

Funambol DS Server

Administration Guide

Version 3.0
March 2006



Important Information

© Copyright Funambol, Inc. 2006. All rights reserved.

The information contained in this publication is subject to US and international copyright laws and treaties. Except as permitted by law, no part of this document may be reproduced or transmitted by any process or means without the prior written consent of Funambol, Inc.

Funambol, Inc. has taken care in preparation of this publication, but makes no expressed or implied warranty of any kind. Funambol, Inc. does not guarantee that any information contained herein is and will remain accurate or that use of the information will ensure correct and faultless operation of the relevant software, service or equipment.

Funambol, Inc., its agents and employees shall not be held liable for any loss or damage whatsoever resulting from reliance on the information contained herein.

Funambol and Sync4j are trademarks and registered trademarks of Funambol, Inc.

All other products mentioned herein may be trademarks of their respective companies.

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



Contents

Chapter 1	Installation and Configuration	1
	Introduction	2
	Prerequisites	2
	Obtaining the Software	3
	Installing the Funambol DS Server	4
	Installing the Server With Bundled Software	4
	Installing the Server Without Bundled Software	5
	Installing the Funambol Administration Tool	7
	Starting and Stopping the Funambol DS Server	8
	Starting the Server	8
	Stopping the Server	9
	Starting and Stopping the Administration Tool	10
	Configuring the Funambol DS Server	11
	Specifying the Public IP Address	11
	Changing the Device-ID	12
	Changing the HTTP Server Port	13
	Changing the Database	14
Chapter 2	Using the Funambol Administration Tool	17
	Introduction	18
	Main Window	18
	Server Login	19
	Server Settings	21
	Log Settings	26
	Managing Users	29
	Managing Devices	31
	Managing Principals	33
Chapter 3	Managing Modules	37
	Understanding Modules	38
	Standard Modules	39
	Managing SyncSources	40
	Installing Modules	41
	Installing a Module	41



Chapter 4	Using the Funambol DB Connector	43
	Overview	44
	Using the DB Connector	45
	Adding a Table SyncSource	47
	Configuring a Single Table Synchronization	49
	Configuring a Partitioned Table Synchronization	53
Appendix A	Supplemental Information	57
	Resources	58
	Related Documentation	58
	Other Resources	58
	Default Databases	59
	Calendar	59
	Contacts	59
	Notes	59
	Tasks	59
	Briefcase	60
	Install Properties	61
	Connection Factory Sample	62



Chapter 1 **Installation and Configuration**

The chapter provides details for obtaining, installing, configuring and running the Funambol DS Server and the Funambol Administration Tool.

Topics

- *Introduction, page 2*
- *Installing the Funambol DS Server, page 4*
- *Starting and Stopping the Funambol DS Server, page 8*
- *Configuring the Funambol DS Server, page 11*



Introduction

This document is intended for those who manage the Funambol DS Server, and includes the following information:

- Installing and configuring the Funambol DS Server
- Installing and using the Funambol Administration Tool
- Managing Funambol Modules
- Managing Funambol Connectors

Prerequisites

The following software should be installed before installing the Funambol DS Server:

- **Java Development Kit** – Java Development Kit 1.4.x or 1.5.x. Ensure that the `JAVA_HOME` environment variable points to the top directory of the JDK.
- **Application Server** – Application servers tested with the Funambol DS Server include Apache Tomcat version 5.0.x and JBoss versions 3.0.x, 3.2.x, and 4.0.x. Other servers may function but have not been tested. Ensure that the `J2EE_HOME` environment variable points to the top directory of the application server.
- **JDBC Compatible Database** – The use of a database by the Funambol DS Server requires that you have created a database with the appropriate permissions for connecting, creating, deleting, reading and writing tables. You also need the JDBC driver for the database, and the connection URL and login information.

NOTE: The Funambol DS Server software is available in a *bundled* package that includes all the software needed to get started with the server; however, a complete JDK is not provided, nor the JDBC driver for databases other than that for Hypersonic. For details, see “Obtaining the Software” on page 3.

For additional information on the above software, see “Resources” on page 58.



Obtaining the Software

You have the following options when you obtain the software.

Bundled Package or Individual Downloads

The Funambol DS Server software is available in a *bundled* package that includes prerequisite software (i.e., Apache Tomcat 5.0, JRE 1.5.0, and Hypersonic) along with the Funambol Administration Tool, a Java GUI client, and a PIM web demo. This package is available for the convenience of those who want to quickly implement an operational synchronization server.

NOTE: When you install the bundled package, all components are automatically installed (i.e., you cannot select a subset of components to install).

The Funambol DS Server, Funambol Administration Tool, Funambol Connectors, Client Plug-ins, and other Funambol components are also available as individual downloads.

Stable or Developer Release

The following releases are available from the Funambol product release website:

- **Stable release** – contains bundled and individual downloads that are tested for the feature set defined for that version.
- **Developer (beta) release** – contains bundled and individual downloads that include work in progress, and may have additional features. The developer release supports normal Funambol DS Server functionality; however, open issues may be present.

Once you have determined whether to download the bundle or individual components, and whether to use the stable or developer release, you can locate the appropriate download(s) for your operating system.



Installing the Funambol DS Server

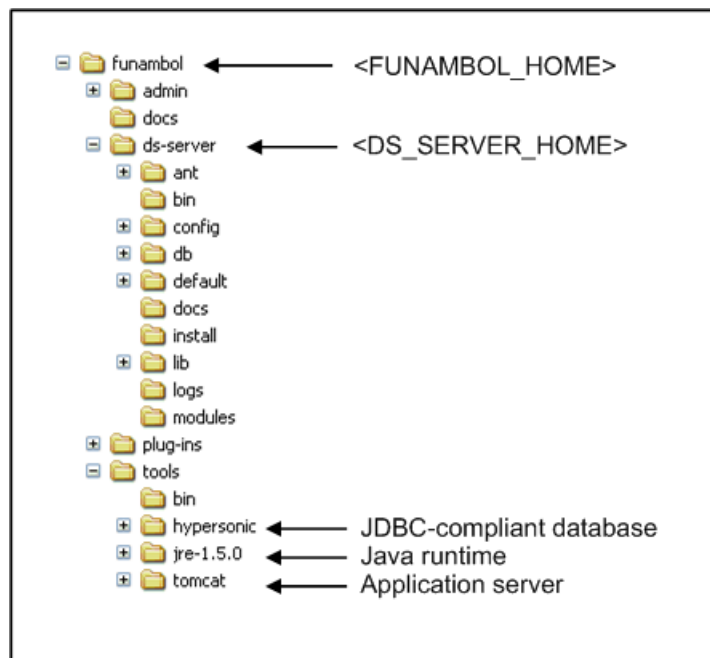
This section describes how to install the Funambol DS Server and the Funambol Administration Tool.

Installing the Server With Bundled Software

When you install bundled software, the Funambol Administration Tool and prerequisite software are automatically installed with the server.

Windows

1. Run the executable installation file.
2. Review the license agreement. To accept the terms, select the checkbox and click **Next**.
3. Accept the suggested **Destination Folder**, i.e., the \Funambol directory, that we will refer to as <FUNAMBOL_HOME> in this document. The installer will create subdirectories for bundle components under the top-level directory, including a subdirectory for the Funambol DS Server that we will refer to as <DS_SERVER_HOME>, as shown below:



Note the prerequisite application server, Java runtime, and database will be installed in the <FUNAMBOL_HOME>\tools directory.

4. When prompted for a **Program Group**, specify Funambol (if not already specified).
5. At the completion of the installation, you are given the option to view the readme file and to start the server.



If you choose to start the server, the Funambol icon (two circular arrows) appears in the system tray. The color of the icon indicates the status of the server.

- Green – the server is running.
- Red – the server is stopped.
- Yellow – the server is starting, wait for the icon to become green to access the server.

If the Funambol icon is green, you can verify the server is running by pointing a browser to `http://localhost:8080/funambol`, where a welcome page to the Funambol DS Server should display.

Linux

1. Execute the downloaded file in a shell as follows:

```
> sh funambol-<version>.bin
```
2. Review the license agreement. To accept the terms, select the checkbox and click **Next**.
3. Specify a top-level directory in which to install the bundled software. If not specified, the installation defaults to `funambol`.

Installing the Server Without Bundled Software

This section assumes that you have an application server, JDK/JRE, and a JDBC-compliant database installed on your system, or will obtain and install those components.

Windows and Unix / Linux

1. Unpack the compressed file. This creates a top-level directory named `Funambol` that we will refer to as `<FUNAMBOL_HOME>` in this document. The server is in the `\ds-server` subdirectory and we will refer to this subdirectory as `<DS_SERVER_HOME>`.
2. Using a text editor, open the `<DS_SERVER_HOME>\install.properties` file. Note that commented lines are preceded by the number (#) symbol. We suggest that you take a few moments to read the comments about each parameter.
3. Locate the line `dbms=` and specify the name of your JDBC-compliant database.
4. Locate the following lines and specify values appropriate for your database:

```
jdbc.classpath=
jdbc.driver=
jdbc.url=
jdbc.user=
jdbc.password=
```
5. Save and close `install.properties`. For additional details on this file, see “Install Properties” on page 61.



6. Verify that your `J2EE_HOME` environment variable points to the top-level directory where your application server resides, and the `JAVA_HOME` environment variable points to the top-level directory where the JDK/JRE resides.
7. Run the server installation script. Open a command prompt window and type the following at the prompt:

Windows:

```
> cd <DS_SERVER_HOME>
> bin\install <application_server>
```

Example:

```
> cd "program files\funambol\ds-server"
> bin\install jboss
```

Unix/Linux:

```
> cd <DS_SERVER_HOME>
> sh bin/install.sh <application_server>
```

You will be prompted several times as to whether you want to create/recreate the database; type `y` (yes) for all questions.

8. For instructions on starting the server, see "Starting the Server" on page 8.



Installing the Funambol Administration Tool

When you do not use the bundled software, you need to install the Funambol Administration Tool separately. You can install the Administration Tool on the same machine as the Funambol DS Server, or on a remote machine. Before proceeding, ensure that the version of the Administration Tool matches that of the Funambol DS Server.

