Funambol DS Server

Module Development Guide

Version 2.3
March 2006

Important Information

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

Contents

Introduction

This document describes how to create a *module* that extends the functionality of the Funambol DS Server. For example, a module may consist of a packaged set of files, including classes, configuration files, server beans, and initialization SQL scripts, that are embedded into the Funambol DS Server to provide access to a specific database for data synchronization. In general, a module can be viewed as a container for anything related to server extensions.

We will use the following terms and concepts in developing the sample module:

Connector: A server extension that provides support for data synchronization with a specific data source. For example, the Funambol Visual DB Connector provides a GUI and runtime classes for the synchronization of generic data stored in a RDMS. Alternatively, a Connector could support a data source that stores email messages, calendar events, or other data types.

SyncSource: A key component of a Connector, it defines the way a set of data is made accessible to the Funambol DS Server for synchronization. A SyncSource *type* is a template from which an instance of a SyncSource is created. For example, the FileSystemSyncSource type defines how data stored in directories in a file system can be accessed by the Funambol DS Server. This SyncSource type does not represent a specific instance of the FileSystemSyncSource, so it does not identify a directory to be used for synchronization. To specify such a directory, you create an instance of the FileSystemSyncSource and configure it with the desired directory. For additional information, see the *Funambol DS Server Developer's Guide*.

This tutorial will guide you through the development, packaging, installation and testing of a module. The module contains a simple SyncSource and is comprised of code classes, configuration files and database initialization scripts.

Module Development Overview

We will develop the sample module using the steps below:

1. Create the following, in sequence:
   – Module source directory structure
   – Dummy SyncSource type
   – Dummy SyncSource configuration panel
   – SQL scripts for registering the module
   – Module archive file
2. Install the module
3. Create a SyncSource instance
4. Test the module with a SyncML client

Prerequisites

This tutorial assumes a working knowledge of Java, Ant and SQL. The system requirements are as follows:

- Funambal DS Server
- Funambol Java Command Line Client Example
- Java 2 SDK version 1.4.x
- Jakarta Ant

