TigaseDoc

Release 0.1

Tigase, Inc.

Jun 23, 2023

CONTENTS

Welcome to Tigase TTS-NG guide

CHAPTER

ONE

OVERVIEW

TTS-NG is a NextGeneration version of Tigase TestSuite intended to run automated functionality tests. It's based on TestNG framework (similar to junit, serving as a basic test framework) and JaXMPP library (to provide client-side functionality)

CHAPTER

TWO

RUNNING TTS-NG

2.1 Basics

As TTS-NG is based on the known framework running the test in the basic form is simply a matter of executing particular test case class(es). The easiest way to do that is to use Maven - you can either run all tests

```
mvn test
```

or for particular package/class

```
mvn test -Dtest=tigase.tests.util.*
mvn test -Dtest=tigase.tests.util.RetrieveVersion
```

This will run particular test(s) using server details (details about server configuration are described in *TTS-NF Configuration*). It's possible to override any of the server configuration options from command line by simply appending `-D<property_name>=<property_value>` to the above options.

2.2 Using test-runner script

For the convenience of running automatic tests it's recommended to use `$ ./scripts/tests-runner.sh` BASH script, which automates whole process of setting up the server (and database, with correct configuration and components being enabled), running desired tests as well as generating summary page.

Running it without any parameter will print a help with description of the possible options:

```
$ ./scripts/tests-runner.sh
Can't find settings file: ./scripts/tests-runner-settings.sh using defaults
Tigase server home directory: tmp/tigase-server
Version: 7.2.0-SNAPSHOT-b4823
Output dir: files/test-results/7.2.0-SNAPSHOT-b4823
Run selected or all tests for defined server
----
  --all-tests [(database)] :: Run all available tests
          (database) is an array, if database is missing tests will be run against all␣
↪configured ones
                      possible values: derby, mysql, postgresql, sqlserver, mongodb
  --custom <package.Class[#method]]> [(database)] :: Run defined test, accepts wildcards,
↪ eg.:
          --custom tigase.tests.util.RetrieveVersion
  --help :: print this help
```

(continues on next page)

(continued from previous page)

```
----
  Special parameters only at the beginning of the parameters list
  --debug|-d                    Turns on debug mode
  --skip-rebuild-tts|-srb       Turns off rebuilding TTS-NG and only runs already build␣
↪tests
  --skip-summary-page-get|-sp   Turns off automatic generation of Summary Page
  --download-latest|-dl         Turns on downloading latest Tigase Server release
  --reload-db|-db               Turns on reloading database
  --start-server|-serv          Turns on starting Tigase server
-----------
  Other possible parameters are in following order:
  [server-dir] [server-ip]
```

Majority of those are self-explanatory. By default, the script will only rebuild all test cases, run them and then generate Summary page and place the output in the `files` subdirectory (if not configured otherwise, see *Test Runner settings*).

After `--all-tests` and `--custom <test_case>` options it's possible to specify a space-delimited list of databases for which tests should be run (they will match database and server IPs defined in *Test Runner settings*).

By appending following options

- `-dl` (or full variant: `--download-latest`) - you will instruct the script to download latest version of the server and unpack it to `tmp/tigase-server` sub directory
- `-db` (or full variant: `--reload-db`) - you will instruct the script to download prepare the configured database (drop it if it exists and load current schema)
- `-serv` (or full variant: `--start-server`) - you will instruct the script start the server from the configured location and utilize recommended settings (located in `src/test/resources/server/etc/init.properties`)

2.2.1 Test Runner settings

It's possible to adjust default Test Runner settings by copying distribution settings and adjusting it to your needs:

```
$ cp scripts/tests-runner-settings.dist.sh scripts/tests-runner-settings.sh
```

Following configuration options are available:

- database configuration:
  - `db_name` - name of the database to be created,
  - `db_user`, `db_pass` - name and password of the basic user, which will be used by Tigase,
  - `db_root_user`, `db_root_pass` - name and password of the database *root* user, which will be used to create all necessary databases and grant roles.
- databases selection
  - `DATABASES=("derby" "mysql" "postgresql" "sqlserver" "mongodb")` - a list of databases which will be tested
  - `DATABASES_IPS=("127.0.0.1" "127.0.0.1" "127.0.0.1" "sqlserverhost" "127.0.0.1")` - a list of IPs of the databases which will be used while testing particular database, i.e. if you have a list of 3 databases in `DATABASE` for each item/index respective item from this array will be used (so for first item `derby` from `DATABASES`, first item from `DATABASES_IPS` will be used;

  - IPS=("127.0.0.1" "127.0.0.1" "127.0.0.1" "127.0.0.1" "127.0.0.1") - a list of IPs of the servers which will be used while testing particular database, i.e. if you have a list of 3 databases in DATABASE for each item/index respective item from this array will be used (so for first item derby from DATABASES, first item from IPS will be used as a server IP to which connection will be made.

