package's client log.                                                                                                |
| 00288              | Failed to purge the package's error history.                                                                                             |
| 00289              | Failed to purge the package's subscription.                                                                                              |
| 00290              | Failed to purge devices.                                                                                                                 |
| 00291              | Failed to purge users.                                                                                                                   |
| 00292              | Failed to replace Hybrid App ( <code>{error_sub}</code> ) certificate.                                                                   |
| 00293              | Failed to replace Hybrid App certificate. The Hybrid App does not support certificate-based authentication.                              |
| 00294              | Messaging-based synchronization server Hybrid App context variable ( <code>{error_sub}</code> ) update failed. The parameter is invalid. |
| 00297              | Failed to do the operation, because the login user does not have authorization to do the operation.                                      |

## Backward Compatibility

---

When upgrading from a previous version of SAP Mobile Platform, certain APIs are no longer supported, or are supported with limitations.

These APIs are no longer supported:

- `SUPMobileWorkflow`: all methods of this class throw an `UnsupportedOperationException` when called. Use the new `SUPMobileHybridApp` API instead.
- `SUPDeviceUser`: all methods of this class throw an `UnsupportedOperationException` when called.
- `SUPApplication`: the following methods of this class throw an exception when called.
  - `getApplicationConnectionTemplateSettings`
  - `getApplicationConnectionSettings`
  - `createApplicationConnectionTemplate`
- `SUPServerConfiguration`: the following methods of this class throw an `UnsupportedOperationException` when called.

## Management API

- getReplicationSyncServerConfiguration
- updateReplicationSyncServerConfiguration
- getMessagingSyncServerConfiguration
- updateMessagingSyncServerConfiguration
- getConsolidatedDatabaseConfiguration
- getAdministrationListenerConfiguration
- updateAdministrationListenerConfiguration
- getHTTPListenerConfigurations
- addHTTPListenerConfiguration
- deleteHTTPListenerConfiguration
- updateHTTPListenerConfiguration
- getSecureHTTPListenerConfigurations
- addSecureHTTPListenerConfiguration
- deleteSecureHTTPListenerConfiguration
- updateSecureHTTPListenerConfiguration
- getSSLSecurityProfileConfigurations
- addSSLSecurityProfileConfiguration
- deleteSSLSecurityProfileConfiguration
- updateSSLSecurityProfileConfiguration
- getKeyStoreConfiguration
- updateKeyStoreConfiguration
- getTrustStoreConfiguration
- updateTrustStoreConfiguration
- getApplePushNotificationConfigurations
- addApplePushNotificationConfiguration
- deleteApplePushNotificationConfiguration
- updateApplePushNotificationConfiguration

These APIs are supported with limitations:

Table 349. SUPDomain

| Method | Reason              | Notes                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Deploy | Package unification | <p>You can still deploy the deployment units of the previous version. However, the package type passed to this method is ignored.</p> <p>Clients of the previous version see the newly deployed package as an RBS package, however, the SAP Mobile Server treats it as a unified package.</p> <p>Packages deployed prior to the upgrade retain their type information for the older version of the client.</p> |

Table 350. SUPPackage

| Method          | Reason              | Notes                                                                                                                                                                                                                                                                    |
|-----------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| getProperties() | Package unification | <p>Clients of the previous version see the newly deployed package as an RBS package, however, the SAP Mobile Server treats it as a unified package.</p> <p>Packages deployed prior to the upgrade retain their type information for the older version of the client.</p> |

---

**Note:** If using the 2.0 version of the Management API client to connect to a SAP Mobile Platform 2.1 installation, you must get the uaf-client.jar shipped with SAP Mobile Platform 2.0.1 in the Management API client libraries folder.

---



# Notification API

You can use the Notification API to make requests from the EIS for SAP Mobile Platform to send push notifications to devices. The Notification API is an HTTP API that consists of a URL and, depending on notification mode may contain HTTP headers and a body. The header fields send notification data from the backend as a generic HTTP header for any device, or as device-specific HTTP headers.

## Notification Mode

---

The notification mode specifies how native notifications or payload push notifications to registered devices are delivered. You select the notification mode when you create an application connection.

- Only native notifications – allows third party applications or EIS to deliver native notifications directly through the HTTP notification channel: BlackBerry (BES), Apple (APNS), or Android (GCM) to the device.  
In this notification mode, you send notification data through the request headers. There is no body in this notification mode.
- Only online/payload push – allows third party applications or EIS to deliver data payload push notifications to an online device.  
In this notification mode, no headers are required, and you send the business payload through the HTTP body.
- Online/payload push with native notifications – allows both payload and native notifications to be delivered to the device.  
In this notification mode, no headers are required, and you send the business payload through the HTTP body.

---

**Note:** Apple and Google do not recommend payload delivery over their systems:

- Data may not be delivered.
- Data is delivered out of sequence.
- If enabled, the actual payloads must be small.  
For example, GCM makes no guarantees about delivery or order of messages. While you might use this feature to inform an instant messaging application that the user has new messages, you probably would not use it to pass actual messages.

RIM supports guaranteed delivery, including callbacks to notify SAP Mobile Server that the message was delivered or when it failed. However, this is a different message format. RIM messaging has many limitations including packet size, number of packets for a single user that BES keeps, number of packets BES keeps for all users, and so on. Therefore, BES should also only be used to send the notification, but not to send payloads.

Refer to the appropriate platform documentation (APNS, BES, or GCM) for additional information.

---

## Notification URL

---

Send a push notification request from the EIS using an HTTP POST to the push notification URL.

```
http://{unwired_server_name}:{unwired_server_port}/notifications/v1/{appid}
```

Parameters:

- **appid** – The application connection ID of the application instance interacting with the service.

---

**Note:** You can also send the Application Connection ID as the "x-sap-push-deviceID" header.

---

## Notification Headers

---

You can send notification data from the backend as a generic HTTP header for any device, or as device-specific HTTP headers.

### Apple Header Fields

The structure of HTTP headers for sending Apple Push Notification Service (APNS) notifications through SAP Mobile Platform.

- **<x-sup-apns-alert>** – The alert can consist of either a JSON-formatted request, or as these separate headers:
  - **<x-sup-apns-alert-body>** – The text of the alert message.
  - **<x-sup-apns-alert-action-loc-key>** – If a string is specified, displays an alert with two buttons, Close and View. iOS uses the string as a key to get a localized string in the current localization to use for the right button's title instead of "View." If the value is null, the system displays an alert with a single OK button that dismisses the alert when tapped.
  - **<x-sup-apns-alert-loc-key>** – A key to an alert-message string in a `Localizable.strings` file for the current localization.
  - **<x-sup-apns-alert-loc-args>** – Variable string values to appear in place of format specifiers in `<x-sup-apns-alert-action-loc-key>`.
  - **<x-sup-apns-alert-launch-image>** – The filename of an image file in the application bundle; it may include the extension or omit it. The image is used as the launch image when users tap the action button or move the action slider. If this property is not specified, the system either uses the previous snapshot, uses the image identified by the



UILaunchImageFile key in the application's `Info.plist` file, or falls back to `Default.png`. This property was added in iOS 4.0.

- **<x-sup-apns-badge>** – Number to be displayed as the badge on the application icon.
- **<x-sup-apns-sound>** – Name of the sound file in the application bundle.
- **<x-sup-apns-data>** – Custom payload data values that must use the JSON structured and primitive types: dictionary (object), array, string, number, and Boolean.

For more information on APNS headers, see <http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html>

## Google Header Fields

The list of HTTP headers to be used in the HTTP request for sending Google Cloud Messaging (GCM) notifications.

- **<x-sup-gcm-collapsekey>** – An arbitrary string (such as "Updates Available") that is used to collapse a group of like messages when the device is offline, so that only the last message gets sent to the client.
- **<x-sup-gcm-data>** – Payload data, expressed as parameters prefixed with `data` and suffixed as the key.
- **<x-sup-gcm-delaywhileidle>** – Optional. Requests that messages not be delivered until the device becomes active. Represented as `1` for true, or any other value for false. The default value is false.
- **<x-sup-gcm-timetolive>** – Specifies how long (in seconds) the message should be kept on GCM storage if the device is offline.
- When you do not send the the collapse key, the default 'Collapse\_Key' value is used, "Updates Available."

## BlackBerry Header Fields

The list of HTTP headers to be used in the HTTP request for sending BlackBerry notifications.

- **<x-sup-rim-data>** – A binary value (base64 encoded).

## Generic Header Fields

You can use the generic header in the HTTP request to send any type of notification. Unless you require platform-specific features in your notification (for example, the display of a button with specific text), you can more easily use the generic header.

- **<x-sup-data>** – The notification data.

SAP Mobile Platform interprets the header for each device type as follows:

- APNS: custom payload data values that use the JSON structured and primitive types: dictionary (object), array, string, number, and Boolean.

## Notification API

- GCM: payload data, expressed as parameters prefixed with data and suffixed as the key.
- Blackberry: a binary value (base64 encoded).

# Index

- `_acf` variableAbstractAuthzChecker class [Sybase Unwired Platform Security API Reference API] 32
- `_addedPrincipalSet` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 170
- `_addedPrivateCredentialSet` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 170
- `_addedPublicCredentialSet` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 170
- `_attributers` variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 211
- `_attributes` variableAbstractAttributed class [Sybase Unwired Platform Security API Reference API] 151
- `_auditConfigurations` variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 211
- `_authorizers` variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 211
- `_automaticBootstrapComplete` variableAbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] 157
- `_bootstrapBaseFile` variableBootstrap class [Sybase Unwired Platform Security API Reference API] 208
- `_bootstrapBaseURL` variableBootstrap class [Sybase Unwired Platform Security API Reference API] 208
- `_bootstrapProps` variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 95
- `_callback` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
- `_cause` variableSecException class [Sybase Unwired Platform Security API Reference API] 321
- `_causeList` variableSecException class [Sybase Unwired Platform Security API Reference API] 321
- `_certificateAuthenticationEnabled` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
- `_certs` variableCertificatePrincipal class [Sybase Unwired Platform Security API Reference API] 66
- `_charset` variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_children` variableHierarchicalItem< C, P, S > class [Sybase Unwired Platform Security API Reference API] 84
- `_cipherProvider` variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_cipherTransform` variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_code` variableAuthenticationFailureWarningImpl class [Sybase Unwired Platform Security API Reference API] 200
- `_code` variableSecLoginExceptionAuthenticationFailureWarningImpl class [Sybase Unwired Platform Security API Reference API] 259
- `_commitSuccess` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
- `_configCache` variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 96
- `_configProvider` variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 96

## Index

- `_csiConfigOptions`
    - variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 211
- `_dataServiceProviders`
  - variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 212
- `_date` variablePasswordExpirationWarningImpl class [Sybase Unwired Platform Security API Reference API] 231
- `_dateFormatter` variableFactoryHolder class [Sybase Unwired Platform Security API Reference API] 130
- `_destination` variableAuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] 210
- `_documentBuilder` variableFactoryHolder class [Sybase Unwired Platform Security API Reference API] 131
- `_environment` variableAbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] 34
- `_failedBootstrap`
  - variableAbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] 157
- `_failedBootstrapException`
  - variableAbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] 157
- `_filter` variableAuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] 210
- `_formatter` variableAuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] 210
- `_id` variableBasicSecIDPrincipal class [Sybase Unwired Platform Security API Reference API] 205
- `_initProviderMap` variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 96
- `_keyStoreAlias`
  - variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_keyStoreAliasPassword`
  - variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_keyStoreLocation`
  - variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_keyStorePassword`
  - variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_keyStoreProvider`
  - variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 180
- `_keyStoreType`
  - variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 181
- `_loginModuleEntries`
  - variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 212
- `_loginSuccess` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
- `_msg` variableSecWarningImpl class [Sybase Unwired Platform Security API Reference API] 264
- `_name` variableBasicSecNamePrincipal class [Sybase Unwired Platform Security API Reference API] 206
- `_name` variableCertificatePrincipal class [Sybase Unwired Platform Security API Reference API] 66
- `_name` variableHierarchialItem< C, P, S > class [Sybase Unwired Platform Security API Reference API] 84
- `_object` variableBootstrap class [Sybase Unwired Platform Security API Reference API] 208
- `_options` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
- `_parent` variableHierarchialItem< C, P, S > class [Sybase Unwired Platform Security API Reference API] 85
- `_pkg` variableAbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] 34

- `_profilers` variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 212
  - `_repository` variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 96
  - `_requests` variableLogicalAndAuthzRequest class [Sybase Unwired Platform Security API Reference API] 43
  - `_requests` variableLogicalOrAuthzRequest class [Sybase Unwired Platform Security API Reference API] 45
  - `_roleMappers` variableConfigurationProperties class [Sybase Unwired Platform Security API Reference API] 212
  - `_sharedState` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
  - `_stringWriter` variableFactoryHolder class [Sybase Unwired Platform Security API Reference API] 131
  - `_subject` variableAbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] 34
  - `_subject` variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 171
  - `_transformer` variableFactoryHolder class [Sybase Unwired Platform Security API Reference API] 131
  - `_useCert` variableAbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] 181
- A**
- ABSTAIN variableDecision class [Sybase Unwired Platform Security API Reference API] 291
  - AbstractAttributed class [Sybase Unwired Platform Security API Reference API] `_attributes` variable 151
  - AbstractAttributed class [Sybase Unwired Platform Security API Reference API] description 147
  - AbstractAttributer class [Sybase Unwired Platform Security API Reference API] description 151
  - AbstractAuthorizer class [Sybase Unwired Platform Security API Reference API] description 155
  - AbstractAuthzChecker class [Sybase Unwired Platform Security API Reference API] `_acf` variable 32
  - AbstractAuthzChecker class [Sybase Unwired Platform Security API Reference API] description 30
  - AbstractAuthzChecker class [Sybase Unwired Platform Security API Reference API] MESSAGES variable 32
  - AbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] `_environment` variable 34
  - AbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] `_pkg` variable 34
  - AbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] `_subject` variable 34
  - AbstractAuthzRequest class [Sybase Unwired Platform Security API Reference API] description 32
  - AbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] `_automaticBootstrapComplete` variable 157
  - AbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] `_failedBootstrap` variable 157
  - AbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] `_failedBootstrapException` variable 157
  - AbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] `BOOTSTRAP_CONFIGURATION_PROPERTY_FILE` variable 158
  - AbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] `BOOTSTRAP_CONFIGURATION_PROPERTY_URL` variable 158
  - AbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] description 156

