Funambol DS Server

Migration

Version 3.0
May 2006

Published by Funambol, Inc., 643 Bair Island Road, Suite 305, Redwood City, CA 94063

Contents

Introduction

This document is intended to assist developers in migrating applications developed on "Sync4j" 2.3 to "Funambol" 3.0. It provides details on changes in the following areas:

- Package name change
- SyncSource API and related classes
- Other API Changes
- Database table changes
- Backward compatibility of clients

For additional information on the SyncSource API, see the *Funambol DS Server SyncSource API.*

Package Name Change

In the Funambol 3.0 release, all Java packages were renamed from `sync4j.xxx` to `com.funambol.xxx`. This change must be propagated in all sections of your source code where a `sync4j` subpackage is used. In addition, this change must be made in bean and configuration files where appropriate.

SyncSource API and Related Classes

This section describes changes to the SyncSource architecture. It describes the interfaces that were added, and methods that were added, changed or deleted.

Interfaces

The interfaces available in the Funambol 3.0 release are as follows:

- **SyncSource** – the base interface, as in Sync4j 2.3.
- **MergeableSyncSource** – an interface used for SyncSources that support content-merging conflict resolution.
- **FilterableSyncSource** – an interface used for SyncSources that support filtering as specified in the SyncML 1.2 protocol.

SyncSource Interface Methods

The tables in this section describe methods that were added, changed, or deleted in the SyncSource interface.

New Methods

Method	Notes
`addSyncItem()`	Called to add a new item.
`commitSync()`	Called to commit the changes applied during the synchronization session.
`getAllSyncItemKeys()`	Called to retrieve the keys of all items based on the parameters used in the `beginSync` call.
`getSyncItemKeysFromTwin()`	Called to retrieve the keys of the twins of the given item.
`setOperationStatus`	Called to communicate the status of an operation executed on the client.
`updateSyncItem()`	Called to update an existing item.

Changed Methods

Method	Notes

`beginSync()`	The server now passes a `SyncContext` that contains state information, such as the principal, the syncMode. and filter.
	Version 2.3:
	`beginSync(Principal principal, int syncMode)`
	Version 3.0:
	`beginSync(SyncContext context)`

`endSync()`	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method. The SyncSource is responsible for storing the principal for future reference.
	Version 2.3:
	`endSync(Principal principal)`
	Version 3.0:
	`endSync()`

`getDeletedSyncItemKeys()`	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method; a time period `sinceTS - untilTS` is specified.
	Version 2.3:
	`getDeletedSyncItemKeys(Principal principal, Timestamp since)`
	Version 3.0:
	`getDeletedSyncItemKeys(Timestamp sinceTs, Timestamp untilTs)`

`getNewSyncItemKeys()`	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method; a time period `sinceTS - untilTS` is specified.
	Version 2.3:
	`getNewSyncItemKeys(Principal principal, Timestamp since)`
	Version 3.0:
	`getNewSyncItemKeys(Timestamp sinceTs, Timestamp untilTs)`

`getSyncItemFromId()`	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method.
	Version 2.3:
	`getSyncItemFromId(Principal principal, SyncItemKey syncItemKey)`
	Version 3.0:
	`getSyncItemFromId(SyncItemKey syncItemKey)`

Method	Notes

`getUpdatedSyncItemKeys()`	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method; a time period `sinceTS - untilTS` is specified.
	Version 2.3:
	`getUpdatedSyncItemKeys(Principal principal,`
	`                        Timestamp since)`
	Version 3.0:
	`getUpdatedSyncItemKeys(Timestamp sinceTs,`
	`                        Timestamp untilTs)`

`removeSyncItem()`	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method, new arguments added.
	Version 2.3:
	`removeSyncItem(Principal principal,`
	`                SyncItem syncItem)`
	Version 3.0:
	`removeSyncItem(SyncItemKey itemKey, Timestamp t,`
	`                boolean softDelete)`

`setSyncItem()`	This method has been split into two methods: `addSyncItem()` is called to add a new item; `updateSyncItem()` is called to update an existing item.
	The principal is now passed only in the `SyncContext` that is passed to `beginSync()` and is not passed to each method.
	Version 2.3:
	`setSyncItem(Principal principal,`
	`             SyncItem syncItem)`
	Version 3.0:
	`addUpdateSyncItem(SyncItem syncItem)`
	and
	`updateSyncItem(SyncItem syncItem)`

Deleted Methods

Method	Notes
`getAllSyncItems()`	Replaced by `getAllSyncItemKeys()` and then `getSyncItemFromId()`
`getDeletedSyncItems()`	Not used.
`getNewSyncItems()`	Not used.
`getSourceQuery()`	Not used.
`getSyncItemsFromIds()`	Not used.
`getSyncItemFromTwin()`	Replaced by `getSyncItemkeysFromTwin()` and `getSyncItemFromId()`
`getSyncItemsFromTwins()`	Not used.
`getUpdatedSyncItems()`	Not used.
`removeSyncItems()`	Not used.
`setSyncItems()`	Not used.

For additional details, see the *Funambol DS Server SyncSource API*.

