Sync4j 1.1.x Administration Guide

Table of Contents

1. Introduction....3
1.1. Comments and Feedbacks....3
2. Installation....4
2.1. How to Get Sync4j....4
2.2. Installing Sync4j 1.1.x....4
2.2.1. Installing Sync4j 1.1.x Bundled....4
2.2.2. Installing Sync4j 1.1.x Unbundled on Sun J2EE SDK 3.1....5
2.2.3. Installing Sync4j 1.1.z Unbundled on JBoss 3.0.x....5
2.3. The install.properties file....6
2.4. Post Installation Notes....6
3. Starting and Stopping Sync4j 1.1.x....8
3.1. Starting Sync4j 1.1.x....8
3.2. Stopping Sync4j 1.1.x....8
3.2.1. On JBoss 3.0.x....8
3.2.2. On Sun J2EE SDK 1.3.x....8
4. Installing Sync4j Modules....9
4.1. Packaging....9
4.2. Installation....9
4.2.1. Full Installation....9
4.2.2. Modules-only Installation....10
5. Administering Users and Devices....11
5.1. Adding a New Sync4j Principal....11
6. Configuring Sync4j 1.1.x....12
6.1. Sync4.properties....12
6.2. Security....13
6.2.1. JAAS....13
6.2.2. The JAASOfficer....13
6.3. Database....14
6.3.1. Database Creation....14
6.4. Database Schema....14
6.5. Logging....15
6.5.1. Sync4j Logging....15
6.5.2. Enabling the Most Verbose Logging....16
6.5.3. Logging Database Access....16
7. Common Configuration Changes....17
7.1. Logging....17
7.2. Temprary Files....17
7.3. Security....17
7.4. Authentication....17
8. References and Resources....18
8.1. Resources....18

1. Introduction

This document is intended for developers and administrators who have to manage Sync4j 1.1.x. It includes:

- Installing Sync4j 1.1.x
- Starting and stopping Sync4j
- Installing Sync4j modules
- Adding users, devices and principals

1.1. Comments and Feedbacks

The Sync4j 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 1.1.x so that it can handle PDI data represented by vCard and vCalendar objects.

2.1. How to Get Sync4j

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

Sync4j is distributed as an archive file called sync4j-x.y.z.zip where x,y and z are the major, minor and build numbers.

2.2. Installing Sync4j 1.1.x

Sync4j is available in two forms: *bundled* with the JBoss application server[2] and *unbundled*; in the latter case you must have an application server on which deploy the Sync4j server. Currently, Sync4j can be directly deployed on top of JBoss 3.0.x and Sun J2EE SDK 3.1[3].

The unbundled Sync4j (sync4j-{major}.{minor}.{build}.jar) is the base package that can be deployed on supported application servers. The bundled Sync4j (sync4j-jboss-{major}.{minor}.{build}.jar) is a distribution that contains a bundled application server. Currently Sync4j is bundled with JBoss 3.0.8.

The requirements to install Sync4j 1.1.x bundled are:

1. JDK 1.4.x[1]
2. Sync4j 1.1.x bundled archive (sync4j-jboss-1.1.x.zip)

The requirements to install Sync4j 1.1.x unbundled are:

1. JDK 1.4.x[1]
2. Sun J2EESDK 1.3.1[3]
   or
   JBoss 3.0.x[2]
3. Sync4j 1.1.x (sync4j-1.1.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

2.2.1. Installing Sync4j 1.1.x Bundled

To install Sync4j, follow the procedure below:

1. Install the JDK 1.4.x if not already present.
2. Unpack sync4j-jboss-1.1.x.jar in a directory of your choice. We will refer to that directory as the *installation directory*.
3. Under the installation directory you'll find the SYNC4J_HOME directory, which is called *sync4j-1.1*. Go into that directory and run:
   ```
   bin/start.sh (bin\start.cmd)
   ```
   (Make sure that in your environment the JAVA_HOME variable is properly set).
4. Point the browser to http://<server>:8080/sync4j to check that Sync4j is properly installed (you should get the welcome page).