## Index

- AbstractFactoryRetriever class [Sybase Unwired Platform Security API Reference API] description 158
- AbstractFileConfiguration class [Sybase Unwired Platform Security API Reference API] description 159
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_addedPrincipalSet variable 170
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_addedPrivateCredentialSet variable 170
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_addedPublicCredentialSet variable 170
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_callback variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_certificateAuthenticationEnabled variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_commitSuccess variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_loginSuccess variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_options variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_sharedState variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] \_subject variable 171
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] CLEARPASS\_OPTION variable 172
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] description 164
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] PASSWORD\_SHARED\_KEY variable 172
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] STOREPASS\_OPTION variable 172
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] TRYFIRSTPASS\_OPTION variable 172
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] USEFIRSTPASS\_OPTION variable 172
- AbstractLoginModule class [Sybase Unwired Platform Security API Reference API] USERNAME\_SHARED\_KEY variable 173
- AbstractPrincipalContextRetriever class [Sybase Unwired Platform Security API Reference API] description 173
- AbstractPrincipalContextRetriever class [Sybase Unwired Platform Security API Reference API] ejbContextAvailable variable 174
- AbstractProfiler class [Sybase Unwired Platform Security API Reference API] description 175
- AbstractRoleMapper class [Sybase Unwired Platform Security API Reference API] description 176
- AbstractSecureDataServices class [Sybase Unwired Platform Security API Reference API] description 177
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] \_charset variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] \_cipherProvider variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] \_cipherTransform variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] \_keyStoreAlias variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] \_keyStoreAliasPassword variable 180

- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] `_keyStoreLocation` variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] `_keyStorePassword` variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] `_keyStoreProvider` variable 180
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] `_keyStoreType` variable 181
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] `_useCert` variable 181
- AbstractSecureFileConfiguration class [Sybase Unwired Platform Security API Reference API] description 178
- `ACTION_ACCESS` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 189
- `ACTION_ACTIVATE` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 189
- `ACTION_AUTHENTICATION` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_AUTHENTICATION_PROVIDER` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_AUTHORIZATION_AUTHORIZATION` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_AUTHORIZATION_RESOURCE` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_AUTHORIZATION_RESOURCE_PROVIDER` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_AUTHORIZATION_ROLE` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_AUTHORIZATION_ROLE_PROVIDE` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 190
- `ACTION_CREATE` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_CREATE_CIPHER` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_CREATE_DIGEST` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_CREATE_PROVIDER` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_CREATE_SIGNATURE` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_LOGOUT` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_MODIFY` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 191
- `ACTION_MODIFY_PROVIDER` variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- active security providers 556
- `ADMIN_ROLE_NAME` variableSecContext interface [Sybase Unwired Platform Security API Reference API] 311
- administration client API 327
- advanced device properties 677
- AgentContext using 348
- `ALGORITHM_PARAMETERS` variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 235

## Index

- AlgorithmSet class [Sybase Unwired Platform Security API Reference API]
  - cipherTransformation variable 59
- AlgorithmSet class [Sybase Unwired Platform Security API Reference API] description 58
- AlgorithmSet class [Sybase Unwired Platform Security API Reference API]
  - messageDigestAlgorithm variable 59
- alias, certificate 699
- AlreadyExistsException class [Sybase Unwired Platform Security API Reference API] description 55
- APNSAppSettingsVO() 548
- Apple Push Notification
  - add configuration 546
  - APNSAppSettingsVO value object 548
  - certificate names 547
  - delete configuration 546
  - retrieve configurations 545
  - update configuration 547
- Apple push notification properties 674
- application connection templates
  - creating 486
  - deletion 487
  - retrieving 485
  - settings 486
- application connections
  - assign application customization resource bundle 489
  - cloning 476
  - deletion 480
  - locking or unlocking 480
  - re-registers 478
  - registering 477
  - retrieving 475
  - settings 479
  - unassign application customization resource bundle 489
- application connections templates
  - assign application customization resource bundle 489
  - unassign application customization resource bundle 489
- application settings properties 674
- application users
  - deletion 450
  - retrieval of a list 449, 464
- applications
  - adding packages 454
  - Agentry configuration 469
  - Agentry definition files 472
  - Agentry locale files 473
  - Agentry log properties 467
  - Agentry resource files 474
  - application connections 476, 478, 480
  - application users 449, 450
  - connection templates 485–487
  - connections 475, 479, 480
  - creation 447
  - creation for Agentry 455, 456
  - deletion 448
  - disable Agentry applications 461
  - disconnect active users for Agentry 463
  - domain assignment 452
  - domain assignments 453
  - domain unassignment 453
  - download for Agentry 459
  - enable Agentry applications 461
  - licensing 384
  - list Agentry application users 464
  - log properties 466
  - logs for Agentry 465
  - package removal 454
  - registration templates 477, 486
  - retrieval 449
  - retrieving packages 455
  - SDK type for Agentry 460
  - status for Agentry 471
  - update 448
  - update Agentry configuration 470
- ATT\_CERTIFICATE variableConst interface [Sybase Unwired Platform Security API Reference API] 283
- ATT\_COMMONNAME variableConst interface [Sybase Unwired Platform Security API Reference API] 283
- ATT\_DESCRIPTION variableConst interface [Sybase Unwired Platform Security API Reference API] 283
- ATT\_EMAIL variableConst interface [Sybase Unwired Platform Security API Reference API] 283
- ATT\_FIRSTNAME variableConst interface [Sybase Unwired Platform Security API Reference API] 283



- ATT\_ID variableConst interface [Sybase Unwired Platform Security API Reference API] 283
- ATT\_LASTNAME variableConst interface [Sybase Unwired Platform Security API Reference API] 284
- ATT\_NAME variableConst interface [Sybase Unwired Platform Security API Reference API] 284
- ATT\_PASSWORD variableConst interface [Sybase Unwired Platform Security API Reference API] 284
- ATT\_PHONE variableConst interface [Sybase Unwired Platform Security API Reference API] 284
- ATT\_USERNAME variableConst interface [Sybase Unwired Platform Security API Reference API] 284
- ATTRIBUTE\_ALTERNATE\_SUBJECT variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- ATTRIBUTE\_AUTHORIZATION\_ACTION variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- ATTRIBUTE\_AUTHORIZATION\_RESOURCE variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- ATTRIBUTE\_CONTEXT\_ID variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- ATTRIBUTE\_CONTROL\_FLAG variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- ATTRIBUTE\_CSI\_REQUEST\_ID variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 192
- ATTRIBUTE\_FAILURE\_REASON variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 193
- ATTRIBUTE\_PROVIDER\_ID variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 193
- ATTRIBUTE\_ROLE\_NAME variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 193
- ATTRIBUTE\_ROLE\_SCOPE variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 193
- ATTRIBUTE\_SUBJECT\_ID variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 193
- ATTRIBUTE\_SUPPLIED\_CREDENTIALS variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 193
- Attributed interface [Sybase Unwired Platform Security API Reference API] description 277
- Attributer interface [Sybase Unwired Platform Security API Reference API] description 181
- AttributerRegistration interface [Sybase Unwired Platform Security API Reference API] description 187
- AttributerRegistration2 interface [Sybase Unwired Platform Security API Reference API] description 188
- attribution provider 717
- audit provider 554, 591
- AUDIT\_TOKEN\_CONFIG\_PROPERTY variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 235
- AuditConst interface [Sybase Unwired Platform Security API Reference API]
- ACTION\_ACCESS variable 189
- AuditConst interface [Sybase Unwired Platform Security API Reference API]
- ACTION\_ACTIVATE variable 189
- AuditConst interface [Sybase Unwired Platform Security API Reference API]
- ACTION\_AUTHENTICATION variable 190
- AuditConst interface [Sybase Unwired Platform Security API Reference API]
- ACTION\_AUTHENTICATION\_PROVIDER variable 190
- AuditConst interface [Sybase Unwired Platform Security API Reference API]
- ACTION\_AUTHORIZATION\_AUTHORIZATION variable 190

## Index

- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_AUTHORIZATION\_RESOURCE variable 190
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_AUTHORIZATION\_RESOURCE\_PROVIDER variable 190
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_AUTHORIZATION\_ROLE variable 190
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_AUTHORIZATION\_ROLE\_PROVIDER variable 190
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_CREATE variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_CREATE\_CIPHER variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_CREATE\_DIGEST variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_CREATE\_PROVIDER variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_CREATE\_SIGNATURE variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_LOGOUT variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_MODIFY variable 191
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ACTION\_MODIFY\_PROVIDER variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_ALTERNATE\_SUBJECT variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_AUTHORIZATION\_ACTION variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_AUTHORIZATION\_RESOURCE variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_CONTEXT\_ID variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_CONTROL\_FLAG variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_CSI\_REQUEST\_ID variable 192
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_FAILURE\_REASON variable 193
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_PROVIDER\_ID variable 193
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_ROLE\_NAME variable 193
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_ROLE\_SCOPE variable 193
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_SUBJECT\_ID variable 193
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
ATTRIBUTE\_SUPPLIED\_CREDENTIALS variable 193
- AuditConst interface [Sybase Unwired Platform Security API Reference API]  
CLIENT\_RESOURCECLASS\_PREFIX variable 194

- AuditConst interface [Sybase Unwired Platform Security API Reference API] CORE\_RESOURCECLASS\_PREFIX variable 194
- AuditConst interface [Sybase Unwired Platform Security API Reference API] description 189
- AuditConst interface [Sybase Unwired Platform Security API Reference API] PROVIDER\_RESOURCECLASS\_PREFIX variable 194
- AuditConst interface [Sybase Unwired Platform Security API Reference API] RESOURCECLASS\_CORE\_PROFILE variable 194
- AuditConst interface [Sybase Unwired Platform Security API Reference API] RESOURCECLASS\_CORE\_PROVIDE R variable 194
- AuditConst interface [Sybase Unwired Platform Security API Reference API] RESOURCECLASS\_CORE\_SUBJECT variable 194
- AuditDestination interface [Sybase Unwired Platform Security API Reference API] description 194
- AuditFilter interface [Sybase Unwired Platform Security API Reference API] description 196
- AuditFormatter interface [Sybase Unwired Platform Security API Reference API] description 197
- AuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] \_destination variable 210
- AuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] \_filter variable 210
- AuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] \_formatter variable 210
- AuditProviderConfiguration class [Sybase Unwired Platform Security API Reference API] description 210
- AuditToken interface [Sybase Unwired Platform Security API Reference API] description 198
- authentication provider 554, 595
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] description 56
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_ACCOUNT\_DISAB LED variable 56
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_ACCOUNT\_EXPIR ED variable 56
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_ACCOUNT\_LOCK ED variable 56
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_IMPERSONATION\_ ERROR variable 57
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_INVALID\_CREDE N TIALS variable 57
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_PASSWORD\_EXPI RED variable 57
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_PASSWORD\_EXPI RED\_CAN\_CHANGE variable 57
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_TOKEN\_VALIDATI ON\_ERROR variable 57
- AuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] FAILURE\_CODE\_UNSPECIFIED variable 57

## Index

AuthenticationFailureWarningImpl class [Sybase Unwired Platform Security API Reference API] `_code` variable 200

AuthenticationFailureWarningImpl class [Sybase Unwired Platform Security API Reference API] description 198

authorization provider 554, 630

AuthorizationChecker interface [Sybase Unwired Platform Security API Reference API] description 34

AuthorizationCheckerFactory class [Sybase Unwired Platform Security API Reference API] description 35

Authorizer interface [Sybase Unwired Platform Security API Reference API] description 200

AuthzRequest interface [Sybase Unwired Platform Security API Reference API] description 36

AuthzResponse interface [Sybase Unwired Platform Security API Reference API] description 38

AuthzResponseImpl class [Sybase Unwired Platform Security API Reference API] description 38

## B

backward compatibility 717

BasicNamed class [Sybase Unwired Platform Security API Reference API] description 201

BasicSecIDPrincipal class [Sybase Unwired Platform Security API Reference API] `_id` variable 205

BasicSecIDPrincipal class [Sybase Unwired Platform Security API Reference API] description 204

BasicSecNamePrincipal class [Sybase Unwired Platform Security API Reference API] `_name` variable 206

BasicSecNamePrincipal class [Sybase Unwired Platform Security API Reference API] description 205

Bootstrap class [Sybase Unwired Platform Security API Reference API] `_bootstrapBaseFile` variable 208

Bootstrap class [Sybase Unwired Platform Security API Reference API] `_bootstrapBaseURL` variable 208

Bootstrap class [Sybase Unwired Platform Security API Reference API] `_object` variable 208

Bootstrap class [Sybase Unwired Platform Security API Reference API] description 206

BOOTSTRAP\_CONFIGURATION\_PROPERTY\_FILE variableAbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] 158

BOOTSTRAP\_CONFIGURATION\_PROPERTY\_URL variableAbstractBootstrapConfiguration class [Sybase Unwired Platform Security API Reference API] 158

BUILDTIME variableConst interface [Sybase Unwired Platform Security API Reference API] 284

## C

cache group

- performance 511
- statistics 512

cache groups 431

- associated mobile business objects 433
- purge 433
- retrieval 431
- schedule properties 431, 432

CAPABILITY\_EXPIRED\_PASSWORD\_CHANGE variableConst interface [Sybase Unwired Platform Security API Reference API] 284

CAPABILITY\_FINE\_GRAIN\_ACCESS\_CONTROL variableConst interface [Sybase Unwired Platform Security API Reference API] 285

CAPABILITY\_PASSWORD\_CHANGE variableConst interface [Sybase Unwired Platform Security API Reference API] 285

CAPABILITY\_PREFIX variableConst interface [Sybase Unwired Platform Security API Reference API] 285

CAPABILITY\_SELF\_REGISTRATION variableConst interface [Sybase Unwired Platform Security API Reference API] 285

CAPABILITY\_X509\_AUTHENTICATION variableConst interface [Sybase Unwired

- Platform Security API Reference API] 285
- CERT\_VALIDATION\_CRL\_PREFIX
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 235
- CERT\_VALIDATION\_DEF\_ENABLE\_REVOCA  
TION\_CHECKING
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 235
- CERT\_VALIDATION\_DEF\_VALIDATE\_CERTP  
ATH
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 235
- CERT\_VALIDATION\_EFFECTIVE\_DATE
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 235
- CERT\_VALIDATION\_ENABLE\_REVOCATION  
\_CHECKING
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 236
- CERT\_VALIDATION\_TRUSTED\_CERTSTORE  
\_LOCATION
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 236
- CERT\_VALIDATION\_TRUSTED\_CERTSTORE  
\_PASSWORD
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 236
- CERT\_VALIDATION\_TRUSTED\_CERTSTORE  
\_PROVIDER
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 236
- CERT\_VALIDATION\_TRUSTED\_CERTSTORE  
\_TYPE
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 236
- CERT\_VALIDATION\_URI
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 237
- CERT\_VALIDATION\_VALIDATE\_CERTPATH
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 237
- certificate alias 699
- CERTIFICATE\_AUTH\_REGEX\_MATCH
  - variableCertificateAuthenticationLogin
- Module class [Sybase Unwired Platform Security API Reference API] 61
- CERTIFICATE\_SHARED\_KEY
  - variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 237
- CertificateAuthenticationLoginModule class [Sybase Unwired Platform Security API Reference API]
  - CERTIFICATE\_AUTH\_REGEX\_MAT  
CH variable 61
- CertificateAuthenticationLoginModule class [Sybase Unwired Platform Security API Reference API]
  - DEF\_CERTIFICATE\_AUTH\_REGEX\_  
MATCH variable 61
- CertificateAuthenticationLoginModule class [Sybase Unwired Platform Security API Reference API] description 58
- CertificateAuthenticationLoginModule class [Sybase Unwired Platform Security API Reference API]
  - MESSAGE\_DIGEST\_ALGORITHM\_1  
variable 62
- CertificateBlob class [Sybase Unwired Platform Security API Reference API] description 62
- CertificateCallback class [Sybase Unwired Platform Security API Reference API] description 48
- CertificateCallbackHandler class [Sybase Unwired Platform Security API Reference API] description 49
- CertificateCredential class [Sybase Unwired Platform Security API Reference API]
  - DEFAULT\_CREDENTIAL\_NAME  
variable 282
- CertificateCredential class [Sybase Unwired Platform Security API Reference API] description 281
- CertificateIDPrincipal class [Sybase Unwired Platform Security API Reference API] description 64
- CertificatePrincipal class [Sybase Unwired Platform Security API Reference API]
  - \_certs variable 66
- CertificatePrincipal class [Sybase Unwired Platform Security API Reference API]
  - \_name variable 66

