Sync4j

Sync4j SyncServer 4.0.x Administration Guide

Table of Contents

1. Introduction

This document is intended for developers and administrators who have to manage Sync4j SyncServer 4.0.6. included in Sync4j 2.2 beta 2.

The document includes:

- Installing Sync4j SyncServer 4.0.6
- Starting and stopping Sync4j SyncServer
- Installing Sync4j SyncServer modules
- Adding users, devices and principals

1.1. Comments and Feedbacks

The Sync4j SyncServer team wants to hear from you! Please submit your questions, comments, feedbacks or testimonials to sync4j-users@lists.sourceforge.net.

2. Installation

This section describes how to install and configure Sync4j SyncServer 4.0.6.

2.1. How to get Sync4j SyncServer

Check the Sync4j homepage (http://www.sync4j.org) for information about the current version and for downloading instructions.

2.2. The Sync4j bundle package and the unbundled applications

Sync4j SyncServer is available in two forms: within the *Sync4j bundle* and as a separate package (*unbundled).* The *Sync4j bundle* includes everything you need to run and manage SyncServer, including the SyncAdmin and the Tomcat server. In order to run the *unbundled* version, you must have an application server on which you deploy SyncServer. Currently, SyncServer has been tested and certified to work on top of JBoss 3.0.x[2], JBoss 3.2.x[2] and Tomcat 5.0.x[5].

2.3. Installing Sync4j bundle

2.3.1. Windows instructions

1. Download sync4j-2.2b2.exe from www.sync4j.org and run it. It will install the software.
2. Follow the instructions during the installation. The server will be started automatically and you will see the Quick Start page. Follow the instructions on the page to run the PIM demos and see the product in action.

2.3.2. Unix/Linux instructions

1. Download sync4j-2.2b2.bin from www.sync4j.org and run it (you may need to make it executable). It will unpackage the software.
1. To start the server, open a shell and cd to the $SYNC4J_HOME directory; then run tools/bin/startall.sh
2. Open the Quick Start document in the docs directory and follow the instructionsciao.

2.4. Installing SyncServer unbundled

The requirements to install Sync4j SyncServer 4.0.6 unbundled are:

1. JDK 1.4.x[1]
2. JBoss 3.0.x[2]
   or
   JBoss 3.2.x[2]
   or
   Tomcat 5.0.x[5]
3. Sync4j SyncServer 4.0.x (syncserver-4.0.x.zip)

The installation procedure is made up of a combination of shell and Ant[4] scripts performing the following tasks:

- Updating configuration files accordingly to user's parameters
- Packaging for deployment on the chosen application server
- Database tables creation
- Deployment on the chosen application server

2.4.1. Installing Sync4j SyncServer 4.0.x Unbundled on JBoss 3.0.x or JBoss 3.2.x

To install SyncServer, follow the procedure below:
1. Install the JDK 1.4.x if not already present.
2. Install JBoss if not already present.
3. Unpack syncserver-4.0.x.jar in a directory of your choice. We will refer to that directory as SYNCSERVER_HOME.
4. Set up your database so that it can be accessed with a dedicated user (e.g. sync4j). This user needs to be granted permissions for connection, creating, deleting and selecting tables.
5. Customize *install.properties* to reflect your system.
6. On unix systems, give execution permission to the executable scripts in bin and ant/bin. Use the command (from SYNCSERVER_HOME):
   ```
   chmod +x bin/*.sh ant/bin/*
   ```
7. From SYNCSERVER_HOME, run:
   ```
   bin/install.sh jboss30 (bin\install.cmd jboss30)
   ```
   or
   ```
   bin/install.sh jboss32 (bin\install.cmd jboss32)
   ```
   (Make sure that the environment variables JAVA_HOME and J2EE_HOME point respectively to your JDK/JRE home and to your JBoss home).
   You will be asked if you want to create the database for SyncServer and some SyncServer modules. Respond yes ('y') to all questions.
8. Start Sync4j SyncServer: from SYNCSERVER_HOME run:
   ```
   bin/start.sh (bin\start.cmd)
   ```
9. Point the browser to *http://<server>:<port>/sync4j* to check that Sync4j SyncServer is properly installed (you should get the welcome page).

2.4.2. Installing Sync4j SyncServer 4.0.x Unbundled on Tomcat 5.0.x

To install SyncServer, follow the procedure below:
10. Install the JDK 1.4.x if not already present.
11. Install Tomcat 5.0.x if not already present.
12. Unpack syncserver-4.0.x.jar in a directory of your choice. We will refer to that directory as SYNCSERVER_HOME.
13. Set up your database so that it can be accessed with a dedicated user (e.g. sync4j). This user needs to be granted permissions for connection, creating, deleting and selecting tables.
14. Customize *install.properties* to reflect your system.

15. On unix systems, give execution permission to the executable scripts in bin and ant/bin. Use the command (from SYNCSERVER_HOME):

```
chmod +x bin/*.sh ant/bin/*
```

16. From SYNCSERVER_HOME, run:

```
bin/install.sh tomcat50 (bin\install.cmd tomcat50)
```

(Make sure that the environment variables JAVA_HOME and J2EE_HOME point respectively to your JDK/JRE home and to your Tomcat home).
You will be asked if you want to create the database for SyncServer and some SyncServer modules. Respond yes ('y') to all questions.