Windows

1. Run the executable installation file.
2. Review the license agreement. To accept the terms, select the checkbox and click **Next**.
3. Accept the suggested **Destination Folder**, i.e., the \Funambol\admin directory.
4. Accept the suggested **Program Group**, i.e., Funambol\Funambol Administration Tool.
5. At the completion of the installation, you are given the option to view the readme file and to start the tool.

Unix / Linux

The Administration Tool download for Unix/Linux is a tarball installation file. Perform the following:

```
> cd <FUNAMBOL_HOME>/admin  
> gunzip funambol-admin-<version>.tar.gz  
> tar xvf funambol-admin-<version>.tar
```



Starting and Stopping the Funambol DS Server

The methods for starting and stopping the Funambol DS Server depend on how the application server associated with the server starts and stops J2EE applications. In this section, we assume the Funambol DS Server is installed as a standalone application; therefore, when the Funambol DS Server is stopped, the entire application server is stopped, and when it is started, the entire application server is started.

Starting the Server

The following commands start the server:

	Bundled Installation	Unbundled Installation
Windows	Start > All Programs > Funambol > Data Synchronization Server > Start	Command line: > cd <DS_SERVER_HOME> > bin\start.cmd
Unix / Linux	Command line: > cd <FUNMAMBOL_HOME> > sh tools/bin/funambol.sh start	Command line: > cd <DS_SERVER_HOME> > sh bin/start.sh

If you have installed the bundled software, when you start the server, the Funambol icon (two circular arrows) appears in the system tray. The color of the icon indicates the status of the server.

- Green – the server is running.
- Red – the server is stopped.
- Yellow – the server is starting, wait for the icon to become green to access the server.

If the Funambol icon is green, you can verify the server is running by pointing a browser to <http://localhost:8080/funambol>, where a welcome page to the Funambol DS Server should display. You can also right-click the icon to open a menu that allows you to start or stop the server.

NOTE: If you have installed the server as an individual component (not the bundled software), the Funambol icon does not display in the system tray.



Stopping the Server

Windows (bundled installation)

The following actions stop the server:

- Start > All Programs > Funambol > Data Synchronization Server > Stop
- Right-click the icon in system tray and select **Stop**

NOTE: When you right-click the icon in the system tray, you can also select Exit. This terminates the service with which you can see if the server is still running, but does not stop the server, and thus avoids stopping the server if the user disconnects from the desktop.

Unbundled Installation

The method of stopping the Funambol DS Server depends on the application server. For JBoss, perform the following:

If it is running in foreground, press **Ctrl+C**; if this fails, determine the process id and kill it with an operation system command or tool.

For Tomcat 5.0.x, perform the following:

Windows

```
> cd <CATALINA_HOME>  
> bin\shutdown.cmd
```

Unix / Linux

```
> cd <CATALINE_HOME>  
> bin/shutdown.sh
```

In either case, <CATALINA_HOME> is the Tomcat installation directory, i.e., <FUNAMBOL_HOME>\tools\tomcat.



Starting and Stopping the Administration Tool

To start the Administration Tool, perform the following:

Windows

Start > All Programs > Funambol > Administration Tool

Unix / Linux

```
> cd <FUNAMBOL_HOME>/admin  
> bin/funamboladmin
```

To stop the Administration Tool, perform the following on the Administration Tool main menu:

File > Exit



Configuring the Funambol DS Server

This section describes the following Funambol DS Server configuration tasks:

- Specifying the Public IP Address
- Changing the Device-ID
- Changing the HTTP Server Port
- Changing the Database

To perform the first two tasks, you use the Administration Tool to modify the appropriate *server JavaBean*. Server JavaBeans are JavaBeans used server-side. A bean configuration is stored in a serialized form of the bean itself. In this way, a bean can be instantiated, configured and serialized to persist its configuration. Later, the bean can be deserialized in memory as a properly configured instance. For additional details on server JavaBeans, see the *Funambol DS Server Developer's Guide*.

To perform the remaining tasks, you modify configuration or properties files.

Specifying the Public IP Address

The default installation provides access to the Funambol DS Server via the localhost; however, in most situations it is necessary to change the IP/hostname to access the server from a remote system. Before proceeding, see “Introduction” on page 18 for general information on using the Administration Tool, and “Server Engine Parameters” on page 22 for information on server JavaBeans.

To change the IP/hostname, perform the following:

1. Start the DS Server.
2. Start the Administration Tool and login to the server.
3. In the navigation pane, expand the server and double-click **Server settings**.
4. In the Server Settings panel, locate the Server URI property, specify the desired IP address or hostname and click **Save**.

In the Output message pane, the message “Server configuration saved.” is displayed.



Changing the Device-ID

In the default Funambol DS Server installation, your device-id is rewritten as the generic device-id `syncml-phone`. This enables all devices to work directly with the username/password combination of `guest/guest`. In a production environment, different usernames would be used in association with real device-ids. To use real device-ids, you need to reset the *Pipeline Manager* (see below).

The Funambol distribution provides a non-rewriting `PipelineManagerGeneric` server JavaBean, as well as a `PipelineManager` server JavaBean (the default); the latter implements the `syncml-phone` device-id override. You have to reset this property to the `com/funambol/server/engine/pipeline/PipelineManagerGeneric.xml` server JavaBean to use real device-ids.

Before proceeding, see “Introduction” on page 18 for general information on using the Administration Tool, and “Server Engine Parameters” on page 22 for information on server JavaBeans.

To change the Pipeline Manager, perform the following:

1. Start the DS Server.
2. Start the Administration Tool and login to the server.
3. In the navigation pane, expand the server and double-click **Server settings**.
4. In the Server Settings panel, locate the Pipeline manager property, specify `com/funambol/server/engine/pipeline/PipelineManagerGeneric.xml` and click **Save**.

In the output message pane, the message “Server configuration saved.” is displayed.

Understanding the Pipeline Manager

The Pipeline Manager constructs and manages input and output pipelines. It is configured with a list of components that build up the input pipeline, and a list of components that build up the output pipeline. A message processing pipeline modifies incoming and outgoing messages, i.e., the message is modified before it goes into the synchronization engine, and the message returned by the synchronization engine is changed before it is sent to the client. The duties of the Pipeline Manager are as follows:

- Create input and output pipelines at initialization
- Provide a way to start the input or output pipeline
- Coordinate the execution of the components in the pipelines
- Keep the “message processing context” (the state of one pipeline execution)

Input and output pipeline components are called *synclets*. For additional details on the Pipeline Manager, see the *Funambol Architecture* document.



Changing the HTTP Server Port

To change the port to which the Funambol DS Server listens, you change the value of the HTTP server port.

Tomcat Example (non-bundled version)

Use a text editor to access the `<TOMCAT_HOME>\conf\server.xml` file and find the following lines:

```
<!-- Define a non-SSL 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"
```

Modify the `Connector port` property to the desired value, save the configuration file, and restart the server.

NOTE: In the bundled version, `server.xml` file is `<FUNAMBOL_HOME>\tools\tomcat\conf\` directory and contains additional comment lines shown below:

```
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080 -->
<!-- Funambol comment: don't modify or remove this Funambol comment! ##### -->
<Connector port="8080"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    debug="0" connectionTimeout="20000"
    disableUploadTimeout="true" />
<!-- Funambol comment: don't modify or remove this Funambol comment! ##### -->
<!-- Note : To disable connection timeouts, set connectionTimeout value
    to 0 -->
```

These lines comment out the above example and are used by the Funambol icon in the system tray.

JBoss Example

1. Access the `deploy/http-invoker.sar/META-INF/jboss-service.xml` file and change any occurrence of port 8080 to the desired value.
2. Access the `deploy/jbossweb-tomcatXX.sar/META-INF/jboss-service.xml` file and change any occurrence of port 8080 to the desired value.
3. Restart the server.



Changing the Database

The bundled distribution includes the Hypersonic database; however, you can use any JDBC-compliant database. We recommend that you use a database you are familiar with, or in a production environment, one that is most suitable. You can use either of the following methods to change the database:

- Modify the install properties file (recommended).
- Modify the application server specific database configuration.

To Modify the Install Properties File

1. Use a text editor to open the `<DS_SERVER_HOME>\install.properties` file. Note that commented lines are preceded by the number (#) symbol.
2. Locate the line `dbms=` and specify the name of your JDBC-compliant database.
3. Locate the following lines and specify values appropriate for your database:
`jdbc.classpath=`
`jdbc.driver=`
`jdbc.url=`
`jdbc.user=`
`jdbc.password=`
4. Save and close `install.properties`. For additional details on this file, see “Install Properties” on page 61.
5. Verify that your `J2EE_HOME` environment variable points to the top-level directory where your application server resides, and the `JAVA_HOME` environment variable points to the top-level directory where the JDK/JRE resides.
6. Run the server installation script. Open a command prompt window and type the following at the prompt:

Windows:

```
> cd <DS_SERVER_HOME>
> bin\install <application_server>
```

Example:

```
> cd "program files\funambol\ds-server"
> bin\install jboss
```

Unix/Linux:

```
> cd <DS_SERVER_HOME>
> sh bin/install.sh <application_server>
```

You will be prompted several times as to whether you want to create/recreate the database; type y (yes) for all questions.



7. Verify the server is running by pointing a browser to `http://<server>:<port>/funambol`, where a welcome page to the Funambol DS Server should display.

To Modify the Application Server Specific Database Configuration

After the Funambol DS Server is installed, the configuration of database access is delegated to the application server. The FunambolDS Server uses the JNDI name `jdbc/fnbls` to acquire a connection from the application server.

For example, with JBoss 3.2, the database connection settings are stored in `J2EE_HOME/server/funambol/deploy/funambol-ds.xml`. To configure a new data source, edit the file by changing the following values:

```
<connection-url>
<driver-class>
<user-name>
<password>
```

In addition, in order to tell the application server where to find the JDBC driver classes, the file `{FUNAMBOL_HOME}/bin/start.bat/sh` must be edited and the driver classpath must be appended to the environment variable `JBoss_CLASSPATH`. Refer to your application server's documentation for details on JDBC configuration.

Logging Database Access

The Funambol DS Server does not create a log of database access directly from the classes that use JDBC. A more generic approach is based on P6Log, 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, visit <http://www.p6spy.com/documentation/index.htm>.