## Index

- CertificatePrincipal class [Sybase Unwired Platform Security API Reference API] description 65
- CertificateTools class [Sybase Unwired Platform Security API Reference API] description 66
- CertificateValidation interface [Sybase Unwired Platform Security API Reference API] description 209
- CertificateValidationException class [Sybase Unwired Platform Security API Reference API] description 209
- CertificateValidationLoginModule class [Sybase Unwired Platform Security API Reference API] description 68
- CHECK\_IMPERSONATION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 237
- cipherTransformation variableAlgorithmSet class [Sybase Unwired Platform Security API Reference API] 59
- CLEARPASS\_OPTION variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 172
- CLEARTEXT\_HASH\_ALGORITHM variablePasswordUtils class [Sybase Unwired Platform Security API Reference API] 111
- client logs 435–437
- CLIENT\_RESOURCECLASS\_PREFIX variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 194
- ClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] description 70
- ClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] PROP\_HTTP\_VALUES\_AS\_NAME\_PRINCIPALS variable 74
- ClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] PROP\_HTTP\_VALUES\_AS\_ROLE\_PRINCIPALS variable 74
- ClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] PROP\_HTTP\_VALUES\_TO\_NAMED\_CREDENTIAL\_MAPPING variable 74
- com.sybase.security.core.CertificateAuthenticationLoginModule 615
- com.sybase.security.core.CertificateValidationLoginModule 598
- com.sybase.security.core.DefaultAuditFilter 593
- com.sybase.security.core.FileAuditDestination 591
- com.sybase.security.core.NoSecAuthorizer 630
- com.sybase.security.core.NoSecLoginModule 595
- com.sybase.security.core.PreConfiguredUserLoginModule 618
- com.sybase.security.core.XmlAuditFormatter 594
- com.sybase.security.http.HttpAuthenticationLoginModule 621
- com.sybase.security.ldap.LDAPAuthorizer 631
- com.sybase.security.ldap.LDAPLoginModule 600
- com.sybase.security.os.NTPProxyLoginModule 609
- com.sybase.security.sap.SAPSSOTokenLoginModule 612
- commit() 532
- Comparator< T extends Named > class [Sybase Unwired Platform Security API Reference API] description 202
- CompatibleConfiguration interface [Sybase Unwired Platform Security API Reference API] description 89
- CompositeAuthzRequest interface [Sybase Unwired Platform Security API Reference API] description 39
- CONFIG\_ATTRIBUTER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 237
- CONFIG\_ATTRIBUTER\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 237
- CONFIG\_AUDIT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 238
- CONFIG\_AUDIT\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 238
- CONFIG\_AUDIT\_DESTINATION variableProviderConst interface [Sybase

- Unwired Platform Security API Reference API] 238
- CONFIG\_AUDIT\_FILTER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 238
- CONFIG\_AUDIT\_FILTER\_PROVIDER\_DEFAULT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 238
- CONFIG\_AUDIT\_FORMATTER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 238
- CONFIG\_AUDIT\_FORMATTER\_PROVIDER\_DEFAULT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 238
- CONFIG\_AUTHORIZER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_AUTHORIZER\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_CONTROLFLAG variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_CSI variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_CSI\_CONFIG\_OPTION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_GLOBAL\_OPTIONS\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_GLOBAL\_PRESERVED\_OPTIONS\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 239
- CONFIG\_ID variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_LOGINMODULE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_LOGINMODULE\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_OPTIONS variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_PROFILER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_PROFILER\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 240
- CONFIG\_ROLEMAPPER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 241
- CONFIG\_ROLEMAPPER\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 241
- CONFIG\_SECURE\_DATASERVICE\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 241
- CONFIG\_SECURE\_DATASERVICE\_PROVIDER\_BASE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 241
- ConfigKey class [Sybase Unwired Platform Security API Reference API] description 88
- CONFIGURATION\_PROVIDER variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 96
- CONFIGURATION\_PROVIDER\_DEFAULT variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 96
- CONFIGURATION\_PROVIDER\_PROPERTY variableConst interface [Sybase Unwired Platform Security API Reference API] 286

## Index

- CONFIGURATION\_PROVIDER\_PROPERTY\_DEFAULT variableConst interface [Sybase Unwired Platform Security API Reference API] 286
- CONFIGURATION\_XML\_VALIDATION\_PROPERTY variableXmlConfiguration class [Sybase Unwired Platform Security API Reference API] 139
- CONFIGURATION\_XML\_VALIDATION\_PROPERTY\_DEFAULT variableXmlConfiguration class [Sybase Unwired Platform Security API Reference API] 139
- ConfigurationParser class [Sybase Unwired Platform Security API Reference API] description 210
- ConfigurationProblem class [Sybase Unwired Platform Security API Reference API] description 215
- ConfigurationProblem class [Sybase Unwired Platform Security API Reference API] Severity() enumeration 216
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_attributers variable 211
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_auditConfigurations variable 211
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_authorizers variable 211
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_csiConfigOptions variable 211
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_dataServiceProviders variable 212
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_loginModuleEntries variable 212
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_profilers variable 212
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] \_roleMappers variable 212
- ConfigurationProperties class [Sybase Unwired Platform Security API Reference API] description 211
- ConfigurationValidationService interface [Sybase Unwired Platform Security API Reference API] description 216
  - connecting
    - SAP Mobile Server 348
  - connection properties 675
  - consolidated database
    - retrieve configuration 535
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_CERTIFICATE variable 283
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_COMMONNAME variable 283
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_DESCRIPTION variable 283
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_EMAIL variable 283
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_FIRSTNAME variable 283
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_ID variable 283
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_LASTNAME variable 284
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_NAME variable 284
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_PASSWORD variable 284
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_PHONE variable 284
  - Const interface [Sybase Unwired Platform Security API Reference API] ATT\_USERNAME variable 284
  - Const interface [Sybase Unwired Platform Security API Reference API] BUILDTIME variable 284
  - Const interface [Sybase Unwired Platform Security API Reference API] CAPABILITY\_EXPIRED\_PASSWORD\_CHANGE variable 284



- Const interface [Sybase Unwired Platform Security API Reference API]  
CAPABILITY\_FINE\_GRAIN\_ACCESS\_CONTROL variable 285
- Const interface [Sybase Unwired Platform Security API Reference API]  
CAPABILITY\_PASSWORD\_CHANGE variable 285
- Const interface [Sybase Unwired Platform Security API Reference API]  
CAPABILITY\_PREFIX variable 285
- Const interface [Sybase Unwired Platform Security API Reference API]  
CAPABILITY\_SELF\_REGISTRATION variable 285
- Const interface [Sybase Unwired Platform Security API Reference API]  
CAPABILITY\_X509\_AUTHENTICATION variable 285
- Const interface [Sybase Unwired Platform Security API Reference API]  
CONFIGURATION\_PROVIDER\_PROPERTY variable 286
- Const interface [Sybase Unwired Platform Security API Reference API]  
CONFIGURATION\_PROVIDER\_PROPERTY\_DEFAULT variable 286
- Const interface [Sybase Unwired Platform Security API Reference API]  
CONTEXT\_RETRIEVER\_PROVIDER\_PROPERTY variable 286
- Const interface [Sybase Unwired Platform Security API Reference API]  
CORE\_CAPABILITY\_ATTRIBUTION variable 286
- Const interface [Sybase Unwired Platform Security API Reference API]  
CORE\_CAPABILITY\_AUTHENTICATION variable 286
- Const interface [Sybase Unwired Platform Security API Reference API]  
CORE\_CAPABILITY\_AUTHORIZATION variable 287
- Const interface [Sybase Unwired Platform Security API Reference API]  
CORE\_CAPABILITY\_CRYPTOGRAPHY variable 287
- Const interface [Sybase Unwired Platform Security API Reference API]  
CORE\_CAPABILITY\_PROVIDER\_CLASS variable 287
- Const interface [Sybase Unwired Platform Security API Reference API] description 282
- Const interface [Sybase Unwired Platform Security API Reference API]  
FACTORY\_RETRIEVER\_PROVIDER\_PROPERTY variable 287
- Const interface [Sybase Unwired Platform Security API Reference API]  
LOG\_WARNINGS\_PROPERTY variable 287
- Const interface [Sybase Unwired Platform Security API Reference API]  
LOG\_WARNINGS\_PROPERTY\_DEFAULT variable 288
- Const interface [Sybase Unwired Platform Security API Reference API]  
MAX\_WARNINGS\_PROPERTY variable 288
- Const interface [Sybase Unwired Platform Security API Reference API]  
MAX\_WARNINGS\_PROPERTY\_DEFAULT variable 288
- Const interface [Sybase Unwired Platform Security API Reference API] OP\_DECRYPT variable 288
- Const interface [Sybase Unwired Platform Security API Reference API] OP\_ENCRYPT variable 288
- Const interface [Sybase Unwired Platform Security API Reference API] OP\_SIGN variable 288
- Const interface [Sybase Unwired Platform Security API Reference API] OP\_VERIFY variable 288
- Const interface [Sybase Unwired Platform Security API Reference API]  
REGEX\_AUDIT\_ATTRIBUTE\_MASK\_PROPERTY variable 289
- Const interface [Sybase Unwired Platform Security API Reference API]  
REGEX\_AUDIT\_ATTRIBUTE\_MASK\_PROPERTY\_DEFAULT variable 289
- Const interface [Sybase Unwired Platform Security API Reference API]  
RESERVED\_CAPABILITY\_PREFIX variable 289

## Index

- Const interface [Sybase Unwired Platform Security API Reference API] RESERVED\_CORE\_CAPABILITY\_PREFIX variable 289
  - Const interface [Sybase Unwired Platform Security API Reference API] SHORTVERSION variable 289
  - Const interface [Sybase Unwired Platform Security API Reference API] VERSION variable 289
  - Const interface [Sybase Unwired Platform Security API Reference API] VERSIONID variable 289
  - Const interface [Sybase Unwired Platform Security API Reference API] WARNINGS\_LOG\_LEVEL\_PROPERTY variable 290
  - Const interface [Sybase Unwired Platform Security API Reference API] WARNINGS\_LOG\_LEVEL\_PROPERTY\_DEFAULT variable 290
  - CONTEXT\_RETRIEVER\_PROVIDER\_PROPERTY variableConst interface [Sybase Unwired Platform Security API Reference API] 286
  - ContextRetriever interface [Sybase Unwired Platform Security API Reference API] description 217
  - ContextRetriever2 interface [Sybase Unwired Platform Security API Reference API] description 218
  - ContextRetrieverPrincipal interface [Sybase Unwired Platform Security API Reference API] description 218
  - contexts
    - introducing 342
  - CORE\_CAPABILITY\_ATTRIBUTION
    - variableConst interface [Sybase Unwired Platform Security API Reference API] 286
  - CORE\_CAPABILITY\_AUTHENTICATION
    - variableConst interface [Sybase Unwired Platform Security API Reference API] 286
  - CORE\_CAPABILITY\_AUTHORIZATION
    - variableConst interface [Sybase Unwired Platform Security API Reference API] 287
  - CORE\_CAPABILITY\_CRYPTOGRAPHY
    - variableConst interface [Sybase Unwired Platform Security API Reference API] 287
  - CORE\_CAPABILITY\_PROVIDER\_CLASS
    - variableConst interface [Sybase Unwired Platform Security API Reference API] 287
  - CORE\_RESOURCECLASS\_PREFIX
    - variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 194
  - createDomain(name) 355
  - createDomainAdministrator(DomainAdministratorVO domainAdministrator) 357
  - createSecurityConfiguration(name) 356
  - CREDENTIAL\_NAME variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 241
  - CSI core logging configuration 24
  - CSI internationalization 23
  - CSI localization 23, 24
  - CSI SDK 3
  - CSI SDK API javadoc 26
  - CSI SDK prerequisites 4
  - CSI SDK quick start 26
  - CURRENT\_SUBJECT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 241
  - custom provider 4
  - custom settings properties 676
  - customization resource bundle
    - assign to application connection 489
    - delete from application 490
    - deploy to an application 488
    - export from an application 488
    - retrieval 487
    - unassign from application connection 489
  - CVPLMRolePrincipal class [Sybase Unwired Platform Security API Reference API] description 70
  - CVPLMUserPrincipal class [Sybase Unwired Platform Security API Reference API] description 71
- ## D
- data change notification
    - history 509
    - performance 509

- Decision class [Sybase Unwired Platform Security API Reference API] ABSTAIN variable 291
- Decision class [Sybase Unwired Platform Security API Reference API] DENY variable 291
- Decision class [Sybase Unwired Platform Security API Reference API] description 290
- Decision class [Sybase Unwired Platform Security API Reference API] NOTAPPLICABLE variable 291
- Decision class [Sybase Unwired Platform Security API Reference API] PERMIT variable 291
- DECRYPT variableOperation class [Sybase Unwired Platform Security API Reference API] 294
- DEF\_CERTIFICATE\_AUTH\_REGEX\_MATCH variableCertificateAuthenticationLoginModule class [Sybase Unwired Platform Security API Reference API] 61
- DEF\_CHECK\_IMPERSONATION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 242
- DEF\_ENABLE\_CERTIFICATE\_AUTHENTICATION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 242
- DEF\_VALIDATED\_CERT\_IS\_IDENTITY variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 242
- DEFAULT\_CONFIGURATION\_NAME variableSecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] 300
- DEFAULT\_CREDENTIAL\_NAME variableCertificateCredential class [Sybase Unwired Platform Security API Reference API] 282
- DEFAULT\_ENCODING variablePasswordUtils class [Sybase Unwired Platform Security API Reference API] 111
- DEFAULT\_HASH\_ALGORITHM variablePasswordUtils class [Sybase Unwired Platform Security API Reference API] 111
- DEFAULT\_IDENTITY\_OPTION variableNoSecLoginModule class [Sybase Unwired Platform Security API Reference API] 104
- DEFAULT\_PACKAGE variableRoleMapper interface [Sybase Unwired Platform Security API Reference API] 254
- DEFAULT\_USE\_USERNAME\_AS\_IDENTITY variableNoSecLoginModule class [Sybase Unwired Platform Security API Reference API] 104
- DefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] description 74
- DefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] OPTION\_CASE\_SENSITIVE\_FILTERING variable 76
- DefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] OPTION\_FILTER variable 76
- DefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] OPTION\_FILTER\_DEFAULT variable 77
- deleteDomainAdministrator (DomainAdministratorVO domainAdministrator) 358
- deleteDomains 355
- deleteOutboundEnabler 551
- deleteOutboundEnablerCertificateFile 553
- deleteOutboundEnablerCertificateFiles 553
- deleteOutboundEnablerProxies 400
- deleteOutboundEnablerProxy 400
- deleteOutboundEnablers 551
- deleteRelayServer 390
- deleteRelayServers 390
- deleteSecurityConfigurations(names) 357
- DENY variableDecision class [Sybase Unwired Platform Security API Reference API] 291
- deprecated methods 717
- device information properties 677
- device notification
  - history 510
  - performance 510
- device user registration properties 679
- DigitalSignature interface [Sybase Unwired Platform Security API Reference API] description 218

