Sync4j

Sync4j Server DS Administration Guide
Version 2.3

04/05/05

Table of Contents

1. Introduction

This document is intended for developers and administrators who have to manage Sync4j Server DS. Since version 2.3 some changes have been done to the configuration of the server and therefore, this document cannot be used for some parts in combination with versions from before 2.3. The document will guide you through the following items:

- Installation of the Sync4j Server DS
- Starting and stopping Sync4j Server DS
- Installation of Sync4j Server DS modules
- Administration of users, devices and principals with the Sync4j Administrator
- Reconfiguring the Sync4j Server DS

Related Documents

The following documents are related to this quick start guide:

- Sync4j Server DS Developer Guide, this guide provides more detail on how to develop your own extenstions for the Sync4j Server DS.
- Sync4j Server DS Architecture Document, this guide provides generic information for the complete Sync4j framework.

1.1. Definitions

This section provides information on how to interpret the use of references to directories.

INDICATOR	***meaning***
<SYNC4J_HOME>	The top level directory of the bundled Sync4j directory. For a bundled installation this is ***'path/to/Sync4j/'*** .
<SYNCSERVER_HOME>	The directory where the syncserver implementation resides. For a bundled installation this is <SYNC4J_HOME>/***syncserver-<version>****.*
<SYNC4J_ADMIN_HOME>	The directory where the Sync4j Admintool resides. For a bundled installation this is <SYNC4j_HOME>/***syncadmin-<version>***.
<CATALINA_HOME>	The directory where Tomcat is installed. For a bundled installation this is <SYNC4J_HOME>/***tools/tomcat***.

INDICATOR	***meaning***
<J2EE_HOME>	This is the directory where the application server resides. For a Tomcat or bundled version this is equal to <CATALINA_HOME> For a jBoss installation this is the directory where jBoss resides.
<JAVA_HOME>	This is the directory where the Java Development Kit is installed. For more information is refered to the JDK documentation.
<JRE_HOME>	This is the directory where the Java Runtime Environment is installed. For more information is refered to the JRE documentation.

1.2. Download the Sync4j Server DS

The Sync4j homepage (http://www.sync4j.org) provides you general information about the project, documentation, support, development and downloads. Click for the downloads on the 'Download'.

First a selection of stable release or developer release is needed.

- The **stable version** is tested for the provided feature set.
- The **developer version** is work in progress, but may have additional features you also need. It must be noted that a released developer release will work for the common use of the Sync4j Server DS, it may have still unknown problems.

Both releases have a bundled package or inidividual packages. The bundled package is a convenience distribution for users that do not want to install all individual package or for instance, would like to operate Sync4j quickly.

- The **bundled version** is set of packages composed of the Sync4j Server DS, the Sync4j Admin, the Sync4j Demo client and the dependant packages Tomcat, JRE, and HSQLDB. This package is a convenient package if you want ot get quickly an operational synchronization server.
- The individual packages require that you download all components you need, but it also requires that a pre-installed application server, JDK/JRE, and a JDBC compliant Database. For the Sync4j Server DS this is referred to as the **unbundled version**.

2. Installation of a Default Sync4j Server DS

This section describes how to get you a Sync4j Server DS up and running from download to readiness for the first synchronization. For more information of the choice of packages is refered to section 'Download the Sync4j Server DS'.

2.1. Installing Sync4j bundle

This section will guide you through the installation of the bundled Sync4j Server DS. The installation for Windows and Linux vary and, therefore, goto your platform specific instructions.

NOTE: The '<version>' must be replacewith the version number you have downloaded.

2.1.1. Windows instructions

1. Download sync4j-<version>.exe from www.sync4j.org
2. Execute the downloaded binary in order to install the server.
3. Follow the instructions during the installation.
    - The first screen will ask you if you agree to the license. It is suggested to review it and decide if you want to accept it.
    - After you agreed with the license you can determine where the package must be installed.
    - In the last screen you can select if you want the server to be started automatically and if you to view the quick start. If you selected the automatical start of the server you will get the Sync4j Server DS icon in the System Tray. If it turns green the server is ready to be used. For more information on starting and stopping is refered to section 3.
4. For more information about how to perform synchronizations is refered to the respective sections later in this document.

2.1.2. Unix/Linux instructions

These instructions are for the Sync4j Server DS bundlle with the Linux OS.

1. Download sync4j-<version>.bin from www.sync4j.org on you system.
2. Execute the downloaded file in a shell.

   shell) **sh sync4j-<version>.bin**

- This will ask you first for agreement with the license. It is suggested to review it and decide if you want to accept it.
- After you agreed with the license you can determine where the package must be installed by providing the 'prefix'. and then unpackage the software in prefix/Sync4j. This will be your <SYNC4J_HOME>.

NOTE: The installation procedure also prints out the configured server name which is the URL that you MUST use also in the mobile device. However, there must be a connection be possible from the device to the server. For instance, the use of localhost enables ONLY access from the same system.

3. Your installation is now ready for starting the server and performing your first synchronizations. For more information about how to start the server and how to perform synchronizations is refered to the respective sections later in this document.

2.2. Installing Sync4j Server DS unbundled

The unbundled installation requires that you will install all the packages seperately and some of those may already have been installed. In general to get the Sync4j Server DS ready for use, you have to install an application server, JDK/JRE, a JDBC compliant Database and the Sync4j Server DS. However, you may want to install more Sync4j components like the Sync4j Administrator.

2.3. Installation of the required packages for the Sync4j Server DS

Java Development Kit

A Java Development Kit 1.4.x must be installed and can be downloaded from http://java/.sun/com. For installation is refered to the installation produre that comes with your JDK.
Ensure the JAVA_HOME environment variabale is set (it should point to the top directory of the JDK) because it is required by the Sync4j Server DS.

Application Server

An application server must be installed and one of the following application servers is suggested:

- JBoss 3.0.x
- JBoss 3.2.x
- Tomcat 5.0.x

Other application server may work, but the Sync4j Server DS has not been tested with them. For the specific installation and configuration procedures of the application server is refered to the documentation of that application server.
Ensure your J2EE_HOME environment variabale is set (it should point to the top directory of the application server) because it is required by the Sync4j Server DS.

JDBC compatible database

A JDBC compliant database must be installed, for installation and configuration is refered to the documentation of the database of your choice. A good choice could be the PostgreSQL database which is also used by the developers of Sync4j.
The usage of the database by Sync4j Server DS requires that you have created a database and a user that can access this database with the granted permissions for connecting, creating, deleting, reading and and writing tables.
Default values used by the Sync4j Server DS are, database=sync4j and username=sync4j, but you are free to choose which ever names you like, but remember them for later during the Sync4j Server DS installation.

2.3.1. Installation of the Sync4j Server DS

The Sync4j Server DS is a component of your application server which perform the SyncML Server functionality.

To install Sync4j Server DS, follow the procedure below:

1. Unpack the downloaded Sync4j Server DS in a directory of your choice. We will refer to that directory as SYNCSERVER_HOME and change to that directory

   shell) **cd <SYNCSERVER_HOME>**

2. Customize *install.properties* to reflect your system.
   Two configuration parts are needed to be correct,
   - the server-name property that must be the URL via which your clients will access the Sync4j Server DS. If you client is on the same system you can use the localhost as hostname of the URL.
   - your choosen JDBC compliant database must match the properties 'dbms', 'jdbc.classpath',' jdbc.driver', 'jdbc.url', 'jdbc.user', 'jdbc.password''.
3. **JBOSS:** Verify that your environment variables JAVA_HOME and J2EE_HOME point respectively to your JDK/JRE home and to your JBoss home.
   TOMCAT: Verify that your environment variables JAVA_HOMEpoint to your JDK/JRE home and J2EE_HOME and CATALINA_HOME point to your Tomcat home.

4. Deploy/install the Sync4j Server DS. During the installation you will be asked if you want to create the database for Sync4j Server DS and some Sync4j Server DS modules. If this is your first (from scratch) installation you MUST respond with yes ('y') to all questions.

   JBOSS
   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/install.cmd jboss**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/install.sh jboss**

   TOMCAT
   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/install.cmd tomcat**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/install.sh tomcat**

3. Starting and Stopping Sync4j Server DS

This section explains how to start and stop Sync4j Server DS.

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

3.1. Bundled Windows installation

The server is started from the Desktop with

Start -> 'All Programs' -> Sync4j -> SyncServer -> Start

This will give you the Sync4j icon in the system-tray (lower right corner) with two curved arrows. The color of the icon indicates the status of the server.

Green indicates the Sync4j Server DS is running.
Red indicates the Sync4j Server DS is not running.
Yellow, indicates the Sync4j Server DS is starting: wait until they become green to access it.

If the icon is green you can verify the access to the server by pointing your browser to *http://<server>:<port>/sync4j* where you should get the welcome page.

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 Sync4j Server DS .

If you want to manually stop the Sync4j Server DS from the Desktop use

Start -> 'All Programs' -> Sync4j -> SyncServer -> Stop

NOTE: You may also use the 'exit' option of the Status icon. This will terminate the service with which you can see if the server is still running. By terminating this service you do NOT stop the server. This is done to avoid a server from stopping if the user disconnects from the Desktop.

3.2. Bundled Linux/Unix installation

To start Sync4j Server DS bundled follow the procedure below:

1. Start Sync4j Server DS:

    shell) **cd <BUNDLED_HOME>**
    shell dir=<BUNDLED_HOME>) **sh tools/bin/sync4j.sh start**

2. Point the browser to *http://<server>:<port>/sync4j* to check that Sync4j Server DS is properly started (you should get the welcome page).

3.3. Unbundled Installation

3.3.1. Starting Sync4j Server DS

To start the unbundled version one should have installed the Sync4j Server DS as was documented in the previous section and have made sure that the environment is created correctly.

1. The following commands would start the Sync4j Server DS.

    (for windows)
    shell dir=<SYNCSERVER_HOME>) **bin/start.cmd**

    (for UNIX)
    shell dir=<SYNCSERVER_HOME>) **bin/start.sh**

2. Point the browser to *http://<server>:<port>/sync4j* to check that Sync4j Server DS is properly started (you should get the welcome page).

3.3.2. Stopping Sync4j Server DS

Stopping the Sync4j Server DS depends on the application server used.

JBOSS

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.

Tomcat 5.0.x

(for windows)
shell dir=<CATALINA_HOME>) **bin/shutdown.cmd**

(for UNIX)

shell dir=<CATALINE_HOME>) **bin/shutdown.sh**

4. Administering Sync4j Server DS

The following sections describe how to administer Sync4j Server DS through Sync4j Admin, the Sync4j Server DS administration interface.