17. Start Sync4j SyncServer: from SYNCSERVER_HOME run:

```
bin/start.sh (bin\start.cmd)
```

18. Point the browser to *http://<server>:<port>/sync4j* to check that Sync4j SyncServer is properly installed (you should get the welcome page).

2.5. The install.properties file

This file is used by the installation procedure as the central repository of configuration information that are needed to properly set up a working SyncServer installation. It is a standard Java properties file containing the properties described in Table 1.

Property	Description	Default Value
context-path	The context path to be used to configure the web container for the SyncServer module. SyncServer will respond to URLs starting with this context path.	/sync4j
dbms	Name of the database where SyncServer tables will be created. One of: • ansisql99 • hypersonic • mysql • oracle • postgresql • sybase • sqlserver	postgresql
jdbc.classpath	Classpath including the JDBC driver for the database if not included in the system classpath.	
jdbc.driver	JDBC driver class.	org.postgresql.Driver
jdbc.password	Database user password	sync4j
jdbc.url	JDBC connection URL	jdbc:postgresql://localhost/sync4j
modules-to-install	Comma separated list of SyncServer modules to install.	foundation-1.4,pdi-1.3
server-name	The server URI that will be specified in the SyncML messages. The server will respond only to messages addressed to this URI.	server-name=http://localhost:8080/sync4j/sync

Table 1 - install.properties properties

For a new SyncServer installation, you have usually to change only the database access configuration and the server-name if you plan to access the server from a remote machine.

3. Starting and Stopping Sync4j SyncServer 4.0.x

This section explains how to start and stop Sync4j SyncServer 4.0.x

The way Sync4j SyncServer 4.0.x is started and stopped usually depends on how the application server on top of which SyncServer is running starts and stops J2EE applications. In this section, we assume Sync4j SyncServer 4.0.x is installed as a standalone application, therefore when SyncServer is stopped, the entire application server is stopped and when it is started, the entire application server is started.

3.1. Windows installation (bundle)

You can manage the SyncServer via the Status icon in the tray (lower right corner). To start the Status icon, from the Start menu, select All Programs → Sync4j → SyncServer → Status (in tray)

You will see the Sync4j icon () in the tray (lower right corner) with two arrows. If the arrows are green, the SyncServer is running. If the icons are red, the SyncServer is not running. If the icons are yellow, the SyncServer is starting: wait until they become green to access it.

Clicking on the Status icon in the tray with the right mouse button will open a menu that will allow you to Start or Stop the SyncServer.

If you want to manually start the SyncServer, select All Programs → Sync4j → SyncServer → Start.
If you want to manually stop the SyncServer, select All Programs → Sync4j → SyncServer → Stop.

3.2. Linux/Unix installation (bundle)

To start Sync4j SyncServer 4.0.x bundled follow the procedure below:

1. cd to SYNC4J_HOME
2. run tools/bin/startall.sh

3.3. Unbundled

3.3.1. Starting Sync4j SyncServer 4.0.x

To start Sync4j SyncServer 4.0.x follow the procedure below:

1. Make sure the following environment variables are correctly set:
    • JAVA_HOME -> the JDK installation directory
    • J2EE_HOME -> the Sync4j SyncServer installation directory
2. From *<SYNCSERVER_HOME>*, run:
   `bin/start.sh (bin\start.cmd)`

3.3.2. Stopping Sync4j SyncServer 4.0.x

Kill the application server process. If it is running in foreground, pressing Ctrl+C should be sufficient: otherwise you have to discover the process id and kill it with an operation system command or tools.

On Tomcat 5.0.x
From theTomcat home and run `bin/shutdown.sh (bin\shutdown.cmd)`

4. Administering Sync4j SyncServer

The following sections describe how to administer SyncServer through SyncAdmin, the SyncServer administration interface.

4.1. The Sync4j SyncServer Administration Tool

Most of the more commons operations such as adding users, devices and principals can be done with the Sync4j SyncServer Administration Tool (a.k.a. SyncAdmin). This section explains how to install and start the SyncAdmin Tool.

With the SyncAdmin Tool, you can:

- Add/Edit/Delete/Search users
- Add/Edit/Delete/Search devices
- Add/Edit/Delete/Search principals
- Manage SyncServer settings (logging)
- Display installed modules/connectors/SyncSource types
- Create/Edit/Delete SyncSources

4.1.1. SyncAdmin download and installation

If you installed Sync4j bundle, SyncAdmin is already installed. You can skip this section.

If you installed SyncServer unbundled, you can get SyncAdmin from the Download section in www.sync4j.org.

Windows installation
The SyncAdmin Tool is delivered in the form of an executable installation file named *SyncAdmin-x.y.exe*, where *x/y* are the major and minor version numbers. To install the program, just run the executable and follow the installation instructions on the screen.

Linux/Unix installation
The SyncAdmin Tool is delivered in the form of an executable installation file named *SyncAdmin-x.y.tgz*, where *x/y* are the major and minor version numbers. Ungzip and untar the package in the directory of choice.