A simple example for configuring the Funambol DS Server to use P6Spy is as follows:

1. Download and install P6Spy using the above link.
2. Copy `spy.jar` into the `<JAVA_HOME>\jre\lib\ext` directory.
3. Copy the file `<DS_SERVER_HOME>\lib\funambol-sqllog.jar` into the `<JAVA_HOME>\jre\lib\ext` directory (this contains an adapter for P6Spy to the standard Java logging system).
4. Append the `<DS_SERVER_HOME>\lib\logging` directory to the application server `CLASSPATH` (this allows P6Spy to access its configuration file `spy.properties`).





Chapter 2 **Using the Funambol Administration Tool**

Topics

- *Introduction, page 18*
- *Server Settings, page 21*
- *Managing Users, page 29*
- *Managing Devices, page 31*
- *Managing Principals, page 33*



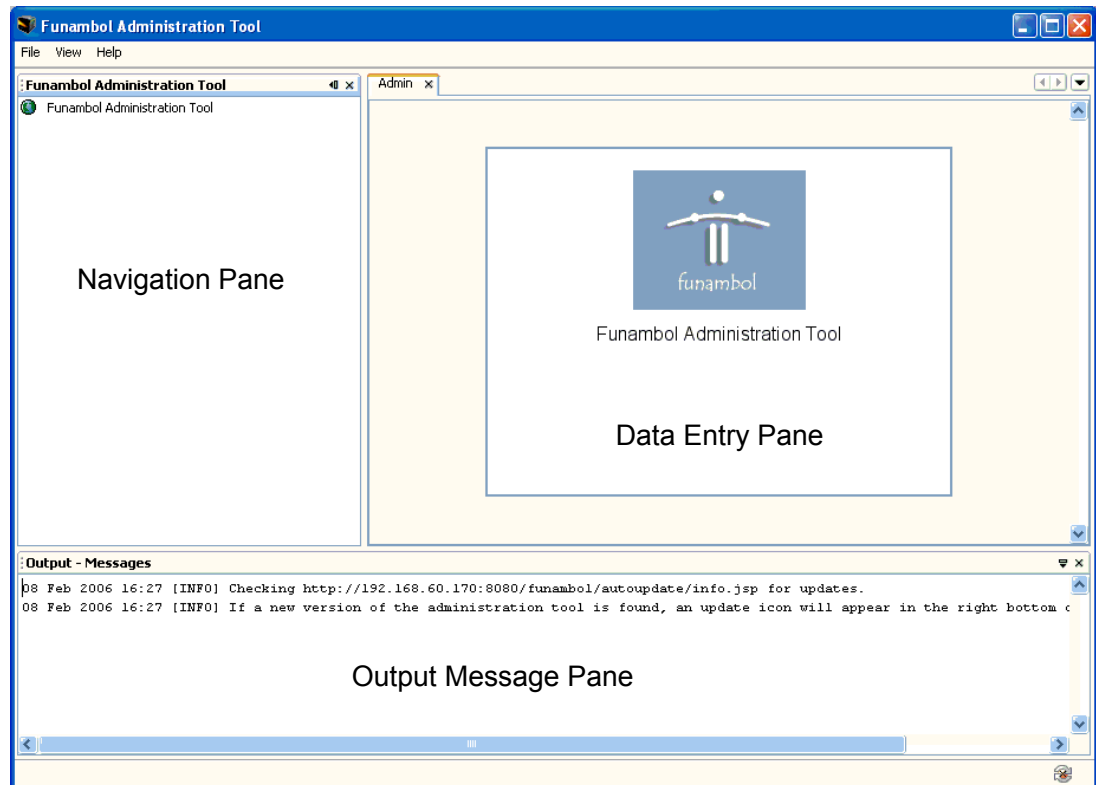
Introduction

The Funambol Administration Tool is the administrative interface to the Funambol DS Server. You can use the Funambol Administration Tool to perform the following tasks:

- Manage Funambol DS Server settings
- Add, edit, and delete users, devices, and principals
- Display installed modules, connectors, and SyncSource types
- Create, edit, and delete SyncSources

Main Window

When you start the Administration Tool, the following window displays:



The main window is partitioned into the following panes:

- Navigation pane – select the server, users, devices, principals or modules to configure.
- Data entry pane – add, edit, delete or search for the item selected in the navigation pane.
- Output Messages pane – displays Administration Tool status messages.



Server Login

To perform server administrative tasks, you first need to login to the server. On the main menu bar, select:

File > Login

Alternatively, you can double-click **Funambol Administration Tool** in the navigation pane. The Login window displays:

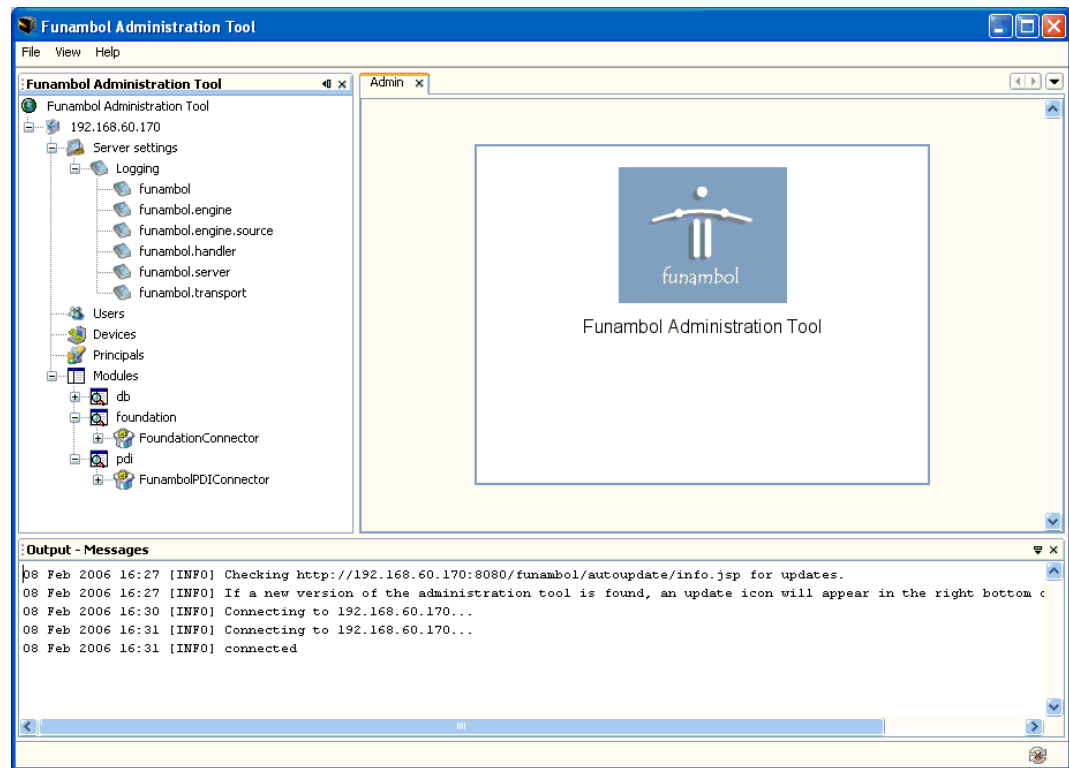
Login Parameters

Parameter	Description
Hostname/IP	Server host or server IP address. If the server is installed on the same machine, <i>localhost</i> will work.
Port	Port number for accessing the server. The standard port for the bundled installation using Tomcat is 8080.
User name	User name for server administration. Default: admin.
Password	Password for server administration: Default: sa. It is strongly advised to change the default password as soon as possible.

After specifying the above values, click **Login**.



After login, your server displays as the root of the tree in the navigation pane. Expand the server to displays server settings, users, devices, principals and modules as shown below:





Server Settings

The primary configuration file for the Funambol DS Server, <DS_SERVER_HOME>\config\Funambol.xml., contains global server settings. The Administration Tool provides an interface to this file that you access by double-clicking **Server settings** in the navigation pane. The following server parameters display:

Capabilities

Manufacturer :	<input type="text" value="Funambol"/>
Model :	<input type="text" value="DS Server"/>
Software version :	<input type="text" value="5.0"/>
Hardware version :	<input type="text" value="-"/>
Firmware version :	<input type="text" value="-"/>
OEM :	<input type="text" value="-"/>
Device id :	<input type="text" value="funambol"/>
Device type :	<input type="text" value="server"/>
DTD version :	<input type="text" value="1.2"/>

Server Capabilities Parameters

Parameter	Description
Manufacturer	Software maker.
Model	Software product /component.
Software version	Software version number.
Hardware version	Hardware version number.
Firmware version	Firmware version number
OEM	OEM
Device id	ID of the device.
Device type	Type of the device.
DTD version	The supported SyncML DTD version.

Engine

Server URI :	http://R505:8080/funambol/ds	
Officer :	com/funambol/server/security/DBOfficer.xml	
Logging configuration :	com/funambol/server/logging/Logging.xml	
Pipeline manager :	mbol/server/engine/pipeline/PipelineManager.xml	
Handler :	com.funambol.server.session.SyncSessionHandler	
Persistence store manager :	/funambol/server/store/PersistentStoreManager.xml	
Device inventory :	om/funambol/server/inventory/DeviceInventory.xml	
Data transformer manager :	/engine/transformer/DataTransformerManager.xml	Configure
Strategy :	com.funambol.server.engine.Sync4jStrategy	
User manager :	com/funambol/server/admin/DBUserManager.xml	
Min. value for max. msg size :	2500	

[Save](#)

Server Engine Parameters

NOTE: For parameters that are server JavaBeans, the specified value is interpreted as the name of a JavaBean, and is searched for in the configpath as the name of a serialized object. If no serialized object is found, the value is considered equal to the name of a class and is searched for in the classpath.

Parameter	Description
Server URI	This parameter serves two purposes: 1) to give a unique identifier to the server that can be used by a client to ensure it is talking to the intended server; 2) to tell the client the base URL to which the SyncML messages of a synchronization session are addressed. This is used by the server to relate an incoming message to a specific synchronization session.
Officer	Server JavaBean representing the security officer. The officer is responsible for authenticating users and authorizing access to DS server resources. The default officer authenticates the user in the Funambol DS Server database, but this component can be customized in order to perform the authentication against a different system. In such a case, this parameter will reflect the Server JavaBean of the custom component.
Logging configuration	Server JavaBean for logging configuration.