## Index

DISABLE\_PASS\_THRU  
    variableXMLFileRoleMapper class  
    [Sybase Unwired Platform Security API  
    Reference API] 142

documentation roadmap 1

DOE-C package  
    subscriptions 426

domain administrators 357, 358

domain management 402, 403

## E

EIS  
    connection properties 679

ejbContextAvailable  
    variableAbstractPrincipalContextRetriever class [Sybase Unwired Platform Security API Reference API] 174

elementContent variableSimpleHandler class [Sybase Unwired Platform Security API Reference API] 125

ENABLE\_CERTIFICATE\_AUTHENTICATION  
    variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 242

ENCRYPT variableOperation class [Sybase Unwired Platform Security API Reference API] 294

ENCRYPTED\_OPTIONS variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 242

EncryptionTools class [Sybase Unwired Platform Security API Reference API] description 219

endpoint  
    creation 407, 409  
    deletion 408, 410  
    retrieval 406  
    update 408, 411

endpoint template  
    retrieval 409

enterprise information systems  
    See EIS

error codes 702  
    introducing 345

error localization 25

ExpiringCredential interface [Sybase Unwired Platform Security API Reference API] description 77

exportPackage(fileName, name, exportOptions) 406

ExternalConfigurationService interface [Sybase Unwired Platform Security API Reference API] description 224

## F

FACTORY\_LEVEL\_SHARED\_STATE  
    variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 243

FACTORY\_RETRIEVER\_PROVIDER\_PROPERTTY variableConst interface [Sybase Unwired Platform Security API Reference API] 287

FactoryHolder class [Sybase Unwired Platform Security API Reference API]  
    \_dateFormatter variable 130

FactoryHolder class [Sybase Unwired Platform Security API Reference API]  
    \_documentBuilder variable 131

FactoryHolder class [Sybase Unwired Platform Security API Reference API]  
    \_stringWriter variable 131

FactoryHolder class [Sybase Unwired Platform Security API Reference API]  
    \_transformer variable 131

FactoryHolder class [Sybase Unwired Platform Security API Reference API] description 130

FactoryRetriever interface [Sybase Unwired Platform Security API Reference API] description 226

FAILURE\_CODE\_ACCOUNT\_DISABLED  
    variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 56

FAILURE\_CODE\_ACCOUNT\_EXPIRED  
    variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 56

FAILURE\_CODE\_ACCOUNT\_LOCKED  
    variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 56

FAILURE\_CODE\_IMPERSONATION\_ERROR  
    variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 57

FAILURE\_CODE\_INVALID\_CREDENTIALS  
    variableAuthenticationFailureWarning

- interface [Sybase Unwired Platform Security API Reference API] 57
  - FAILURE\_CODE\_PASSWORD\_EXPIRED
    - variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 57
  - FAILURE\_CODE\_PASSWORD\_EXPIRED\_CANN\_CHANGE
    - variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 57
  - FAILURE\_CODE\_TOKEN\_VALIDATION\_ERROR
    - variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 57
  - FAILURE\_CODE\_UNSPECIFIED
    - variableAuthenticationFailureWarning interface [Sybase Unwired Platform Security API Reference API] 57
  - FieldFilter 516
  - FILE\_NAME\_SYSTEM\_PROPERTY
    - variablePropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] 121
  - FILE\_NAME\_SYSTEM\_PROPERTY
    - variableXmlConfiguration class [Sybase Unwired Platform Security API Reference API] 140
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] description 77
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] OPTION\_AUDIT\_FILE variable 80
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] OPTION\_COMPRESSION\_THRESHOLD variable 80
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] OPTION\_DELETION\_THRESHOLD variable 81
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] OPTION\_ENCODING variable 81
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] OPTION\_ERROR\_THRESHOLD variable 81
  - FileAuditDestination class [Sybase Unwired Platform Security API Reference API] OPTION\_MAXIMUM\_LOG\_SIZE variable 81
  - FineGrainedAuthzChecker class [Sybase Unwired Platform Security API Reference API] description 40
  - FineGrainedAuthzRequest class [Sybase Unwired Platform Security API Reference API] description 40
  - FormatHelper interface [Sybase Unwired Platform Security API Reference API] description 132
  - framework, diagram of 342
- ## G
- generateSAPAuditMeasurement 386
  - getAuthenticationCacheTimeout() 359
  - getDomainAdministrators() 357
  - getDomains() 355
  - getLicensingInfo() 384
  - getOutboundEnablerProxy 397
  - getOutboundfEnablerProxies 397
  - getProperties() 350
  - getRelayServer 387
  - getRelayServers 387
  - getRelayServers() 383
  - getSecurityConfigurations() 356
  - getServers() 354
  - getSUPApplication(ClusterContext clusterContext) 446
  - getSUPCluster(ClusterContext clusterContext) 353
  - getSUPDomain(DomainContext domainContext) 402
  - getSUPDomainLog(DomainContext domainContext) 522
  - getSUPMobileBusinessObject(MBOContext mboContext) 441
  - getSUPMobileHybridApp(ClusterContext clusterContext) 578
  - getSUPMobileWorkflow(ClusterContext clusterContext) 567
  - getSUPMonitor(ClusterContext clusterContext) 492
  - getSUPOperation(OperationContext operationContext) 444

## Index

getSUPPackage(PackageContext packageContext)  
421  
getSUPSecurityConfiguration(SecurityContext  
securityContext) 555  
getSUPServer(ServerContext serverContext) 350  
getSUPServerConfiguration(ServerContext  
serverContext) 531  
getSUPServerLog(ServerContext serverContext)  
516

## H

hasMap variableSimpleHandler class [Sybase  
Unwired Platform Security API  
Reference API] 125  
HierarchicalItem< C, P, S > class [Sybase Unwired  
Platform Security API Reference API]  
\_children variable 84  
HierarchicalItem< C, P, S > class [Sybase Unwired  
Platform Security API Reference API]  
\_name variable 84  
HierarchicalItem< C, P, S > class [Sybase Unwired  
Platform Security API Reference API]  
\_parent variable 85  
HierarchicalItem< C, P, S > class [Sybase Unwired  
Platform Security API Reference API]  
description 82  
HierarchicalItem< C, P, S > class [Sybase Unwired  
Platform Security API Reference API]  
MESSAGES variable 85  
HTTP listener  
add configuration 537  
delete configuration 537  
retrieve configuration 536  
update 538, 540  
HTTP\_PROPERTIES\_COOKIES\_SHARED\_KEY  
variableProviderConst interface [Sybase  
Unwired Platform Security API  
Reference API] 243  
HTTPS listener  
add configuration 539  
delete configuration 540  
retrieve configuration 539  
Hybrid App  
replace certificate 589  
unblock queue 588  
Hybrid App package  
assignment 585–587  
context variables 581  
deletion 580

devices 586  
e-mail settings 587  
error list 582  
installation 579  
matching rule 584  
matching rules 581  
properties 583  
queue items 582  
retrieval 579  
Hybrid App packages 578

## I

IDENTITY\_OPTION variableNoSecLoginModule  
class [Sybase Unwired Platform Security  
API Reference API] 105  
importPackage(fileName, overwrite) 405  
interfaces  
introducing 343  
SUPAgentry Application 455, 456, 459–461,  
463, 465–467, 471–474  
SUPApplication 447–450, 452–455, 464, 469,  
470, 475–480, 485–490  
SUPCluster 352–360, 383–386, 493–495, 523,  
526–530  
SUPDomain 402–412, 441  
SUPDomainLog 522, 523, 525–528  
SUPHybridApp 578, 581  
SUPMobileBusinessObject 441–444  
SUPMobileHybridApp 579–585, 587–589  
SUPMobileWorkflow 567–577, 584, 586, 587  
SUPMonitor 492, 496, 499, 501–513  
SUPObjectFactory 446, 531, 590  
SUPOperation 444–446  
SUPPackage 421–437, 439, 440  
suppkg 424  
SUPRelayServer 386–390  
SUPSecurityConfiguration 555–564  
SUPServer 350–352, 397–400, 549, 551, 553  
SUPServerConfiguration 531–547, 549–553  
SUPServerLog 516–522

## J

JAXP\_SCHEMA\_LANGUAGE  
variableXmlConfiguration class [Sybase  
Unwired Platform Security API  
Reference API] 140  
JAXP\_SCHEMA\_SOURCE  
variableXmlConfiguration class [Sybase

Unwired Platform Security API  
Reference API] 140

JCESecureDataServices class [Sybase Unwired  
Platform Security API Reference API]  
description 85

JDBC properties 679

## K

key store  
retrieve configuration 543  
update configuration 543

## L

licensing 384

log appenders 519, 520

log buckets 521

log entry  
export 530  
retrieval 527

log filters  
retrieval 526

log profile  
retrieval 523

log settings 519

log store policy  
retrieval 528  
update 529

LOG\_WARNINGS\_PROPERTY variableConst  
interface [Sybase Unwired Platform  
Security API Reference API] 287

LOG\_WARNINGS\_PROPERTY\_DEFAULT  
variableConst interface [Sybase Unwired  
Platform Security API Reference API]  
288

LogAppenderVO 519

LogBucketVO 519

LogicalAndAuthzChecker class [Sybase Unwired  
Platform Security API Reference API]  
description 42

LogicalAndAuthzRequest class [Sybase Unwired  
Platform Security API Reference API]  
\_requests variable 43

LogicalAndAuthzRequest class [Sybase Unwired  
Platform Security API Reference API]  
description 42

LogicalOrAuthzChecker class [Sybase Unwired  
Platform Security API Reference API]  
description 44

LogicalOrAuthzRequest class [Sybase Unwired  
Platform Security API Reference API]  
\_requests variable 45

LogicalOrAuthzRequest class [Sybase Unwired  
Platform Security API Reference API]  
description 44

LogicalRole class [Sybase Unwired Platform  
Security API Reference API] description  
87

## M

management API 327

Management API, enhancements for 328

MAX\_WARNINGS\_PROPERTY variableConst  
interface [Sybase Unwired Platform  
Security API Reference API] 288

MAX\_WARNINGS\_PROPERTY\_DEFAULT  
variableConst interface [Sybase Unwired  
Platform Security API Reference API]  
288

message localization 25

MESSAGE\_DIGEST\_ALGORITHM\_1  
variableCertificateAuthenticationLogin  
Module class [Sybase Unwired Platform  
Security API Reference API] 62

MessageDigest interface [Sybase Unwired Platform  
Security API Reference API] description  
226

messageDigestAlgorithm variableAlgorithmSet  
class [Sybase Unwired Platform Security  
API Reference API] 59

MESSAGES variableAbstractAuthzChecker class  
[Sybase Unwired Platform Security API  
Reference API] 32

MESSAGES variableHierarchialItem< C, P, S >  
class [Sybase Unwired Platform Security  
API Reference API] 85

messaging package  
subscriptions 425, 426

metadata  
introducing 345

metadata:cluster configuration 646

metadata:security configuration 590

metadata:server configuration 667

metadata:server log configuration 671

mobile business object  
data refresh error history 443  
endpoints 442  
operations 444

## Index

- properties 442
- mobile business objects 434
- mobile workflow
  - replace certificate 577
  - unblock queue 577
- mobile workflow package
  - assignment 574, 575
  - context variables 570, 573, 584
  - deletion 569
  - devices 574
  - e-mail settings 576
  - error list 570
  - matching rule 572
  - matching rules 569
  - properties 572
  - queue items 571
  - retrieval 568
  - unassignment 575, 586
- Mobile Workflow package
  - installation 568
- mobile workflow packages 567
- MonitoredObject 496
- monitoring
  - cache group 511, 512
  - current messaging requests 501
  - current replication requests 505
  - data 495
  - data change notification 509
  - data export 513
  - data, large volume 496
  - device notification performance 510
  - messaging history, detailed 502
  - messaging history, summary 502
  - messaging performance data 503
  - profiles 493–495
  - queue data 513
  - replication history, detailed 506
  - replication history, summary 506
  - replication performance 507
  - security log history 499
  - statistics, messaging 504
  - statistics, replication 508

**N**

- Named interface [Sybase Unwired Platform Security API Reference API] description 291
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - \_bootstrap variable 95

- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - \_configCache variable 96
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - \_configProvider variable 96
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - \_initProviderMap variable 96
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - \_repository variable 96
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - CONFIGURATION\_PROVIDER variable 96
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - CONFIGURATION\_PROVIDER\_DEF AULT variable 96
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - description 87
- NamedConfiguration class [Sybase Unwired Platform Security API Reference API]
  - REPOSITORY\_URL\_SYSTEM\_PROPERTY variable 97
- NamedCredential interface [Sybase Unwired Platform Security API Reference API]
  - description 292
- NamedCredentialImpl class [Sybase Unwired Platform Security API Reference API]
  - description 97
- NamedCredentialProvider interface [Sybase Unwired Platform Security API Reference API] description 227
- NO\_ROLE variableRoleMapper interface [Sybase Unwired Platform Security API Reference API] 254

node

- Agentry status 471
- NoSecAttributer class [Sybase Unwired Platform Security API Reference API] description 98
- NoSecAuthorizer class [Sybase Unwired Platform Security API Reference API] description 99
- NoSecLoginModule class [Sybase Unwired Platform Security API Reference API]