2.2.2. Installing Sync4j 1.1.x Unbundled on Sun J2EE SDK 3.1

To install Sync4j, follow the procedure below:

1. Install the JDK 1.4.x if not already present.
2. Install the J2EE SDK if not already present.
3. Unpack sync4j-1.1.x.jar in a directory of your choice. We will refer to that directory as SYNC4J_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 connecting, creating, deleting and selecting tables.
5. Customize *install.properties* to reflect your system
6. Start the Sun J2EE RI server.
7. On unix systems, give execution permission to the executable scripts in bin and ant/bin. Use the command (from SYNC4J_HOME):
   ```
   chmod +x bin/*.sh ant/bin/*
   ```
8. From SYNC4J_HOME, run:
   ```
   bin/install.sh sunri (bin\install.cmd sunri)
   ```
   (Make sure that the environment variables JAVA_HOME and J2EE_HOME point respectively to your JDK home and to your J2EE SDK home)
   You will be asked if you want to create the database for Sync4j and some Sync4j modules. Respond yes ('y') to all questions.
9. Edit *{J2EE_HOME}/config/resource.properties* and add the following lines:
   ```
   #
   # Added for Sync4j
   #
   jdbcDataSource.5.name=jdbc/sync4j
   jdbcDataSource.5.url={the jdbc url}
   jdbcDriver.5.name={the jdbc driver}
   ```
10. Edit *{J2EE_HOME}/bin/userconfig.bat/sh* and append to the J2EE_CLASSPATH the complete classpath of your jdbc driver.
11. Stop the Sun J2EE RI server
12. Start Sync4j: from SYNC4J_HOME run:
    ```
    bin/start.sh (bin\start.cmd)
    ```
13. Point the browser to *http://<server>:<port>/sync4j* to check that Sync4j is properly installed (you should get the welcome page).

2.2.3. Installing Sync4j 1.1.x Unbundled on JBoss 3.0.x

To install Sync4j, follow the procedure below:

1. Install the JDK 1.4.x if not already present.
2. Install JBoss 3.0.x if not already present.
3. Unpack sync4j-0.1.x.jar in a directory of your choice. We will refer to that directory as SYNC4J_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 SYNC4J_HOME):
   ```
   chmod +x bin/*.sh ant/bin/*
   ```

7. From SYNC4J_HOME, run:
   `bin/install.sh jboss (bin\install.cmd jboss)`
   (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 Sync4j and some Sync4j modules. Respond yes ('y') to all questions.
8. Start Sync4j: from SYNC4J_HOME run:
   `bin/start.sh (bin\sstart.cmd)`
9. Point the browser to *http://<server>:<port>/sync4j* to check that Sync4j is properly installed (you should get the welcome page).

2.3. 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 Sync4j 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 Sync4j module. Sync4j will respond to URLs starting with this context path.	/sync4j
dbms	Name of the database where Sync4j 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 Sync4j modules to install.	pdi-1.0
server-name	The server URI that will be specified in the SyncML messages. The server will respond only to messages addressed to this URI.	http://localhost

Table 1 - install.properties properties

For a new Sync4j installation, you have usually to change only the database access configuration.

2.4. Post Installation Notes

The standard Sync4j installation configures Sync4j with two FileSystem SyncSources to store PDI (Personal Data Information) vCard and vCalendar items. Items are stored in the *<SYNC4J_HOME>/db directory* under, respectively, *contact* and *calendar* subdirectories: the directory name represents the name of the database and each file represents one contact or calendar card (the filename is the card id).
In a new Sync4j installation, those directories contain a few sample contacts/appointments (Figure 1).

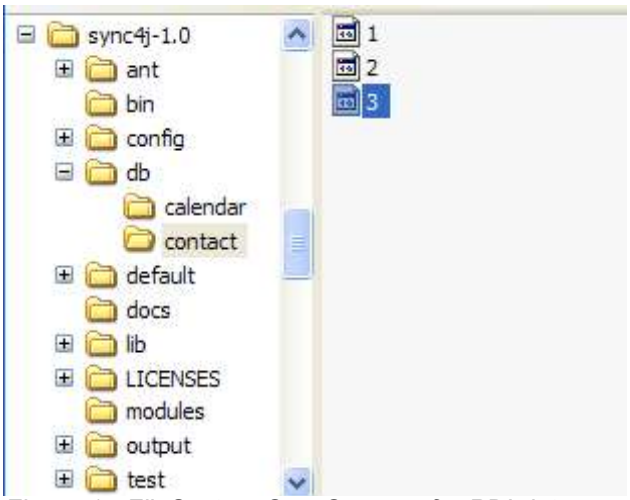

Figure 1 - FileSystem SyncSources for PDI data

3. Starting and Stopping Sync4j 1.1.x

This section explains how to start and stop Sync4j 1.1.x

The way Sync4j 1.1.x is started and stopped usually depends on how the application server on top of which Sync4j is running starts and stops J2EE applications. In this section, we assume Sync4j 1.1.x is installed as a standalone application, therefore when Sync4j is stopped, the entire application server is stopped and when it is started, the entire application server is started.