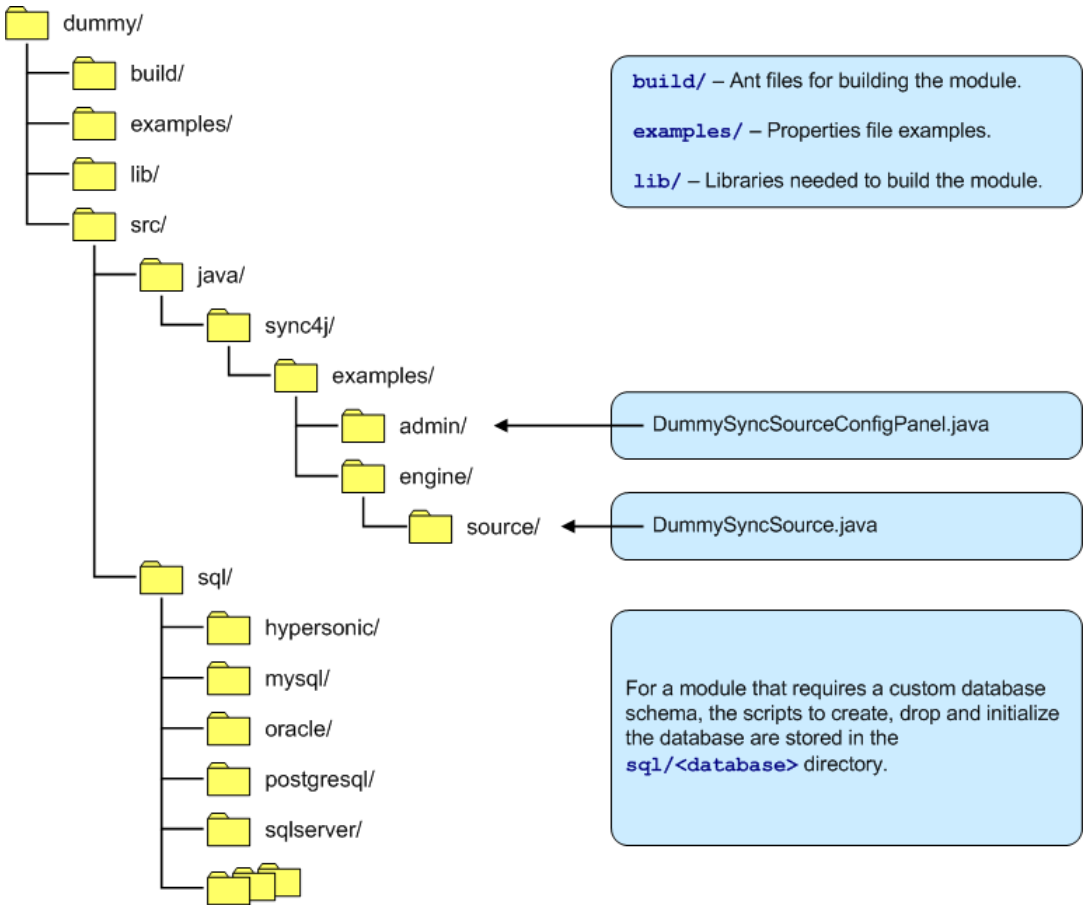
Obtaining the Tutorial Software

Files used in developing the sample module are available for download from the following site:

`http://forge.objectweb.org/project/download.php?group_id=96&file_id=5806`

Creating the Module Source Directory Structure

Unzip the compressed file in a directory of your choice. The structure containing the source, configuration and script files will be created as follows:



Creating a Dummy SyncSource Type

The SyncSource type is the primary component of the module; it defines the way a set of data is made accessible to the Funambol DS Server for synchronization. We will use the SyncSource type as a template for implementing an instance of a SyncSource later in the tutorial.

Our component, `DummySyncSource`, is a simple example; it only displays a message when one of its methods is called, and it always returns the same items. Sample code for `DummySyncSource` is available in the `\dummy\src\java\sync4j\examples\engine\source` directory.

Reviewing the Code

The class structure (methods) reflects the SyncSource interface. In addition, it extends `AbstractSyncSource` so that it inherits common methods. For details, see the *Funambol DS Server Developer's Guide*.

The constructor creates some note items that are stored in the instance variables `newItems`, `deletedItems` and `updatedItems`. These are returned when requested by `get[All/Updated/New/Deleted]Items()`.

Items are created in `createItem()`: given the item identifier (the item key), the content and the state, it instantiates a new `SyncItemImpl` (a simple implementation of the SyncItem interface) and sets the `BINARY_PROPERTY` to the note content.

NOTE: Some of the above methods are not currently executed by the Funambol DS Server engine since they are intended for future implementations of the engine. Specifically, methods that work on `SyncItemKeys` instead of `SyncItems` are not currently used.

Creating a Dummy SyncSource Configuration Panel

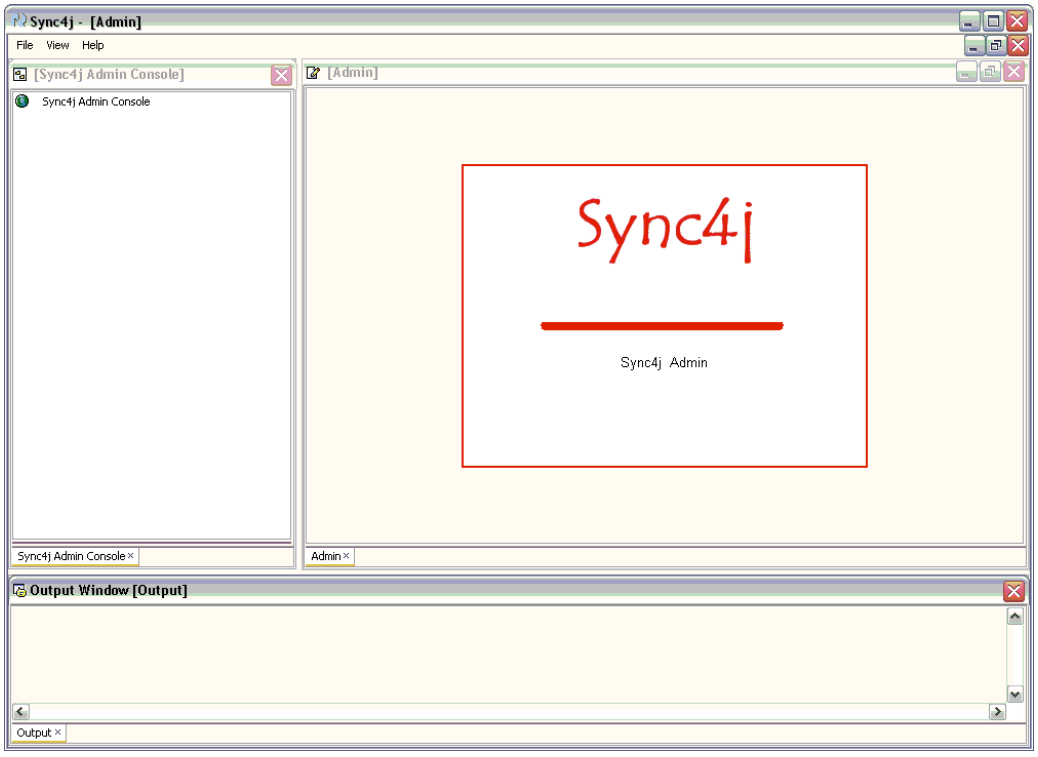
You use the Administration Tool to configure a SyncSource. In this section we briefly introduce the Administration Tool, then we will create an extension specifically for configuring the Dummy SyncSource.

