Sync4j Java API J2ME Developers Guide

Sync4j
http://www.sync4j.org

Funambol
http://www.funambol.com

Revision History

Name	Date	Reason for Change	Ver./Rev.
Daniele Pagani	04 May 2004	Original draft	1.0
Fabio Maggi	06 Agoust 2004	Add device charset, SyncServer configuration file	1.1

Table of Contents

1. Overview....4
1.1. SyncClient API Architecture.... 4
2. Data Synchronization API....5
2.1. The SyncManagerFactory.... 5
2.2. The SyncManager.... 5
2.3. The SyncSource.... 5
2.4. The Sync Process.... 6
2.5. Configuring the Sync Manager.... 7
3. Running the SyncClientDemo Application....8

1. Overview

The Sync4 SyncClient API is the means application developers can embed and interact with the Sync4 platform in order to take advantage of its powerful data synchronization features.

This document explains, from a developer point of view, the architecture and the use of the Sync4j SyncClient API 2.7 for J2ME.

1.1. Sync4j Java API J2ME Architecture

The Sync4j Java API J2ME is built up of two main modules: data synchronization and device management; they are layered as shown in Figure 1, where the device management layer is responsible for device and application configuration management and the data synchronization layer is responsible for everything regarding the SyncML protocol and the data synchronization process.

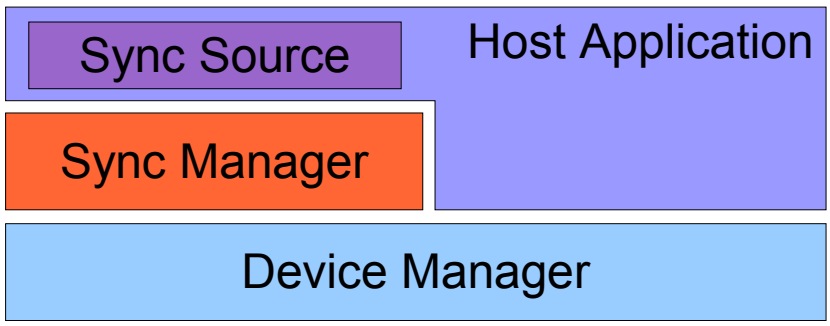

Figure 1 - Sync4j Java API architecture

The host application can access the services provided by both modules: the *Sync Manager* when a synchronization has to be performed and the *Device Manager* when the configuration must be read, manipulated or written. In addition, the Device Manager is intended to store host application configuration information, enabling the application to be transparently managed remotely with the SyncML Device Management features that will be implemented in a next release of the API.

A *Sync Source* is a host application module that groups callback functions called by Sync Manager to interact with the application data sources. The way the Sync Source access the external data source is application specific and transparent to the synchronization engine.

2. Data Synchronization API

The synchronzation API is grouped under the package *sync4j.syncclient*.
The most important classes for a quick start are the Sync Manager Factory , implemented in *sync4j.syncclient.spds.SyncManagerFactory* and the Sync Manager, defined by *sync4j.syncclient.spds.SyncManager*. The latter is the driver of any synchronization operation. They are described in the following sections.

2.1. The SyncManagerFactory

sync4j.syncclient.spds.SyncManagerFactory is a factory for SyncManager objects. It istantiates and configures a new SyncManager with the information provided in the given properties.

The SyncManagerFactory is used to get a new instance of a SyncManager as in the example below:

```
SyncManagerFactory syncMngr = SyncManagerFactory.getSyncManager("MyDataStore",
properties)
```

getSyncManager() is a factory method that creates a new SyncManager bound to the given DataSource.

Error conditions are signalled throwing *sync4j.syncclient.spds.SyncException* for synchronization problems.

See the javadoc documentation for the published API.

2.2. The SyncManager

The SyncManager is the object used by client applications to start a new synchronization process.

When the sync() method is called

```
syncMngr.sync();
```

the synchronization manager starts a new SyncML session to the server synchronizing the data source specified during sync manager construction.

2.3. The SyncSource

The concept of SyncSource is inherited by the Sync4j Java API J2SE. It is an abstraction on a data source layer. In Sync4j Java API J2ME, however, the real implementation of this object is hidden to the developer and embedded into the SyncManager. This has been done for the following reasons:

- provide a light API with small memory footprint;
- give to developers a simple environment hiding the complexity of the data synchronization and SyncML communication.

2.4. The Sync Process

From the host application developer perspective, the interaction with the synchronization engine is limited to firing the synchronization process calling *sync()*. However, under the covers, a lot of work happens. The main tasks performed during a sync execution are:

- synchronization initialization
- client modifications detection
- SyncML synchronization with the server
- server modifications execution

In order to make it possible, the synchronization engine interacts with the host application in two of the above tasks: *client modifications detection* and server *modifications execution* where the methods of the synchronizing sync source are called.

An important aspect of the synchronization process is the concept of fast and slow synchronization.
Fast synchronization can be performed when client and server rely on their respective state, because, for example, they have synchronized recently. In this case only the differences (the modifications) since the last synchronization are exchanged.
When for any reason, client and server are not confident about their respective state, fast synchronization cannot be done and *slow synchronization* is performed. In this case, the client sends its database content to the server, who compares the received information with its local database and then sends back the operations the client has to apply in order to be again up to date and in sync.

The synchronization process tasks are briefly described in the following.