4.1.2. SyncAdmin Start

Windows installation
From the Start menu, select All Programs → Sync4j → SyncAdmin

Linux/Unix installation

Go to the SyncAdmin installation directory and run bin/syncadmin.sh.

4.2. SyncAdmin Interface

Once started, you will see a window like the one in Figure 1.
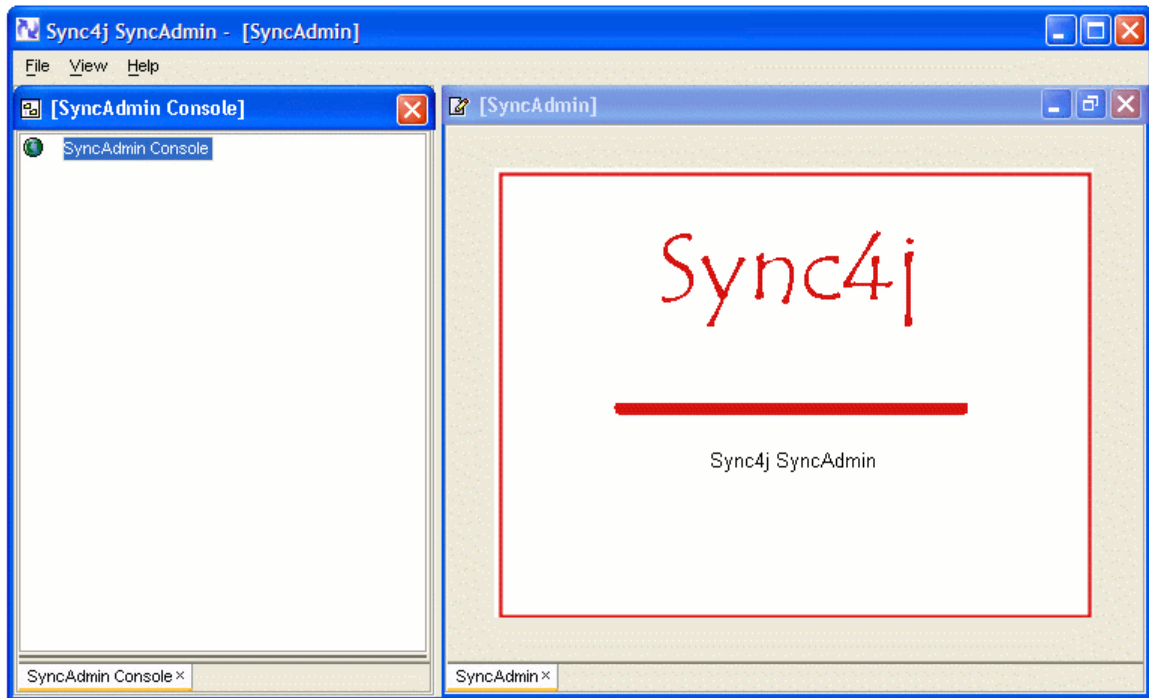
Figure 1 - Sync4j SyncServer Administration tool

The first thing to do before any administration operation is to log on the server. Select File/Login and you will seeet the form of Figure 2.

Figure 2 - Login Form

Fill the login form with the following information:

Host Name: <yourserverhost> or <yourserverip> (localhost would work if you installed the server on you machine; 8080 is the standard port of the bundled installation with Tomcat)
User: admin
Password: sa

and press Login. That user is the default administrator user. Make sure to change the default password as soon as possible.