Using the Administration Tool

To start the Administration Tool, perform the following:

1. Start the Funambol DS Server by selecting **Start > All Programs > Sync4j > Sync4j Server DS > Start**.

2. Start the Administration Tool by selecting **Start > All Programs > Sync4j > Sync4j Admin**. The SyncAdmin window displays.

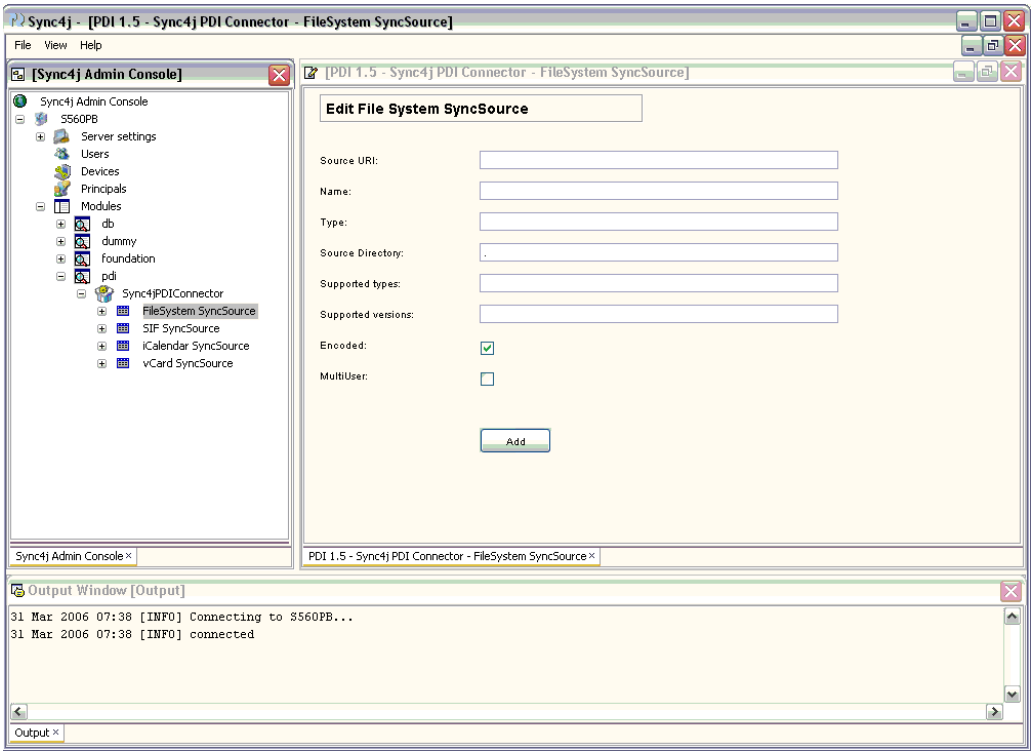


3. On the Main Menu bar, select **File > Login**. The Login window displays. Verify the fields are populated as follows, or specify these values:

   Hostname/IP: `<localhost>` (should be your machine name)
   Port: `8080`
   User name: `admin`
   Password: `sa`

   Click **Login**. The Output window in the lower pane should display "connected."