Parameter	Description
Pipeline manager	Server JavaBean for the Pipeline Manager configuration. The Pipeline Manager constructs and manages input and output pipelines. It is configured with a list of components that build up the input pipeline, and a list of components that build up the output pipeline. A message processing pipeline modifies incoming and outgoing messages, i.e., the message is modified before it goes into the synchronization engine, and the message returned by the synchronization engine is changed before it is sent to the client.
Handler	SyncSession handler. The session handler is the server component that handles the entire synchronization session. It defaults to <code>com.funambol.server.session.SyncSessionHandler</code> , but it can be replaced by the server JavaBean representing the custom component.
Persistence store manager	Server JavaBean representing the persistent store manager, which is the component responsible of accessing the persistent store (i.e., the database used internally by the DS server). The default implementation is configured by the server JavaBean <code>com.funambol/server/store/PersistentStoreManager.xml</code> and stores and reads the data to and from a JDBC database; a custom implementation can be configured by specifying a custom server bean.
Device inventory	<p>Server JavaBean for Device Inventory configuration. The Device Inventory component is responsible for storing information about devices and their capabilities.</p> <p>Open source: The default implementation is <code>com.funambol/server/inventory/DeviceInventory.xml</code> and it currently does not perform any action. This is a base for custom developments.</p> <p>Enterprise: The default implementation is <code>com.funambol/server/inventory/PSDeviceInventory.xml</code>, which uses the <code>PersistentStoreManager</code> to store and retrieves the devices and their capabilities.</p>
Data transformer manager	<p>Server JavaBean for Data Transformer Manager configuration. The Data transformer manager is the component responsible for applying transformations to the data provided by a connector. Common transformations are encoding of binary data (e.g., B64 encoding), encryptions, conversions between vcard/icalendar and SIF-C/SIF-E, and so on. It defaults to <code>com.funambol/server/engine/ttransformer/DataTransformerManager.xml</code>.</p> <p>To specify configuration parameters, click Configure (see “Data Transformer Manager Configuration Parameters” on page 25).</p>



Parameter	Description
Strategy	<p>Server JavaBean representing a SyncStrategy object. The Strategy component is responsible for defining the behavior of the synchronization engine. One of the main roles of this component is to resolve the conflicts that may arise when the same information is modified by different sources.</p> <p>It defaults to the server JavaBean <code>com.funambol.server.engine.Sync4jStrategy</code>.</p>
User manager	<p>Server JavaBean representing the user manager. This component is responsible of accessing the user database. It defaults to <code>com/funambol/server/admin/DBUserManager.xml</code>, which accesses the user in the local Funambol DS Server database. However, it can be customized in order to access a different database, such as a LDAP server. In that case, this parameter will reflect the custom server bean.</p>
Min. value for max. msg size	<p>The minimum MaxMsgSize supported by the server. If a client requests a MaxMsgSize less than this value, the request is rejected. When starting a synchronization, a device can tell the server not to send messages bigger than a certain byte amount; however, there is a lower limit under which the server may not be able to work. This limit can be influenced by some internal processing or by the amount of metadata the server needs to return. This parameter specifies the lower limit the server can accept.</p>

DataTransformer Manager Configuration

Transformers for incoming items

Name	Class
b64	b64decoder
des	desdecoder

Transformers for outgoing items

Name	Class
b64	b64encoder

Transformations required

Source URI	Transformation
cal	b64
card	b64

Data Transformer Manager Configuration Parameters

Parameter	Description
Transformers for incoming items	Specifies the name and class of transformers for incoming items.
Transformers for outgoing items	Specifies the name and class of transformers for outgoing items.
Transformations required	Specifies the source URI of items that require transformation, and the name of the transformer to use.

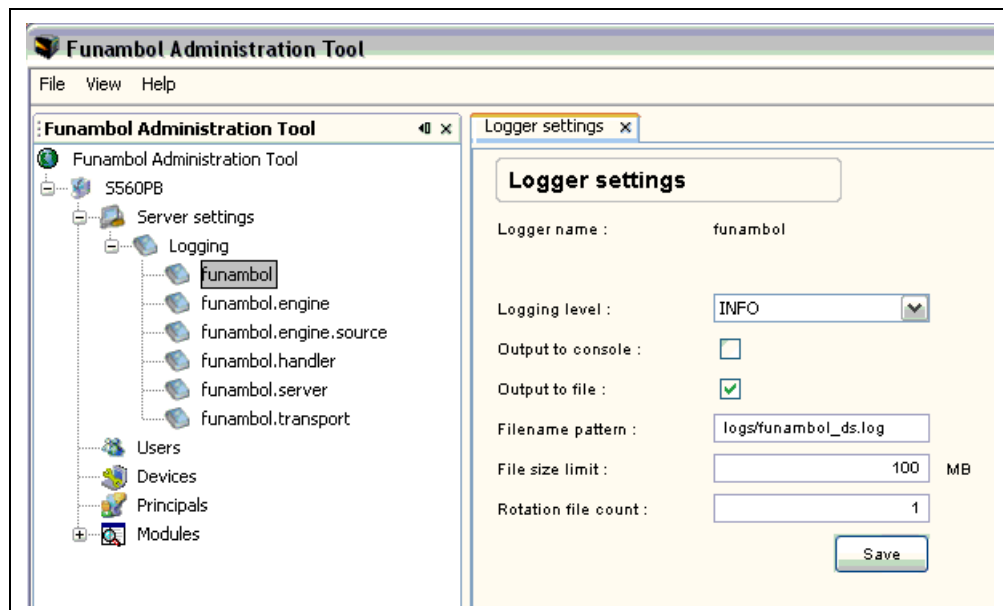
Log Settings

A basic installation includes a single log file called `funambol_ds.log`, stored in the `<DS_SERVER_HOME>\logs` directory. However, you can specify logging for specific server components, with the output going to separate files if desired. The system provides *loggers* for this purpose, as follows:

Logger	Description / Filename
funambol	The standard system logger. Filename: <code>funambol_ds.log</code>
funambol.engine	Logger for the engine. Filename: <code>funambol_ds.engine.log</code>
funambol.engine.source	Logger for the engine source. Filename: <code>funambol_ds.engine.source.log</code>
funambol.handler	Logger for the handler. Filename: <code>funambol_ds.handler.log</code>
funambol.server	Logger for the server. Filename: <code>funambol_ds.server.log</code>
funambol.transport	Logger for the transport. Filename: <code>funambol_ds.transport.log</code>

The loggers for specific parts of the server are included in the `funambol` logger by default. If you want to use the logger for a server part, uncheck the **Same as funambol** checkbox in the settings for that logger. This allows you to specify a different log level, output type, and so on for this component. For details, see “Specifying Logger Settings for a Server Component” on page 28.

To specify the level of information, output type, and so on for the standard system logger, access the navigation pane and expand the server as follows: **Server settings > Logging** and select **funambol**. The following logging parameters display:





Logger Settings Parameters

Parameter	Description
Logger name	Name of the logger.
Logging level	<p>The level of information logged. Valid values: NONE = no information logged; ERROR = only errors are logged; INFO = basic info and errors are logged; ALL = info, errors and debug information are logged. Default: INFO.</p> <p>For server problems, or to debug the server or a syncsource, use ALL, since this provides the most information. You should also use ALL if you wish to submit a log file for consideration on Funambol mailing lists.</p>
Output to console	Specifies if the log should be viewed on the standard console. Default: not selected.
Output to file	Specifies if the log should be stored in a file (defined in the Filename pattern field). Default: selected.
Filename pattern	<p>Defines the name of the file where the log is stored when the Output to file field is checked. It consists of a string that includes the following:</p> <ul style="list-style-type: none"> " / " – 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 "%"
File size limit	The maximum size of the log file in MB. Default: 100.
Rotation file count	Logging can either be written to a specified file, or 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. Default: 1.



Specifying Logger Settings for a Server Component

The logger settings for the Funambol engine are shown below. If you uncheck the **Same as funambol** checkbox, you can specify log settings specific to this server component.

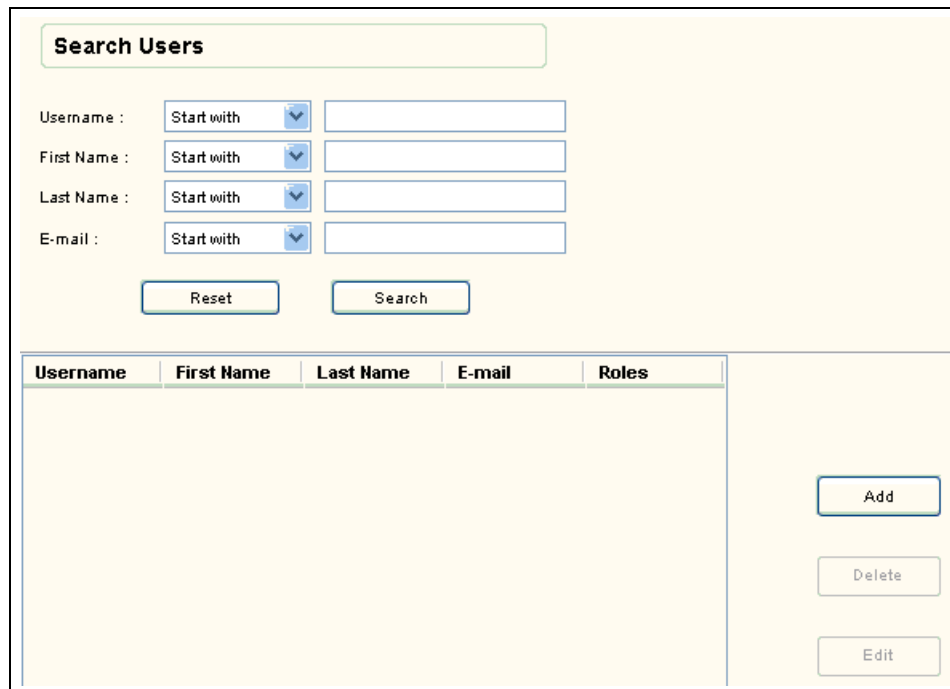
Logger settings

Logger name :	funambol.engine	
Same as funambol :	<input checked="" type="checkbox"/>	
Logging level :	INFO ▼	
Output to console :	<input type="checkbox"/>	
Output to file :	<input type="checkbox"/>	
Filename pattern :	s/funambol_ds.engine.log	
File size limit :	100	MB
Rotation file count :	1	

Save

Managing Users

The user settings specify role-based access to the Funambol DS Server. To access these settings, double-click **Users** in the navigation pane. The following window displays:



Search Users

Username :

First Name :

Last Name :

E-mail :

Username	First Name	Last Name	E-mail	Roles

On this window you can perform the following:

- **Search** – to search for a user, specify a value for the Username, First or Last Name, or E-mail address and click **Search**. The table below the search form displays the results of your query.
- **Add User** – to add a new user, click **Add**. You can also right-click **Users** in the navigation pane and select **Add User**. The Add User window displays; for details, see “Add User Parameters” on page 30.
- **Edit User** – to edit the information about a user, select a row in the table below the search form and click **Edit**, or double-click the row. The User Details window displays; click **Save** to update your changes.
- **Delete User** – to delete a user, select a row in the table below the search form and click **Delete**. A confirmation window displays; click **OK** to delete the user.