Synchronization initialization
In this phase the synchronization engine prepares a new synchronization session, communicating to the server which sources it wants to synchronize and for which user. The server evaluates the request and responds a status message in which it allows or denies the request.

The Sync Manager synchronized the sources registered in the way described in the Sync Sources section.

Client modifications detection
Here there are two possibilities: in the case of fast sync, the Sync Manager asks the registered Sync Sources which items have changed since the last synchronization; in the case of slow sync, the Sync Manager asks for all items in the data store. As said, in this phase, the Sync Manager calls back the SyncSource's methods *getXXXSyncItem()*, which return the modified (or all) items.

SyncML synchronization with the server
This is the process of exchanging database modifications through the SyncML protocol. This task is hidden to the host application developer.

Server modification execution
This is the phase where server side modifications must be applied to the locale data store. Again, the Sync Manager delegates the SyncSources to execute the changes.

The synchronization process flow looks like Figure 2.

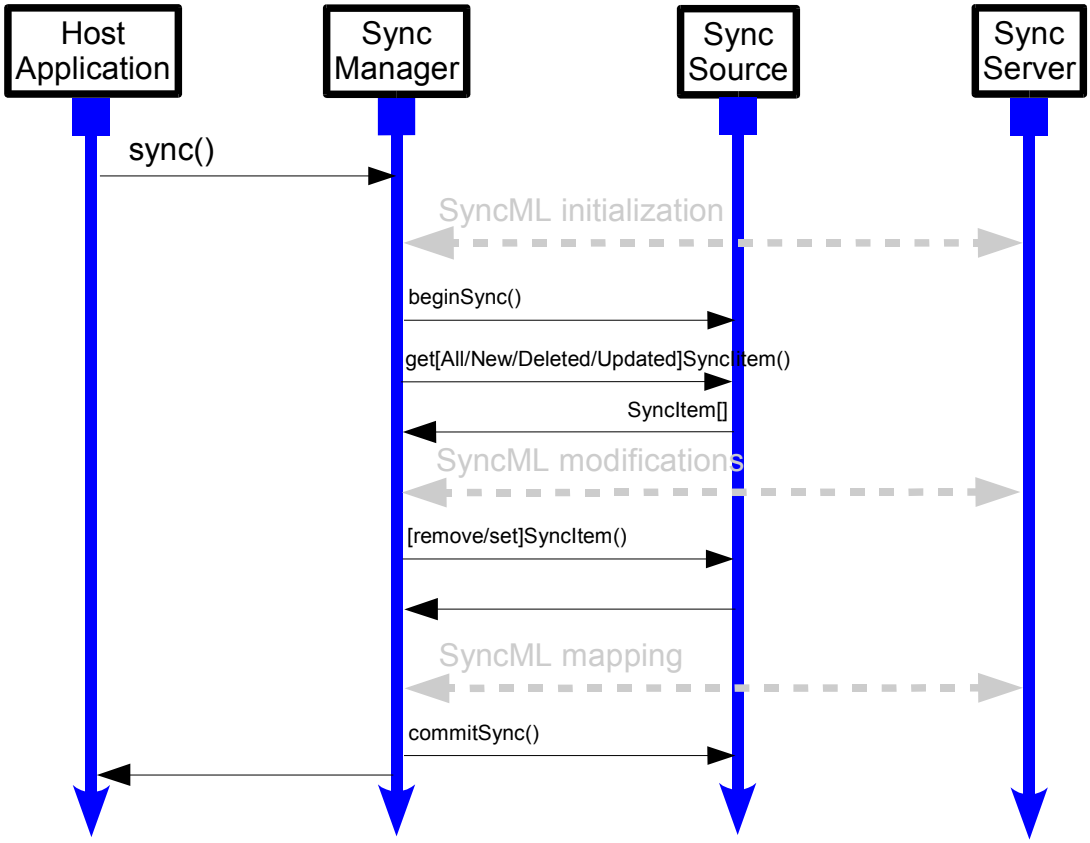

Figure 2 - Synchronization process flow

2.5. Configuring the Sync Manager

Sync Manager requires few configuration parameters such as the url of the SyncML server, which Sync Sources must be synchronized and so on, and the name of the DataStore where the SyncManager has to performe storing and retrieving data process.

Sync Manager makes use of the following configuration parameters:

Property	*Description*
dataStoreName	The name of the DataStore to sync
login	username:password
server-url	The URL of the SyncServer
charset	Device charset. Optional property, if not present set default device charset.
encode	Base64 encoding [true / false] [default: false]

Note that tipically login and serverUrl are sets on the jad file of the application. For more details see the example below.

3. Running the SyncClientDemo Application

In this section, we are going to run the SyncClientDemo application provided with the Sync4j Java API J2ME. Our test application is a J2ME application, composed of the following files:

- SyncClientDemo.jad
- SyncClientDemo.jar

You can find all those files in the examples directory of the Sync4j Java API installation directory.

In sync4j.syncclient.midlet.SyncClientDemoMIDlet.java default sourceURI is ./briefcase, therefore SyncClientDemo synchronize to server ./briefcase SyncSource and the content on SyncServer DS .../db/briefcase folder.installo

You have simply to open with a text editor the SyncClientDemo.jad file and verify that the the following parameters are correctly sets:

- login: guest:guest
- serverUrl: http://localhost:8080/sync4j/sync (or the URL where you have installed the sync4j server)

Then, run the application from your device or emulator (e.g.: J2ME Wireless Toolkit).