4.1. The Sync4j Server DS Administration Tool

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

With the Sync4j Admin Tool, you can:

- Add/Edit/Delete/Search users
- Add/Edit/Delete/Search devices
- Add/Edit/Delete/Search principals
- Manage Sync4j Server DS settings (like logging)
- Display installed modules/connectors/SyncSource types
- Create/Edit/Delete SyncSources

4.1.1. Sync4j Admin download and installation

If you installed Sync4j bundle, Sync4j Admin is already installed and you may skip this section. Otherwise, you can get Sync4j Admin from the Download section in www.sync4j.org. If you will install the SyncAdmin you need to use the version that is released in combination with the Sync4j Server DS you use. This is needed because between version do exist little differences causing the Sync4j Admin tool not towork properly.

Windows installation

1. Download sync4j-<version>.exe from www.sync4j.org
2. Execute the downloaded binary in order to install the server.
3. Follow the instructions during the installation.
    - The first screen will ask you if you agree to the license. It is suggested to review it and decide if you want to accept it.
    - After you agreed with the license you can determine where the package must be installed.
    - In the last screen you can select if you want to see the quick start.

Linux/Unix installation

The Sync4jAdmin Tool is delivered in the form of an tarball installation file named s*ync4j-admin-<version>.tgz*.

```
shell) cd <SYNC4J_ADMIN_HOME>
shell dir=<SYNC4J_ADMIN_HOME>) gunzip sync4j-admin-<version>.tar.gz
shell dir=<SYNC4J_ADMIN_HOME>) tar xvf sync4j-admin-<version>.tar
```

4.1.2. Sync4j Admin Start

Windows installation

The sync4j admintool is started from the

Start -> 'All Programs' -> Sync4j -> Admin <version> -> Admin

Linux/Unix installation

1. The following commands would start the Sync4j Server DS.

```
shell) cd <SYN4J_ADMIN_HOME>
shell dir=<SYNC4j_ADMIN_HOME>) bin/syncadmin.sh
```

4.2. Sync4j Admin Interface

Once started, you will see a window like the one in Figure 1.

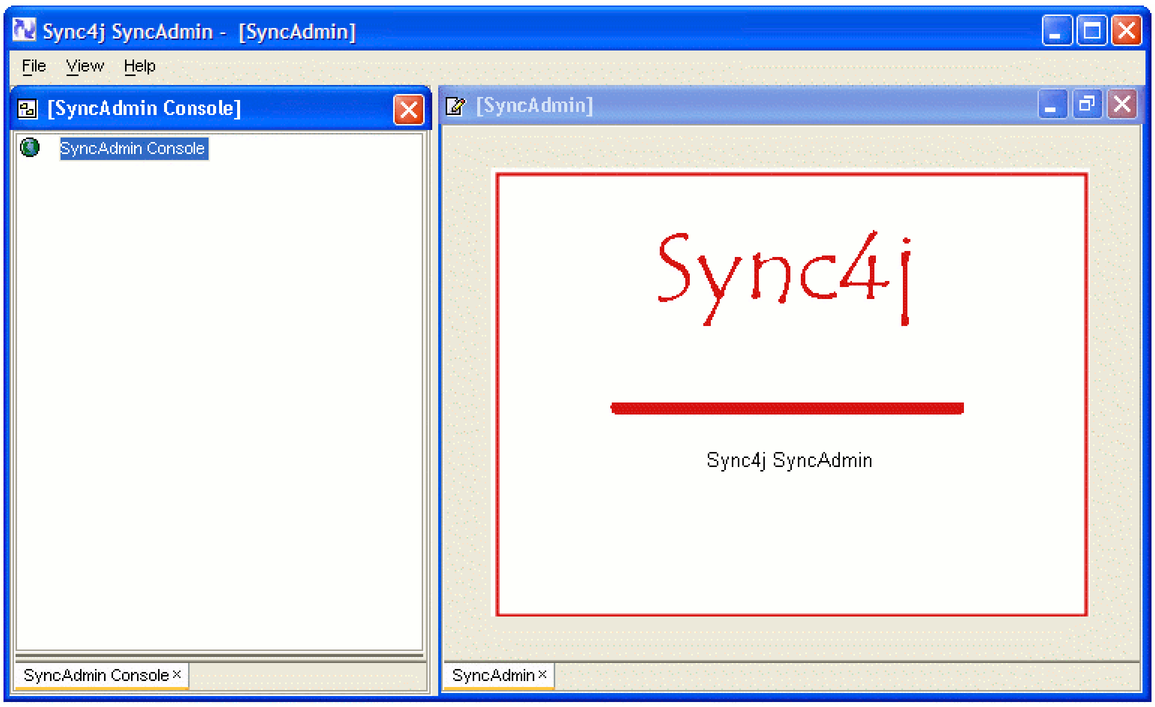
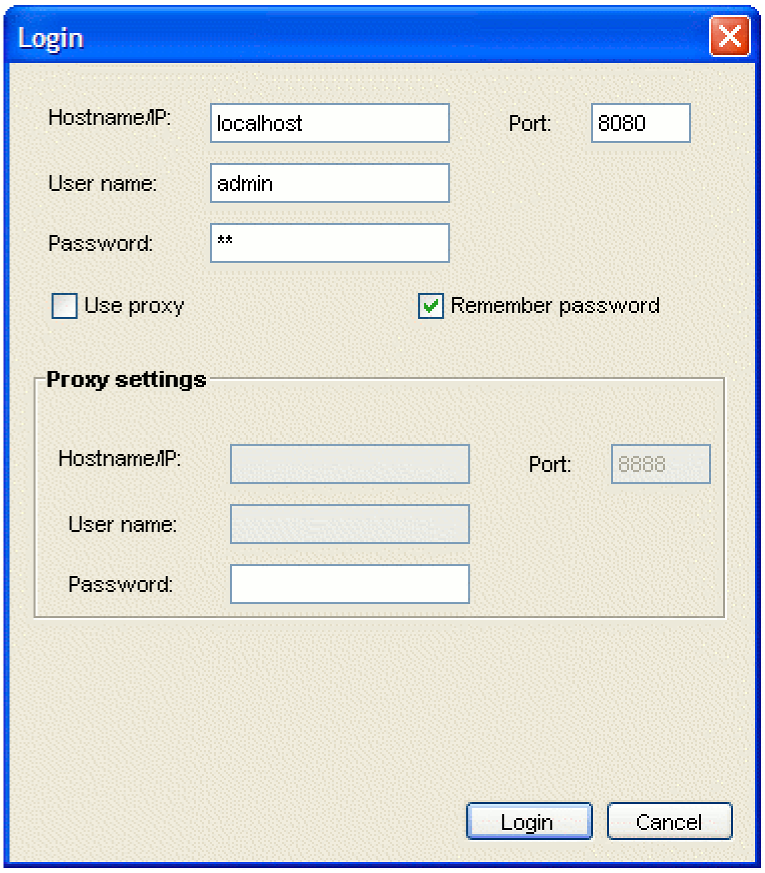


Figure 1 - Sync4j Server DS Administration tool

Before you can perform any administrative operation is to log on the server.

Select **'File'** -> **'Login'** and you will see the form of Figure 2.



Hostname/IP:	localhost	Port:	8080
User name:	admin		
Password:	**		
☐ Use proxy		☑ Remember password	

Proxy settings

Hostname/IP:		Port:	8888
User name:			
Password:			

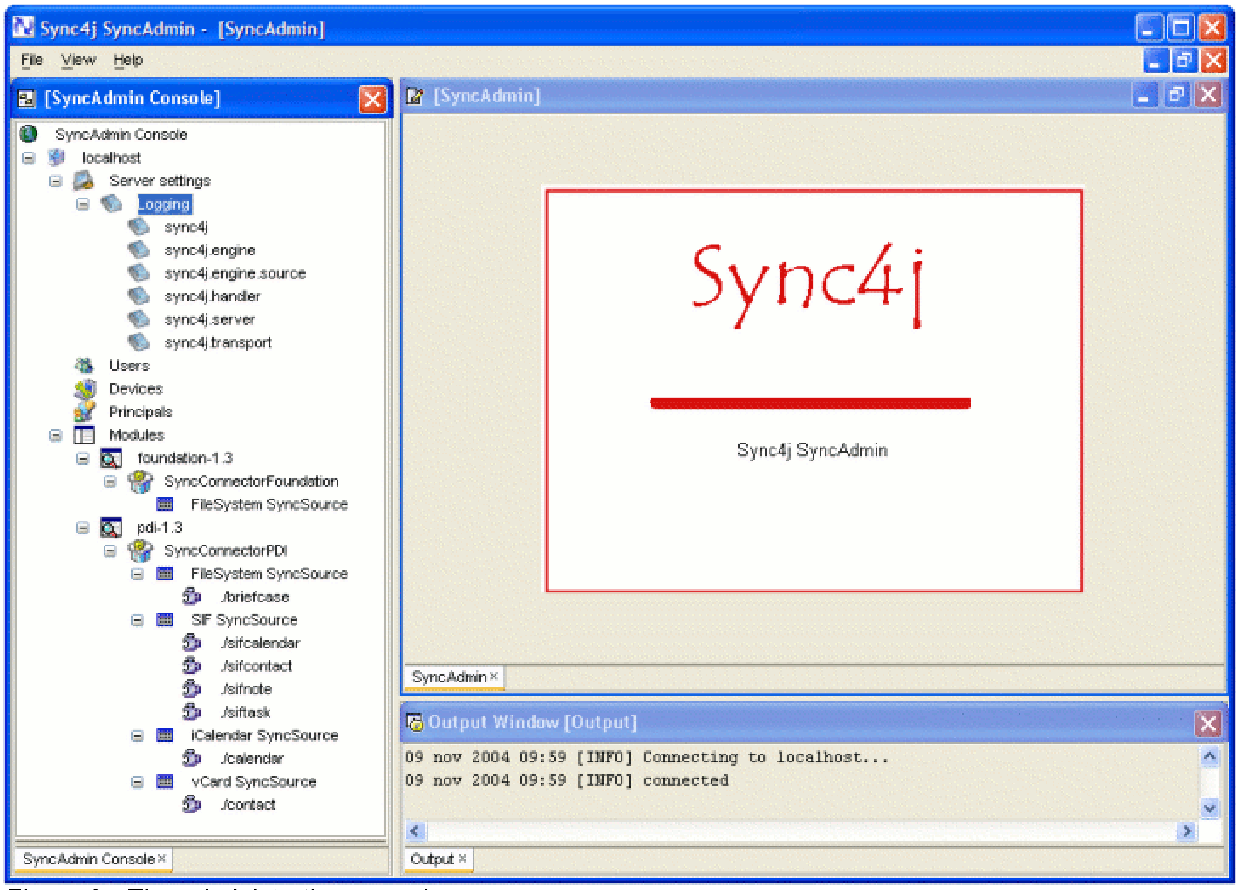
Figure 2 - Login Form

Fill the login form with the following information:

- Host Name: <yourserverhost> or <yourserverip>
  localhost would work if you installed the server on the same machine; 8080 is the standard port of the bundled installation with Tomcat. On some systems due a problem with Java in combination with DNS a connection may not be made if the hostname is used. For those systems it is advised to use the IP address, for instance, **127.0.0.1** instead of **localhost**.
- User: <username> the default username defined for administartion purposes is **admin**.
- Password: <password> the default username defined for administartion purposes is **sa**.

and press Login.