Add User

Username :

Password :

Confirm password :

First Name :

Last Name :

E-mail :

Roles :

User

Administrator

Add

Add User Parameters

Parameter	Description
Username	Username.
Password	User's password.
Confirm password	Confirmation of the user's password.
First Name	User's first name.
Last Name	User's last name.
Email	User's email address.
Roles	Role assigned to the user. Valid values: User = can perform synchronizations with the server; Administrator = can perform administrative tasks in addition to synchronizations.



Managing Devices

The device settings specify information about the devices that can connect to the Funambol DS Server. To specify these settings, double-click **Devices** in the navigation pane. The following window displays:

Search Devices

ID :

Type :

Description :

ID	Type	Timezone	Description

On this window you can perform the following:

- **Search** – to search for a device, specify a value for the ID, Type, or Description and click **Search**. The table below the search form displays the results of your query.
- **Add Device** – to add a new device, click **Add**. You can also right-click **Devices** in the navigation pane and select **Add Device**. The Add Device window displays; for details, see “Add Device Parameters” on page 32.
- **Edit Device** – to edit the information about a device, select a row in the table below the search form and click **Edit**, or double-click the row. The Device Details window displays; click **Save** to update your changes.
- **Delete Device** – to delete a device, select a row in the table below the search form and click **Delete**. A confirmation window displays; click **OK** to delete the device.



Add Device

ID :

Type :

Timezone :

Not specified

▼

☐ Convert dates to this timezone

Charset :

UTF-8

▼

Address :

Msisdn :

Notification Builder :

Notification Sender :

Description :

Add

Add Device Parameters

Parameter	Description
ID	The device ID, e.g., the phone IMEI for SyncML phones.
Type	The device type.
Timezone	The timezone associated with the device.
Charset	The character set used for communication with the device. Valid values: UTF-8, UTF-16, ISO-8859-1, US-ASCII.
Address	IP address of the device (if applicable).
Msisdn	Msisdn of the device (i.e., the phone number)
Notification Builder	The builder (server component) used to create notification messages for this device. For example, <code><DS_SERVER_HOME\config\com\funambol\server\notification\DSNotificationBuilder.xml</code>
Notification Sender	The sender (server component) used to send notification messages to this device. For example, <code><DS_SERVER_HOME\config\com\funambol\server\notification\WAPSender.xml</code>
Description	Informational text, e.g., John Smith's phone.



Managing Principals

A user can use multiple devices for data synchronization, for example, a SyncML phone, Outlook and a Pocket PC PDA. In addition, multiple users can synchronize from a single device, if they share it. Therefore, the Funambol DS Server is built around the concept of a *principal*, a concept that associates a user with a device. A principal is a tuple (user,device).

Search Principals

Principal Id :

Start with

Username :

Start with

Device Id :

Start with

Reset

Search

Principal Id	Username	Device Id

Add

Delete

Details

On this window you can perform the following:

- **Search** – to search for a principal, specify a value for the Principal Id, Username, or Device Id and click **Search**. The table below the search form displays the results of your query.
- **Add Principal** – to add a new principal, click **Add**. You can also right-click **Principals** in the navigation pane and select **Add Principal**. The Add Principal window displays; for details, see “Add Principal Parameters” on page 34.
- **Delete Principal** – to delete a principal, select a row in the table below the search form and click **Delete**. A confirmation window displays; click **OK** to delete the principal.
- **Last Synchronization Timestamps** – To display the information about the last synchronization by a principal, select a row in the table below the search form and click **Details**. The Last Synchronization Timestamps window displays. For details, see “Last Synchronization Timestamps” on page 35.



Add Principal

Search Users

Username :

Start with

First Name :

Start with

Last Name :

Start with

E-mail :

Start with

Reset

Search

Username	Name	E-mail
----------	------	--------

Search Devices

ID :

Start with

Type :

Start with

Description :

Start with

Reset

Search

ID	Type	Description
----	------	-------------

Add Principal

Add Principal Parameters

Parameter	Description
Search Users	Search by Username, First or Last Name, or E-mail. The table below the search form displays the results of your query
Search Devices	Search by ID, Type, or Description. The table below the search form displays the results of your query

To add a principal, search for the desired user and device, select each from the respective search results table, and click **Add Principal**.

NOTE: To facilitate use of the Funambol DS Server by first-time users, the Funambol PDI Connector module includes a Synclet that bypasses the device check. When 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).



Last Synchronization Timestamps

Last Synchronization Timestamps

Principal Id : 2
Username : guest
Device Id : so-pim-demo

Database	Sync Type	Status	Client anchor	Server anchor	Start	End
soal	TWO-WAY	200	1140001843667	1140001843466	2006-02-15 03...	2006-02-15 0...
soard	TWO-WAY	200	1140001843667	1140001843466	2006-02-15 03...	2006-02-15 0...

Reset

The table provides the following information about the last synchronizations of the selected principal:

Column	Description
Database	SyncSource with which the principal synchronized.
Sync Type	Type of synchronization.
Status	Sync status code.
Client anchor	The client anchor last used.
Server anchor	The server anchor last used.
Start	Start time of synchronization.
End	End time of synchronization.

The **Reset** button deletes the information on the last synchronization, which means that the next time the principal synchronizes, a slow sync will be performed.





Chapter 3 **Managing Modules**

The chapter provides details for installing modules to extend the functionality of the Funambol DS Server.

Topics

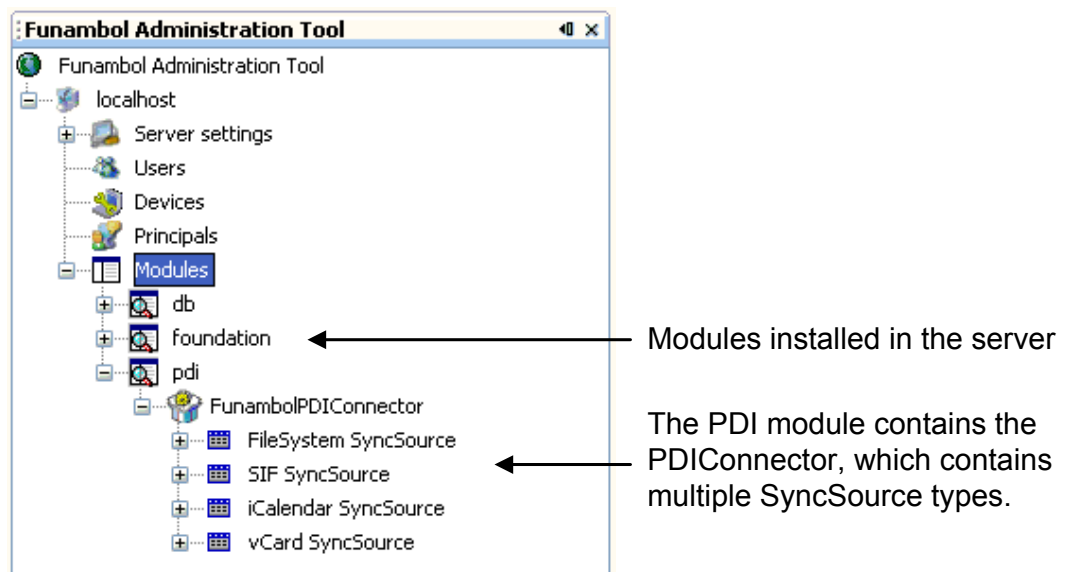
- *Understanding Modules, page 38*
- *Installing Modules, page 41*

Understanding Modules

The following concepts are used for defining server extensions, external data connectivity, and data synchronization:

- **Module** – a server extension that adds new functionality or modifies the existing behavior of a Funambol DS Server component. A module is a package consisting of connectors, SyncSource types, synclets, configuration files, database scripts and so on.
- **Connector** – a server extension that integrates the Funambol DS Server with an external source of data, providing support for data synchronization with that source. It contains everything required for the configuration and runtime execution of the integration module, including configuration files, code, software interfaces, and graphical user interfaces for SyncSource configuration. In addition, a connector defines SyncSource types.
- **SyncSource Type** – a template from which an instance of a SyncSource is created. It represents a specific kind of SyncSource, such as a FileSystem SyncSource that defines how data stored in directories in a file system can be accessed by the Funambol DS Server. Since the SyncSource type does not represent a specific instance, in the case of the FileSystem SyncSource, it does not identify a directory to be used for synchronization. To specify such a directory, you create an instance of the FileSystem SyncSource and configure it with the desired directory. Another example of a SyncSource type is an Exchange Server SyncSource for accessing a Microsoft Exchange account.
- **SyncSource** – the basic synchronization unit, it defines the way a set of data is made accessible to the Funambol DS Server for synchronization. A SyncSource is the entity with which a client requests synchronization. A SyncSource is uniquely identified by the server by a source URI, which the client uses to address it.