After being logged in, you will see the name of your server as the root of the navigation tree on the left panel. Double click on it (or single click on the + sign at the left of your server name) and you will see the interface displayed in Figure 3.
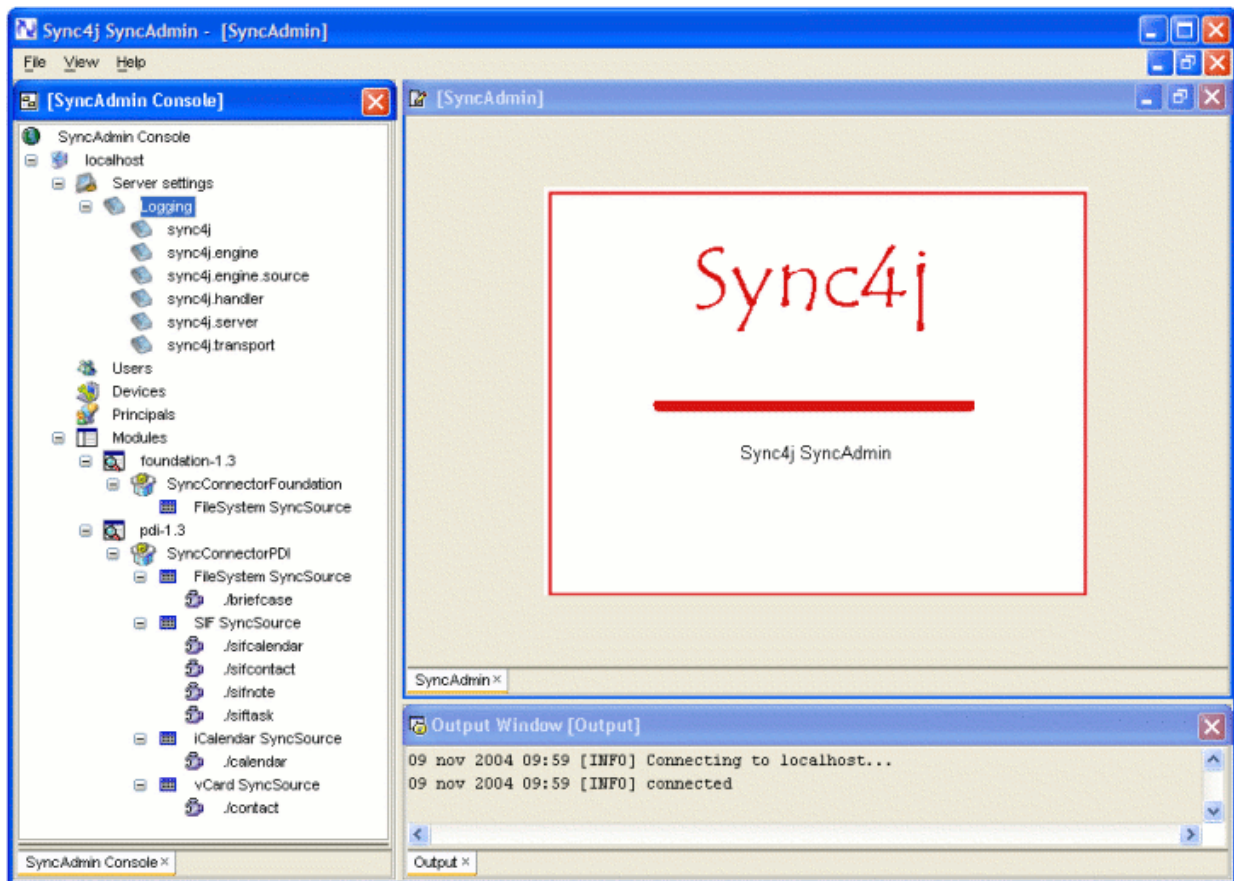
Figure 3 - The administration console

4.3. Users and Roles

In Sync4j SyncServer, users can play one or more of the following roles:

- User
- Administrator

The users with role *User* are the ones enabled to the synchronization. The users with role *Administrator* are the ones that can administer the Sync4j SyncServer installation.

To add a new user, click on the Users node of the SyncAdmin Console and select *Add*.

You can also search and edit existing users. From the *Users* context menu input some of the parameters for the query (such as Username, First Name, Last Name or E-mail) and select *Search*. The table below the search form will be filled with the results of your query.

To edit the information about a user, select a row and press *Edit* (or double-click on it). Press the *Save* button to update your changes.

To delete a user, select a row and press *Delete*.

4.4. Devices

A user connects to the server with a device, such as a SyncML phone or a Sync4j SyncClient PIM.

To add a new device, click on the Devices node of the SyncAdmin Console and select *Add*. You have to insert the device ID (e.g. the phone IMEI number for SyncML phones), the device type (it is a free field, you can put whatever you think reasonable, e.g. *Nokia 6600*) and a description (e.g. *John's phone*).

You can also search and edit existing devices. From the *Devices* context menu input some of the parameters for the query (such as ID, Type or Description) and select *Search*. The table below the search form will be filled with the results of your query.

To edit the information about a device, select a row and press *Edit* (or double-click on it). Press the *Save* button to update your changes.

To delete a device, select a row and press *Delete*.

4.5. Principals

A user can synchronize its data from multiple devices, for example a SyncML phone, Outlook and a Pocket PC PDA. Multiple users can synchronize from a single device, if they share it. Therefore, SyncServer is built around the concept of a *Principal*, which is a more generic concept than a user or a device. A principal is a couple *(user-device)*.

For example, if John uses a Nokia 6600 phone to synchronize its contact to an Outlook client, you will have the following:

Users:
John

Devices:
Nokia 6600, IMEI xyz
SyncClient PIM Outlook

Principals:
John – Nokia 6600
John – SyncClient PIM Outlook

Another example: if John and Susan use the same Nokia 6600 phone to synchronize their contact, you will have the following:

Users:
John
Susan

Devices:
Nokia 6600, IMEI xyz

Principals:
John – Nokia 6600
Susan – Nokia 6600

Note: in order to facilitate the usage of SyncServer by first-time users, the pdi-1.3 module includes a Synclet which bypasses the device check. Once you add a user, you can synchronize your data with any device. In a real-world scenario, you would need to add the device ID (such as the phone IMEI).

To add a new principal, select *Add Principal* from the *Principals* node context menu. You will get a form split in two search forms. You can search users and devices, then select a particular user and a particular device and press *Add Principal*.

You can also search and edit existing principals. From the *Principals* context menu input some of the parameters for the query (such as PrincipalID, Username or DeviceID) and select *Search*. The table below the search form will be filled with the results of your query.

To delete a principal, select a row and press *Delete*.

