Funambol Web Services Connector Quick Start Guide

Table of Contents

1. Introduction

The purpose of this document is to describe how to manage and administer the Web Services SyncSource component using the SyncAdmin console.

This document is intended to be read by the administration users.

1.1. Related Documents

The following documents are related to this design document:

[1] Funambol Interchange Format
[2] Internet Calendaring and Scheduling Core Object Specification - [RFC 2445]
[3] Funambol Modules Development Guide
[4] Funambol SyncServer 5.0.x Administration Guide

2. Setting Web Services Connector

2.1. Funambol Web Services Connector Installation Procedure

The Funambol Web Services Connector is distributed as a standard Funambol module [3]. The distribution contains the following files:

- funambol-webservices-<major>.<minor>.<buildnumber>.s4j (the module)
- the release notes
- the readme.txt
- this guide

To install the module you have to follow this steps:

1) put the s4j file in the directory

<installation dir>/ds-server/modules

2) modify “install.properties” file adding “funambol-webservices-3.x.x” to the modules list:

modules-to-install=foundation-3.0.1,pdi-3.0.1,pimweb-3.0.1,funambol-db-3.0.1**,funambol-webservices-3.x.x**

3) start installation modules command.

For more details about the Funambol module installation see [4].

2.2. Configuring Web Services SyncSource

To set up Web Services SyncSources go in the Administration Console and check the following tree structure

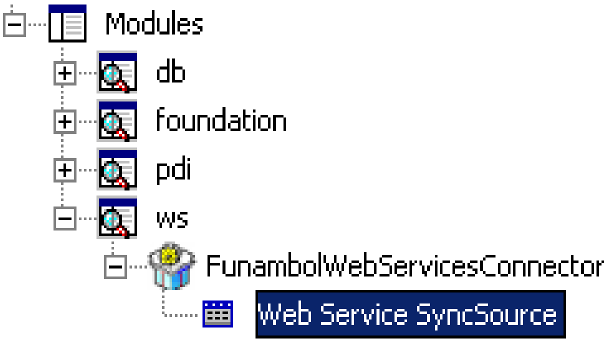


Figure 1: Web Services SyncSource tree

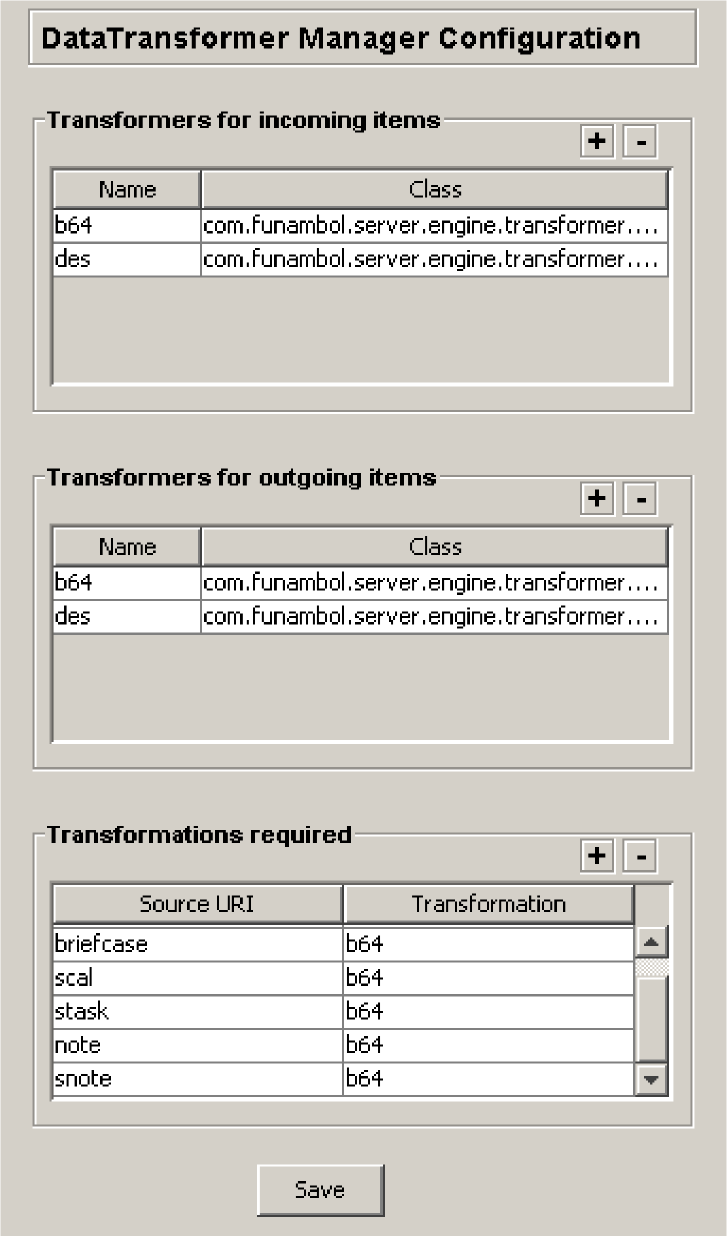
All SyncSources have the following properties:

Property	*Description*
Source URI	The sync source URI [i.e. "./contacts"].
Name	The SyncSource name. [i.e. "contacts"]
Type	The data content type [i.e. "SIF-E, SIF-C, vcard, i-cal, ..."]
Web Service URL	The web service URL [i.e http://localhost:8080/example-ws/services/SyncSourceService]
Enable Filtering	If the web service knows how to handle Funambol filter capabilities, you can enable the filter, if not, disable it.

2.3. Enabling Data Transformation

Some devices communicates in a different language(i.e. encoded, encrypted) with server. Server needs to know about it, you have to set for each sync source that needs data transformation, the specific type of it.

To do so, open Funambol Administration Tool, go to Server settings and in data transformation manager field, press Configure button, then you will see the DataTransforer Manager Configuration panel.



DataTransformer Manager Configuration

Transformers for incoming items

Name	Class
b64	com.funambol.server.engine.transformer....
des	com.funambol.server.engine.transformer....

Transformers for outgoing items

Name	Class
b64	com.funambol.server.engine.transformer....
des	com.funambol.server.engine.transformer....

Transformations required

Source URI	Transformation
briefcase	b64
scal	b64
stask	b64
note	b64
snote	b64

Figure 2: DataTransformer Manager Configuration panel

Here you can set in the 3th table the transformations required for each sync source with the Transformers available in the tables above.

3. Using the Funambol Web Services Connector

The purpose of this chapter is to describe the environment that the Funambol Web Services Connector provides.

3.1. Real Environment

The system architecture of the Funambol Web Services Connector in a real installation is drown in Figure 3

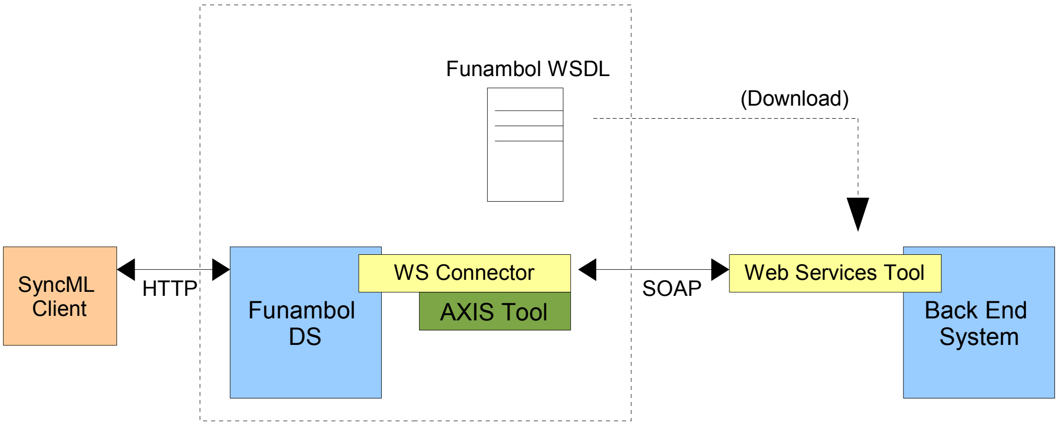


Figure 3: Funambol WS Architecture

Steps:

1) The “Back End Owner” has to:
   1. download the s4j wsdl file
      - The wsdl is automatically generated by the web application provided together with the connector for test purposes (example-ws.war). Deploy the example-ws.war in an application server. The wsdl can be now downloaded from: *http://<your-application-server-host>/example-ws/services/SyncSourceService?wsdl*
   2. create the “Web Service Tool” that exchanges the data with the back end system
2) Funambol Web Service Connector is administered with a dedicated plug-in for the Funambol Server DS Administration tool by which an administrator can create/modify/delete WebServiceSyncSources.

3) When Funanbol Server DS receives a synchronization request from a mobile client addressed to one of the WebServiceSyncSource, it uses the Funambol Web Service Connector to establish connection with a Web Service Tool using SOAP protocol [Simple Object Access Protocol] embedded in Axis Web Services engine.