- DEFAULT\_IDENTITY\_OPTION variable 104
  - NoSecLoginModule class [Sybase Unwired Platform Security API Reference API]
    - DEFAULT\_USE\_USERNAME\_AS\_IDENTITY variable 104
  - NoSecLoginModule class [Sybase Unwired Platform Security API Reference API]
    - description 101
  - NoSecLoginModule class [Sybase Unwired Platform Security API Reference API]
    - IDENTITY\_OPTION variable 105
  - NoSecLoginModule class [Sybase Unwired Platform Security API Reference API]
    - USE\_USERNAME\_AS\_IDENTITY variable 105
  - NOTAPPLICABLE variableDecision class [Sybase Unwired Platform Security API Reference API] 291
- O**
- OCSP 661
  - Online Certificate Status Protocol 661
  - OP\_DECRYPT variableConst interface [Sybase Unwired Platform Security API Reference API] 288
  - OP\_ENCRYPT variableConst interface [Sybase Unwired Platform Security API Reference API] 288
  - OP\_SIGN variableConst interface [Sybase Unwired Platform Security API Reference API] 288
  - OP\_VERIFY variableConst interface [Sybase Unwired Platform Security API Reference API] 288
  - Operation class [Sybase Unwired Platform Security API Reference API] DECRYPT variable 294
  - Operation class [Sybase Unwired Platform Security API Reference API] description 293
  - Operation class [Sybase Unwired Platform Security API Reference API] ENCRYPT variable 294
  - Operation class [Sybase Unwired Platform Security API Reference API] SIGN variable 294
  - Operation class [Sybase Unwired Platform Security API Reference API] VERIFY variable 294
  - operations
    - playback error history 446
    - properties 445
  - OPTION\_AUDIT\_FILE
    - variableFileAuditDestination class [Sybase Unwired Platform Security API Reference API] 80
  - OPTION\_CASE\_SENSITIVE\_FILTERING
    - variableDefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] 76
  - OPTION\_COMPRESSION\_THRESHOLD
    - variableFileAuditDestination class [Sybase Unwired Platform Security API Reference API] 80
  - OPTION\_DELETION\_THRESHOLD
    - variableFileAuditDestination class [Sybase Unwired Platform Security API Reference API] 81
  - OPTION\_ENCODING
    - variableFileAuditDestination class [Sybase Unwired Platform Security API Reference API] 81
  - OPTION\_ERROR\_THRESHOLD
    - variableFileAuditDestination class [Sybase Unwired Platform Security API Reference API] 81
  - OPTION\_FILTER variableDefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] 76
  - OPTION\_FILTER\_DEFAULT
    - variableDefaultAuditFilter class [Sybase Unwired Platform Security API Reference API] 77
  - OPTION\_MAXIMUM\_LOG\_SIZE
    - variableFileAuditDestination class [Sybase Unwired Platform Security API Reference API] 81
  - OPTION\_REDUCE\_THREAD\_CONTENTION
    - variableXmlAuditFormatter class [Sybase Unwired Platform Security API Reference API] 135
  - OPTION\_REDUCE\_THREAD\_CONTENTION\_DEFAULT
    - variableXmlAuditFormatter class [Sybase Unwired Platform Security API Reference API] 135
  - OptionMapHelper class [Sybase Unwired Platform Security API Reference API] description 228
  - outbound enabler
    - delete 551

## Index

- start 549
- stop 549
- outbound enabler certificate
  - delete 553
- outbound enabler proxy
  - create 398
  - delete 400
  - retrieve 397
  - update 399
- P**
- package
  - deletion 405
  - deployment 404
  - export 406
  - import 405, 441
  - retrieval 403
- package management 421, 422
  - client logs 435–437
  - mobile business objects 434
  - personalization keys 435
- packages
  - add applications 439
  - remove applications 439
- PASSWORD\_SHARED\_KEY
  - variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 172
- PasswordException class [Sybase Unwired Platform Security API Reference API] description 106
- PasswordExpirationWarning interface [Sybase Unwired Platform Security API Reference API] description 105
- PasswordExpirationWarningImpl class [Sybase Unwired Platform Security API Reference API] \_date variable 231
- PasswordExpirationWarningImpl class [Sybase Unwired Platform Security API Reference API] description 230
- PasswordUtils class [Sybase Unwired Platform Security API Reference API] CLEARTEXT\_HASH\_ALGORITHM variable 111
- PasswordUtils class [Sybase Unwired Platform Security API Reference API] DEFAULT\_ENCODING variable 111
- PasswordUtils class [Sybase Unwired Platform Security API Reference API] DEFAULT\_HASH\_ALGORITHM variable 111
- PasswordUtils class [Sybase Unwired Platform Security API Reference API] description 105
- PasswordUtils class [Sybase Unwired Platform Security API Reference API] SALT\_LENGTH variable 111
- PERMIT variableDecision class [Sybase Unwired Platform Security API Reference API] 291
- personalization keys 434
- PhysicalRole class [Sybase Unwired Platform Security API Reference API] description 112
- ping() 351
- PRECONFIG\_PASSWORD\_OPTION
  - variablePreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] 116
- PRECONFIG\_ROLES\_OPTION
  - variablePreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] 116
- PRECONFIG\_USERNAME\_OPTION
  - variablePreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] 116
- PreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] description 112
- PreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] PRECONFIG\_PASSWORD\_OPTION variable 116
- PreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] PRECONFIG\_ROLES\_OPTION variable 116
- PreConfiguredUserLoginModule class [Sybase Unwired Platform Security API Reference API] PRECONFIG\_USERNAME\_OPTION variable 116
- PreConfigUserPrincipal class [Sybase Unwired Platform Security API Reference API] description 113

- PreConfigUserRolePrincipal class [Sybase Unwired Platform Security API Reference API] description 113
- PrefixMap< T > class [Sybase Unwired Platform Security API Reference API] description 231
- PRESERVED\_OPTIONS variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 243
- Principal class [Sybase Unwired Platform Security API Reference API] description 101
- Profiler interface [Sybase Unwired Platform Security API Reference API] description 233
- ProfilerImpl class [Sybase Unwired Platform Security API Reference API] description 116
- PROP\_HTTP\_VALUES\_AS\_NAME\_PRINCIPALS variableClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] 74
- PROP\_HTTP\_VALUES\_AS\_ROLE\_PRINCIPALS variableClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] 74
- PROP\_HTTP\_VALUES\_TO\_NAMED\_CREDENTIAL\_MAPPING variableClientValuePropagatingLoginModule class [Sybase Unwired Platform Security API Reference API] 74
- properties
  - connection reference 679
- PropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] description 119
- PropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] FILE\_NAME\_SYSTEM\_PROPERTY variable 121
- PropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] RESOURCE\_NAME\_SYSTEM\_PROPERTY variable 121
- PropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] RESOURCE\_NAME\_SYSTEM\_PROPERTY\_DEFAULT variable 121
- provider error handling 24
- provider error reporting 25
- provider localization 23
- provider warning reporting 25
- PROVIDER\_RESOURCECLASS\_PREFIX variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 194
- PROVIDER\_SERVICES variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 243
- ProviderConfiguration class [Sybase Unwired Platform Security API Reference API] description 212
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] ALGORITHM\_PARAMETERS variable 235
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] AUDIT\_TOKEN\_CONFIG\_PROPERTY variable 235
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] CERT\_VALIDATION\_CRL\_PREFIX variable 235
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] CERT\_VALIDATION\_DEF\_ENABLE\_REVOCATION\_CHECKING variable 235
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] CERT\_VALIDATION\_DEF\_VALIDATE\_CERTPATH variable 235
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] CERT\_VALIDATION\_EFFECTIVE\_DATE variable 235
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] CERT\_VALIDATION\_ENABLE\_REVOCATION\_CHECKING variable 236
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] CERT\_VALIDATION\_TRUSTED\_CERTSTORE\_LOCATION variable 236
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]

## Index

- CERT\_VALIDATION\_TRUSTED\_CER  
TSTORE\_PASSWORD variable 236
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CERT\_VALIDATION\_TRUSTED\_CER  
TSTORE\_PROVIDER variable 236
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CERT\_VALIDATION\_TRUSTED\_CER  
TSTORE\_TYPE variable 236
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CERT\_VALIDATION\_URI variable 237
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CERT\_VALIDATION\_VALIDATE\_CE  
RTPATH variable 237
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CERTIFICATE\_SHARED\_KEY  
variable 237
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CHECK\_IMPERSONATION variable  
237
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_ATTRIBUTER variable 237
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_ATTRIBUTER\_BASE  
variable 237
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT\_BASE variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT\_DESTINATION  
variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT\_FILTER variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT\_FILTER\_PROVIDE  
R\_DEFAULT variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT\_FORMATTER  
variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUDIT\_FORMATTER\_PRO  
VIDER\_DEFAULT variable 238
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUTHORIZER variable 239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_AUTHORIZER\_BASE  
variable 239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_CONTROLFLAG variable  
239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_CSI variable 239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_CSI\_CONFIG\_OPTION  
variable 239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_GLOBAL\_OPTIONS\_BASE  
variable 239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_GLOBAL\_RESERVED\_OP  
TIONS\_BASE variable 239
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_ID variable 240
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_LOGINMODULE variable  
240
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_LOGINMODULE\_BASE  
variable 240
- ProviderConst interface [Sybase Unwired Platform  
Security API Reference API]
  - CONFIG\_OPTIONS variable 240

- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_PROFILER variable 240
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_PROFILER\_BASE variable 240
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_PROVIDER variable 240
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_ROLEMAPPER variable 241
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_ROLEMAPPER\_BASE variable 241
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_SECURE\_DATASERVICE\_PROVIDER variable 241
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CONFIG\_SECURE\_DATASERVICE\_PROVIDER\_BASE variable 241
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CREDENTIAL\_NAME variable 241
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 CURRENT\_SUBJECT variable 241
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 DEF\_CHECK\_IMPERSONATION variable 242
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 DEF\_ENABLE\_CERTIFICATE\_AUTHENTICATION variable 242
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 DEF\_VALIDATED\_CERT\_IS\_IDENTITY variable 242
- ProviderConst interface [Sybase Unwired Platform Security API Reference API] description 234
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 ENABLE\_CERTIFICATE\_AUTHENTICATION variable 242
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 ENCRYPTED\_OPTIONS variable 242
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 FACTORY\_LEVEL\_SHARED\_STATE variable 243
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 HTTP\_PROPERTIESCOOKIES\_SHARED\_KEY variable 243
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 PRESERVED\_OPTIONS variable 243
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 PROVIDER\_SERVICES variable 243
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SEC\_CONTEXT variable 244
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SECURED\_PROPERTY\_SUFFIX variable 244
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SECURITY\_CIPHER\_ALIAS\_PASSWORD variable 244
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SECURITY\_CIPHER\_KEYSTORE\_ALIAS variable 244
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SECURITY\_CIPHER\_KEYSTORE\_LOCATION variable 244
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SECURITY\_CIPHER\_KEYSTORE\_PASSWORD variable 244
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]  
 SECURITY\_CIPHER\_KEYSTORE\_PROVIDER variable 245
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]

## Index

- SECURITY\_CIPHER\_KEYSTORE\_TYPE variable 245
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_CIPHER\_PROVIDER variable 245
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_CIPHER\_USE\_CERT variable 245
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_CIPHER\_XFORM variable 245
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_ALIAS\_PASSWORD variable 245
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_ALIAS variable 246
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_LOCATION variable 246
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_PASSWORD variable 246
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_PROVIDER variable 246
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_TYPE variable 246
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_PROVIDER variable 246
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_CIPHER\_USE\_CERT variable 247
- SECURITY\_PROFILE\_CIPHER\_XFORM variable 247
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_MESSAGEDIGEST\_ALGORITHM variable 247
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_MESSAGEDIGEST\_PROVIDER variable 247
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_ALGORITHM variable 247
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_ALIAS\_PASSWORD variable 247
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_ALIAS variable 248
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_LOCATION variable 248
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_PASSWORD variable 248
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_PROVIDER variable 248
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_TYPE variable 248
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_PROVIDER variable 248
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]
  - SECURITY\_PROFILE\_SIGNATURE\_USE\_CERT variable 249
- ProviderConst interface [Sybase Unwired Platform Security API Reference API]

- VALIDATED\_CERT\_IS\_IDENTITY variable 249
  - ProviderConst interface [Sybase Unwired Platform Security API Reference API]
    - VOTE\_ABSTAIN variable 249
  - ProviderConst interface [Sybase Unwired Platform Security API Reference API] VOTE\_NO variable 249
  - ProviderConst interface [Sybase Unwired Platform Security API Reference API] VOTE\_YES variable 249
  - ProviderConst interface [Sybase Unwired Platform Security API Reference API] WARNING\_MANAGER variable 249
  - ProviderInfo interface [Sybase Unwired Platform Security API Reference API] description 249
  - ProviderServices interface [Sybase Unwired Platform Security API Reference API] description 251
  - proxy properties 678
  - Proxy properties 701
- R**
- REGEX\_AUDIT\_ATTRIBUTE\_MASK\_PROPER TY variableConst interface [Sybase Unwired Platform Security API Reference API] 289
  - REGEX\_AUDIT\_ATTRIBUTE\_MASK\_PROPER TY\_DEFAULT variableConst interface [Sybase Unwired Platform Security API Reference API] 289
  - relay server
    - create 388
    - delete 390
    - retrieve 387
    - start configuration management 386
    - update 389
  - Relay Server Outbound Enabler 550–553
  - Relay Server Outbound Enablers 549
  - Relay Servers 383
  - replication package
    - subscriptions 427, 428
  - REPOSITORY\_URL\_SYSTEM\_PROPERTY variableNamedConfiguration class [Sybase Unwired Platform Security API Reference API] 97
  - RESERVED\_CAPABILITY\_PREFIX variableConst interface [Sybase Unwired Platform Security API Reference API] 289
  - RESERVED\_CORE\_CAPABILITY\_PREFIX variableConst interface [Sybase Unwired Platform Security API Reference API] 289
  - RESOURCE\_NAME\_SYSTEM\_PROPERTY variablePropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] 121
  - RESOURCE\_NAME\_SYSTEM\_PROPERTY variableXmlConfiguration class [Sybase Unwired Platform Security API Reference API] 140
  - RESOURCE\_NAME\_SYSTEM\_PROPERTY\_DE FAULT variablePropertiesConfiguration class [Sybase Unwired Platform Security API Reference API] 121
  - RESOURCE\_NAME\_SYSTEM\_PROPERTY\_DE FAULT variableXmlConfiguration class [Sybase Unwired Platform Security API Reference API] 140
  - RESOURCECLASS\_CORE\_PROFILE variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 194
  - RESOURCECLASS\_CORE\_PROVIDER variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 194
  - RESOURCECLASS\_CORE\_SUBJECT variableAuditConst interface [Sybase Unwired Platform Security API Reference API] 194
  - restart() 352
  - result sorting 497
  - resume() 354
  - role mappings 429, 430
  - ROLE\_MAP\_FILE variableXMLFileRoleMapper class [Sybase Unwired Platform Security API Reference API] 143
  - RoleAuthzChecker class [Sybase Unwired Platform Security API Reference API] description 45
  - RoleAuthzRequest class [Sybase Unwired Platform Security API Reference API] description 46

## Index

RoleCheck interface [Sybase Unwired Platform Security API Reference API] description 122

RoleCheckAuthorizer class [Sybase Unwired Platform Security API Reference API] description 122

RoleMapAdministrable interface [Sybase Unwired Platform Security API Reference API] description 253