4.6. Server Settings (logging)

The Server Setting node allows changes and modification of the server settings. In the 2.2 release, only logging can be managed. In future release, more parameters will be manageable from the SyncAdmin.

4.6.1. Logging

Sync4j basic installation includes a single log file, called syncserver.log, stored under the logs directory. However, the system can be configured to specify different levels of log from different components, with the output going to different files of the standard console.

If you do not have specific needs for specific components logging, you just have to manage the sync4j log node under Logging.

The following parameters are supported:

Parameter	Description	Default
Logging level	NONE: no information is logged ERROR: only errors are logged INFO: basic info and errors are logged ALL: info, errors and debug information are logged	INFO
Output to console	It specifies if the log should be visualized on the standard console or not	No
Output to file	It specifies if the log should be stored on a specific file in the file system	Yes
Filename pattern	A filename pattern, which defines the name of the file where the log will be stored (if Output to file is checked). It consists of a string that includes the following "/" the local pathname separator "%t" the system temporary directory "%h" the value of the "user.home" system property "%g" an automatically generated number to distinguish rotated logs "%u" a unique number to resolve conflicts "%%" translates to a single percent sign "%"	log/syncserver.log
File size limit	The maximum file size of the log, in MB	100
Rotation file count	Logging can either be written to a specified file, or it can be written to a rotating set of files. For a rotating set of files, as each file reaches the file size limit, it is closed, rotated out, and a new file opened. Successively older files are named by replacing the %g placeholder in the filename pattern with "0", "1", "2", etc.	1

If you want to manage the logs generated by different Sync4j components (usually only of interest for advanced users), you can select one of the components between:

1. sync4j.engine
2. sync4j.engine.source
3. sync4j.handler
4. sync4j.server
5. sync4j.transport

If the "Same as sync4j" checkbox is checked, the log of the component will be included in the generic sync4j log, as defined above. If you want to manage it separately, uncheck the "Same as sync4j" checkbox. You can thus specify a different log level for this component, a different log file and so on.

4.7. Modules, Sync Connectors, SyncSource Types and SyncSources

SyncServer uses many concepts to group together features and configuration settings.

Module: it is a package used to group together and distribute SyncServer extensions. A Module contains SyncConnectors, Synclets, configuration files, database scripts and so on.

SyncConnector: it is a server extension that integrates SyncServer with an external source of data. It contains everything required for the configuration and the runtime execution of the integration module. This includes basic configuration files, code, software interfaces and graphical user interfaces for the SyncSources configuration. In addition, a SyncConnector defines the SyncSource types, which are the kind of SyncSources an administrator can create and configure.

SyncSource Type: it represents a specific kind of SyncSource, such as File System SyncSource (to access the file system), Exchange Server SyncSource (to access a Microsoft Exchange account) and so on.

SyncSource: is the minimal synchronization unit. A SyncSource represents the entity a client can request to synchronize. A SyncSource is uniquely identified in the server through a source URI, which is the key the client must use to address it.

4.7.1. Standard modules

A standard SyncServer installation comes with two installed modules: *Foundation* and *PDI.*

The Foundation module contains internal components used by the server and the SyncConnectorFoundation. This just defines a SyncSource type called FileSystem SyncSource that can be used to create new file system SyncSources.

The PDI (Personal Data Interchange) module includes the PDI connector and, again, gives the ability to create new file system SyncSources. This module is intended for easily testing SyncServer with PIM synchronization (contacts and calendar). By default, the PDI module brings two preconfigured file system SyncSources, *./contact* and *./calendar*, which are ready to be synced just configuring the clients. The SyncClient PIM demo and the PIM Web Demo use this module. For more information on how to see the PDI module in action, please see the Quick Start Guide, which includes a step-by-step example of a contact synchronization.

4.7.2. SyncSource management

To edit a SyncSource click on its source URI in the SyncAdmin Console. The fields have the following meaning (the example values refer to the vCard SyncSource in the PDI module):

URI: the source URI (e.g. ./contact)
Name: the source mnemonic name (a description of the SyncSource, e.g. contact)
Type: files content mime type (e.g. text/x-vcard)
Source Directory: where files are stored and read
Supported types: comma separated list of the supported mime types; they are sent in the server capabilities packet (e.g. text/x-vcard,text/vcard)
Supported versions: comma separated list of the mime type versions; for each mime type specified in the supported types, a version number must appear here (e.g. 2.1,3.0 means support for vCard 2.1 and 3.0)

Encoded: whether the files content must be Base64 encoded. The feature is useful if you are building a SyncClient and you are planning to transfer binary files. If your SyncSource is meant to synchronize with phones, you should leave this option unchecked.

5. Installing new Sync4j Modules

A Sync4j module is a pluggable extension provided either by the Sync4j development team, a third party or developed by yourself. It is the way you can add new functionalities or modify the standard behavior of a Sync4j SyncServer component.

More details on how to develop a Sync4j module can be found in the Sync4j SyncServer 4.0.x Developer's Guide. This section, instead, explains how to install a Sync4j module.