3.2. Test Environment

The system architecture in a test environment is drawn in Figure 4

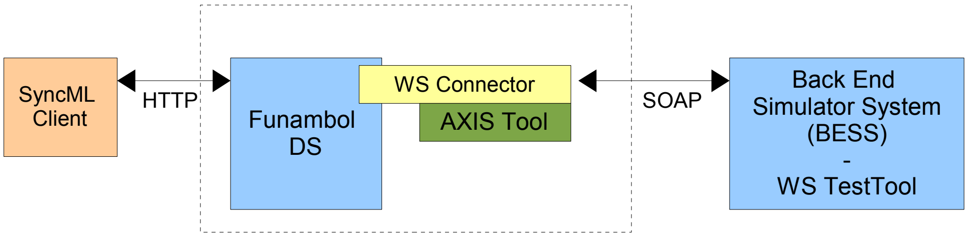


Figure 4: Funambol WS Architecture

Steps:

1) Funambol provides a test web application in order to test the main feature of the WS connector. This tool simulates a customer back end system.
   1. install a tomcat server (you can use the same application server of the Funambol DS)
   2. deploy the “example-ws.war” web application (you can copy the war archive in the “<installation-dis>\webapps”
   3. run the tomcat server in order to run a *Back End Simulator System (BESS)*
   4. *configure the Back End Simulator System (BESS). After installation you should have in “<installation-dis>\webapps\example-ws\” the “config.ini” file where you can set the properties: “format”, “type” and BESS database “location”.*
   - The wsdl of this test web-service can be downloaded from: *http://<your-application-server-host>/example-ws/services/SyncSourceService?wsdl*, and can be used to create other web-services for the Funambol Web Services Connector.

2) Create the *WebServiceSyncSources* Funambol WS Connector (at the moment we can test the contact entity)
   1. create the SyncSource
   2. specify the SIF-C
   3. add the syncsource in the DataTransformation table

3) Run the *Funambol Java GUI Plug-in 3.0.x*
   1. configure the java gui with the guest/guest user
      1. eventually check if in the Administration Tool if the in the “settings” panel there is the default officer (UserProvisioningOfficer.xml)
   2. set the WS SyncSource

3.3. Some basic test

To test web services connector, Funambol provides the *Back End Simulator System (BESS).*

Please note that to reproduce the test cases scenarios in server side, you have to manipulate the items in BESS database (you can find or set its location in *<installation-dir>\webapps\example-ws\config.ini file).*

BESS's functionality:

- If you want the server to send to the client an item as “**created**”, your have to create a file in the database's “new” folder. The name of the file represent the item's key and the content of the file is the items content.
- If you want the server to send an item as “**updated**” to the client, you have to put it in folder “updated” , that you can find in BESS database directory.
- If you want to **delete** an item, you have to put it in the “deleted” folder.
- If an newly created item is sent from client to server, after the sync it will be found in the the database's “new” folder
- If an updated item is sent from client to server, after the sync it will be found in the the database's “updated” folder
- If an item that exist on both client and server is deleted from the client, after sync this will be deleted from the database's “synchronized” folder
- **After a sync the items from the database's “new” and “update” folders must be moved in the “synchronized” folder, otherwise, for the next sync will be considered as “newly created” or “updated”, respectively**.

Test Case 0 – Initialization

- the client and the server are empty
- sync
- on the server will be the directory *<installation-dir>\webapps\example-ws\db*

Test case 1 – new items on JAVAGUI

- Create 2 items on the client
- sync
- check the db directory on the server

note: move the items from “new” to “synchronized” folder

Test case 2 – new items on BESS

- Create 2 items on the server (create a file in the database's “new” folder, i.e. you can copy/paste the client items an then change “firstname”, ”lastname” and “file as”)
- sync
- check the items on the client

Test case 3 – delete the item on BESS

- delete item on the BESS (move the item from “synchronized” to “deleted” folder)
- sync
- check the item list on the client

Test case 4 – delete the item on JAVAGUI

- delete item on the client
- sync
- check the item list on the server and check if the item is deleted from “synchronized” folder

Test case 5 – get All items

- Create 3 items on the server (you can save the used items in a tmp directory and now you can copy it in the “new” folder) - the client is empty
- sync
- the server sends the items on the client

Test case 6 – modify the item on JAVAGUI

- modify item on the client
- sync
- check the modified item on the server in the “updated” folder

Test case 7 – modify the item on BESS

- modify item on the BESS (move from “synchronized” to “updated” folder)
- sync
- check the modified item on the client