RoleMapper interface [Sybase Unwired Platform Security API Reference API] DEFAULT\_PACKAGE variable 254

RoleMapper interface [Sybase Unwired Platform Security API Reference API] description 254

RoleMapper interface [Sybase Unwired Platform Security API Reference API] NO\_ROLE variable 254

RoleMapperAdmin class [Sybase Unwired Platform Security API Reference API] description 123

RoleMapperConfig class [Sybase Unwired Platform Security API Reference API] description 141

RoleMappings class [Sybase Unwired Platform Security API Reference API] description 129

RolePackage class [Sybase Unwired Platform Security API Reference API] description 129

## S

SALT\_LENGTH variablePasswordUtils class [Sybase Unwired Platform Security API Reference API] 111

SAP connection properties 696

SAP Mobile Server  
connecting to 348

SAP Mobile Server:configuration 530

SAP SAP® Data Orchestration Engine Connector connections 696

SAP SAP® Data Orchestration Engine Connector properties 696

SAP/R3 properties 695

saveOutboundEnablerProxy 398

saveRelayServer 388

SEC\_CONTEXT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 244

SecAdminContext interface [Sybase Unwired Platform Security API Reference API] description 294

SecConfiguration interface [Sybase Unwired Platform Security API Reference API] description 297

SecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] DEFAULT\_CONFIGURATION\_NAME variable 300

SecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] description 299

SecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] SELECTOR\_ALTERNATE\_REPOSITORY variable 300

SecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] SELECTOR\_CONFIGURATION\_NAME variable 300

SecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] SELECTOR\_FRESH\_CONFIGURATION variable 301

SecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] SELECTOR\_PRIMARY\_REPOSITORY\_PREFIX variable 301

SecConfiguration3 interface [Sybase Unwired Platform Security API Reference API] description 301

SecConfigurationValidatingProvider interface [Sybase Unwired Platform Security API Reference API] description 254

SecContext interface [Sybase Unwired Platform Security API Reference API] ADMIN\_ROLE\_NAME variable 311

SecContext interface [Sybase Unwired Platform Security API Reference API] description 302

SecContextFactory class [Sybase Unwired Platform Security API Reference API] description 311

SecContextProvider interface [Sybase Unwired Platform Security API Reference API] description 255



- SecEnvironment interface [Sybase Unwired Platform Security API Reference API] description 319
- SecException class [Sybase Unwired Platform Security API Reference API] `_cause` variable 321
- SecException class [Sybase Unwired Platform Security API Reference API] `_causeList` variable 321
- SecException class [Sybase Unwired Platform Security API Reference API] description 319
- SecIDPrincipal interface [Sybase Unwired Platform Security API Reference API] description 257
- SecLoginExceptionAuthenticationFailureWarningImpl class [Sybase Unwired Platform Security API Reference API] `_code` variable 259
- SecLoginExceptionAuthenticationFailureWarningImpl class [Sybase Unwired Platform Security API Reference API] description 257
- SecLoginExceptionWarningImpl class [Sybase Unwired Platform Security API Reference API] description 259
- SecNamePrincipal interface [Sybase Unwired Platform Security API Reference API] description 260
- SecProfile interface [Sybase Unwired Platform Security API Reference API] description 321
- SecProvider interface [Sybase Unwired Platform Security API Reference API] description 260
- SecProviderCapabilities interface [Sybase Unwired Platform Security API Reference API] description 261
- SecProviderPersistence interface [Sybase Unwired Platform Security API Reference API] description 262
- SecResource interface [Sybase Unwired Platform Security API Reference API] description 322
- SecSubject interface [Sybase Unwired Platform Security API Reference API] description 323
- SECURED\_PROPERTY\_SUFFIX variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 244
- SecureDataServices interface [Sybase Unwired Platform Security API Reference API] description 263
- Security API 3, 26
  - security configurations 356, 357, 423, 424
    - retrieval 411
    - retrieval of default 412
    - set default 412
    - update 412
  - security providers 554, 556
  - security providers:active 556
  - security settings properties 679
- SECURITY\_CIPHER\_ALIAS\_PASSWORD variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 244
- SECURITY\_CIPHER\_KEYSTORE\_ALIAS variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 244
- SECURITY\_CIPHER\_KEYSTORE\_LOCATION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 244
- SECURITY\_CIPHER\_KEYSTORE\_PASSWORD variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 244
- SECURITY\_CIPHER\_KEYSTORE\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 245
- SECURITY\_CIPHER\_KEYSTORE\_TYPE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 245
- SECURITY\_CIPHER\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 245
- SECURITY\_CIPHER\_USE\_CERT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 245
- SECURITY\_CIPHER\_XFORM variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 245

- Unwired Platform Security API Reference API] 245
- SECURITY\_PROFILE\_CIPHER\_ALIAS\_PASS\_WORD variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 245
- SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_ALIAS variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 246
- SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_LOCATION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 246
- SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_PASSWORD variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 246
- SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 246
- SECURITY\_PROFILE\_CIPHER\_KEYSTORE\_TYPE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 246
- SECURITY\_PROFILE\_CIPHER\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 246
- SECURITY\_PROFILE\_CIPHER\_USE\_CERT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 247
- SECURITY\_PROFILE\_CIPHER\_XFORM variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 247
- SECURITY\_PROFILE\_MESSAGEDIGEST\_ALGORITHM variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 247
- SECURITY\_PROFILE\_MESSAGEDIGEST\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 247
- SECURITY\_PROFILE\_SIGNATURE\_ALGORITHM variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 247
- SECURITY\_PROFILE\_SIGNATURE\_ALIAS\_PASSWORD variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 247
- SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_ALIAS variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 248
- SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_LOCATION variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 248
- SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_PASSWORD variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 248
- SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 248
- SECURITY\_PROFILE\_SIGNATURE\_KEYSTORE\_TYPE variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 248
- SECURITY\_PROFILE\_SIGNATURE\_PROVIDER variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 248
- SECURITY\_PROFILE\_SIGNATURE\_USE\_CERT variableProviderConst interface [Sybase Unwired Platform Security API Reference API] 249
- SecurityProviderVO 554, 555
- SecWarning interface [Sybase Unwired Platform Security API Reference API] description 325
- SecWarningImpl class [Sybase Unwired Platform Security API Reference API] \_msg variable 264
- SecWarningImpl class [Sybase Unwired Platform Security API Reference API] description 263
- SELECTOR\_ALTERNATE\_REPOSITORY variableSecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] 300

- SELECTOR\_CONFIGURATION\_NAME
  - variableSecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] 300
- SELECTOR\_FRESH\_CONFIGURATION
  - variableSecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] 301
- SELECTOR\_PRIMARY\_REPOSITORY\_PREFI
  - X variableSecConfiguration2 interface [Sybase Unwired Platform Security API Reference API] 301
- ServerComponentVO 530, 531
- ServerContext
  - using 348
- ServletRequestCallback class [Sybase Unwired Platform Security API Reference API] description 50
- ServletRequestCallbackHandler class [Sybase Unwired Platform Security API Reference API] description 51
- setAuthenticationCacheTimeout() 359
- setTimeZone 385
- Severity() enumerationConfigurationProblem class [Sybase Unwired Platform Security API Reference API] 216
- SHORTVERSION variableConst interface [Sybase Unwired Platform Security API Reference API] 289
- SIGN variableOperation class [Sybase Unwired Platform Security API Reference API] 294
- SimpleHandler class [Sybase Unwired Platform Security API Reference API] description 124, 136
- SimpleHandler class [Sybase Unwired Platform Security API Reference API]
  - elementContent variable 125
- SimpleHandler class [Sybase Unwired Platform Security API Reference API] hasMap variable 125
- SOAP Web Services properties 699
- SortedField 497
- SSL
  - mutual authentication 699
- SSL security profile
  - add configuration 541
  - delete configuration 542
  - retrieve configuration 541
  - update 542
- SSOTokenCredential interface [Sybase Unwired Platform Security API Reference API] description 325
- start() 351
- startOutboundEnablers 549
- stop() 352
- stopOutboundEnablers 549
- STOREPASS\_OPTION
  - variableAbstractLoginModule class [Sybase Unwired Platform Security API Reference API] 172
- subscription template 429
- SUPApplication interface 447–450, 452–456, 459–461, 463–467, 469–481, 485–490
- SUPApplication.
  - getAgentryApplicationDefinitionFiles 472
- SUPApplication.
  - getAgentryApplicationResourceFiles 474
- SUPApplication.addApplication Packages 454
- SUPApplication.assignCustomizationResourceBundle 489
- SUPApplication.assignDomainsToApplication 452
- SUPApplication.cloneApplicationConnections 476
- SUPApplication.createApplication 447, 455, 456
- SUPApplication.createApplicationConnectionTemplate 486
- SUPApplication.deleteApplicationConnections 480
- SUPApplication.deleteApplicationConnectionTemplates 487
- SUPApplication.deleteApplications 448
- SUPApplication.deleteApplicationUsers 450
- SUPApplication.deleteCustomizationResourceBundle 490
- SUPApplication.deployCustomizationResourceBundle 488
- SUPApplication.disconnectAgentryActiveUsers 463
- SUPApplication.downloadApplication 459
- SUPApplication.exportCustomizationResourceBundle 488
- SUPApplication.getAgentryActiveUsers 464
- SUPApplication.getAgentryApplicationConfiguration 469
- SUPApplication.getAgentryApplicationLocalFiles 473

## Index

- SUPApplication.getAgetryApplicationLogSetting 466
- SUPApplication.getAgetryApplicationNodeStatuses 471
- SUPApplication.getApplicationConnections 475
- SUPApplication.getApplicationConnectionTemplates 485
- SUPApplication.getApplicationDomainAssignments 453
- SUPApplication.getApplicationPackages 455
- SUPApplication.getApplications 449
- SUPApplication.getApplicationUsers 449
- SUPApplication.getClientSDKType 460
- SUPApplication.getCustomizationResourceBundle 487
- SUPApplication.lockApplicationConnection 480
- SUPApplication.registerApplicationConnections 477
- SUPApplication.removeApplicationPackages 454
- SUPApplication.reregisterApplicationConnections 478
- SUPApplication.rollAgetryApplicationLog 465
- SUPApplication.setAgetryApplicationStatus 461
- SUPApplication.unassignCustomizationResourceBundle 489
- SUPApplication.unassignDomainsToApplication 453
- SUPApplication.unlockApplicationConnection 480
- SUPApplication.updateAgetryApplicationConfiguration 470
- SUPApplication.updateAgetryApplicationLogSetting 467
- SUPApplication.updateApplication 448
- SUPApplication.updateApplicationConnectionSettings 479
- SUPApplication.updateApplicationConnectionTemplateSettings 486
- SUPCluster interface 352, 354–360, 383–386, 493–495, 522, 523, 526–530
- SUPCluster.createMonitoringProfile(MonitoringProfileVO mpvo) 493
- SUPCluster.deleteMonitoringData(startTime, endTime) 495
- SUPCluster.deleteMonitoringProfile(name) 495
- SUPCluster.exportDomainLogEntry 530
- SUPCluster.exportTraceEntries 522
- SUPCluster.getAuthenticationLockDuration 360
- SUPCluster.getDomainLogEntry 527
- SUPCluster.getDomainLogFilters() 526
- SUPCluster.getDomainLogProfiles() 523
- SUPCluster.getDomainLogStorePolicy() 528, 529
- SUPCluster.getMonitoringProfile(name) 493
- SUPCluster.getMonitoringProfiles() 493
- SUPCluster.getTraceConfigs 384
- SUPCluster.getTraceEntries 522
- SUPCluster.setAuthenticationLockDuration 360
- SUPCluster.setTraceConfigs 384
- SUPCluster.updateMonitoringProfile(MonitoringProfileVO monitoringProfile) 494
- SUPDeviceUser 446
- SUPDomain interface 402–412, 441
- SUPDomain.createEndpoint 407
- SUPDomain.createEndpointTemplate 409
- SUPDomain.deleteEndpoint 408
- SUPDomain.deleteEndpointTemplate 410
- SUPDomain.deletePackage(name) 405
- SUPDomain.deployPackage 404
- SUPDomain.enable(false) 403
- SUPDomain.enable(true) 402
- SUPDomain.getEndpoints 406
- SUPDomain.getEndpointTemplates 409
- SUPDomain.getPackages() 403
- SUPDomain.getSecurityConfigurations() 411, 412
- SUPDomain.setSecurityConfigurations (names) 412
- SUPDomain.updateEndpoint(EndpointVO endpoint) 408
- SUPDomain.updateEndpointTemplate(EndpointVO endpoint) 411
- SUPDomainLog 522
- SUPDomainLog interface 522, 523, 525–528
- SUPDomainLog.createDomainLogProfile 523
- SUPDomainLog.deleteDomainLogFilters 527
- SUPDomainLog.deleteDomainLogProfiles 526
- SUPDomainLog.deleteLog 528
- SUPDomainLog.saveDomainLogFilters 526
- SUPDomainLog.updateDomainLogProfile 525
- SUPHybridApp interface 579
- SUPMobileBusinessObject interface 441–444
- SUPMobileBusinessObject.getDataRefreshErrors(startTime, endDate) 443
- SUPMobileBusinessObject.getEndpoint() 442
- SUPMobileBusinessObject.getOperations() 444
- SUPMobileBusinessObject.getProperties() 442
- SUPMobileBusinessObject() 441
- SUPMobileHybridApp 578