3.1. Starting Sync4j 1.1.x

To start Sync4j 1.1.x follow the procedure below:

1. Make sure the following environment variables are correctly set:
   - JAVA_HOME -> the JDK installation directory
   - J2EE_HOME -> the Funambol Sync4j installation directory
2. From <*Sync4j_HOME*>, run:
   `bin/start.sh (bin\start.cmd)`

3.2. Stopping Sync4j 1.1.x

3.2.1. On JBoss 3.0.x

To stop Sync4j 1.1.x follow the procedure below:

1. Point the browser to the url:
   `http://<server>:<port>/jmx-console/HtmlAdaptor?action=invokeOpByName&name=jboss.system:type=Server&methodName=shutdown`

3.2.2. On Sun J2EE SDK 1.3.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.

4. Installing Sync4j Modules

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

More details on how to develop a Sync4j module can be found in the Sync4j 1.1.x Developer's Guide. This section, instead, explains how to install new and existing Sync4j modules.

4.1. Packaging

A Sync4j module is packaged as a zip or jar archive that you have to expand in your <SYNC4J_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 Sync4j enterprise archive (a J2EE ear file). It is represented by classes, configuration and initialization files that are processed by the installation procedure.

4.2. Installation

Sync4j modules can be installed in two ways: fully reinstalling the entire Sync4j as described in a previous section 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 Sync4j.

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

```
modules-to-install=pdi-1.0
```

tells the installation procedure to include in Sync4j the PDI module version 1.0.

4.2.1. Full Installation

After setting modules-to-install in install.property you just run the installation procedure `bin/install.sh <application server>` (`bin\install.cmd <application server>`).
Note that because this is a fully Sync4j 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.

With 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.

4.2.2. 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 Sync4j 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.

5. Administering Users and Devices

Sync4j 1.1.x keeps no real users database. This is not the role of a synchronization engine and is best performed by dedicated databases or directory services. Instead, Sync4j takes advantage of the JAAS (Java Authentication and Authorization Services) architecture implemented in the majority of the application servers on the market. This way, the duty of managing users, storing password, authenticating and authorizing users to a service is demanded to the application server itself, without duplicating the users database.

In Sync4j, data are associated to a *Principal*, which is a more generic concept than a person. A principal may represent a user, a device, an application and so on. In Sync4j a principal is a couple (userid-deviceid), covering the cases were a user can use different devices and a device can be used by many users.

The following sections describe how to insert a new principal to the Sync4j database so that it can be used to retrieve personalized information.

5.1. Adding a New Sync4j Principal

As said, a principal is an association between a userid (or username) and a device id. Therefore, to make this association we need to insert a new user into the *sync4_user* table and a new device into the *sync4j_device* table. Then, we can insert the principal in *sync4j_principal*.

For example, suppose we want to insert the principal (jdoe, IMEI:351111103384988). We use the following SQL statements:

```
insert into sync4j_user (username, email, first_name, last_name)
  values('jdoe', 'john_doe@somewhere.com', 'John', 'Doe');

insert into sync4j_device (deviceid, description, type)
  values('IMEI:351111103384988', 'John Doe''s mobile phone', 'Nokia 7650');

insert into sync4j_principal (principalid, username, device)
  values('103', 'jdoe', 'IMEI:351111103384988');
```

Note that user's email and first and last name are inserted for benefit of the administrator only: they are not required or used by Sync4j, but inserted just to make easier for administrative staff to find a particular person.

6. Configuring Sync4j 1.1.x

Sync4j 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 Sync4j: 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 Sync4j aspects, starting with the principal configuration file Sync4j.properties.