5.1. Packaging

A Sync4j module is packaged as a zip or jar archive that you have to expand in your <SYNCSERVER_HOME> directory. The archive might contain many files, but the most important one is located under the *modules* subdirectory and is called accordingly to the following pattern:

```
modules/{modulename}-{major}.{minor}.s4j
```

Where *modulename* is the name of the module and *major/minor* are the major and minor version numbers. The s4j module file contains the part of the module that must become part of the SyncServer enterprise archive (a J2EE ear file). It is represented by classes, configuration and initialization files that are processed by the installation procedure.

5.2. Installation

Sync4j modules can be installed in two ways: fully reinstalling the entire SyncServer or installing just the modules. In either methods, the installation file install.properties must be configured with the list of the modules to be included in SyncServer.

In install.properties of a standard installation, the line:

```
modules-to-install=foundation-1.4,pdi-1.3
```

tells the installation procedure to include in the SyncServer final ear (the SyncServer package for the J2EE application server) the modules Foundation 1.4 and PDI 1.3.

5.2.1. Full Installation

After properly setting modules-to-install in install.property, run the installation procedure `bin/install.sh <application server>` (`bin\install.cmd <application server>`).
Note that because this is a fully SyncServer installation you will be asked if you want to rebuild the database: choose 'n' (do not rebuild the database) to keep the existing users, mappings and last syncs information.

Through this method, the installation procedure installs each module in the list; you will be notified of any module installation by proper messages on the screen. Again, for each module, you will also be asked if you want to rebuild the module database. Choose 'y' or 'n' depending on the need of recreating and initializing the module database tables.

5.2.2. Modules-only Installation

This method is not very different from the full installation method. Simply, you have to call a different script.

After setting *modules-to-install* in install.property, call `bin/install-modules.sh <application server>` (`bin\install-modules.cmd <application server>`). This procedure skips the SyncServer installation and just installs each module in the modules-to-install list. You will be notified of every module installation by proper messages on the screen. For each module, you will also be asked if you want to rebuild the module database. Choose 'y' or 'n' depending on the need of recreating and initializing the module database tables.

6. Configuring Sync4j SyncServer 4.0.x

SyncServer is - by design - very flexible and configurable in many of its modules. One of the design goal of the product is to provide a framework that can be used to implement any kind of synchronization service.

There are two configuration techniques used by SyncServer: properties files and server JavaBeans. The former is based on classic properties files that can be read by a *java.util.Properties* object. The *server JavaBeans* configuration type is represented by serialized Java beans stored in the so called *configuration path* (see the Developer's Guide for details).

The following sections describe how to configure many SyncServer aspects, starting with the principal configuration file Sync4j.properties.

6.1. Sync4.properties

This is the main SyncServer configuration file and is located in `<SYNCSERVER_HOME>/config`

Sync4j.properties defines the following properties:

Property	Description	Default
server.uri	The server URI that identifies the server. SyncServer will refuse all synchronization messages addressed to a server URI different from the one indicated by this property.	http://localhost:8080/sync4j/sync
server.id	The server unique identifier. Used for instance in server authentication.	sync4j
syncml.dtdversion	The supported SyncML dtd version	01.01.00
engine.manifacturer	The manufacturer included in the server capabilities.	SyncServer
engine.modelname	The model name included in the server capabilities.	-
engine.oem	The oem name included in the server capabilities.	-
engine.firmwareversion	The firmware version included in the server capabilities.	-
engine.softwareversion	The server version included in the server capabilities.	The server version
engine.hardwareversion	The hardware version included in the server capabilities.	-
engine.deviceid	The device id included in the server capabilities.	Sync4j SyncServer
engine.devicetype	The device type included in the server capabilities	-

Property	Description	Default
engine.strategy	The JavaBeans representing a *sync4j.framework.engine.SyncStrategy* object. This property is interpreted as the name of a JavaBeans. The given value is searched in the configpath as the name of a serialized object. If no serialized object are found, the value is considered equal to the name of a class and it will be searched for in the classpath.	sync4j.server.engine.Sync4jStrategy
engine.store	The JavaBeans representing the persistent store manager.	sync4j/server/store/PersistentStoreManager.xml
security.officer	The JavaBeans representing the security officer (see the next section).	sync4j/server/security/DBOfficer.xml
engine.pipeline	The PipelineManager configuration	sync4j/framework/engine/pipeline/PipelineManager.xml
User.manager	The JavaBeans representing the user manager	sync4j/server/admin/DBUserManager.xml
minMaxMsgSize	The minumum MaxMsgSize supoprted by the server. If a client requests a MaxMsgSize less than this value, the request is rejected.	3000

6.2. Authentication and authorization

SyncServer does not implement complex authentication and authorization mechanisms. The basic implementation simply checks for login and password as defined in the User table, the deviceID in the Device table and that there is a principal with that user/device combination (unless the Synclet for bypassing the deviceID check is installed).

A more complex authentication and authorization mechanism might be required for enterprise and carrier deployments. Usually this is accomplished by dedicated software such as a directory service, where a single sign-on strategy is preferable (the user logs in the system with the same password for every service). A classic problem when applications keep users and user information in a proprietary database is the synchronization between the local database and the corporate database. This is even more problematic when single sign on is required and user logins and passwords must be stored and verified only in one place. Therefore, SyncServer is capable to rely on an external system to perform authentication and authorization, thus bypassing the issue. SyncServer bases its security services on the Java Authentication and Authorization Service architecture provided out of the box with the JDK 1.4.x.