4. To configure a SyncSource, navigate in the left pane to the desired SyncSource. For example, to configure a File System SyncSource, expand the **localhost** tree as follows: **localhost > Modules > pdi > Sync4jPDIConnector**, then select **FileSystem SyncSource**. The Edit File System SyncSource screen displays, as shown below:



You use this screen to specify configuration values and add an instance of the SyncSource type. We will create a screen similar to this for the Dummy SyncSource.

Creating the Configuration Panel

To create a configuration panel for Dummy SyncSource, we create an extension of `sync4j.syncadmin.ui.SourceManagementPanel` and call it `DummySyncSourceConfigPanel.java`.

Sample code for `DummySyncSourceConfigPanel` is available in the `\dummy\src\java\sync4j\examples\admin` directory.

Creating SQL Scripts for Registering the Module

To make the Funambol DS Server aware of a module containing a connector and SyncSource type, we register these items in the Sync4j database using SQL scripts. You can support multiple databases by storing the script(s) specific to each in the `\dummy\src\sql\<database_vendor>` directory, where `database_vendor` is the vendor's name. This name is also specified in the `install.properties` file, where it identifies the database the Funambol DS Server uses. For each database, we could create the following scripts:

- `drop_schema.sql` – cleans up existing database tables, if any
- `create_schema.sql` – creates new database tables, if required
- `init_schema.sql` – populates the database

For our sample module we will use Hypersonic. The only required script for the tutorial is `init_schema`, which includes the following SQL statements:

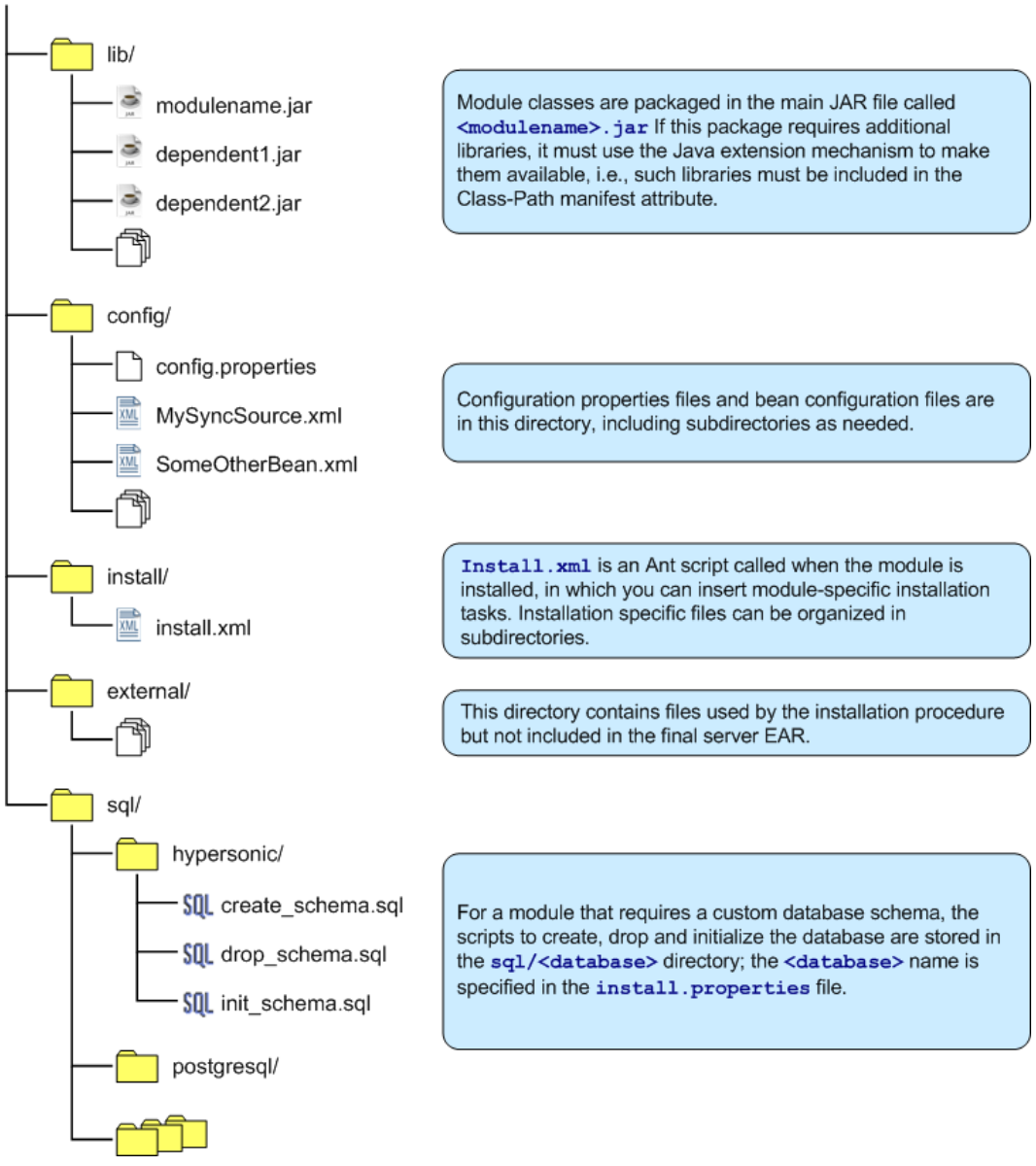
init_schema:

```
--
-- Initialization data for the Dummy module
-- @version $Id: init_schema.sql,v 1.8 2005/11/28 15:17:27 Exp $
--

--
-- Module structure registration
--

delete from sync4j_sync_source_type where id='dummy';
insert into sync4j_sync_source_type(id, description, class, admin_class)
values('dummy','Dummy
SyncSource','sync4j.examples.engine.source.DummySyncSource','sync4j.examples
.admin.DummySyncSourceConfigPanel');

delete from sync4j_module where id='dummy';
insert into sync4j_module (id, name, description)
values('dummy','dummy','Dummy');

delete from sync4j_connector where id='dummy';
insert into sync4j_connector(id, name, description, admin_class)
values('dummy','Sync4jDummyConnector','Sync4j Dummy Connector','');

delete from sync4j_connector_source_type where connector='dummy' and
sourcetype='dummy';
insert into sync4j_connector_source_type(connector, sourcetype)
values('dummy','dummy');

delete from sync4j_module_connector where module='dummy' and
connector='dummy';
insert into sync4j_module_connector(module, connector)
values('dummy','dummy');
```

The above SQL commands inform the Funambol DS Server there is a new module called **dummy**, which contains a Connector called **dummy**, which in turn contains a SyncSource type called **dummy**. The SyncSource type is specified by the SyncSource class `sync4j.examples.engine.source.DummySyncSource` and the configuration panel by `sync4j.examples.admin.DummySyncSourceConfigPanel`.

Creating the Module Archive File

In this step we will automate the process of compiling the classes and packing everything into module archive (for additional details on module archives, see Chapter 6 of the *Funambol DS Server Developer's Guide*). The internal structure of the archive file is shown below:

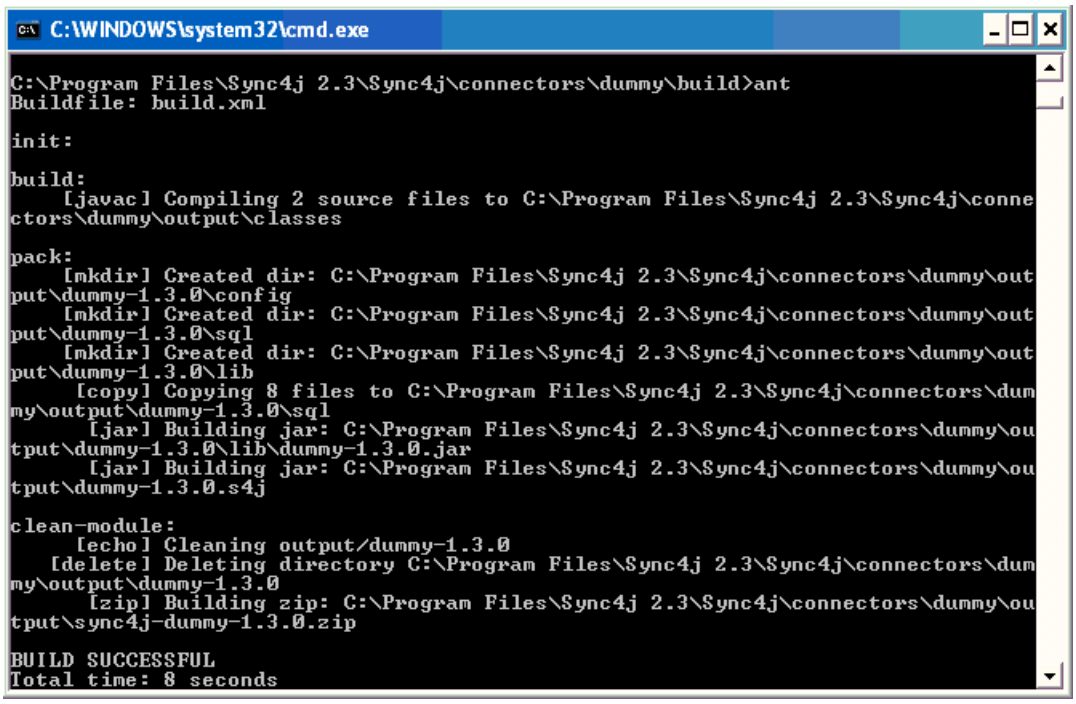


We will use Jakarta Ant to build the module archive, but you can use your preferred tool or IDE, as long as you produce a single `.s4j` file and maintain the structure shown above. Sample code for the build file we will use, `build.xml`, is available in the `\dummy\build` directory.

To perform the build, go to the `\dummy\build` directory and run the command (with Jakarta Ant in your path):

```
> ant
```

The output should appear similar to the following:



The build process creates the directory `\output` containing the `dummy-1.3.0.s4j` module archive file.

Installing the Module

In this procedure we will use `<DS-SERVER_HOME>` to represent the directory containing the Funambol DS Server (e.g., `C:\Program Files\sync4j\ds-server`).

1. Copy the `dummy-1.3.0.s4j` module archive file to the `<DS-SERVER_HOME>\modules` directory.
2. Using a text editor, open the `<DS-SERVER_HOME>\install.properties` file.
3. Find the line that begins `modules-to-install=` in the Module definitions section. This line specifies, in a comma-separated list, the modules to install during installation.
4. Add `dummy-1.3.0` to the comma-separated list (without the `.s4j` filename extension).
5. Save and close `install.properties`.
6. On **Windows**, open a command prompt window by selecting **Start > All Programs > Accessories > Command Prompt** and run the server installation script by typing the following at the prompt:

   ```
   cd <DS-SERVER_HOME>
   bin\install <application_server>
   ```

   Alternatively, you can install just the modules with the following command:

   ```
   bin\install-modules <application_server>
   ```

   Unix/Linux: use the command `bin/install.sh <application_server>` or `bin/install-modules.sh <application_server>`.

Creating a Dummy SyncSource Instance

We will use the Administration Tool to create an instance of Dummy SyncSource, as follows:

1. Start the Funambol DS Server by selecting **Start > All Programs > Sync4j > Sync4j Server DS > Start**.

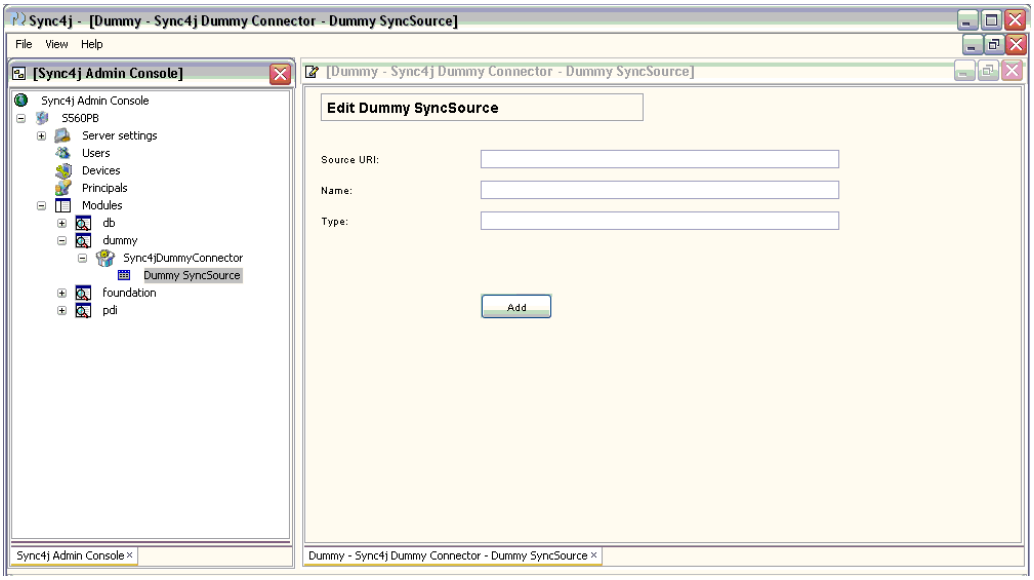
2. Start the Funambol Administration tool by selecting **Start > All Programs > Sync4j > Sync4j Admin**. The SyncAdmin window displays.

3. On the Main Menu bar, select **File > Login**. The Login window displays. Verify the fields are populated as follows, or specify these values:

   Hostname/IP: `<localhost>` (should be your machine name)
   Port: `8080`
   User name: `admin`
   Password: `sa`

   Click **Login**. The Output Window in the lower right pane should display "connected."

4. In the left pane, expand the **localhost** tree as follows: **localhost > Modules > dummy > Sync4jDummyConnector**, then select **Dummy SyncSource**. The Edit Dummy SyncSource screen displays in the right pane, as shown below:



5. Specify the following field values:

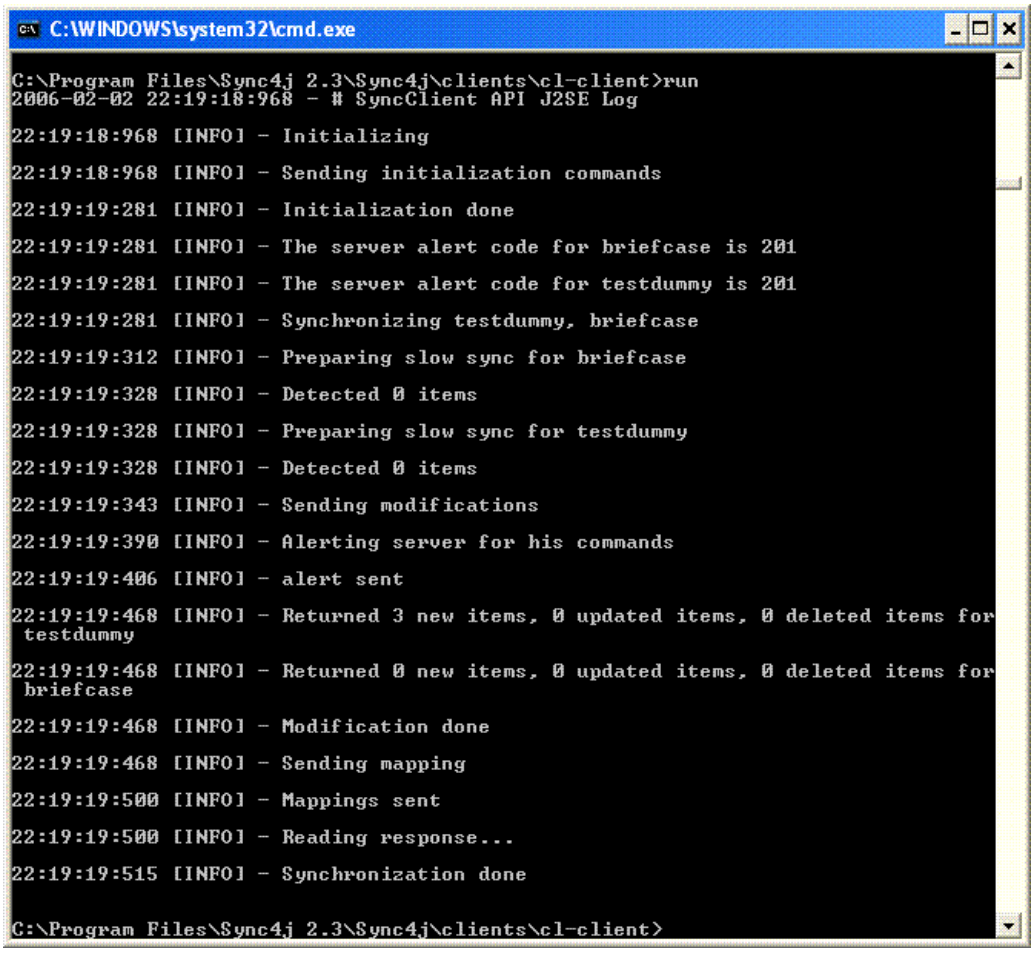
   Source URI: `testdummy`
   Name: `testdummy`
   Type: `text/plain`