You view the modules installed in the server in the navigation pane, as shown below:

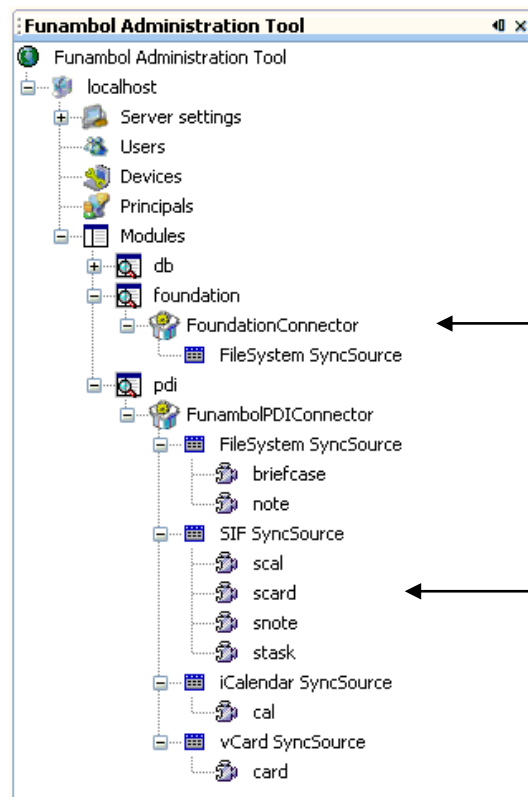




Standard Modules

The Funambol DS Server includes the following modules:

- **Foundation module** – contains the Funambol FoundationConnector. It also defines the FileSystem SyncSource type that can be used to create file system SyncSources.
- **PDI (Personal Data Interchange) module** – contains the Funambol PDIConnector and the FileSystem SyncSource type for creating file system SyncSources. This module is intended to provide easy implementation of PIM data synchronization (contacts and calendar). The PDI module provides pre-configured SyncSources ready for synchronization by just configuring the clients (see “Default Databases” on page 59). The Java GUI Plug-in and the PIM Web Demo use this module. For a demonstration of the PDI module in use, see the *Funambol DS Server Quick Start Guide*, which includes a step-by-step example of contact synchronization.



← The Foundation module contains the FoundationConnector, which contains the FileSystem SyncSource type.

The PDI module contains the PDIConnector, which contains multiple SyncSource types.

← Each SyncSource type can be used to create a SyncSource of that type. For example, the SIF SyncSource type is used to create the scal, scard, snote, and stask SyncSources.

- **Database module** – contains the Funambol DBConnector used for integrating the Funambol DS Server with databases. For details, see Chapter 4, Using the Funambol DB Connector.



Managing SyncSources

To add or edit a SyncSource, double-click the SyncSource type (to add) or the SyncSource (to edit) in the navigation pane. A window such as the following displays:

Edit File System SyncSource

Source URI:

Name:

Type:

Source Directory:

Supported types:

Supported versions:

Encoded: ☒

MultiUser: ☐

File System SyncSource Parameters

Field	Description
Source URI	The case-sensitive identifier of the SyncSource.
Name	The descriptive name of the SyncSource.
Type	MIME type of the file's content, e.g., text/x-vcard.
Source Directory	Directory where files are stored and read.
Supported types	Comma-separated list of supported MIME types, sent in the server capabilities packet. Example: text/x-vcard,text/vcard (see Supported versions)
Supported versions	Comma-separated list of MIME type versions. For each MIME type specified in the Supported types field, a version number must be specified. Example: 2.1,3.0 means support for vCard 2.1 and 3.0.
Encoded	Specifies whether the file's content must be Base64 encoded. This feature is useful if you are building a SyncClient and plan to transfer binary files. If your SyncSource is meant to synchronize with phones, you should leave this option unchecked.
Multiuser	



Installing Modules

A module is usually distributed as a zip or jar archive. The archive might contain several files, but the primary file has the following filename syntax,

`{modulename}-{versionnumber}.s4j`

where *modulename* is the name of the module and *versionnumber* is the version.

Example: Funambol Email Connector Module filename

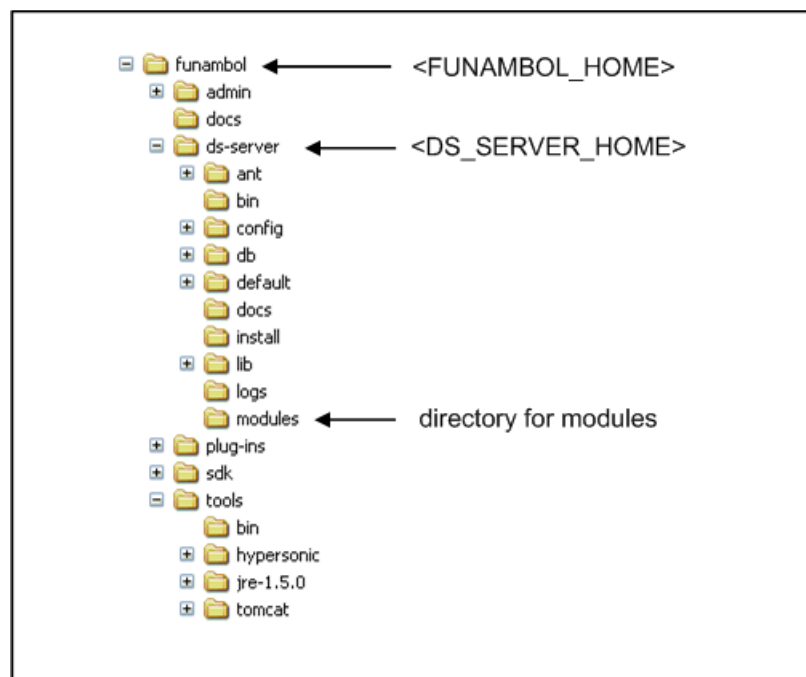
`funambol-email-3.0.2.s4j`

The `.s4j` module file contains the part of the module that becomes part of the Funambol DS Server enterprise archive (a J2EE ear file). It contains classes, configuration files, and initialization files that are processed by the installation procedure. The `.s4j` module file must be copied to the `<DS_SERVER_HOME>\modules` directory to be processed by the installation procedure.

NOTE: Funambol modules can be installed either by re-installing the entire Funambol DS Server, or by using a separate script that installs only modules.

Installing a Module

1. Unpack the module archive file.
2. Copy the `funambol-{modulename}-{versionnumber}.s4j` file to the `<DS_SERVER_HOME>\modules` directory.





3. Using a text editor, open the `<DS_SERVER_HOME>\install.properties` file.
4. Find the line that begins `modules-to-install=` in the Module definitions section. This line specifies, in a comma-separated list, the modules to install during installation.
5. Add `funambol-{modulename}-{versionnumber}` to the comma-separated list without the `.s4j` filename extension.
6. Save and close `install.properties`.
7. Install the module. Open a command prompt window and run the installation script by typing the following at the prompt:

Windows

```
> cd <DS_SERVER_HOME>
> install-modules.cmd <application_server>
```

Unix / Linux

```
> cd <DS_SERVER_HOME>
> install-modules.sh <application_server>
```

During the installation you are prompted to create (i.e., rebuild) the database for the server. Type **n** (no) if you have data in the database that you do not want to lose, such as your existing users, mappings and previous sync information.

In addition, as the installation procedure installs each module, you are prompted to rebuild that module's database. Accept or decline as appropriate, but for the module that is being installed for the first time, you must type **y** (yes).



Chapter 4 **Using the Funambol DB Connector**

The chapter describes how you can use the Funambol DB Connector to synchronize data between client and server databases.

Topics

- *Overview, page 44*
- *Configuring a Single Table Synchronization, page 49*
- *Configuring a Partitioned Table Synchronization, page 53*

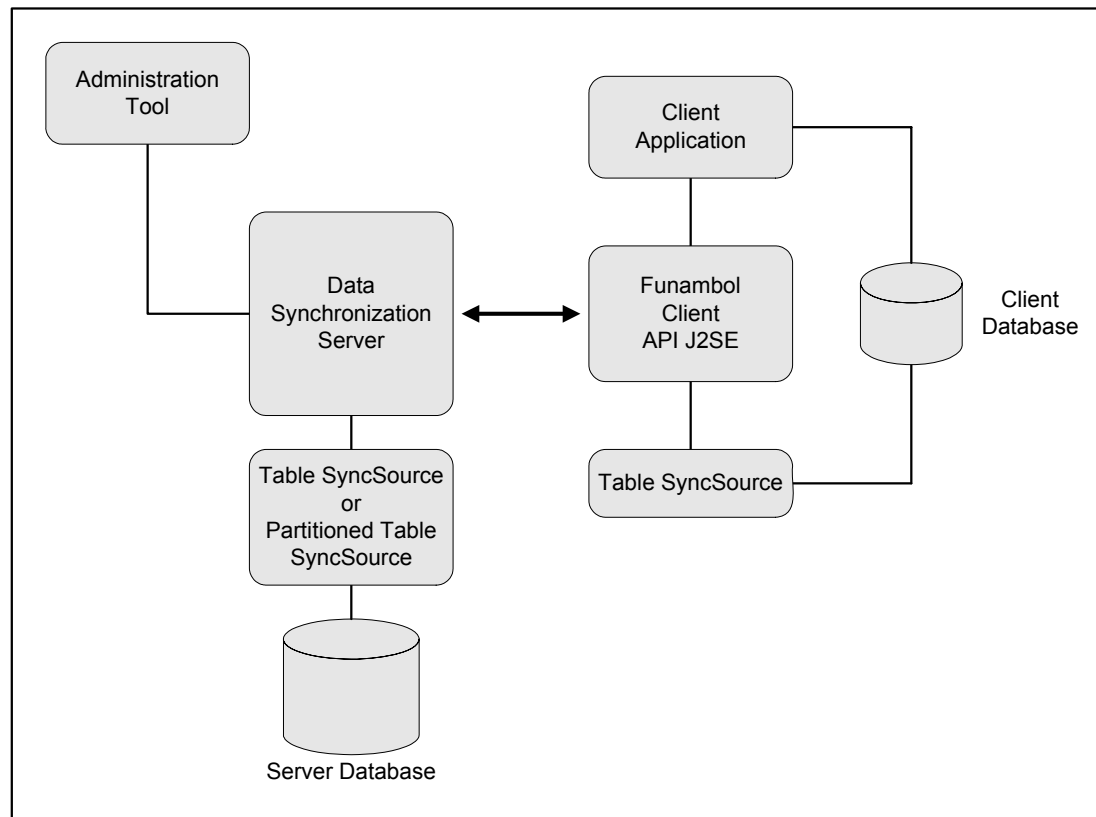


Overview

The Funambol DB Connector is a server extension that you can use to perform table-to-table database synchronization. This connector is included in a default module in the Funambol DS Server, and provides the following SyncSource types:

- **Table SyncSource** – used to create a SyncSource for synchronizing the data in a table that exists in both a client and server database.
- **Partitioned Table SyncSource** – used to create a SyncSource for synchronizing data that on the server side is stored in a data table and in a *partitioning* table. The partitioning table is used to identify the rows that belongs to a particular user, so that only the data of a given user is synchronized into the client table. Note that on the client only the data table is synchronized, since the partitioning table is not needed.

