Funambol JSON API testing tool user guide

Last modified: October 29, 2008

Table of Contents

1. Introduction

This guide is intended for developers working on implementing the Funambol json API.

Using Funambol json testing tool the developer will be able to test the insert/remove/update and synchronization features of the server side developed api.

2. Installation & configuration

2.1. Installing

Funambol JSON testing tool is distributed as a .zip package, in order to use it you must unzip it, the scripts to run the application are placed under Funambol/json-test-api/bin/ . Executing the startup script will immediately start the tests.

. Linux

In linux systems the zip package can be unzipped using the “unzip” tool: *unzip Funambol-JSON-test-api-1.0.0.zip* . In order to start the tests the user should go th bin directory (cd Funambol/json-test-api/bin/) and run the start script (*./run.sh*)

. Windows

In windows it's possible to use any compression tool that supports “zip” files, when extracted the files the user should locate the “Funambol” directory and then reach the startup scripts directory (*cd Funambol\json-test-api\bin*), in order to run the tests the user should execute the start script (*run.cmd*)

2.2. Configuring

Under the “config” dir you will find two files: one for logging configurations (log4j.xml) and the other for server related properties.

2.2.1. Logging

In the logging file it's possible to configure the log that will appear in the console and will be recorded in the log file. The configuration follows the log4j standard : http://logging.apache.org/log4j/1.2/

. raising verbosity

By default the log level is set to info, in this mode the user will only see the tests being executed and the errors(if any).

```
<category name="funambol.json-test-api">
       <priority value="info"/>
       <appender-ref ref="json-test-api-file"/>
   </category>
   <category name="funambol.json-test-api">
       <priority value="info"/>
       <appender-ref ref="json-test-api-console"/>
   </category>
```

While developing the server side json API it's useful to set the log level to “trace”

```
<category name="funambol.json-test-api">
    <priority value="trace"/>
```

```
        <appender-ref ref="json-test-api-file"/>
    </category>
    <category name="funambol.json-test-api">
        <priority value="trace"/>
        <appender-ref ref="json-test-api-console"/>
    </category>
```

This way you will be able to see all the data exchanged with the server(sessionid's, request bodys request headers etc)

2.2.2. Server properties

Here you will be able so set up your server settings: server url, username and password

```
server-uri=http://localhost:8080/jsonsync
username=your_username
password=your_password
method=standard
```

. Method property

The method property in the server.properties file is used in order to support two different server implementations. This happens because some backends might not be able to differentiate between new and updated items, if this is the case both new and updated items will be returned as new items and therefore method property should be set to “alternate” otherwise “standard” should be used.

3. Tests

The Funambol JSON tool ships with 15 default tests per source, in order to run this tests you just have to configure you server definitions and run the init script.

NOTE: Running all the tests (60 considering 15 per source) will take 25-30 minutes. See bellow how to turn off tests, if you want to run just a sub-set of the tests.

3.1. Structure

All the tests are placed under “testcases” directory, and all directories inside “testcases” will be consider as a source in server side (eg. testcases/contacts , testcases/notes, etc). Any directory inside a source will be considered the test name. Inside each directory representing a test name are the test files.

testcases

- contacts *[this is the syncsource name server side]*
  - integrity1
    - operations.properties
    - item1.json
    - item2.json
  - sync1
    - operations.properties
    - item1.json
    - item1_added.json
- events
  - 0001_test
    - operations.properties
    - item1.json
    - item2.json

the “item*.json” files contain a json item that should have the item type of the source beeing tested, the “item*_added”, “item*_updated” and “item*_removed” contain a item key that can be used as a reference for some operations(more info bellow).

3.2. Turning off tests

If you don't have a complete server side implementation of the JSON API you will need to turn off some tests, you can do this by changing the source our test name from “sourcename” to “_sourcename” in this case the source/test wont be executed.

3.3. Writing new tests

There are the 2 ways to reference items: <instruction>:item1 and <instruction>:item1_added/<instruction>:item1_updated the first one refers to a json representation of an item, the second refers to a file containing a key(with is created when an item is inserted or updated)

Instruction	Description	Syntax
login	performs the login in the json server	login
logout	logs out from the json server	logout
beginSync	tell the server that a sync session is going to begin	begynSync:<syncType> allowed types : “two-way”, “full”, “one-way-from-server”, “one-way-from-client”, “refresh-from-server”, “refresh-from-client”
endSync	tell the server that a sync session had ended.	endSync
getAllSyncItemKeys	get all the keys for the items in the server.	getAllSyncItemKeys
getDeletedSyncItemKeys	get all the keys of the deleted items since a specified time.	getDeletedSyncItemKeys
getUpdatedSyncItemKeys	get all the keys of the updated items since a specified time.	getUpdatedSyncItemKeys
getNewSyncItemKeys	get all the keys of the new items since a specified time.	getNewSyncItemKeys
getSyncItemFromId	get the item associated with a specific key.	GetSyncItemFromId:<key_file>
removeSyncItem	remove the item with a specific key.	removeSyncItem:<key_file>
addSyncItem	insert a specified item in the server and return it with a key.	addSyncItem:<item_file>
updateSyncItem	update a specified item in the server for a specific key.	updateSyncItem:<key_file>:<item_file>
getSyncItemKeysFromTwin	search for application defined equivalent items for a specified item. The key of the second parameter must be one of the returned twin items	getSyncItemKeysFromTwin:<item_fike:<key_file>
compareClientServer	compares the item on client with the item on the server the item returned by the server must have all the same keys and values as the item on client side	compareClientServer:<key_file>:<item_file>
deleteAllItems	Deletes all the items server side, for the source being tested	deleteAllItems

All the commands described could be inserted in the “operation.properties” file that must be inside any test directory, also all the resources needed during the test (.json files) should be present in this directory.

4. Running the tests

As stated before In order to start the tests, the startup scripts under bin/ need to be executed. According to the log level the user will see only the test names and error or all the communications.

NOTE: in order to have correct synchronization results the testing framework MUST run on the same machine as the one where the testing API is running.