- SUPMobileHybridApp interface 578–585, 587–589
- SUPMobileHybridApp.configureEmail(configurationXML) 587
- SUPMobileHybridApp.deleteMobileHybridApp(hybridAppID) 580
- SUPMobileHybridApp.enableEmail(enable) 587
- SUPMobileHybridApp.getDeviceMobileHybridAppStatus(hybridAppID) 585
- SUPMobileHybridApp.getEmailConfiguration() 587
- SUPMobileHybridApp.getMobileHybridAppContextVariables(MobileHybridAppIDVO workflowID) 581
- SUPMobileHybridApp.getMobileHybridAppErrorList 582
- SUPMobileHybridApp.getMobileHybridAppList() 579
- SUPMobileHybridApp.getMobileHybridAppMatchingRule(MobileHybridAppIDVO hybridAppID) 581
- SUPMobileHybridApp.getMobileHybridAppQueueItems 582
- SUPMobileHybridApp.installMobileHybridApp(zippedWorkflowPackage) 579
- SUPMobileHybridApp.isEmailEnabled() 587
- SUPMobileHybridApp.replaceMobileHybridAppCertificate 589
- SUPMobileHybridApp.testEmailConnection(configXml) 587
- SUPMobileHybridApp.unblockHybridAppQueueForDevices 588
- SUPMobileHybridApp.updateMobileHybridAppDisplayNames 583
- SUPMobileHybridApp.updateMobileHybridAppMatchingRule 584
- SUPMobileWorkflow interface 567–577, 584, 586, 587
- SUPMobileWorkflow.assignMobileWorkflowToDevices 574
- SUPMobileWorkflow.configureEmail(configurationXML) 576
- SUPMobileWorkflow.deleteMobileWorkflow(workflowID) 569
- SUPMobileWorkflow.enableEmail(enable) 576
- SUPMobileWorkflow.getDeviceMobileWorkflowStatus(workflowID) 574
- SUPMobileWorkflow.getDeviceWorkflowAssignments(applicationConnectionNumericID) 575, 587
- SUPMobileWorkflow.getEmailConfiguration() 576
- SUPMobileWorkflow.getMobileWorkflowContextVariables(MobileWorkflowIDVO workflowID) 570
- SUPMobileWorkflow.getMobileWorkflowErrorList 570
- SUPMobileWorkflow.getMobileWorkflowList() 568
- SUPMobileWorkflow.getMobileWorkflowMatchingRule(MobileWorkflowIDVO workflowID) 569
- SUPMobileWorkflow.getMobileWorkflowQueueItems 571
- SUPMobileWorkflow.installMobileWorkflow(zippedWorkflowPackage) 568
- SUPMobileWorkflow.isEmailEnabled() 576
- SUPMobileWorkflow.replaceMobileWorkflowCertificate 577
- SUPMobileWorkflow.testEmailConnection(configXml) 576
- SUPMobileWorkflow.unassignMobileWorkflowFromDevices 575, 586
- SUPMobileWorkflow.unblockWorkflowQueueForDevices 577
- SUPMobileWorkflow.updateMobileWorkflowContextVariables 573, 584
- SUPMobileWorkflow.updateMobileWorkflowDisplayName 572
- SUPMobileWorkflow.updateMobileWorkflowIconIndex 572
- SUPMobileWorkflow.updateMobileWorkflowMatchingRule 572
- SUPMonitor interface 492, 496, 499, 501–513
- SUPMonitor.exportCacheGroupMBOStatistics 513
- SUPMonitor.exportCacheGroupPackageStatistics 513
- SUPMonitor.exportCacheGroupPerformance 513
- SUPMonitor.exportCacheGroupStatistics 513
- SUPMonitor.exportDataChangeNotificationHistory 513
- SUPMonitor.exportDataChangeNotificationPerformance 513
- SUPMonitor.exportDeviceNotificationHistory 513

## Index

- SUPMonitor.exportDeviceNotificationPerformance 513
- SUPMonitor.exportMessagingHistoryDetail 513
- SUPMonitor.exportMessagingHistorySummary 513
- SUPMonitor.exportMessagingPerformance 513
- SUPMonitor.exportMessagingQueueStatistics 513
- SUPMonitor.exportMessagingRequests 513
- SUPMonitor.exportMessagingStatistics 513
- SUPMonitor.exportOperationStatistics 513
- SUPMonitor.exportReplicationHistoryDetail 513
- SUPMonitor.exportReplicationHistorySummary 513
- SUPMonitor.exportReplicationPerformance 513
- SUPMonitor.exportReplicationRequests 513
- SUPMonitor.exportReplicationStatistics 513
- SUPMonitor.exportSecurityLogHistory 513
- SUPMonitor.getCacheGroupPackageStatistics 512
- SUPMonitor.getCacheGroupPerformance 511
- SUPMonitor.getDataChangeNotificationHistory 509
- SUPMonitor.getDataChangeNotificationPerformance 509
- SUPMonitor.getDeviceNotificationHistory 510
- SUPMonitor.getDeviceNotificationPerformance 510
- SUPMonitor.getMessagingHistoryDetail 502
- SUPMonitor.getMessagingPerformance 503
- SUPMonitor.getMessagingQueueStatistics (startTime, endTime) 513
- SUPMonitor.getMessagingRequests(MonitoredObject monitoredObjects) 501
- SUPMonitor.getMessagingStatistics 504
- SUPMonitor.getReplicationHistoryDetail 506
- SUPMonitor.getReplicationHistorySummary 506
- SUPMonitor.getReplicationPerformance 507
- SUPMonitor.getReplicationRequests(MonitoredObject monitoredObjects) 505
- SUPMonitor.getReplicationStatistics 508
- SUPMonitor.getSecurityLogHistory 496, 499
- SUPMonitor.getSecurityLogHistoryCount 499
- SUPMonitor.SecurityLogHistoryCount 496
- SUPMonitored.getMessagingHistorySummary 502
- SUPObjectFactory
  - introducing 345
- SUPObjectFactory interface 446, 531, 590
- SUPObjectFactory.shutdown() 590
- SUPOperation 444
  - SUPOperation interface 444–446
  - SUPOperation.getEndpointVO() 445
  - SUPOperation.getPlaybackErrors (startDate, endDate) 446
  - SUPOperation.getProperties() 445
- SUPPackage interface 421–437, 439, 440
  - SUPPackage.createRBSSubscriptionTemplate 429
  - SUPPackage.deleteClientLogs 436
  - SUPPackage.enable(false) 422
  - SUPPackage.enable(true) 422
  - SUPPackage.exportClientLogs 437
  - SUPPackage.getApplications() 440
  - SUPPackage.getCacheGroupMBOs(cacheGroupName) 433
  - SUPPackage.getCacheGroups() 431
  - SUPPackage.getCacheGroupSchedule(cacheGroupName) 431
  - SUPPackage.getClientLogs() 435
  - SUPPackage.getMBSSubscriptions() 425
  - SUPPackage.getMobileBusinessObjects() 434
  - SUPPackage.getPackageUsers 440
  - SUPPackage.getPersonalizationKeys() 435
  - SUPPackage.getRBSSubscriptions(syncGroup, user) 427
  - SUPPackage.getRoleMappings() 429
  - SUPPackage.getSecurityConfiguration() 423
  - SUPPackage.getSyncGroups() 424
  - SUPPackage.purgeCacheGroup(cacheGroupName, dateThreshold) 433
  - SUPPackage.removeMBSSubscriptions(clientIds) 425
  - SUPPackage.removeRBSSubscription(syncGroup, clientId) 428
  - SUPPackage.removeRBSSubscriptions(syncGroup) 428
  - SUPPackage.resetMBSSubscriptions(clientIds) 426
  - SUPPackage.resumeMBSSubscriptions(clientIds) 426, 427
  - SUPPackage.setCacheGroupSchedule(cacheGroupName, CacheGroupScheduleVO cacheGroupSchedule) 432
  - SUPPackage.setRoleMappings(roleMappingVO rmvos) 430
  - SUPPackage.setSecurityConfiguration 424
  - SUPPackage.setSyncGroupChangeDetectionInterval 424
  - SUPPackage.setSyncTracingStatus(false) 423
  - SUPPackage.setSyncTracingStatus(true) 423

- SUPPackage.suspendMBSSubscriptions(clientIds) 426
- SUPPackage() 421
- suppkg interface 424
- SUPRelayServer 386
- SUPRelayServer interface 386–390
- SUPSecurityConfiguration interface 555–564
- SUPSecurityConfiguration.addActiveAttributionProvider(SecurityProviderVO securityProvider) 560
- SUPSecurityConfiguration.addActiveAuditProvider(SecurityProviderVO securityProvider) 561
- SUPSecurityConfiguration.addActiveAuthenticationProvider(SecurityProviderVO securityProvider) 559
- SUPSecurityConfiguration.addActiveAuthorizationProvider(SecurityProviderVO securityProvider) 559
- SUPSecurityConfiguration.commit() 556
- SUPSecurityConfiguration.deleteActiveAuditProvider 562
- SUPSecurityConfiguration.deleteActiveAuthenticationProvider 562
- SUPSecurityConfiguration.deleteActiveAuthorizationProvider 562
- SUPSecurityConfiguration.getActiveAuditProvider 557
- SUPSecurityConfiguration.getActiveAuthenticationProvider 557
- SUPSecurityConfiguration.getActiveAuthorizationProvider 557
- SUPSecurityConfiguration.getInstalledAuditFormatterProviders() 564
- SUPSecurityConfiguration.getInstalledAuthenticationProviders() 564
- SUPSecurityConfiguration.getInstalledAuthorizationProviders() 564
- SUPSecurityConfiguration.refresh() 555
- SUPSecurityConfiguration.updateActiveAuditProvider 558
- SUPSecurityConfiguration.updateActiveAuthenticationProvider 558
- SUPSecurityConfiguration.updateActiveAuthorizationProvider 558
- SUPSecurityConfiguration.validate() 562
- SUPServer interface 350–354, 397–400, 549, 551, 553
- SUPServerConfiguration 530
- SUPServerConfiguration interface 531–547, 549–553
- SUPServerConfiguration.addApplePushNotificationConfiguration 546
- SUPServerConfiguration.addHTTPListenerConfiguration(serverComponent) 537
- SUPServerConfiguration.addOutboundEnablerCertificateFile 552
- SUPServerConfiguration.addSecureHTTPListenerConfiguration(serverComponent) 539
- SUPServerConfiguration.addSSLSecurityProfileConfiguration(serverComponent) 541
- SUPServerConfiguration.deleteApplePushNotificationConfiguration(apnsConfigName, restart) 546
- SUPServerConfiguration.deleteHTTPListenerConfiguration(serverComponentID) 537
- SUPServerConfiguration.deleteSecureHTTPListenerConfiguration(serverComponentID) 540
- SUPServerConfiguration.deleteSSLSecurityProfileConfiguration(serverComponentID) 542
- SUPServerConfiguration.getAdministrationListenerConfiguration() 535
- SUPServerConfiguration.getApplePushNotificationCertificateNames() 547
- SUPServerConfiguration.getApplePushNotificationConfigurations(true) 545
- SUPServerConfiguration.getConsolidatedDatabaseConfiguration() 535
- SUPServerConfiguration.getHTTPListenerConfigurations() 536
- SUPServerConfiguration.getKeyStoreConfiguration() 543
- SUPServerConfiguration.getMessagingSyncServerConfiguration() 534
- SUPServerConfiguration.getOutboundEnabler() 549
- SUPServerConfiguration.getOutboundEnablerCertificateFiles() 553
- SUPServerConfiguration.getOutboundEnablers() 549
- SUPServerConfiguration.getReplicationSyncServerConfiguration() 533
- SUPServerConfiguration.getSecureHTTPListenerConfigurations() 539
- SUPServerConfiguration.getSSLSecurityProfileConfigurations() 541

## Index

- SUPServerConfiguration.getTrustStoreConfiguration() 544
- SUPServerConfiguration.refresh() 531
- SUPServerConfiguration.saveOutboundEnabler 550
- SUPServerConfiguration.updateApplePushNotificationConfiguration 547
- SUPServerConfiguration.updateHTTPListenerConfiguration(serverComponentID, serverComponent) 538, 540
- SUPServerConfiguration.updateKeyStoreConfiguration(ServerComponentVO serverComponent) 543
- SUPServerConfiguration.updateMessagingSyncServerConfiguration(ServerComponentVO serverComponent) 534
- SUPServerConfiguration.updateOutboundEnabler 551
- SUPServerConfiguration.updateReplicationSyncServerConfiguration(ServerComponentVO serverComponent) 533
- SUPServerConfiguration.updateSSLSecurityProfileConfiguration(serverComponentID, serverComponent) 542
- SUPServerConfiguration.updateTrustStoreConfiguration(ServerComponentVO serverComponent) 544
- SUPServerConfiguration.updateupdateAdministrationListenerConfiguration(serverComponentID, serverComponent) 536
- SUPServerLog 516, 519
- SUPServerLog interface 516–521
- SUPServerLog.deleteLog() 518
- SUPServerLog.getActiveLogAppenders() 519, 520
- SUPServerLog.getLogEntries 517
- SUPServerLog.refresh() 519
- SUPServerLog.setLogPosition 517
- SUPServerLog.updateActiveLogAppender 520
- SUPServerLog.updateActiveLogBucket 521
- SUPWorkflow 566
- suspend() 354
- Sybase Unwired Platform Security API Reference
  - API AbstractAttributed class 147
- Sybase Unwired Platform Security API Reference
  - API AbstractAttributer class 151
- Sybase Unwired Platform Security API Reference
  - API AbstractAuthorizer class 155
- Sybase Unwired Platform Security API Reference
  - API AbstractAuthzChecker class 30
- Sybase Unwired Platform Security API Reference
  - API AbstractAuthzRequest class 32
- Sybase Unwired Platform Security API Reference
  - API AbstractBootstrapConfiguration class 156
- Sybase Unwired Platform Security API Reference
  - API AbstractFactoryRetriever class 158
- Sybase Unwired Platform Security API Reference
  - API AbstractFileConfiguration class 159
- Sybase Unwired Platform Security API Reference
  - API AbstractLoginModule class 164
- Sybase Unwired Platform Security API Reference
  - API AbstractPrincipalContextRetriever class 173
- Sybase Unwired Platform Security API Reference
  - API AbstractProfiler class 175
- Sybase Unwired Platform Security API Reference
  - API AbstractRoleMapper class 176
- Sybase Unwired Platform Security API Reference
  - API AbstractSecureDataServices class 177
- Sybase Unwired Platform Security API Reference
  - API AbstractSecureFileConfiguration class 178
- Sybase Unwired Platform Security API Reference
  - API AlgorithmSet class 58
- Sybase Unwired Platform Security API Reference
  - API AlreadyExistsException class 55
- Sybase Unwired Platform Security API Reference
  - API Attributed interface 277
- Sybase Unwired Platform Security API Reference
  - API Attributer interface 181
- Sybase Unwired Platform Security API Reference
  - API AttributerRegistration interface 187
- Sybase Unwired Platform Security API Reference
  - API AttributerRegistration2 interface 188
- Sybase Unwired Platform Security API Reference
  - API AuditConst interface 189
- Sybase Unwired Platform Security API Reference
  - API AuditDestination interface 194
- Sybase Unwired Platform Security API Reference
  - API AuditFilter interface 196
- Sybase Unwired Platform Security API Reference
  - API AuditFormatter interface 197