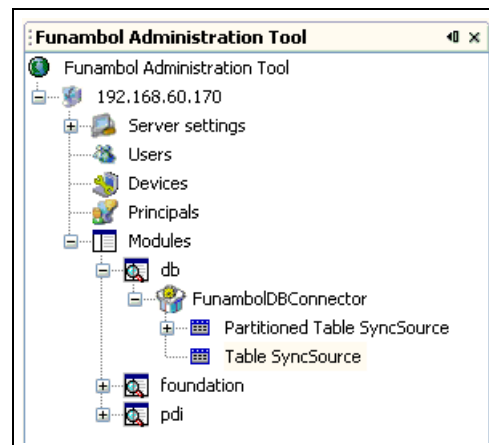
These SyncSources are the key components for enabling the Funambol DS Server to synchronize data between client and server databases, as illustrated below:





Using the DB Connector

You access the DB Connector using the navigation pane:



To add a Table SyncSource instance, double-click the Table SyncSource type or right-click it and select **Add SyncSource**. To edit a Table SyncSource instance, double-click it or right-click and select **Edit** (or select **Delete** to delete). The Edit Table SyncSource window displays:

Edit Table SyncSource

Source URI:
Name:
Type:
JNDI Name datasource:

Table Info

Load data from db

Press this button to select the desired table and load its fields

Table name:
Key field:
Last update timestamp field:
Last update type field:
Principal field:

Fields Mapping:

Server	Client	Binary data

New

Remove

Add



The Edit Partitioned Table SyncSource window has additional parameters. For details, see “Configuring the Server Partitioned Table SyncSource” on page 55.

Edit Table SyncSource Parameters

Parameter	Description
Source URI	The case-sensitive identifier of the SyncSource.
Name	The descriptive name of the SyncSource.
Type	MIME type of the content, e.g., text/plain.
JNDI Name datasource	The JNDI name of the datasource for accessing the database.
Table name	The name of the database table.
Key field	The primary key of the table.
Last update timestamp field	The field containing the last modification timestamp.
Last update type field	The field containing the last modification type. Valid values: N = new; U = updated; D = deleted.
Principal field	The field containing the principal ID. If not specified, the table data will not be filtered based on the principal.
Field Mapping	The mapping of server and client fields, where each row represents a mapping of a server field to a client field. Select the checkbox in the Binary data column to mark a field as binary (e.g., a photo of a contact). Binary fields are encoded before sent between client and server.

Edit Table SyncSource Buttons

- **Load table from db** – Displays a window that allows you to connect with a database and select a table to synchronize with. This action populates the fields in the Table Info area. For example, the Table name field displays the name of the selected table, and the Key field list box contains fields for selection. In addition, the Field Mapping table is populated with server and client fields, with a checkbox for each row to designate binary data fields.
- **New** – Adds a new row (mapping) to the Field Mapping table.
- **Remove** – Removes the selected row (mapping) from the Field Mapping table.
- **Add** – Adds a SyncSource (i.e., an instance of the SyncSource type) configured with the specified parameter values.



Adding a Table SyncSource

To add a Table SyncSource, perform the following:

1. Double-click **Table SyncSource** in the navigation pane.
2. Specify the Source URI, Name, Type, and JNDI Name datasource.
3. Click **Load data from db**. The following window displays:

Load Data from DB Parameters

Parameter	Description
Jar driver	JAR file containing the database driver, including the path. You can use the "... " button to browse to the desired file; you can specify multiple files.
Driver	Database driver class.
URL	Database URL.
User	User ID for accessing the database.
Password	Password for accessing the database.
Tables List	List of tables in the specified database.
Columns List	List of columns in the selected table.

4. Specify values for accessing the database and click **Connect**. This action populates the Tables List with all tables accessible with the given values.
5. Select the desired table in the Tables List. This action populates the Columns List with all columns in the selected table.

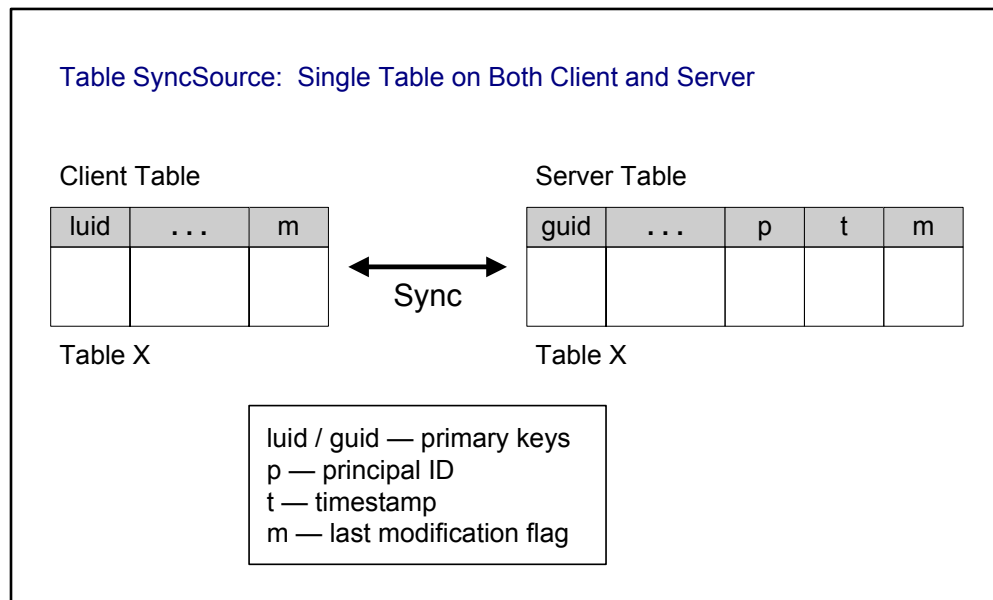


6. Click **OK** or **Cancel** to cancel the operation. When you click **OK**, you return to the Edit Table SyncSource window. The parameters in the Table Info area, such as Table name, Key field, and so on, are now populated with the fields of the selected table. In addition, the Field Mapping table is populated with server and client fields, with a checkbox for each row to designate binary data fields.
7. Select the desired field for the parameters in the Table Info area. The fields you select are highlighted in the Field Mapping table. For example, when you make a selection for Key field, the selected field is highlighted in the table.
8. Add or remove mappings in the Field Mapping table as desired, using the **New** or **Remove** buttons.
9. Click **Add** to create the SyncSource instance.



Configuring a Single Table Synchronization

To set up synchronization between client and server for a single table, you use the Table SyncSource type provided in the DB Connector. In this example, the entire table is synchronized, and data is filtered based on the principal ID, such that each principal only synchronizes the data that belongs to it. The table fields on the client and on server are shown below:



On the client, the table must have fields containing the following:

- Primary key of the table
- Last modification type, where the valid values are N, U, and D. For details, see “Edit Table SyncSource Parameters” on page 46.

On the server, the table must have fields containing the following:

- Primary key of the table.
- Last modification timestamp.
- Last modification type, where the valid values are N, U, and D.
- A field containing the ID of the principal associated with a record is optional. If non-existent, table data will not be filtered based on the principal.

To review the architecture of the synchronization, see “Overview” on page 44. You create a Table SyncSource for the server and a Table SyncSource for the client to access the data in their respective databases.



Configuring the Server Table SyncSource

To create an instance of the server Table SyncSource, see “Adding a Table SyncSource” on page 47. A sample configuration is shown below:

Parameter	Value	
Source URI	./contactDB	
Name	contactDB	
Type	text/plain	
JNDI Name datasource	jdbc/fnblds	
Table name	contact	
Key field	id_contact	
Last update timestamp field	update_date	
Last update type field	update_type	
Principal field	id_principal	
Field 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	None	

For parameter definitions, see “Edit Table SyncSource Parameters” on page 46.

Configuring the Client Table SyncSource

The DB Connector provides a client Table SyncSource for use with the Funambol Client API J2SE. This SyncSource is implemented in *com.funambol.db.client.engine.source.TableSyncSource*. The underlying table must have a field to handle the state of each record. This field is configurable into the management panel of the SyncSource.



The client Table SyncSource is configured with the following parameters:

Parameter	Description
tableName	The name of the table.
keyField	The primary key field.
updateTypeField	The field containing the last update type.
fieldsList	A comma-separated list of the fields.
binaryFields	A comma-separated list of the binary fields.

A sample of the configuration file (including standard SyncSource parameters) is shown below:

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

Database Access

A DBMS capable of running on a PDA usually has restrictions and limitations. For example, it may not be available via JNDI or may not allow more than one open connection at a time. In addition, the overhead for getting a connection may be substantial. Therefore, the Table SyncSource class can use an external connection factory for the connection to the database.