6.1. Sync4.properties

This is the main Sync4j configuration file and is located in `<SYNC4J_HOME>/config`. That location can be changed editing the EJB descriptor file *default/config/xml/META-INF/ejb-jar.xml* and changing the value of the environment entry *server/config_uri*.

Sync4j.properties defines the following properties:

Property	Description	Default
server.uri	The server URI that identifies the server. Sync4j will refuse all synchronization messages addressed to a server URI different from the one indicated by this property.	http://localhost:8080
syncml.dtdversion	The supported SyncML dtd version	1.1
engine.manifacturer	The manufacturer included in the server capabilities.	Sync4j
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.	1.1
engine.hardwareversion	The hardware version included in the server capabilities.	-
engine.deviceid	The device id included in the server capabilities.	-
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/JAASOfficer.xml
engine.pipeline	The PipelineManager configuration	sync4j/framework/engine/pipeline/PipelineManager.xml

6.2. Security

Sync4j does not implement complex authentication and authorization mechanisms; usually this is accomplished by dedicated software such as directory services. In addition, 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. Instead, Sync4j 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.
Sync4j 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

Sync4j should work with any database for which a JDBC driver exists. After Sync4j is installed the database access configuration is delegated to the application server. Sync4j 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.

Note that the installation procedure for JBoss, configure the application server automatically setting the required configuration files based on the JDBC information found in *install.properties*.

6.3.1. Database Creation

The installation procedure creates the database schema required by Sync4j. 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.

6.4. Database Schema

The database schema used by Sync4 is depicted in Figure 2.

Below, you find a brief description of the tables:

- **User** (`sync4j_user`): contains the users's name and details;
- **Device** (`sync4j_device`): contains the devices Sync4j can deal with. The id is what is specified by the <Source><LocURI> element in the <SyncHeader> of the SyncML message;
- **Principal** (`sync4j_principal`): is the entity that can make a synchronization request. Conceptually, the principle is the couple (username, device);

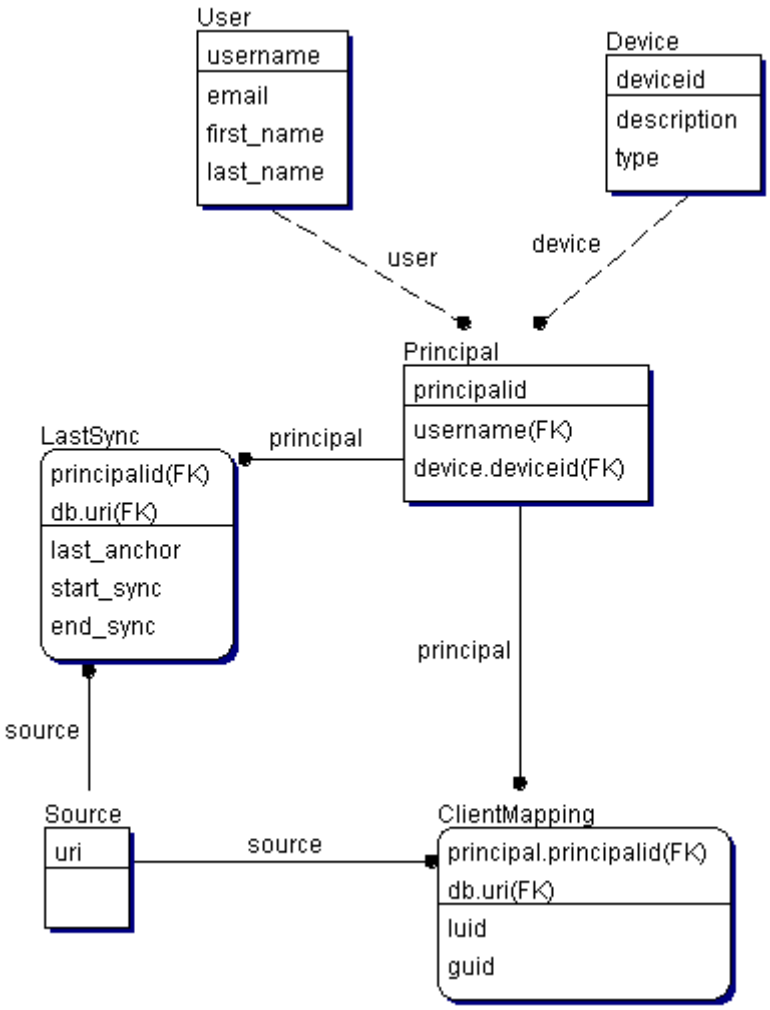

Figure 2 - Engine database schema

- **LastSync** (`sync4j_last_sync`): stores the synchronization anchors of the most recent synchronization of a particular database by a particular principal;
- **ClientMapping** (`sync4j_client_mapping`): stores the LUID-GUID mappings for a particular database and principal;
- **Source** (`sync4j_sync_source`): contains the known sync sources Sync4j can deal with. A source is identified by its URI, which is used also as a key in the referring tables.