6.2.1. JAAS

The Java Authentication and Authorization Service (JAAS) was introduced as an optional package (extension) to the Java 2 SDK, Standard Edition (J2SDK), v 1.3 and has now been integrated into the J2SDK, v 1.4.

JAAS can be used for two purposes:

- for users authentication of users, to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet; and
- for authorization of users to ensure they have the access control rights (permissions) required to do the actions performed.

JAAS authentication is performed in a pluggable fashion. This allows applications to remain independent from underlying authentication technologies. New or updated authentication technologies can be plugged under an application without requiring modifications to the application itself. Applications enable the authentication process by instantiating a LoginContext object, which in turn references a Configuration to determine the authentication technology/technologies, or LoginModule (s), to be used in performing the authentication. Typical LoginModules may prompt for and verify a username and password. Others may read and verify a voice or fingerprint sample.
Once the user or service executing the code has been authenticated, the JAAS authorization component works in conjunction with the core Java 2 access control model to protect access to sensitive resources.
For additional information about JAAS see

http://java.sun.com/j2se/1.4.1/docs/guide/security/jaas/JAASRefGuide.html

and

http://java.sun.com/j2se/1.4.1/docs/guide/security/jaas/tutorials/index.html .

6.2.2. The JAASOfficer

sync4j.framework.security.JAASOfficer is an implementation of *sync4j.framework.security.Officer* that delegates to JAAS the authentication and authorization functionality.

In order to use this implementation, the system property *java.security.auth.login.config* must be set accordingly to what specified in the JAAS documentation or in the documentation of the application server in use.

SyncServer implements also an empty *LoginModule* that always authenticates and authorizes the users. This module is under the package *sync4j.framework.security.jaas* and is plugged in the JAAS configuration adding the following lines to the login configuration file:

```
sync4j {
      sync4j.framework.security.jaas.SimpleLoginModule   required   required
debug=true;
}
```

See the documentation of the application server in use for details on how to use that login module.

6.3. Database

SyncServer should work with any database for which a JDBC driver exists. After SyncServer is installed, the database access configuration is delegated to the application server. SyncServer uses the JNDI name *jdbc/sync4j* to acquire a connection from the application server. For example, with the Sun J2EE reference implementation, the database connection settings are stored in *{J2EE_HOME}/ config/resource.properties* as a numbered list of datasources and driver definitions. To configure a new datasource, it is sufficient to edit the file and add the following lines:

```
jdbcDataSource.{n}.name=jdbc/sync4j
jdbcDataSource.{n}.url={the jdbc url}
jdbcDriver.{n}.name={the jdbc driver}
```

Where *n* is the next number greater than the maximum existing number.

In addition, in order to tell the application server where to find the JDBC driver classes, the file *{J2EE_HOME}/bin/userconfig.bat/sh* must be edited and the driver classpath must be appended to the environment variable *J2EE_CLASSPATH*.

Please read the documentation of your application server to see how it performs JDBC configuration.

6.3.1. Database Creation

The installation procedure creates the database schema required by SyncServer. There are specific SQL scripts for the most common database systems; the script to be used is specified by the property *dbms* in *install.properties*. There are the scripts for the following databases:

• Standard SQL 99
• Hypersonic
• Informix
• MySQL
• Oracle
• PostgreSQL
• SQLServer
• Sybase

If your database is not in the list try with the SQL 99.

6.3.2. Logging Database Access

SyncServer does not log database access directly from the classes that use JDBC. Instead, a more generic approach is taken, which is based on P6Log (http://p6spy.sourceforge.net), an open source application that logs all JDBC transactions in a seamless manner for the target application. You just need to configure the application server to use the P6Spy JDBC driver instead of the database driver. P6Spy is configured to access the real database. For information on how to install and configure P6Spy, go to http://www.p6spy.com/documentation/index.htm.

For the sake of simplicity, a short list of the steps required to configure SyncServer to use P6Spy is presented here.

1. Download and install P6Spy from the above link
2. Copy spy.jar in your {JAVA_HOME}/jre/lib/ext
3. Copy {SYNCSERVER_HOME}/lib/sync4j-sqllog.jar in {JAVA_HOME}/jre/lib/ext (this contains an adapter for P6Spy to the standard java logging system)
4. Append to the application server CLASSPATH the directory {SYNCSERVER_HOME}/lib/logging (this will allow P6Spy to access its configuration file spy.properties).

7. Common Configuration Changes

This section describes some common configuration changes that you might want to undertake to customize a bundle SyncServer instalaltion.

7.1. Changing the HTTP listening port

To change the HTTP port the server listens to, follow the steps below.