It is strongly advised to change the default password as soon as possible.



```
09 nov 2004 09:59 [INFO] Connecting to localhost...
09 nov 2004 09:59 [INFO] connected
```

Figure 3 - The administration console

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.

4.3. Server Settings

The first node under the hostname is the **'Server Setting'** which allows you to manipulate global server settings. In the current release one can only modify logging settings, but in future releases more parameters will be manageable from the Sync4j Admin.

4.3.1. Logging

Sync4j basic installation includes a single log file, called syncserver.log, stored under the <SYNCSERVER_HOME>/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 can manage the sync4j log node under Logging. By default the server comes with the following loggers:

1. sync4j

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

The first logger – **sync4j** – is the most generic one that is used for the server. The others are more for specific parts of the server to provide more logging for advanced usage. These may be considered as sub-loggers of the toplevel logger **sync4j**.

The following table shows the parameters that can be managed on a logger level basis.

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

All sub-loggers of the toplevel **sync4j** logger have an extra option in which you easier can select the logger configuration of the parent logger. The **'Same as sync4j'**-checkbox enables this and will then include all logging 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.

In case you have server problems or would like to debug the server or a syncsource, it is advised to switch all logging to the log level 'ALL', since that provides the most information of a synchronization. This log level one must also use if you would like to submit a log file for a problem you encounter and would like to ask questions about it on the sync4j mailinglists.

4.4. Users

The second node enables user management. The user management is applies a role based access to the Sync4j Server DS. The two roles recognized are:

- User
- Administrator

The users with role *User* are enabled to perform synchronizations with the server. The users with role *Administrator* may as well perform synchronizations as administrative tasks with Sync4j Server DS installation.

Adding a new user.

To add a new user, click on the Users node of the Sync4j Admin 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*.

4.5. 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 Sync4j Admin 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*.

4.6. 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, Sync4j Server DS is built around the concept of a *Principal*, which is a more generic concept than a user or a device. A principal is a tuple *(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 Sync4j Server DS 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*.

4.7. Modules, Sync Connectors, SyncSource Types and SyncSources

Sync4j Server DS uses many concepts to group together features and configuration settings.

Module: it is a package used to group together and distribute Sync4j Server DS extensions. A Module contains SyncConnectors, Synclets, configuration files, database scripts and so on.

SyncConnector: it is a server extension that integrates Sync4j Server DS 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 default Sync4 Sync4j Server DS - both bundled and individual - distribution comes with two 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 Sync4j Server DS 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 Sync4j Admin 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 Server DS component.

More details on how to develop a Sync4j module can be found in the Sync4j Server DS 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}-{version.number}.s4j
```

Where *modulename* is the name of the module and *version number* 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. Either by installing reinstalling the entire Sync4j Server DS or just installing the modules but for both ways the installation file install.properties must be configured with the proper list of the Sync4j modules.

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