- Sybase Unwired Platform Security API Reference  
API AuditProviderConfiguration class  
210
- Sybase Unwired Platform Security API Reference  
API AuditToken interface 198
- Sybase Unwired Platform Security API Reference  
API AuthenticationFailureWarning  
interface 56
- Sybase Unwired Platform Security API Reference  
API AuthenticationFailureWarningImpl  
class 198
- Sybase Unwired Platform Security API Reference  
API authorization package 28
- Sybase Unwired Platform Security API Reference  
API AuthorizationChecker interface 34
- Sybase Unwired Platform Security API Reference  
API AuthorizationCheckerFactory class  
35
- Sybase Unwired Platform Security API Reference  
API Authorizer interface 200
- Sybase Unwired Platform Security API Reference  
API AuthzRequest interface 36
- Sybase Unwired Platform Security API Reference  
API AuthzResponse interface 38
- Sybase Unwired Platform Security API Reference  
API AuthzResponseImpl class 38
- Sybase Unwired Platform Security API Reference  
API BasicNamed class 201
- Sybase Unwired Platform Security API Reference  
API BasicSecIDPrincipal class 204
- Sybase Unwired Platform Security API Reference  
API BasicSecNamePrincipal class 205
- Sybase Unwired Platform Security API Reference  
API Bootstrap class 206
- Sybase Unwired Platform Security API Reference  
API callback package 48
- Sybase Unwired Platform Security API Reference  
API  
CertificateAuthenticationLoginModule  
class 58
- Sybase Unwired Platform Security API Reference  
API CertificateBlob class 62
- Sybase Unwired Platform Security API Reference  
API CertificateCallback class 48
- Sybase Unwired Platform Security API Reference  
API CertificateCallbackHandler class 49
- Sybase Unwired Platform Security API Reference  
API CertificateCredential class 281
- Sybase Unwired Platform Security API Reference  
API CertificateIDPrincipal class 64
- Sybase Unwired Platform Security API Reference  
API CertificatePrincipal class 65
- Sybase Unwired Platform Security API Reference  
API CertificateTools class 66
- Sybase Unwired Platform Security API Reference  
API CertificateValidation interface 209
- Sybase Unwired Platform Security API Reference  
API CertificateValidationException class  
209
- Sybase Unwired Platform Security API Reference  
API CertificateValidationLoginModule  
class 68
- Sybase Unwired Platform Security API Reference  
API  
ClientValuePropagatingLoginModule  
class 70
- Sybase Unwired Platform Security API Reference  
API Comparator< T extends Named >  
class 202
- Sybase Unwired Platform Security API Reference  
API CompatibleConfiguration interface  
89
- Sybase Unwired Platform Security API Reference  
API CompositeAuthzRequest interface  
39
- Sybase Unwired Platform Security API Reference  
API ConfigKey class 88
- Sybase Unwired Platform Security API Reference  
API ConfigurationParser class 210
- Sybase Unwired Platform Security API Reference  
API ConfigurationProblem class 215
- Sybase Unwired Platform Security API Reference  
API ConfigurationProperties class 211
- Sybase Unwired Platform Security API Reference  
API ConfigurationValidationService  
interface 216
- Sybase Unwired Platform Security API Reference  
API Const interface 282
- Sybase Unwired Platform Security API Reference  
API ContextRetriever interface 217
- Sybase Unwired Platform Security API Reference  
API ContextRetriever2 interface 218
- Sybase Unwired Platform Security API Reference  
API ContextRetrieverPrincipal interface  
218
- Sybase Unwired Platform Security API Reference  
API core package 53

## Index

- Sybase Unwired Platform Security API Reference
  - API CVPLMRolePrincipal class 70
- Sybase Unwired Platform Security API Reference
  - API CVPLMUserPrincipal class 71
- Sybase Unwired Platform Security API Reference
  - API Decision class 290
- Sybase Unwired Platform Security API Reference
  - API DefaultAuditFilter class 74
- Sybase Unwired Platform Security API Reference
  - API DigitalSignature interface 218
- Sybase Unwired Platform Security API Reference
  - API EncryptionTools class 219
- Sybase Unwired Platform Security API Reference
  - API ExpiringCredential interface 77
- Sybase Unwired Platform Security API Reference
  - API ExternalConfigurationService interface 224
- Sybase Unwired Platform Security API Reference
  - API FactoryHolder class 130
- Sybase Unwired Platform Security API Reference
  - API FactoryRetriever interface 226
- Sybase Unwired Platform Security API Reference
  - API FileAuditDestination class 77
- Sybase Unwired Platform Security API Reference
  - API FineGrainedAuthzChecker class 40
- Sybase Unwired Platform Security API Reference
  - API FineGrainedAuthzRequest class 40
- Sybase Unwired Platform Security API Reference
  - API FormatHelper interface 132
- Sybase Unwired Platform Security API Reference
  - API HierarchicalItem< C, P, S > class 82
- Sybase Unwired Platform Security API Reference
  - API JCSESecureDataServices class 85
- Sybase Unwired Platform Security API Reference
  - API LogicalAndAuthzChecker class 42
- Sybase Unwired Platform Security API Reference
  - API LogicalAndAuthzRequest class 42
- Sybase Unwired Platform Security API Reference
  - API LogicalOrAuthzChecker class 44
- Sybase Unwired Platform Security API Reference
  - API LogicalOrAuthzRequest class 44
- Sybase Unwired Platform Security API Reference
  - API LogicalRole class 87
- Sybase Unwired Platform Security API Reference
  - API MessageDigest interface 226
- Sybase Unwired Platform Security API Reference
  - API Named interface 291
- Sybase Unwired Platform Security API Reference
  - API NamedConfiguration class 87
- Sybase Unwired Platform Security API Reference
  - API NamedCredential interface 292
- Sybase Unwired Platform Security API Reference
  - API NamedCredentialImpl class 97
- Sybase Unwired Platform Security API Reference
  - API NamedCredentialProvider interface 227
- Sybase Unwired Platform Security API Reference
  - API NoSecAttributer class 98
- Sybase Unwired Platform Security API Reference
  - API NoSecAuthorizer class 99
- Sybase Unwired Platform Security API Reference
  - API NoSecLoginModule class 101
- Sybase Unwired Platform Security API Reference
  - API Operation class 293
- Sybase Unwired Platform Security API Reference
  - API OptionMapHelper class 228
- Sybase Unwired Platform Security API Reference
  - API PasswordException class 106
- Sybase Unwired Platform Security API Reference
  - API PasswordExpirationWarning interface 105
- Sybase Unwired Platform Security API Reference
  - API PasswordExpirationWarningImpl class 230
- Sybase Unwired Platform Security API Reference
  - API PasswordUtils class 105
- Sybase Unwired Platform Security API Reference
  - API PhysicalRole class 112
- Sybase Unwired Platform Security API Reference
  - API PreConfiguredUserLoginModule class 112
- Sybase Unwired Platform Security API Reference
  - API PreConfigUserPrincipal class 113
- Sybase Unwired Platform Security API Reference
  - API PreConfigUserRolePrincipal class 113
- Sybase Unwired Platform Security API Reference
  - API PrefixMap< T > class 231
- Sybase Unwired Platform Security API Reference
  - API Principal class 101
- Sybase Unwired Platform Security API Reference
  - API Profiler interface 233
- Sybase Unwired Platform Security API Reference
  - API ProfilerImpl class 116
- Sybase Unwired Platform Security API Reference
  - API PropertiesConfiguration class 119
- Sybase Unwired Platform Security API Reference
  - API provider package 143

- Sybase Unwired Platform Security API Reference  
API ProviderConfiguration class 212
- Sybase Unwired Platform Security API Reference  
API ProviderConst interface 234
- Sybase Unwired Platform Security API Reference  
API ProviderInfo interface 249
- Sybase Unwired Platform Security API Reference  
API ProviderServices interface 251
- Sybase Unwired Platform Security API Reference  
API RoleAuthzChecker class 45
- Sybase Unwired Platform Security API Reference  
API RoleAuthzRequest class 46
- Sybase Unwired Platform Security API Reference  
API RoleCheck interface 122
- Sybase Unwired Platform Security API Reference  
API RoleCheckAuthorizer class 122
- Sybase Unwired Platform Security API Reference  
API RoleMapAdministrable interface  
253
- Sybase Unwired Platform Security API Reference  
API RoleMapper interface 254
- Sybase Unwired Platform Security API Reference  
API RoleMapperAdmin class 123
- Sybase Unwired Platform Security API Reference  
API RoleMapperConfig class 141
- Sybase Unwired Platform Security API Reference  
API RoleMappings class 129
- Sybase Unwired Platform Security API Reference  
API RolePackage class 129
- Sybase Unwired Platform Security API Reference  
API SecAdminContext interface 294
- Sybase Unwired Platform Security API Reference  
API SecConfiguration interface 297
- Sybase Unwired Platform Security API Reference  
API SecConfiguration2 interface 299
- Sybase Unwired Platform Security API Reference  
API SecConfiguration3 interface 301
- Sybase Unwired Platform Security API Reference  
API SecConfigurationValidatingProvider  
interface 254
- Sybase Unwired Platform Security API Reference  
API SecContext interface 302
- Sybase Unwired Platform Security API Reference  
API SecContextFactory class 311
- Sybase Unwired Platform Security API Reference  
API SecContextProvider interface 255
- Sybase Unwired Platform Security API Reference  
API SecEnvironment interface 319
- Sybase Unwired Platform Security API Reference  
API SecException class 319
- Sybase Unwired Platform Security API Reference  
API SecIDPrincipal interface 257
- Sybase Unwired Platform Security API Reference  
API  
SecLoginExceptionAuthenticationFailureWarningImpl class 257
- Sybase Unwired Platform Security API Reference  
API SecLoginExceptionWarningImpl  
class 259
- Sybase Unwired Platform Security API Reference  
API SecNamePrincipal interface 260
- Sybase Unwired Platform Security API Reference  
API SecProfile interface 321
- Sybase Unwired Platform Security API Reference  
API SecProvider interface 260
- Sybase Unwired Platform Security API Reference  
API SecProviderCapabilities interface  
261
- Sybase Unwired Platform Security API Reference  
API SecProviderPersistence interface  
262
- Sybase Unwired Platform Security API Reference  
API SecResource interface 322
- Sybase Unwired Platform Security API Reference  
API SecSubject interface 323
- Sybase Unwired Platform Security API Reference  
API SecureDataServices interface 263
- Sybase Unwired Platform Security API Reference  
API security package 26
- Sybase Unwired Platform Security API Reference  
API SecWarning interface 325
- Sybase Unwired Platform Security API Reference  
API SecWarningImpl class 263
- Sybase Unwired Platform Security API Reference  
API ServletRequestCallback class 50
- Sybase Unwired Platform Security API Reference  
API ServletRequestCallbackHandler  
class 51
- Sybase Unwired Platform Security API Reference  
API SimpleHandler class 124, 136
- Sybase Unwired Platform Security API Reference  
API SSOTokenCredential interface 325
- Sybase Unwired Platform Security API Reference  
API SynchronizedFormatHelper class  
131

## Index

Sybase Unwired Platform Security API Reference  
API SynchronizedSecAdminContext  
class 264

Sybase Unwired Platform Security API Reference  
API SynchronizedSecContextImpl class  
267

Sybase Unwired Platform Security API Reference  
API ThreadLocalFormatHelper class  
132

Sybase Unwired Platform Security API Reference  
API UsernamePasswordCallbackHandler  
class 52

Sybase Unwired Platform Security API Reference  
API WarningManager interface 277

Sybase Unwired Platform Security API Reference  
API XmlAuditFormatter class 130

Sybase Unwired Platform Security API Reference  
API XmlConfiguration class 135

Sybase Unwired Platform Security API Reference  
API XMLFileRoleMapper class 141

Sybase Unwired Platform Security API Reference  
API XMLFileRoleMapperAdmin class  
141

synchronization group  
properties 424

synchronization group properties 424

synchronization tracing 423

SynchronizedFormatHelper class [Sybase Unwired  
Platform Security API Reference API]  
description 131

SynchronizedSecAdminContext class [Sybase  
Unwired Platform Security API  
Reference API] description 264

SynchronizedSecContextImpl class [Sybase  
Unwired Platform Security API  
Reference API] description 267

## T

ThreadLocalFormatHelper class [Sybase Unwired  
Platform Security API Reference API]  
description 132

trust store  
retrieve configuration 544  
update configuration 544

TRYFIRSTPASS\_OPTION  
variableAbstractLoginModule class  
[Sybase Unwired Platform Security API  
Reference API] 172

## U

understanding the framework 342

updateDomainAdministrator(DomainAdministrato  
rVO domainAdministrator) 358

updateOutboundEnablerProxy 399

updateRelayServer 389

USE\_USERNAME\_AS\_IDENTITY  
variableNoSecLoginModule class  
[Sybase Unwired Platform Security API  
Reference API] 105

USEFIRSTPASS\_OPTION  
variableAbstractLoginModule class  
[Sybase Unwired Platform Security API  
Reference API] 172

user registration properties 679

USERNAME\_SHARED\_KEY  
variableAbstractLoginModule class  
[Sybase Unwired Platform Security API  
Reference API] 173

UsernamePasswordCallbackHandler class [Sybase  
Unwired Platform Security API  
Reference API] description 52

## V

validate custom provider 6

VALIDATED\_CERT\_IS\_IDENTITY  
variableProviderConst interface [Sybase  
Unwired Platform Security API  
Reference API] 249

validating providers 22

VERIFY variableOperation class [Sybase Unwired  
Platform Security API Reference API]  
294

VERSION variableConst interface [Sybase  
Unwired Platform Security API  
Reference API] 289

VERSIONID variableConst interface [Sybase  
Unwired Platform Security API  
Reference API] 289

VOTE\_ABSTAIN variableProviderConst interface  
[Sybase Unwired Platform Security API  
Reference API] 249

VOTE\_NO variableProviderConst interface  
[Sybase Unwired Platform Security API  
Reference API] 249

VOTE\_YES variableProviderConst interface  
[Sybase Unwired Platform Security API  
Reference API] 249

**W**

- W3C\_XML\_SCHEMA variable
  - XmlConfiguration class [Sybase Unwired Platform Security API Reference API] 140
- WARNING\_MANAGER variable
  - ProviderConst interface [Sybase Unwired Platform Security API Reference API] 249
- WarningManager interface [Sybase Unwired Platform Security API Reference API] description 277
- WARNINGS\_LOG\_LEVEL\_PROPERTY variable
  - Const interface [Sybase Unwired Platform Security API Reference API] 290
- WARNINGS\_LOG\_LEVEL\_PROPERTY\_DEFAULT variable
  - Const interface [Sybase Unwired Platform Security API Reference API] 290

**X**

- XmlAuditFormatter class [Sybase Unwired Platform Security API Reference API] description 130
- XmlAuditFormatter class [Sybase Unwired Platform Security API Reference API]
  - OPTION\_REDUCE\_THREAD\_CONTENTION variable 135
- XmlAuditFormatter class [Sybase Unwired Platform Security API Reference API]
  - OPTION\_REDUCE\_THREAD\_CONTENTION\_DEFAULT variable 135
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - CONFIGURATION\_XML\_VALIDATION\_PROPERTY variable 139
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - ON\_PROPERTY\_DEFAULT variable 139
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API] description 135
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - FILE\_NAME\_SYSTEM\_PROPERTY variable 140
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - JAXP\_SCHEMA\_LANGUAGE variable 140
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - JAXP\_SCHEMA\_SOURCE variable 140
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - RESOURCE\_NAME\_SYSTEM\_PROPERTY variable 140
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - RESOURCE\_NAME\_SYSTEM\_PROPERTY\_DEFAULT variable 140
- XmlConfiguration class [Sybase Unwired Platform Security API Reference API]
  - W3C\_XML\_SCHEMA variable 140
- XMLFileRoleMapper class [Sybase Unwired Platform Security API Reference API] description 141
- XMLFileRoleMapper class [Sybase Unwired Platform Security API Reference API]
  - DISABLE\_PASS\_THRU variable 142
- XMLFileRoleMapper class [Sybase Unwired Platform Security API Reference API]
  - ROLE\_MAP\_FILE variable 143
- XMLFileRoleMapperAdmin class [Sybase Unwired Platform Security API Reference API] description 141