6. Click **Add**.

Testing the Module with a SyncML Client

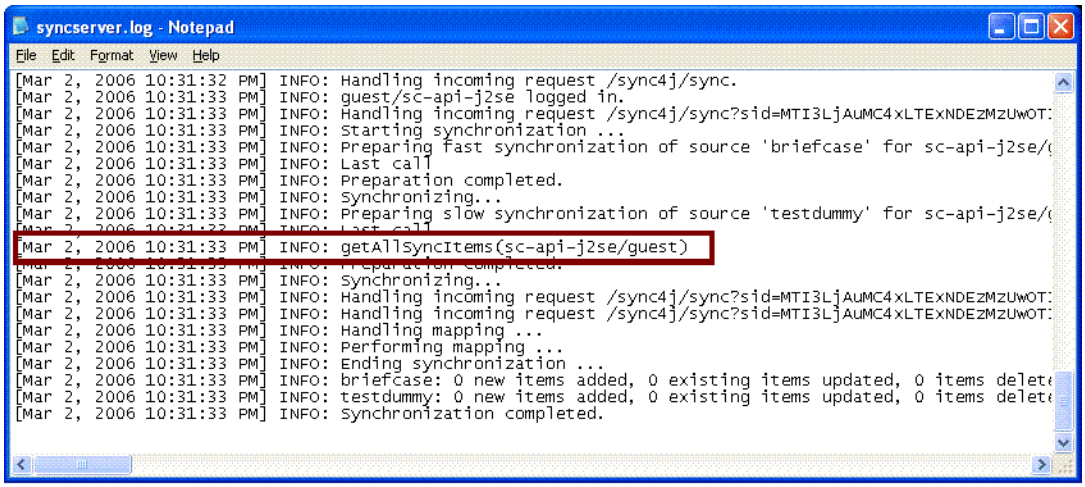
To test the module with a SyncML client, perform the following:

1. Download the Funambol Java Command Line Client Example and unpack the archive.
2. Copy the file `\dummy\examples\dummy.properties` to the `\cl-client\config\spds\sources` directory. Make sure there are no other properties files in this directory.
3. Create the directory `\cl-client\db\dummy`.
4. Verify the `JAVA_HOME` environment property is set correctly.
5. Execute `run.cmd` (or `run.sh` inLinux). The output should appear similar to the following:



If successful, the `\db\dummy` directory contains three new files named **10**, **30** and **40**; these are the items generated by Dummy SyncSource. You can also inspect the content to verify that it corresponds to the text set in the SyncSource code.

You can also view the synchronization results in the server log:



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 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 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*: This document.

Other Resources

This section lists other resources you may find useful.

- For information on Java 2 Standard Edition, visit `http://java.sun.com/j2se`.
- For information on Java 2 Enterprise Edition, visit `http://java.sun.com/j2ee`.
- For information on JBoss, visit `http://www.jboss.org`.
- For information on Apache Tomcat, visit `http://jakarta.apache.org/tomcat`.