Miscellaneous Changes

The following are miscellaneous changes:

- All methods described in this section throw a SyncSourceException.
- Funambol 3.0 introduced the concept of data transformation at an engine level. As a result, if a SyncSource performs base64 encoding/decoding on the item's content, you have the following choices: (1) remove the encoding/decoding functionality from the SyncSource implementation and configure the server accordingly; (2) keep the encoding/decoding functionality inside the SyncSource implementation, in which case no changes on the server configuration are necessary.

Sync4jPrincipal

In the Funambol 3.0 release, the Sync4jPrincipal class groups the following information:

- Principal id
- User as a Sync4jUser
- Device as a Sync4jDevice

In the previous release, the user and device were represented only by their ids. Now the developer has additional information about who is synchronizing.

The following table describes methods that were added to the Sync4jPrincipal class.

New Methods

Method	Notes
`createPrincipal(String userName, String deviceId)`	Called to create a new principal with the given information
`createPrincipal(long id, String userName, String deviceId)`	Called to create a new principal with the given information
`getDevice()`	Called to retrieve the device object.
`getUser()`	Called to retrieve the user object.
`setDevice(Sync4jDevice device)`	Called to set the device object.
`setUser(Sync4jUser user)`	Called to set the user object.

Other API Changes

This section describes additional changes to Funambol APIs.

Administration Tool Panel

The ManagementPanel classes were moved from the `sync4j.syncadmin.ui` package to the `com.funambol.admin.ui` package as a result of the new naming convention.

Officer

If an application needs its own Officer to deal with user information and devices, it will likely use the Sync4jPrincipal object. The interface of the Sync4jPrincipal class does not introduce incompatible changes, and the additional user and device information may be useful for connector developers.

Logging

Logger names were changed from `sync4j.xxx` to `funambol.xxx`. However, if you used the Sync4jLogging class, no changes are necessary to your code.

Database Table Changes

Database table names were changed from `sync4j_<table_name>` to `fnbl_<table_name>`. For connector developers, who usually do not work directly with such tables, this should not impact connector source code. Table name changes are summarized in the following table:

3.0 Table Name	2.3 Table Name
fnbl_user	sync4j_user
fnbl_client_mapping	sync4j_client_mapping
fnbl_connector	sync4j_connector
fnbl_connector_source_type	sync4j_connector_source_type
fnbl_device	sync4j_device
fnbl_device_caps	sync4j_device_caps
fnbl_id	sync4j_id
fnbl_last_sync	sync4j_last_sync
fnbl_module	sync4j_module
fnbl_module_connector	sync4j_module_connector
fnbl_module_sync_source_type	sync4j_module_sync_source_type
fnbl_principal	sync4j_principal
fnbl_role	sync4j_role
fnbl_sync_source	sync4j_sync_source
fnbl_sync_source_type	sync4j_sync_source_type
fnbl_user_role	sync4j_user_role

New Tables

The following tables were added in the 3.0 release:

fnbl_device_datastore	fnbl_ds_ctcap_prop	fnbl_ds_cttype_tx
fnbl_device_ext	fnbl_ds_ctcap_prop_param	fnbl_ds_filter_cap
fnbl_ds_ctcap	fnbl_ds_cttype_rx	fnbl_ds_filter_rx
		fnbl_ds_mem

Client Compatibility

Before discussing client compatibility, we need to make a distinction between clients that fully implement the SyncML standard (such as the mobile phones) and Sync4j 2.3 clients.

For the first category, compatibility should be guaranteed by the protocol implementation and interoperability tests performed by both the clients and the Funambol DS server. Therefore, there should not be incompatibility issues.

Sync4j clients, however, did not implement the entire SyncML standard. The following are the main issues:

- The clients do not send the correct status of the operations performed, and this was ignored in Sync4j 2.3. In Funambol 3.0, however, a correct status is necessary to optimize the synchronization process.
- When sending data base64 encoded, the old client did not specify it with the SyncML Format element. This is required by Funambol 3.0.

The Funambol 3.0 plug-ins (clients are now referred to as plug-ins) are updated and fully interoperable with the server, and we strongly recommend migrating to the latest plug-ins. As an alternative, you can write the synclets necessary to fix the incoming messages as expected by Funambol 3.0 (i.e., as specified by SyncML 1.2).

Resources

This section lists resources you may find useful.

Related Documentation

This section lists documentation resources you may find useful.

Funambol DS Server Documentation

The following documents form the Funambol DS Server documentation set:

- *Funambol DS Server Architectural Overview*: Read this document for an overview of the architecture.
- *Funambol DS Server Administration Guide*: Read this guide to gain an understanding of installation, configuration, and administration.
- *Funambol DS Server Developer's Guide*: Read this guide to understand how to develop extensions to the server.
- *Funambol DS Server SyncSource API*: Read this reference guide for information on the SyncSource interface and related classes.
- *Funambol DS Server Quick Start Guide*: Read this guide to install and run a simple demonstration of synchronizing PIM data using the Funambol DS Server.
- *Funambol DS Server Module Development Tutorial*: Read this tutorial for instructions on packaging, installing and testing modules.