The connection factory is represented by the interface *com.funambol.db.client.ConnectionFactory* and can be provided by the client application. This allows the SyncSource to use an existing database connection. The *com.funambol.db.client.ConnectionFactory* class defines the following methods:

- **getConnection(): Connection** – returns a database connection
- **closeConnection(con: Connection): void** – closes the database connection

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



An example of the device management node `db/connectionfactory` is shown below:

```
className=com.funambol.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 setter methods (set + property name capitalized).

The DB module also provides *com.funambol.db.client.ConnectionFactoryImpl*, a simple connection factory that opens a new connection to the database for all calls to the `getConnection()` method. The method `closeConnection()` closes the connection. The following configuration parameters are used:

Parameter	Description
driver	The driver to access the database.
url	The URL to access the database.
user	The user to access the database.
password	The password to access the database.

For an example of a connection factory with a single connection to the database, see “Connection Factory Sample” on page 62.



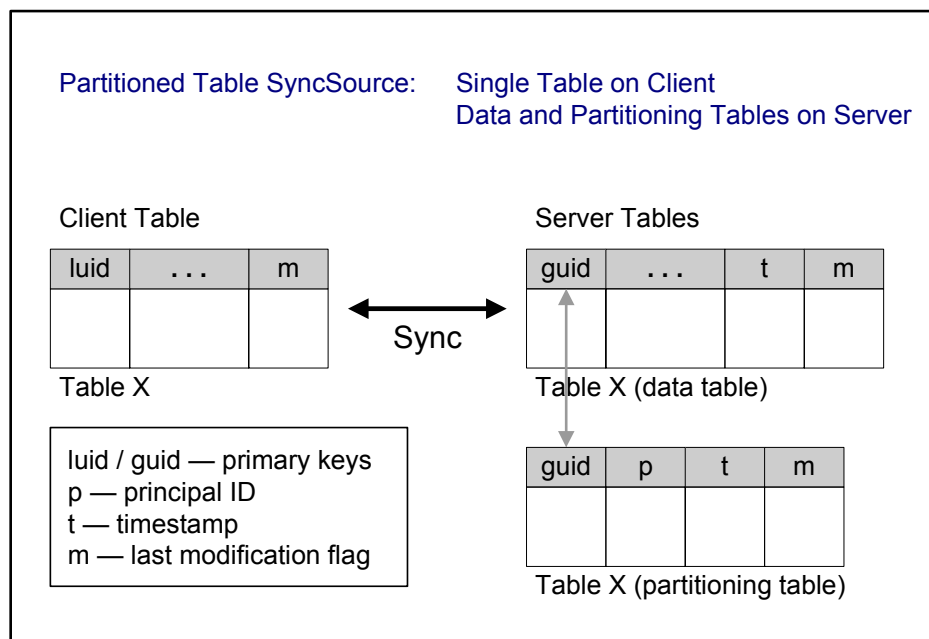
Configuring a Partitioned Table Synchronization

For this synchronization you use the Partitioned Table SyncSource type provided in the DB Connector. In this case, the association between a row of the data table on the server, and the owner (i.e., principal) of that row, is stored in a principal table, referred to as a *partitioning* table. The primary issue is that modifications to both the data table and the partitioning table must be reflected on the client.

For example, suppose the data table contains customer data and the partitioning table contains the sales agent that deals with a particular customer (thus it contains the agent-customer 1:N relationship). A row on the client must be deleted under the following conditions:

- A customer is deleted.
- A customer is associated to another principal.

The table fields on the client and on server are shown below:



On the client, the table must have fields containing the following:

- Primary key of the table
- Last modification type, where the valid values are S, N, U, and D. For details, see “Edit Table SyncSource Parameters” on page 46.



On the server, the data table must have fields containing the following:

- Primary key of the table.
- A “guid” field (can be the preceding field)
- Last modification timestamp.
- Last modification type, where the valid values are S, N, U, and D.

The partitioning table must have fields containing the following:

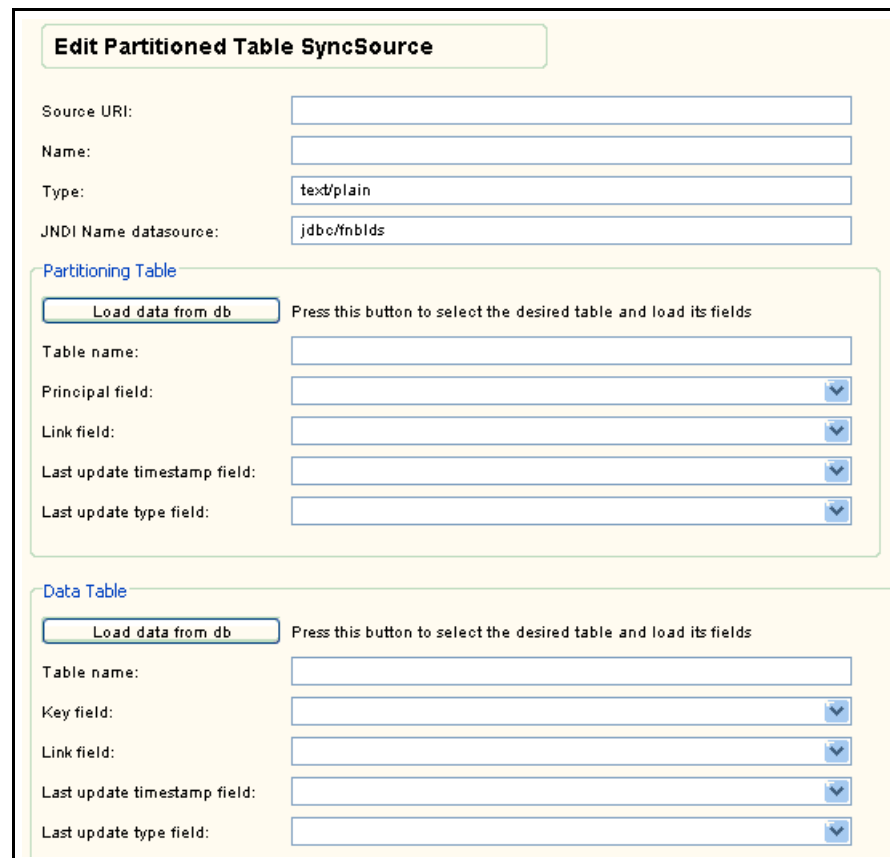
- Principal ID of the principal.
- A “guid” field used to link the principal with a record in the data table
- Last modification timestamp.
- Last modification type, where the valid values are S, N, U, and D.

To review the architecture of the synchronization, see “Overview” on page 44. You create a Partitioned Table SyncSource for the server and a Table SyncSource for the client to access the data in their respective databases.

Configuring the Server Partitioned Table SyncSource

To create an instance of the server Partitioned Table SyncSource, review “Adding a Table SyncSource” on page 47. The procedure is essentially the same, with the following exceptions:

- In Step 1, you select **Partitioned Table SyncSource** in the navigation pane instead of **Table SyncSource**.
- The Edit Partitioned Table SyncSource window has additional parameters; specifically, the **Table Info** area of the Table SyncSource window is replaced with **Partitioning Table** and **Data Table**., and each has a **Load data from db** button for populating the fields, as shown below:



The parameter definitions in “Edit Table SyncSource Parameters” on page 46 apply to this window. An additional **Link field** parameter is defined as follows:

- In the partitioning table, this is the field used to get the records from the data table (the “guid”)
- In the data table, this is the “guid” field (which can be the key field).



A sample configuration is shown below:

Parameter	Value	
Source URI	./customerDB	
Name	customerDB	
Type	text/plain	
JNDI Name datasource	jdbc/fnblbs	
Partitioning Table		
Table name	agent_customer	
Principal field	id_principal	
Link field	id_customer	
Last update timestamp field	update_date	
Last update type field	update_type	
Data Table		
Table name	customer	
Key field	id_customer	
Link field	id_customer	
Last update timestamp field	update_date	
Last update type field	update_type	
Field Mapping	country	country
	street	street
	postal_code	postal_code
	state	state
	name	name
	city	city
Binary fields	None	

Configuring the Client Table SyncSource

The procedure for configuring the client is the same as described in “Configuring the Client Table SyncSource” on page 50.



Appendix A **Supplemental Information**

Topics

- *Resources, page 58*
- *Default Databases, page 59*
- *Install Properties, page 61*
- *Connection Factory Sample, page 62*



Resources

This section lists resources you may find useful.

Related Documentation

This section lists documentation resources you may find useful.

Funambol DS Server Documentation

The following documents form the Funambol DS Server documentation set:

- *Funambol DS Server Administration Guide*: This guide.
- *Funambol DS Server Developer's Guide*: Read this guide to gain an understanding of how to develop extensions to the server.
- *Funambol DS Server Module Development Tutorial*: Read this tutorial to gain an understanding of how to develop, package, install, and test a module.
- *Funambol DS Server Quick Start Guide*: Read this guide to install and run a simple demonstration of synchronizing PIM data using the Funambol DS Server.

Other Resources

This section lists other resources you may find useful.

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



Default Databases

The Funambol DS Server provides the following default databases to which you can synchronize:

NOTE: URIs are case sensitive.

Calendar

Usage: synchronizing calendar data.

URI	MIME-TYPE	Clients
scal	text/x-s4j-sife	All Funambol-developed clients
cal	text/x-vcalendar text/calendar	Most of the common and known clients that are already built into a mobile device.

Contacts

Usage: synchronizing contact data.

URI	MIME-TYPE	Clients
scard	text/x-s4j-sifc	All Funambol-developed clients
card	text/x-vcard text/vcard	Most of the common and known clients that are already built into a mobile device.

Notes

Usage: synchronizing text-based notes.

URI	MIME-TYPE	Clients
snote	text/x-s4j-sifn	All Funambol-developed clients
note	text/plain	Most of the common and known clients that are already built into a mobile device.

Tasks

Usage: synchronizing task data.

URI	MIME-TYPE	Clients
stask	text/x-s4j-sift	All Funambol-developed clients



Briefcase

Usage: synchronizing briefcase data.

URI	MIME-TYPE	Clients
briefcase	Application/*	All Funambol-developed clients



Install Properties

The `<DS_SERVER_HOME>\install.properties` file is the central repository of configuration information that is used by the installation procedure to set up the Funambol DS Server. It is a standard Java properties file that contains the following parameters:

Parameter	Description
context-path	The context path to be used to configure the web container for the Funambol DS Server module. The DS Server will respond to URLs starting with this context path.
dbms	Name of the database where Funambol DS Server tables are created.
jdbc.classpath	Classpath including the JDBC driver for the database, if not included in the system classpath.
jdbc.driver	JDBC driver class.
jdbc.password	Database user password
jdbc.url	JDBC connection URL
modules-to-install	Comma-separated list of Funambol DS Server modules to install. If a module has already been installed, the installation procedure reinstalls it again.
server-name	The server URI that will be specified in SyncML messages. The server will only respond to messages addressed to this URI.



Connection Factory Sample

The following is an example of a simple connection factory that uses a single connection to access to a database.

```
/**
 * Copyright (C) 2003-2006 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 com.funambol.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.
 */
public class SimpleConnectionFactoryImpl implements
com.funambol.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;
}
}
```