7.1.1. Bundled/Tomcat

1. Edit SYNC4J_HOME/tools/tomcat/conf/server.xml and look for the following comment:
   ```
   <!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
   ```
   In the following line, change "8080" to the port of choice. For example:
   ```
   <Connector port="1234"
              maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
              enableLookups="false" redirectPort="8443" acceptCount="100"
              debug="0" connectionTimeout="20000"
              disableUploadTimeout="true" />
   ```
2. Edit SYNCSERVER_HOME/config/Sync4j.properties and change the port in the server.uri parameter value. For example:
   ```
   server.uri=http://localhost:1234/sync4j/sync
   ```
3. Optionally you may want to change the install.properties too so that if you run bin/install[-modules] you'll keep your change. In this case you will have to change the server-name parameter.
4. Restart Sync4j

7.1.2. JBoss+Tomcat

1. Edit deploy/http-invoker.sar/META-INF/jboss-service.xml and change any occurrence of port 8080 to the port of choice.
2. Edit deploy/jbossweb-tomcat41.sar/META-INF/jboss-service.xml and change any occurrence of port 8080 to the port of choice.
3. Edit SYNC4J_HOME/config/Sync4j.properties and change the port in the server.uri parameter value.
4. Optionally you may want to change the install.properties too so that if you run bin/install[-modules] you'll keep your change. In this case you will have to change the server-name parameter.
5. Restart Sync4j

7.2. Changing the IP/hostname

This change is common when you want to access the server from remote clients (SyncClients and mobile phones). Follow the steps below.

1. Edit SYNCSERVER_HOME/config/Sync4j.properties and change the server part of the URL specified by the server.uri parameter to the wanted ip/hostname. For example:
   ```
   server.uri=http://sync.sync4j.org:1234/sync4j/sync
   ```
2. Optionally you may want to change the install.properties too so that if you run bin/install[-modules] you'll keep your change. In this case you will have to change the server-name parameter.

3. Restart Sync4j

7.3. Changing the database

For the sake of easiness of use, Sync4j bundle brings its own database system to store internal data.
A common change is to use the DBMS you are more familiar with. To do so, you have two choices:

1. Change the application server specific database configuration
2. For the supported application server, change the install.properties file and run bin/install[-modules]

For option 1, refer to the application server documentation.
For option 2, follow the steps below.

1. Edit SYNC4SERVER_HOME/install.properties and set the following properties:
   ```
   jdbc.classpath=<the pathname to the JDBC driver>
   jdbc.driver=<the JDBC driver class>
   jdbc.url=<the JDBC connection URL>
   jdbc.user=<the DB user>
   jdbc.password=<the DB user password>
   ```

   For example, for mysql:

   ```
   jdbc.classpath=<somepath>/mysql-connector-java-3.0.8-stable-bin.jar
   jdbc.driver=com.mysql.jdbc.Driver
   jdbc.url=jdbc:mysql://db.sync4j.org/sync4j
   jdbc.user=sync4j
   jdbc.password=sync4jpwd
   ```
2. Set the environment variable J2EE_HOME to SYNC4J/tools/tomcat
3. From SYNCSERVER_HOME run bin/install[-modules].sh/cmd
4. Restart Sync4j

8. Sync4j Licensing

Sync4j has two licensing options, following what is known as "dual licensing" model. After the adoption by MySQL, this model is becoming very successful as a way to support open source development.

The guiding business principle of dual licensing is one of fair exchange, or Quid pro Quo ("something for something"). From a licensing perspective, there are two different products depending on usage and distribution, though technically they have the same source code.

- For those developing open source applications, the Open Source License allows you to offer your software under an open source / free software license (GLP) to all who wish to use, modify, and distribute it freely. The Open Source License allows you to use the software at no charge under the condition that if you use Sync4j in an application you redistribute, the complete source code for your application must be available and freely redistributable under reasonable conditions. Sync4j bases its interpretation of the GPL on the Free Software Foundation's Frequently Asked Questions.

- The Commercial License, which allows you to provide commercial software licenses to your customers or distribute Sync4j-based applications within your organization. This is for organizations that do not want to release the source code for their applications as open source / free software; in other words they do not want to comply with the GNU General Public License (GPL). If you want more information on pricing, please contact license@sync4j.org.

The idea is to get the best out open source (high quality software, a community of people working together, no vendor lock-in), while providing a source of income to pay for the development of the software (yep, Sync4j developers need to eat...).

In their simplest form, the following are general licensing guidelines:

- If your software is licensed under either the GPL-compatible Free Software License as defined by the Free Software Foundation or approved by OSI, then use our GPL licensed version.

- If you distribute a proprietary application in any way, and you are not licensing and distributing your source code under GPL, you need to purchase a commercial license of Sync4j

Commercially licensed customers get commercially supported product with assurances from Funambol. Commercially licensed users are also free from the requirement of making their own application open source.

For OEM's, ISVs, corporate, and government users, a commercial license is the proper solution because it provides you with assurance from the vendor and releases you from the strict requirements of the GPL license.

Nevertheless, you can test Sync4j under the GPL license and inspect the source code before you purchase a commercial non-GPL license.

9. References and Resources

9.1. Resources

[1] http://java.sun.com/j2se
[2] http://www.jboss.org
[3] http://java.sun.com/j2ee
[4] http://ant.apache.org/
[5] http://jakarta.apache.org/tomcat/