6.5. Logging

Sync4j uses the standard Java Logging APIs introduced with the JDK 1.4.x.
For detailed information about the Java Logging APIs, see http://java.sun.com/j2se/1.4.1/docs/guide/util/logging/overview.html.
The output produced by the logging system can be configured in terms of content and writing media (the system output console, the file system, a database, etc.). To configure the JDK logging system, edit the file *`{SYNC4J_HOME}/lib/logging/logging.properties`*.
It is recommended to configure the logging system to output logging information to files instead of to the standard output/error streams. This way, you can also control how big log files can become and how rotate them.

6.5.1. Sync4j Logging

Sync4j uses many logging namespaces, so that you can easily select which module should generate logging and which module should not. The namespaces defined by Sync4j are as follows:

Name	Description
sync4j	It is the default logging namespace, used when no other namespace is specified.
sync4j.engine	Synchronization engine logging information.
sync4j.handler	Session handling logging information.
sync4j.source	SyncSource related logging information.

You can set the verbosity level of a log namespace by setting its *level* property:

```
sync4j.level=INFO
```

If not otherwise specified, the verbosity level is inherited by all subnames (such as sync4j.engine or sync4j.source). To overwrite the inherited value, you can set explicitly the subname level like in the following example:

```
sync4j.engine=ALL
```

In the case above, the default logging level is set to INFO, whilst the engine logging is configured to show any message with any severity. Since logging impacts on performance, on a production system the recommended logging level is SEVERE, so that only errors will get displayed.

6.5.2. Enabling the Most Verbose Logging

In order to get the most verbose logging information, you should follow these steps:

- Edit {SYNC4J_HOME}/lib/logging.properties and set .level, java.util.logging.ConsoleHandler.level or any other handler you are using to ALL (eg.: .level=ALL)
- set sync4j.level to ALL
- Restart Sync4j

6.5.3. Logging Database Access

Sync4j 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 Sync4j 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 {SYNC4J_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 {SYNC4J_HOME}/lib/logging (this will allow P6Spy to access its configuration file spy.properties).

SLQ logging is turned on and off acting on the logging configuration file *{SYNC4J_HOME}/lib/logging/logging.properties.*

7. Common Configuration Changes

This section describes some common configuration changes that you might want to undertake for a production system. Since this chapter is addressed to production environments, it focuses on the JBoss deployment.

7.1. Logging

7.2. Temporary Files

7.3. Security

7.4. Authentication

A common configuration change regards how users are authenticated. The authentication service is abstracted in JBoss (like in Sync4j) through the adoption of the Java Authentication and Authorization Services (JAAS). JBoss provides out of the box login modules for authenticating users by files, relational database and LDAP directory server.
When Sync4j is first installed, JBoss is configured to use the *org.jboss.security.auth.spi.UsersRolesLoginModule* login module, which is based on the use of *users.properties* and *roles.properties; the former stores users and passwords and the latter stores users roles.*

To change the login module to use with Sync4j you need to modify *login-config.xml* located under *server/sync4j/conf* changing the application policy associated to sync4j with the one of your choice.

For example, to authenticate users from the db use:

```
<application-policy name="sync4j">
  <authentication>
    <login-module code = "org.jboss.security.auth.spi.DatabaseServerLoginModule"
                flag = "required"


    >
      <module-option name = "dsJndiName">java:jdbc/sync4j</module-option>
      <module-option name="principalsQuery">SELECT password FROM users WHERE userid = ?</module-option>
      <module-option name="rolesQuery">SELECT role, group FROM roles WHERE userid = ?</module-option>
    </login-module>
  </authentication>
</application-policy>
```

8. References and Resources

8.1. Resources

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