```
modules-to-install=foundation-1.5.1,pdi-1.3.0
```

specifies the installation procedure to include in the Sync4j Server DS final ear (the Sync4j Server DS package for the J2EE application server) the modules Foundation 1.5.1 and PDI 1.3.0. If you want to add other modules, you must add this on this line (comma-sepreated list).

5.2.1. Full Installation

After properly setting modules-to-install in install.property, run the installation procedure.

1. Stop Sync4j Server DS (as described int section 4).

JBOSS

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.

Tomcat 5.0.x

> shell) **cd <CATALINA_HOME>**
>
> (for windows)
> shell dir=<CATALINE_HOME>) **bin/shutdown.cmd**
>
> (for UNIX)
> shelldir=<CATALINE_HOME>) **bin/shutdown.sh**

2. Deploy/install the Sync4j Server DS. During the installation you will be asked if you want to create the database for Sync4j Server DS and some Sync4j Server DS modules.
   NOTE: You may respond yes 'y' to the questions to reinstall, but since you maybe already have valuable data in the DB it would be smart to choose no 'n' (not rebuilding the database). This will keep your existing users, mappings and last syncs information.
   NOTE2: In order to be able do interact with the Database it must be running. If you still use the database that comes default with a bundle you must start the server for this.

> shell) **cd <SYNCSERVER_HOME>**
>
> (for windows)
> shell dir=<SYNCSERVER_HOME>) **bin/install.cmd [jboss | tomcat]**
>
> (for UNIX)
> shell dir=<SYNCSERVER_HOME>) **bin/install.sh [jboss | tomcat]**

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. However, if it is a module that is installed for the first time you must answer 'y', yes.

3. The following commands would start the Sync4j Server DS.

> (for windows)
> shell dir=<SYNCSERVER_HOME>) **bin/start.cmd**
>
> (for UNIX)
> shell dir=<SYNCSERVER_HOME>) **bin/start.sh**

4. Point the browser to new http://<server>:<port>/sync4j to check that Sync4j Server DS is properly installed (you should get the welcome page).

5.2.2. Modules-only Installation

If you have performed only changes to the modules properties of the Sync4j Server DS you can do also only a module installation.

1. Stop Sync4j Server DS (as described in section 4).

JBOSS

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.

Tomcat 5.0.x

shell) **cd <CATALINA_HOME>**

(for windows)
shell dir=<CATALINA_HOME>) **bin/shutdown.cmd**

(for UNIX)
shell dir=<CATALINA_HOME>) **bin/shutdown.sh**

2. Deploy/install the Sync4j Server DS. During the installation you will be asked if you want to create the database for Sync4j Server DS and some Sync4j Server DS modules.
   NOTE: You may respond yes 'y' to the questions to reinstall, but since you may have already valuable data in the DB it would be smart to choose no 'n' (not rebuilding the database). This will keep your existing users, mappings and last syncs information. However, if you install a module for the first time you must answer 'y', yes, at least for that module. If you do not do that, the server does not know abut that module.
   NOTE2: In order to be able do interact with the Database it must be running. If you still use the database that comes default with a bundle you must start the server for this.

   shell) **cd <SYNCSERVER_HOME>**

   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/install-modules.cmd [jboss | tomcat]**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/install-modules.sh [jboss | tomcat]**

3. The following commands would start the Sync4j Server DS.
4. Point the browser to new http://<server>:<port>/sync4j to check that Sync4j Server DS is properly installed (you should get the welcome page).

   shell) **cd <SYNCSERVER_HOME>**

   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/start.cmd**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/start.sh**

6. Sync4j DB Connector Registration

The connector is distributed as a default Sync4j module (see the Developer Guide for details) already available in the Sync4j Server DS package.

Opening the Sync4j Admin you should have the new connector displayed. Expanding the children nodes you should have the tree shown in Figure 4.

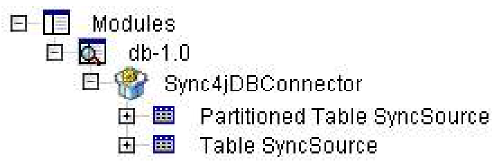


Figure 4 - Sync4j DB Connector registration

6.1. TableSyncSource

This syncsource is used to synchronize a single table.
This scenario is shown in Figure 5, where “luid/guid” are the primary key columns, “p” is the principal id, “t” is the timestamp of the last modification and “m” is the last modification flag.

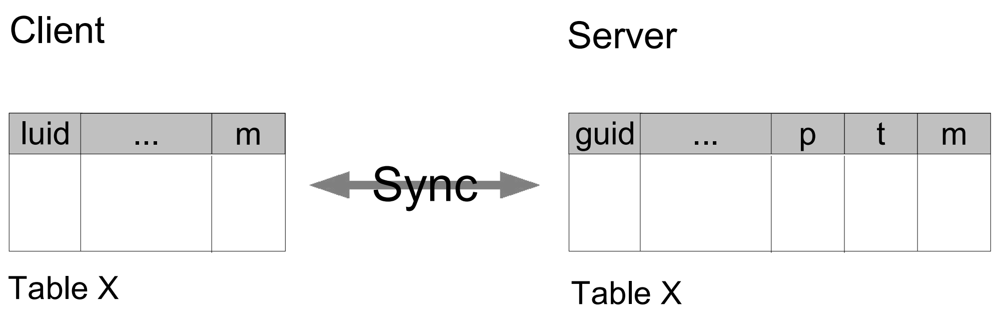


Figure 5 - Single table scenario

In this case, the entire table must be synchronized between client and server. Data are filtered based on the principal so that each principal will synchronize only the data belonging to it.

Server Database

The table that you want to synchronize must have:

- a field containing the primary key of the table
- a field containing the last modification timestamp
- a field containing the last modification type (One between “S”, “N”, “U”, “D”)
- a field containing the principal id of the principal associated with a record. This field is optional. If thereis'nt it, the table data will not be filtered based on the principal.

The table name and the previous fields are configurable.

Client Database

The table that you want to synchronize must have:

- a field containing the primary key of the table
- a field containing the last modification type (One between “S”, “N”, “U”, “D”)

The table name and the previous fields are configurable into the management node of the syncsource.

Figure 6 shows the system architecture.

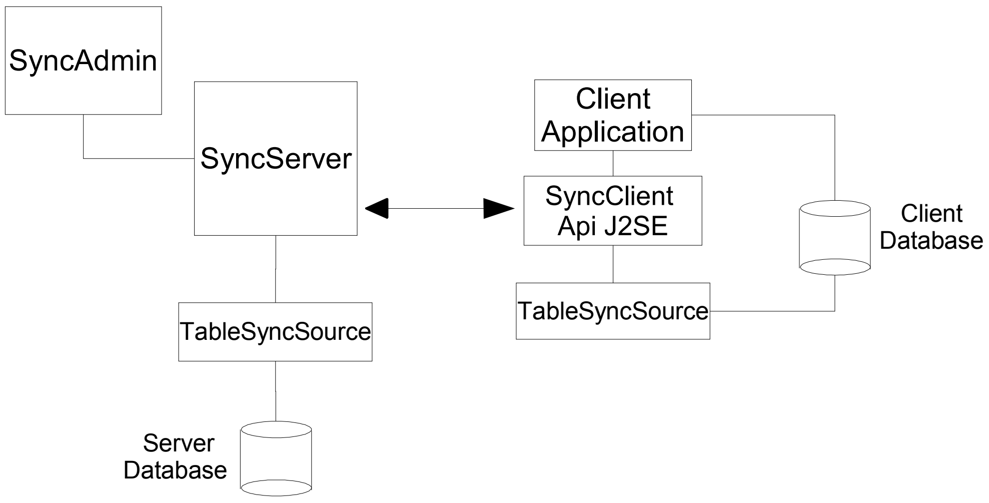


Figure 6 - Sync4j DB Connector high level architecture

The main components are:

1. a server side TableSyncSource to access data on the server database
2. a client side TableSyncSource to access data on the client database

6.1.1. Server side configuration

Server side TableSyncSource is the sync source type that handles the interaction with the underlying database on the server. It defines the following properties.

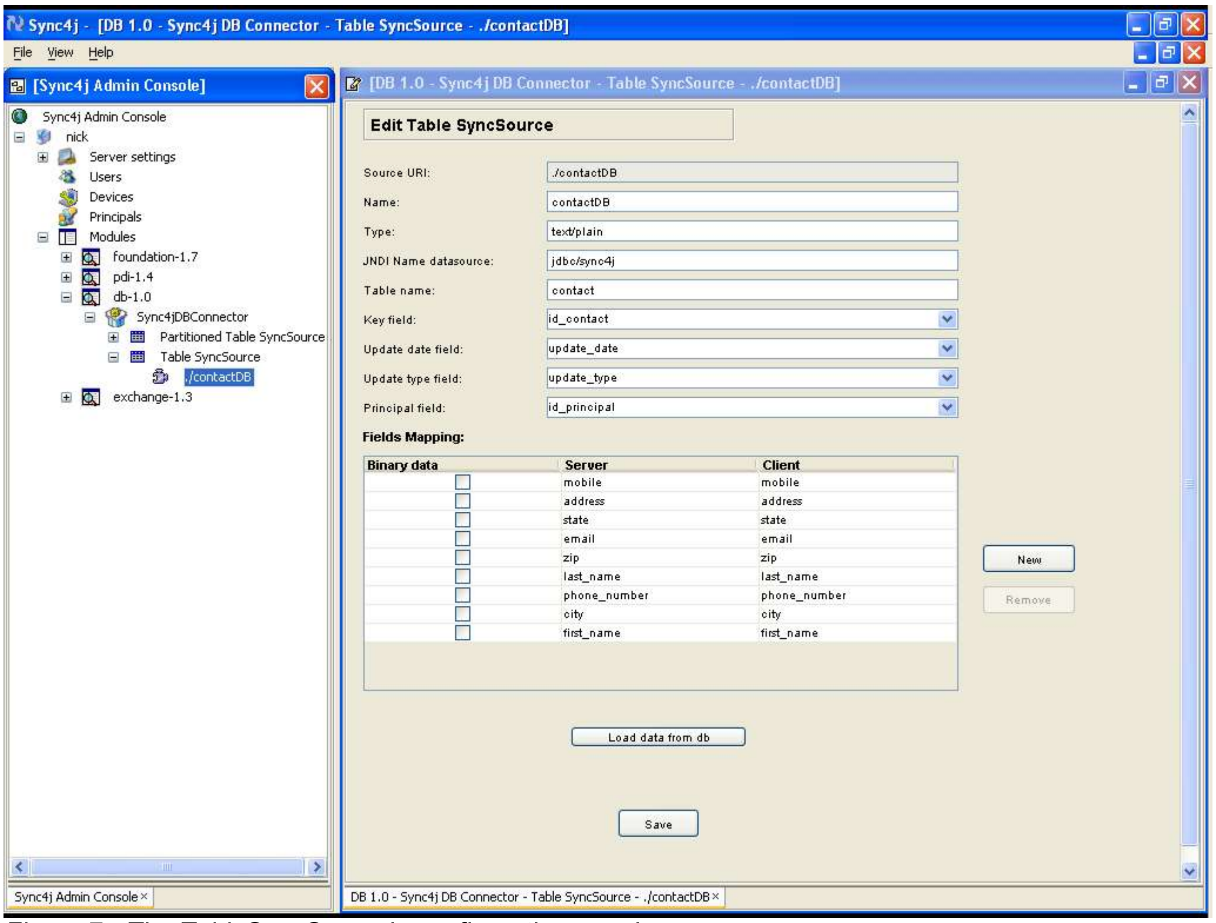
Property	***Description***
Source URI	The sync source URI
Name	The sync source name.
Type	The Data content mime type.
Jndi name	The Jndi name of the datasource to access to the database
Key field	The primary key of the table.
Update date field	The field containing the last modification timestamp.
Update type field	The field containing the last modification type (one beetwen 'S', 'N', 'U', 'D').
Principal field	The field containing principal id. If this property is empty, the table data will not be filtered based on the principal.
Field mapping	The mapping between server fields and client fields.
Binary fields	The list of the fields containing binary data.

Configuration examples

Property	***Value***
Source URI	./contactDB
Name	contactDB
Type	text/plain
Jndi name	jdbc/sync4j

Property	**Value**	
Table name	contact	
Key field	id_contact	
Update date field	update_date	
Update type field	update_type	
Principal field	id_principal	
Fields Mapping	first_name	first_name
	last_name	last_name
	phone_number	phone_number
	mobile	mobile
	email	email
	address	address
	city	city
	zip	zip
	state	state
Binary fields	No fields with binary data	

Configuration panel



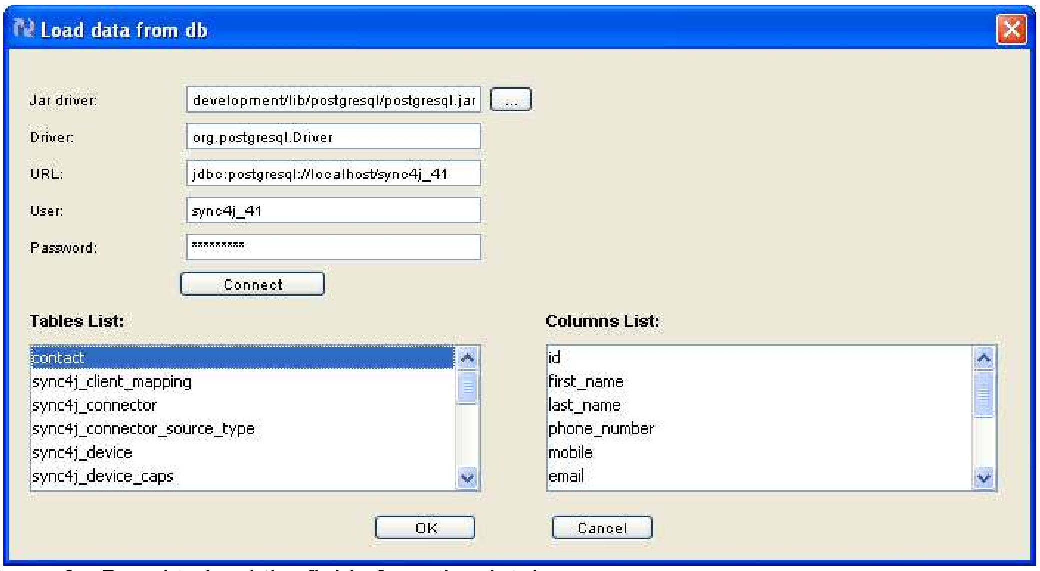
Source URI:	./contactDB
Name:	contactDB
Type:	text/plain
JNDI Name datasource:	jdbc/sync4j
Table name:	contact
Key field:	id_contact
Update date field:	update_date
Update type field:	update_type
Principal field:	id_principal

Fields Mapping:

Binary data	Server	Client
☐	mobile	mobile
☐	address	address
☐	state	state
☐	email	email
☐	zip	zip
☐	last_name	last_name
☐	phone_number	phone_number
☐	city	city
☐	first_name	first_name

Figure 7 - The TableSyncSource's configuration panel

The module provides also a Sync4j Admin configuration panel to create/update a TableSyncSource (see Figure 7).



Load data from db

Jar driver: development/lib/postgresql/postgresql.jar

Driver: org.postgresql.Driver

URL: jdbc:postgresql://localhost/sync4j_41

User: sync4j_41

Password: *********

Tables List:

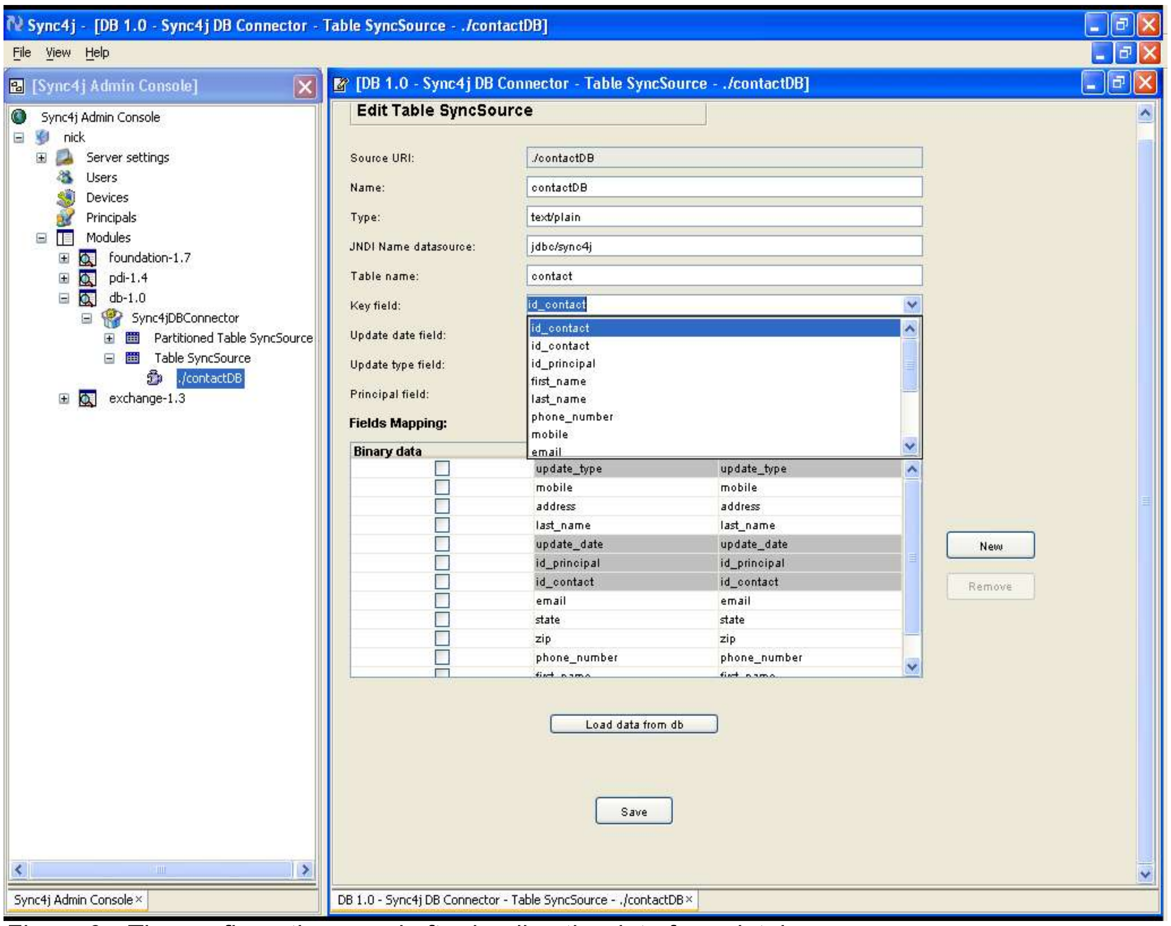
- contact
- sync4j_client_mapping
- sync4j_connector
- sync4j_connector_source_type
- sync4j_device
- sync4j_device_caps

Columns List:

- id
- first_name
- last_name
- phone_number
- mobile
- email

Figure 8 - Panel to load the fields from the database

To simplify the configuration, the panel provides a way to load the fields directly from the database. Pressing the button “Load data from db”, the Sync4j Admin shows the panel of Figure 8.
Using this panel, you can connect directly with the database and select the table you want to synchronize. You must select the jar file with the database driver and insert the driver class, the url, the user and the password to use to get the connection. Pressing the button “Connect”, in the left list, the Sync4j Admin shows all tables accessible with the given credential; selecting a table, in the right list are shown all his columns. Pressing the button “Ok” the columns list is loaded in the configuration panel as shown in Figure 9.



Source URI: ./contactDB
Name: contactDB
Type: text/plain
JNDI Name datasource: jdbc/sync4j
Table name: contact
Key field: id_contact
Update date field:
Update type field:
Principal field:

id_contact
id_contact
id_principal
first_name
last_name
phone_number
mobile
email

Fields Mapping:

Binary data		
☐	update_type	update_type
☐	mobile	mobile
☐	address	address
☐	last_name	last_name
☐	update_date	update_date
☐	id_principal	id_principal
☐	id_contact	id_contact
☐	email	email
☐	state	state
☐	zip	zip
☐	phone_number	phone_number
☐	first_name	first_name

Figure 9 - The configuration panel after loading the data from database

As shown in this figure, all combo box now contain all fields of the selected table. All fields are loaded also in the “Fields Mapping”.
The fields selected in the combo box are marked in the mapping table with a different color to indicate that these fields are already used.
With the “Binary data” checkbox in the mapping table, you can mark a field as binary (as a contact's photo). The binary fields are encoded before send them between server and client.

The button “New” and “Remove” permit to add/remove a mapping.

6.1.2. Client side

Sync4j DB Connector provides also the client side syncsource to use with the SyncClient API J2SE. The syncsource is implemented in *sync4j.db.client.engine.source.TableSyncSource.*

To use this syncsource, the underlying table must have a field to handle the state of each record. This field is configurable into the management node of the syncsource.

A client side TableSyncSource is configured with the properties listed in the table below.

Property	***Description***
tableName	The table name.
keyField	The primary key field.

Property	Description
updateTypeField	The field with the last update type.
fieldsList	The list of the fields (as comma separeted list).
binaryFields	The list of the binary fields (as comma separeted list).

Configuration examples

This is an example of a complete configuration file with also the standard syncsource properties:

```
#
#Fri Feb 18 18:22:15 CET 2005
syncModes=none, slow, two-way, one-way, refresh
name=dbcontact
sourceURI=./dbcontact
sourceClass=sync4j.db.client.engine.source.TableSyncSource
keyField=id
fieldsList=first_name,last_name,phone_number,mobile,email,address,city,zip,state
updateTypeField=update_type
type=text/clear
sync=two-way
tableName=contact
last=1108747334979
description=dbcontact
```

Database Access

An important issue to consider is how to access the database. DBMS capable of running on a limited device such as a PDA usually have restrictions and limitations. For example, they may not be available via JNDI or not allowing more than one connection open at a time. Plus, the overhead of getting a connection could be not negligible. Therefore the TableSyncSource class can use an external *connection factory*, which in turn provides the connection to the database. The connection factory is represented by the interface *sync4j.db.client.ConnectionFactory* and can be provided by the client application. In this way, the sync source will be able to use an already existing database connection.

sync4j.db.client.ConnectionFactory defines the following methods:

Method	Description
getConnection(): Connection	Returns a database connection
closeConnection(con: Connection): void	Closes the database connection

The db module reads the dm node *db/connectionfactory* to obtain an instance of a ConnectionFactory. The properties of this node are used to create a ConnectionFactory; the property *className* contains the class name to use.

This is an example of *db/connectionfactory* management node:

```
className=sync4j.db.client.ConnectionFactoryImpl
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://localhost/testdbcontact
user=testdbcontact
password=testdbcontact
```

All properties except *className* are used to configure the instance calling the respective methods set (set + property name capitalised).

The db module provides also a simple ConnectionFactory *sync4j.db.client.ConnectionFactoryImpl* that opens always a new connection to the database for all calls to the getConnection() method. The method closeConnection(), instead, always closes the connection.
These are all properties available and used by this implementation:

Name	Description
driver	The driver to access to database

Name	Description
url	The url to access to database
user	The user to access to database
password	The password to access to database

In Appendix D is provided an example of ConnectionFactory to have only one connection to the database.

6.2. PartitionedTableSyncSource

This syncsource is used to synchronize principal related data based on a data table and a partitioning table-
In this scenario (shown in Figure 10), the association between a row of a table on the server and the owner of such row is stored in a principal table, called “partitioning table”. In this case, on the client we still have one table, while on the server we have the partitioning table and the data table. The tricky issue that must be addressed here is that both the modifications on the data table and on the partitioning table must be reflected on the client.
For example, suppose that the data table contains customers data and the partitioning table contains which sales agent deals with a particular customer (thus it contains the agent-customer 1:N relationship). A row on the client must be deleted either if:

1. a customer is deleted;
2. a customer is associated to another principal.

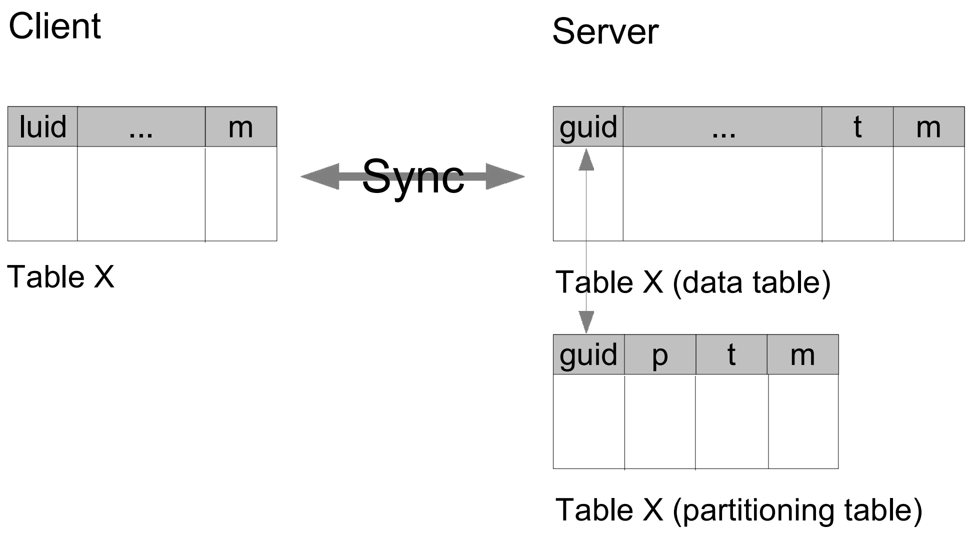


Figure 10 - Partitioned table scenario

Server Database

The partitioning table must have:
- a field containing the principal id of the principal
- a field “guid” used to link the principal with a record of the data table

- a field containing the last modification timestamp
- a field containing the last modification type (One between “S”, “N”, “U”, “D”)
- a field containing the principal id of the principal associated with a record. This field is optional. If thereis'nt it, the table data will not be filtered based on the principal.

The data table must have:
- a field as primary key
- a field “guid” (it can be the previous field)
- a field containing the last modification timestamp
- a field containing the last modification type (One between “S”, “N”, “U”, “D”)

The tables name and the previous fields are all configurable.

Client Database

The table that you want to synchronize must have:
- a field containing the primary key of the table
- a field containing the last modification type (One between “S”, “N”, “U”, “D”)

The table name and the previous fields are configurable into the management node of the syncsource.

Figure 11 shows the system architecture.

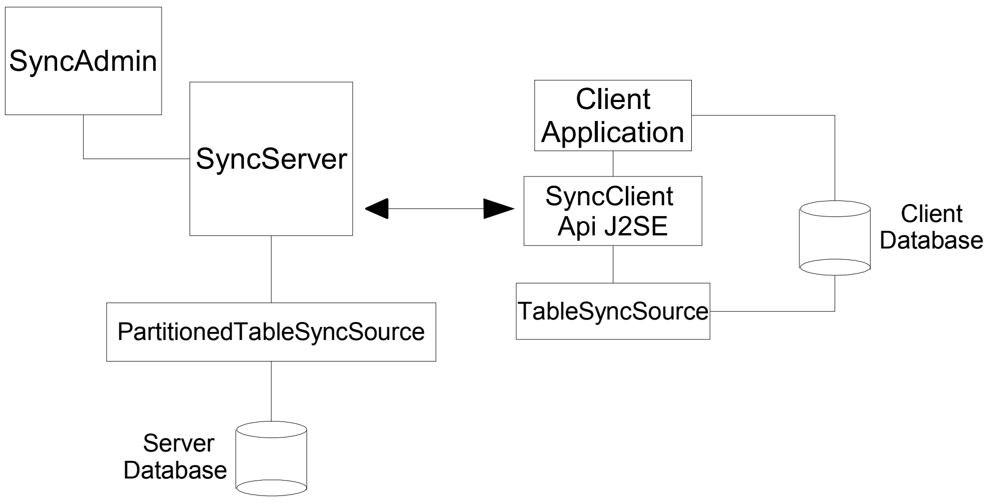


Figure 11 - Sync4j DB Connector high level architecture

The main components are:
3. a server side PartitionedTableSyncSource to access data on the server database
4. a client side TableSyncSource to access data on the client database

Note: the TableSyncSource to use on the client is the same used in the Table Synchronization scenario.

6.2.1. Server side configuration

Server side PartitionedTableSyncSource is the sync source type that handles the interaction with the underlying database on the server. It defines the following properties.

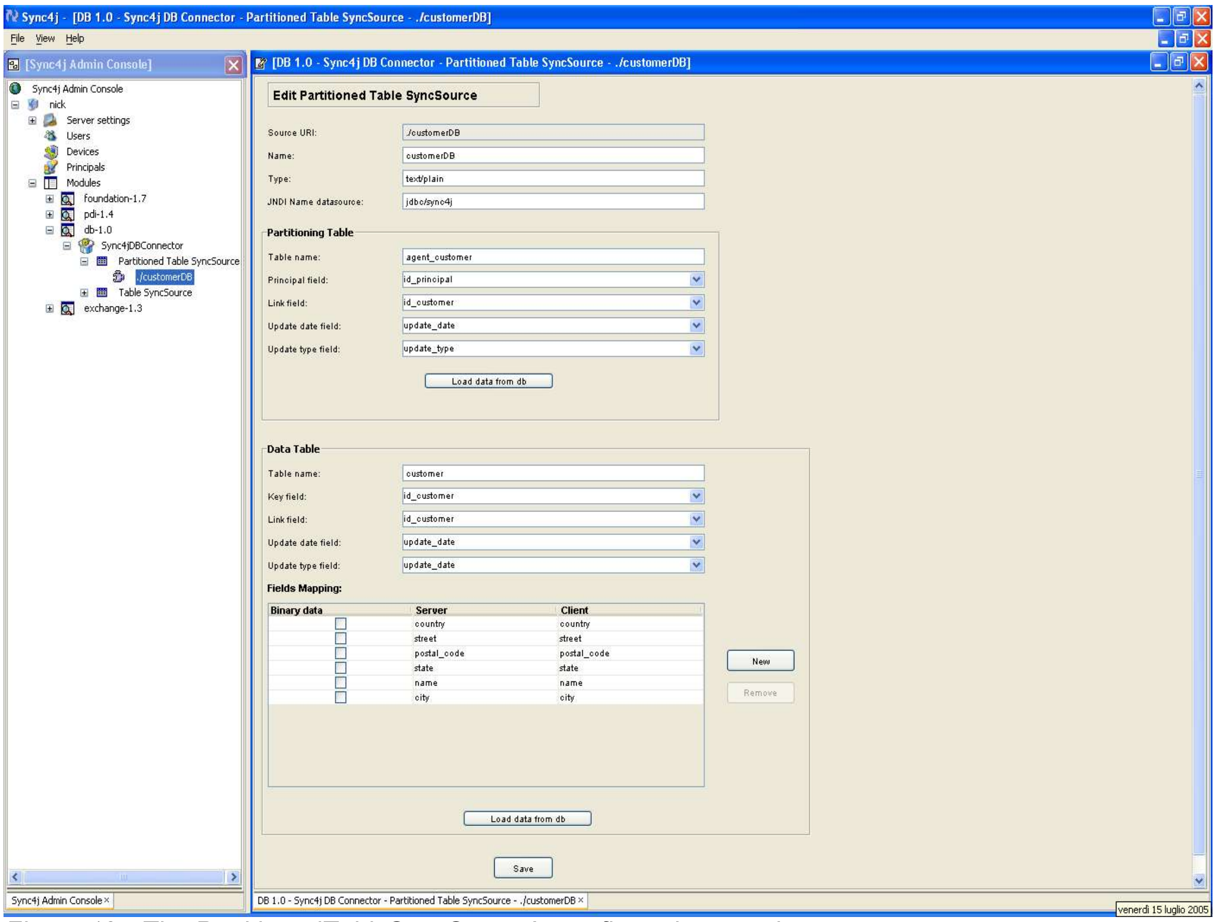
Property	***Description***
Source URI	The sync source URI
Name	The sync source name.
Type	The Data content mime type.
Jndi name	The Jndi name of the datasource to access to the database
Partitioning table	
Table name	The table name
Principal field	The field containing the principal id
Link field	The field used to get the records from the data table (the “guid”)
Update date field	The field containing the last modification timestamp.
Update type field	The field containing the last modification type (one beetwen 'S', 'N', 'U', 'D').
Data table	
Table name	The table name
Key field	The primary key
Link field	The “guid”. It can be the key field (the primary key)
Update date field	The field containing the last modification timestamp.
Update type field	The field containing the last modification type (one beetwen 'S', 'N', 'U', 'D').
Field mapping	The mapping between server fields and client fields.
Binary fields	The list of the fields containing binary data.

Configuration examples

Property	***Value***	
Source URI	./customerDB	
Name	customerDB	
Type	text/plain	
Jndi name	jdbc/sync4j	
Partitioning table		
Table name	agent_customer	
Principal field	id_principal	
Link field	id_customer	
Update date field	update_date	
Update type field	update_type	
Data table		
Table name	customer	
Key field	id_customer	
Link field	id_customer	
Update date field	update_date	
Update type field	update_type	
Field mapping	country	country
	street	street
	postal_code	postal_code
	state	state
	name	name

Property	Value	
	city	city
Binary fields	No fields with binary data	

Configuration panel



[DB 1.0 - Sync4j DB Connector - Partitioned Table SyncSource - ./customerDB]

Edit Partitioned Table SyncSource

Source URI: ./customerDB
Name: customerDB
Type: text/plain
JNDI Name datasource: jdbc/sync4j

Partitioning Table

Table name: agent_customer
Principal field: id_principal
Link field: id_customer
Update date field: update_date
Update type field: update_type

Load data from db

Data Table

Table name: customer
Key field: id_customer
Link field: id_customer
Update date field: update_date
Update type field: update_date

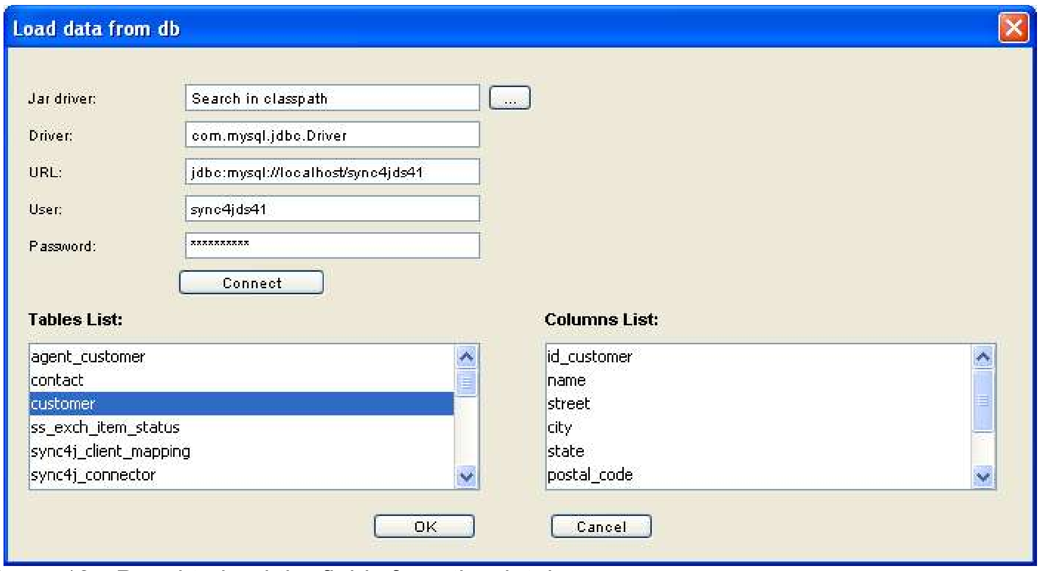
Fields Mapping:

Binary data	Server	Client
☐	country	country
☐	street	street
☐	postal_code	postal_code
☐	state	state
☐	name	name
☐	city	city

New
Remove
Load data from db
Save

Figure 12 - The PartitionedTableSyncSource's configuration panel

The module provides also a Sync4j Admin configuration panel to create/update a PartitionedTableSyncSource (see Figure 12).



Jar driver: Search in classpath
Driver: com.mysql.jdbc.Driver
URL: jdbc:mysql://localhost/sync4jds41
User: sync4jds41
Password: **********

Tables List:

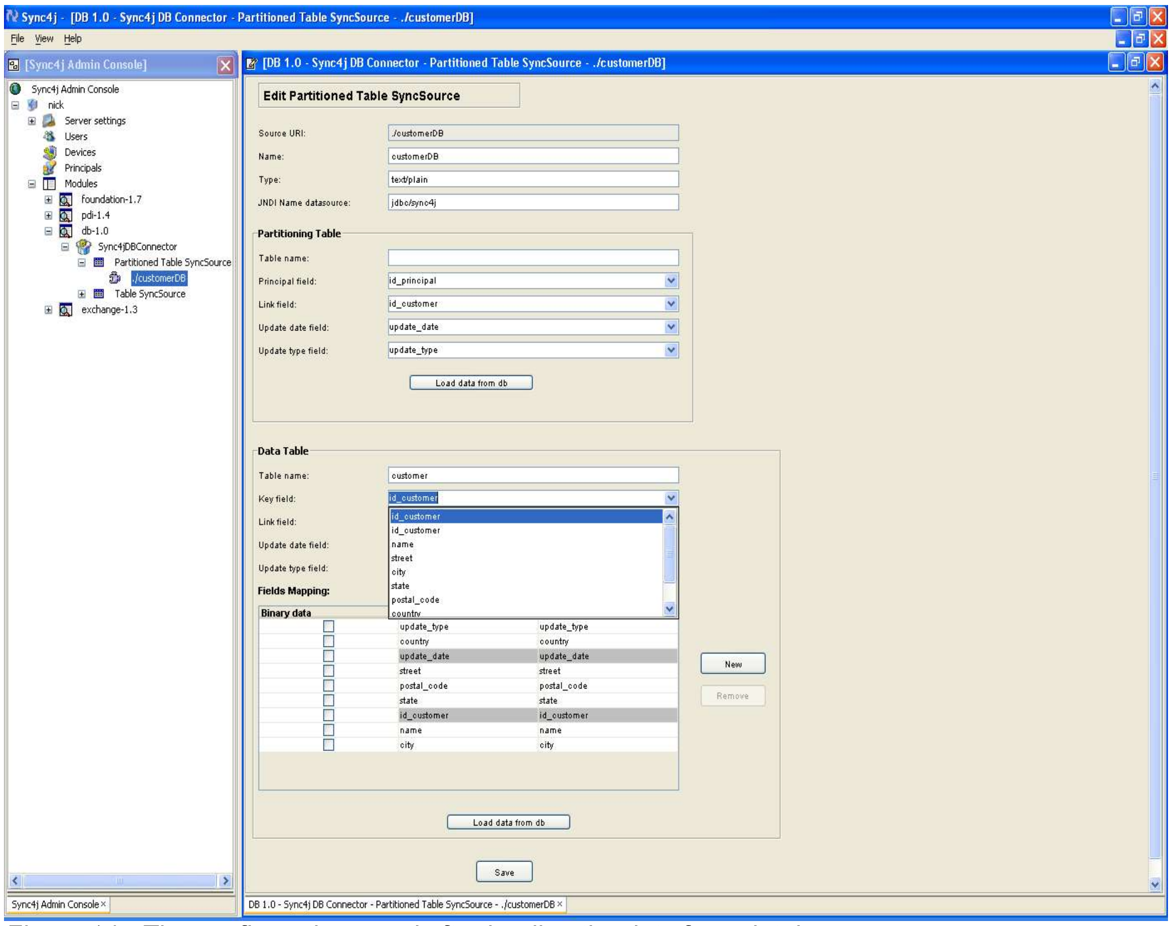
agent_customer
contact
customer
ss_exch_item_status
sync4j_client_mapping
sync4j_connector

Columns List:

id_customer
name
street
city
state
postal_code

Figure 13 - Panel to load the fields from the database

To simplify the configuration, the panel provides a way to load the fields directly from the database. Pressing the button “Load data from db”, the Sync4j Admin shows the panel of Figure 13.
Using this panel, you can connect directly with the database and select the table you want to synchronize. You must select the jar file with the database driver and insert the driver class, the url, the user and the password to use to get the connection. Pressing the button “Connect”, in the left list, the Sync4j Admin shows all tables accessible with the given credential; selecting a table, in the right list are shown all his columns. Pressing the button “Ok” the columns list is loaded in the configuration panel as shown in Figure 14.


Sync4j - [DB 1.0 - Sync4j DB Connector - Partitioned Table SyncSource - ./customerDB]
File View Help
[Sync4j Admin Console]
Sync4j Admin Console
nick
Server settings
Users
Devices
Principals
Modules
foundation-1.7
pdi-1.4
db-1.0
Sync4jDBConnector
Partitioned Table SyncSource
./customerDB
Table SyncSource
exchange-1.3
[DB 1.0 - Sync4j DB Connector - Partitioned Table SyncSource - ./customerDB]
Load data from db
New
Remove
Save
Sync4j Admin Console
DB 1.0 - Sync4j DB Connector - Partitioned Table SyncSource - ./customerDB

Edit Partitioned Table SyncSource

Source URI:	./customerDB
Name:	customerDB
Type:	text/plain
JNDI Name datasource:	jdbc/sync4j

Partitioning Table

Table name:	
Principal field:	id_principal
Link field:	id_customer
Update date field:	update_date
Update type field:	update_type

Data Table

Table name:	customer
Key field:	id_customer
Link field:	
Update date field:	
Update type field:	

id_customer
id_customer
name
street
city
state
postal_code
country

Fields Mapping:

Binary data		
☐	update_type	update_type
☐	country	country
☐	update_date	update_date
☐	street	street
☐	postal_code	postal_code
☐	state	state
☐	id_customer	id_customer
☐	name	name
☐	city	city

Figure 14 - The configuration panel after loading the data from database

As shown in this figure, all combo box now contain all fields of the selected table. All fields are loaded also in the "Fields Mapping".
The fields selected in the combo box are marked in the mapping table with a different color to indicate that these fields are already used.
With the "Binary data" checkbox in the mapping table, you can mark a field as binary (as a contact's photo). The binary fields are encoded before send them between server and client.

The button "New" and "Remove" permit to add/remove a mapping.

6.2.2. Client Side

On the client you have to use the TableSyncSource like in Table Synchronization Scenario.

7. Configuring Sync4j Server DS

This section provides Sync4j Server DS specific configuration asministration tasks. These are tasks that cannot be done with the AdminTool, since they are to be considered core tasks before any comunication with the Sync4j Server DS can be done.

There are two configuration techniques used by Sync4j Server DS : 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).

7.1. Changing the IP/hostname

A default installation provides access to the Sync4j Server DS only for the localhost, but in most situations it is necessary to access the Sync4j SynServer from a remote system. In that case you can
- or edit the configuration file specifying the servername
- or edit the install properties and do a re-installation of the Sync4j Server DS (this method would keep the new server URL also for following server re-installations).

Open in your favorite editor the file named <SYNCSERVER_HOME>/config/Sync4j.xml and change the property server.uri parameter to the your prefered ip/hostname

Default value is:

```
<void property="serverURI">
 <string></string>
</void>
```

A changed value that uses the hostname 'sync.sync4j.org' looks like:

```
<void property="serverURI">
 <string>http://sync.sync4j.org:8080/sync4j/sync</string>
</void>
```

Restart the server as provided in section 4 via a stop and start procedure.

Option 2) Using install.properties

Using the install.properties requires a reinstallation of the server and the steps to perform this is provided below.

1. Customize *<SYNCSERVER_HOME>/install.properties* to reflect your system the new server URL by changing the server-name property to your requirements.
2. Stop Sync4j Server DS as described int section 4.
3. Deploy/install the Sync4j Server DS. During the installation you will be asked if you want to create the database for Sync4j Server DS and some Sync4j Server DS modules.
   NOTE: You may respond yes 'y' to the questions to reinstall, but since you maybe already have valuable data in the DB it would be smart to choose no 'n'.
   NOTE2: In order to be able do interact with the Database it must be running. If you still use the database that comes default with a bundle you must start the server for this.

   shell) **cd <SYNCSERVER_HOME>**

   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/install.cmd [jboss | tomcat]**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/install.sh [jboss | tomcat]**

4. Start Sync4j Server DS as described in section 4.
5. Point the browser to new *http://<server>:<port>/sync4j* to check that Sync4j Server DS is properly installed (you should get the welcome page).

7.2. Changing the HTTP server port

In order to change the port to which the Sync4j Server DS will listen to you need to make first the similar modifcations as provided in the section, 'Changing the IP/hostname'. Only in this case you do not change the servername but the port value of the URL.Those changes are required but only for the internal use of the Sync4j Server DS . Again they are not sufficient for running the server on a different port.

To change the HTTP port the server listens to, follow the steps below. This information one can also find in the application server specific documentation.

Tomcat

1. Edit <SYNC4J_HOME>/tools/tomcat/conf/server.xml and look for the following comment:

Default is:

```
    <!-- Define a non-SSSL Coyote HTTP/1.1 Connector on port 8080 -->
    <Connector port="8080"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        debug="0" connectionTimeout="20000"
        disableUploadTimeout="true" />
```

A changed value that uses the port 80 looks like:

```
<!-- Define a non-SSSL Coyote HTTP/1.1 Connector on port 8080 -->
   <Connector port="80"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" redirectPort="8443" acceptCount="100"
        debug="0" connectionTimeout="20000"
        disableUploadTimeout="true" />
```

2. Restart Sync4j as described in section 4.

Jboss

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. Restart Sync4j as described in section 4.

7.3. Changing the database

The bundled distribution comes with the HSQLDB included in order to get you going. However, this is not an optimal performing DB in production environments. Any JDBC compliant database can be choosen and we advise to use the one with which you are most familiar with our use in the production enviroronment.

option 1) change the application server specific database configuration

After Sync4j Server DS is installed, the database access configuration is delegated to the application server. Sync4j Server DS 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.

option 2) edit the install properties and do a re-installation of the Sync4j Server DS

If you are not familiar with the application server specific DB configuration you can also make changes in the install.properties file. This has the advantage that the new DB configuration also for following server re-installations

1. Customize your <SYNCSERVER_HOME>/install.properties and set the dbms to your preference.

The default for a bundled version looks like:

```
# The DBMS name. One of:
#   - ansisql99
#   - hypersonic
#   - mysql
#   - oracle
#   - postgresql
#   - sybase
#   - sqlserver
#
dbms=hypersonic
```

If you database is not in the list, you could try to use the ***'ansisql99'*** that will work on most ANSI compliant databases.

2. following properties must be edited (note these are below many examples DB configurations):

```
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, these would look like:

```
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
```

NOTE: the jdbc.classpath variable is used by the installer to perform some the required DB (tables and data) setup for the Sync4j Server DS . However, you must add this also to your classpath or insert it in the application server from where it reads all jarfiles. Otherwise, you server cannot access the database.

3. Deploy/install the Sync4j Server DS. During the installation you will be asked if you want to create the database for Sync4j Server DS and some Sync4j Server DS modules.
   NOTE: You must respond yes 'y' to the questions to reinstall, because you choose a new database and would normally not have the tables and initial data yet..
   NOTE2: In order to be able do interact with the Database it must be running. If you still use the database that comes default with a bundle you must start the server for this.

   shell) **cd <SYNCSERVER_HOME>**

   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/install.cmd [jboss | tomcat]**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/install.sh [jboss | tomcat]**

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

7.3.1. Logging Database Access

Sync4j Server DS 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 Server DS 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).

7.4. Device-Ids and Principal

By default the Sync4j Server DS installation in which your device-id gets rewritten into a generic device id '**syncml-phone**'. This enables all devices directly to work with the username/password conmbination, '**guest/guest**'. However, this may not be very useful in production environments where you want to use different username in association with the real device-id.

The Sync4j distribution comes with a non-rewriting PipelineManagerGeneric configuration in which no actions are done within the PipelineManager. For more details about the Pipelinemanager (it functions etc.) is refered to the Developers Guide. IIn that case you can
- or edit the *config/Sync4j.xml*
- or edit the *default/config/common/properties/config/Sync4j.xml.*
The last method would keep the change also for following server re-installations.

Option 1 *config/Sync4j.xml*

Open in your favorite editor the file named <SYNCSERVER_HOME>/config/Sync4j.xml change the property engine.pipeline.

Default value is:

```
<void property="pipelineManager">
 <string>sync4j/framework/engine/pipeline/PipelineManager.xml</string>
</void>
```

A changed value that uses the hostname 'sync.sync4j.org' looks like:

```
<void property="pipelineManager">
 <string>sync4j/framework/engine/pipeline/PipelineManagerGeneric.xml</string>
</void>
```

Restart the server as provided in section 4 via a stop and start procedure.

Option 2) *default/config/common/properties/config/Sync4j.xml*

To change the choosen Pipeline Manager and make it persistent after re-installations of the Sync4j Server DS you need to follow the steps below:

1. Customize *<SYNCSERVER_HOME>/default/config/common/properties/config/Sync4j.properties* to according to option 1.
   Open in your favorite editor the file named <SYNCSERVER_HOME>/config/Sync4j.properties and change the property server.uri parameter to the your prefered ip/hostname
2. Stop Sync4j Server DS as described int section 4.
3. Deploy/install the Sync4j Server DS. During the installation you will be asked if you want to create the database for Sync4j Server DS and some Sync4j Server DS modules.
   NOTE: Since you already have installed the Sync4j Server DS before it is advised to answer 'n', no, in order to maintain your existing data already in the databse. This configuration change does not require a reinstallation of the database data.
   NOTE2: In order to be able do interact with the Database it must be running. If you still use the database that comes default with a bundle you must start the server for this.

   shell) **cd <SYNCSERVER_HOME>**

   (for windows)
   shell dir=<SYNCSERVER_HOME>) **bin/install.cmd [jboss | tomcat]**

   (for UNIX)
   shell dir=<SYNCSERVER_HOME>) **bin/install.sh [jboss | tomcat]**

4. Start Sync4j Server DS as described in section 4.

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

7.4.1. JAAS

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, Sync4j Server DS is capable to rely on an external system to perform authentication and authorization, thus bypassing the issue. Sync4j Server DS bases its security services on the Java Authentication and Authorization Service architecture provided out of the box with the JDK 1.4.x.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.

7.4.2. 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 Server DS 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.

8. Synching with other devices and clients

This section provides additional information of synching other devices beyond the PIMWEB demo and the SyncClient PIM Demo. The Sync4j Outlook Client and and a mobile phone are presented. Even though, you could test with the Sync4j Outlook client with a default installation as long as the client is on the same machine as the server, it is adviced to first reconfigure your server for remote access. For the precise instructions are refered to section 6.1.

8.1. Synchronizing with Outlook

Now that you have the working server with a web interface and a client, it is time to add a new client to the mix. A common one is the SyncClient PIM Outlook. It allows you to synchronize your Outlook contacts and calendar information with the Sync4j Server DS . It works with Outlook 2002 and XP.

First of all, download the SyncClient PIM Outlook from www.sync4j.org. Please make sure to take the version included in the saem version as the server. Install the application and from the

Start menu -> All Programs -> Sync4j -> SyncClient Outlook -> SyncClient Outlook.

NOTE: you are testing a demo. Your contacts and calendar info in Outlook will be modified. Please make absolutely sure you have a backup copy of your Outlook contacts and calendar information before you proceed.

Now click on Synchronize. You will probably see a window from Outlook, asking for the permission of accessing its PIM data. Allow access for 10 minutes, so you do not have to worry about that.

Once the synchronization is completed, the SyncClient Outlook will show the message "Synchronization successfully completed!". If you look in your Outlook contacts, you should see our three demo contacts (Brown Vincent, Doe John, Smith Mike). If you look in the PIM Web demo, you should see your contacts. Now try to modify them on both sides and click on Synchronize again.

8.2. Syncing with a mobile phone.

In order to synchronize with a mobile fine your server must be accessible from the Internet. For access to your system from the internet is refered to your local network administrator. The network administrator maybe must open a firewall, provide a routable IP address or with a non-routable IP address perform Network Address Translations and port forwarding.

A good indicator is that if your IP addresses are:

10.0.0.0	-	10.255.255.255	(10/8 prefix)
172.16.0.0	-	172.31.255.255	(172.16/12 prefix)
192.168.0.0	-	192.168.255.255	(192.168/16 prefix)

you better ask your network administrator. These are IP addresses reserved for private networks and will never be used within the Internet.

8.2.1. Configuring your cellular phone

The SyncML client configuration is quite different from phone to phone. Overall, there are some standard parameters you have to set.

First of all, make sure you can access your WAP Home Page and navigate. If you cannot do it, the phone is probably not configured for data access. In this case, you cannot synchronize your PIM data and you are refered to you mobile service provider for more details of your phone model to access the Internet.

The standard parameters of a SyncML client are described in the following table.

Property	***Description***	***Value***
Server URI	The address of the Sync4j Server DS for the synchronization. It has different names on different devices. Some devices call it Synchronization URL, some just URL, some Home or Homepage. In same cases, the port is kept separated. So you have to put the server URL without the :<port> and put 8080 in a separate field called PORT. Lastly, in same rare cases, the URI needs to be put without the leading http://	http://<hostname_or_ip>:<port>/sync4j/sync
User	The user connecting to the Sync4j Server DS	guest
Password	The password for the user connecting to the Sync4j Server DS	guest
Contacts path	The path to perform the contacts synchronization in Sync4j Server DS . In some cases, the phone will add the leading ./ so there is no need to put it. Often, you will be able to choose on your phone if you want to synchronize contacts (address book) or not.	./contact
Calendar path	The path to perform the calendar synchronization in Sync4j Server DS . In some cases, the phone will add the leading ./ so there is no need to put it. Often, you will be able to choose on your phone if you want to synchronize contacts (address book) or not.	./calendar

The task of configuring a SyncML phone is time consuming and cumbersome. A complete list of all synchronization parameters for every phone tested with Sync4j will be soon available on the www.sync4j.org website. The next step will be to provide a way to configure phones over-the-air. Lastly, we are planning of adding a synchronization portal where you can synchronize your phone and other clients without the need of installing Sync4j Server DS .

Once the first synchronization is completed, if you look in your phone contacts list, you should see our three demo contacts (Brown Vincent, Doe John, Smith Mike). If you look in the PIM Web demo, you should see your contacts. Now try to modify them on both sides and synchronize again.

9. 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.

10. References and Resources

1. Sync4j Server DS Administration Guide, this guide provides more detail on how to adminstrate the server after installation.
2. Sync4j Server DS Developer Guide, this guide provides more detail on how to develop your own extenstions for the Sync4j Server DS .
3. Sync4j Server DS Architecture Document, this guide provides generic information for the complete Sync4j framework.
4. http://java.sun.com/j2se, provides information of the Java 2 Standard Edition
5. http://www.jboss.org, provides information of the Java 2 Standard Edition
6. http://java.sun.com/j2ee, provides information of the Java 2 Enterprise Edition
7. http://jakarta.apache.org/tomcat, provides information of the Java 2 Standard Edition

Appendix A: Databases

The Sync4j Server DS comes with a number of default databases to which one can synchronize. Some databases can be accessed with different URIs and depend on the format of the data that is synchronized. The following databases are set by default:

Calendar

This database is for synchronization of calendar data.

URI	***MIME-TYPE***	***clients***
./sifcalendar	sif/calendar	All sync4j developed clients
./calendar	text/x-vcalendar text/calendar	Most common known clients that are alread built in a mobile device.

Contacts

This database is for synchronization of calendar data.

URI	***MIME-TYPE***	***clients***
./sifcontact	sif/contact	All sync4j developed clients
./contact	text/x-vcard	Most common known clients that are alread built in a mobile device.

Notes

This database is for synchronization of text based notes.

URI	***MIME-TYPE***	***clients***
./notes	Text/plain	All sync4j developed clients

Tasks

This database is for synchronization of text based notes.

URI	***MIME-TYPE***	***clients***
./siftasks	sif/tasks	All sync4j developed clients

Briefcase

This database is for synchronization of text based notes.

URI	***MIME-TYPE***	***clients***
./briefcase	Application/*	All sync4j developed clients

Appendix B: 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 Server DS 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 Server DS module. Sync4j Server DS will respond to URLs starting with this context path.	/sync4j
dbms	Name of the database where Sync4j Server DS 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 Server DS modules to install. If one has already installed the module once, this will simply reinstall it again.	foundation-1.5.1,pdi-1.3
Modules-to-uninstall	Comma separated list of Sync4j Server DS modules to be un-installed. If one does not specify this any previous installed module will reside in the Sync4j Server DS.	
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

Appendix C: Sync4.properties

This is the main Sync4j Server DS 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. Sync4j Server DS 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	1.1
engine.manifacturer	The manufacturer included in the server capabilities.	Sync4j Server DS
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 Server DS
engine.devicetype	The device type included in the server capabilities	-
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

Appendix D: Example of ConnectionFactory

This is an example of a simple ConnectionFactory to use only one connection to access to your database.

```
/**
 * Copyright (C) 2003-2005 Funambol
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

package sync4j.db.client.test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


/**
 * This is an example of a simple ConnectionFactory that guarantees to use only
 * one connection to the database.
 * @author Stefano Nichele
 * @version $Id: SimpleConnectionFactoryImpl.java,v 1.1 2005/02/28 17:10:19 nichele Exp $
 */
public class SimpleConnectionFactoryImpl implements sync4j.db.client.ConnectionFactory {

    // --------------------------------------------------------------- Constants

    // ------------------------------------------------------------ Private data
    private String driver   = null;
    private String url      = null;
    private String user     = null;
    private String password = null;

    private static Connection conn   = null;

    /**
     * Open a new connection if there isn't one.
     * @return Connection
     * @throws SQLException
     */
    public Connection getConnection() throws SQLException {
        if (conn == null) {
            connect();
        }
        return conn;
    }

    /**
     * This implementation doesn't close the connection
     * @param conn Connection
     */
    public void closeConnection(Connection conn) {

    }

    /**
     * Closes the connection
     * @throws SQLException
     */
    public void closeConnection() throws SQLException {
        if (conn != null) {
            conn.close();
        }
    }

    // --------------------------------------------------------- Private Methods

    /**
     * Creates the connection to the database
     * @throws SQLException
     */
    private synchronized void connect() throws SQLException {

        if (conn != null) {
            return ;
        }
        try {
            Class.forName(driver);
            conn = DriverManager.getConnection(url, user, password);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Sets the driver
     * @param driver the driver to set
     */
    public void setDriver(String driver) {
        this.driver = driver;
    }

    /**
     * Sets the user
     * @param user the user to set
     */
    public void setUser(String user) {
        this.user = user;
    }

    /**
     * Sets the password
     * @param password the password to set
     */
    public void setPassword(String password) {
        this.password = password;
    }

    /**
     * Sets the url
     * @param url the url to set
     */
    public void setUrl(String url) {
        this.url = url;
    }

}
```