- server_timeout=15 - a timeout in seconds used to delay subsequent actions/tasks (for example to allow server proper startup)
- server_dir="../tigase-server/server" - server directory which will be used to reload database (if enabled) and start the server (if enabled)
- tigase_distribution_url="https://build.tigase.net/nightlies/dists/latest/tigase-server-dist-enterprise.tar.gz" - link which will be used to download latest release of Tigase XMPP Server
- memory configuration for normal tests: MS_MEM=100 and MX_MEM=1000 (minimum and maximum JVM heap size respectively) and *low memory tests*: SMALL_MS_MEM=10, SMALL_MX_MEM=50 (minimum and maximum JVM heap size respectively)
- ROOT_DIR=./files/ - a root directory where tests results will be stored and where summary page will be placed (in not disabled)

CHAPTER

THREE

TTS-NG CONFIGURATION

3.1 Test Configuration

In order to allow connecting to any server that we want to test we need to provide server details and such details are configured in `src/test/resources/server.properties` file. Through that file it's possible to configure:

- list of configured domains/VHosts (at least one entry is needed):

  `server.domains=localhost`

- define which domain should be used for Two-way-TLS test (such domain must be configured to use certificate from `certs/root_ca.pem` file)

  `server.client_auth.domain=a.localhost`

- a list of cluster nodes to which test should connect (for single node setup leave only one node; at least one entry is needed)

  `server.cluster.nodes=localhost`

- details of the admin account (JID will be created from below username and domain name or, if details are not provided, `admin` name will be used with first item from `server.domains`; default password will be the same as used user-name)

```
test.admin.username=admin
test.admin.domain=localhost
test.admin.password=admin
```

- HTTP-API component
  - API key used to access service:

    `test.http.api-key=test-api-key`

  - port under which service is listening:

    `test.http.port=8088`

- it's possible to override default ports for WebSocket nad BOSH connections: `test.ws.port=5290` and `test.bosh.port=5280` respectively.

- Tigase XMPP Server e-mail monitoring configuration can be set-up using following entries:

```
imap.server=localhost
imap.username=xygoteheyd
imap.password=medkbreqeppbemzinmtu
```

3.2 Recommended server configuration

In case of running tests against Tigase XMPP Server, there are a couple of required configuration changes (that differ from the default server configuration). Recommended server configuration is located under following location: `src/test/resources/server/etc/init.properties`. Please refer to the documentation for the explanation of the settings used if in doubt.

3.3 Test-NG configuration

All tests are grouped in *suites*. All main *suites* are defined in `pom.xml` file in `maven-surefire-plugin` plugin configuration:

```
<suiteXmlFiles>
    <suiteXmlFile>src/test/resources/testng_server.xml</suiteXmlFile>
    <suiteXmlFile>src/test/resources/testng_muc.xml</suiteXmlFile>
    <suiteXmlFile>src/test/resources/testng_pubsub.xml</suiteXmlFile>
    <suiteXmlFile>src/test/resources/testng_http_api.xml</suiteXmlFile>
    <suiteXmlFile>src/test/resources/testng_archive.xml</suiteXmlFile>
    <suiteXmlFile>src/test/resources/testng_custom.xml</suiteXmlFile>
    <suiteXmlFile>src/test/resources/testng_jaxmpp.xml</suiteXmlFile>
</suiteXmlFiles>
```

For the actual semantics of the suite xml files please refer to Test-NG documentation.

CHAPTER

FOUR

BASIC API OVERVIEW

4.1 API Overview

All test cases should extend `tigase.tests.AbstractTest` class, which offers a couple of handy methods that makes writing a test case easier. The most useful and basic are following:

- handling user/admin account/connection:
  - `createAccount()` - returns `AccountBuilder` allowing adjusting settings of particular user account;
  - `getAdminAccount()` and `getJaxmppAdmin()` - get either `Account` object related to admin account or `Jaxmpp` object directly.
  - `removeUserAccount(Jaxmpp jaxmpp)` - delete particular user account
- vhost management:
  - `addVhost(Jaxmpp adminJaxmpp, String prefix)`
  - `removeVhost(Jaxmpp adminJaxmpp, String VHost)`
- user basic xmpp functionality methods:
  - `changePresenceAndWait(Jaxmpp from, Jaxmpp to, Presence.Show p)`
  - `sendAndFail(Jaxmpp from, Jaxmpp to)`
  - `sendAndWait(Jaxmpp j, IQ iq, AsyncCallback asyncCallback)`
  - `sendAndWait(Jaxmpp from, Jaxmpp to, Message message)`
  - `sendAndWait(Jaxmpp from, Jaxmpp to, String message)`
  - `testSendAndWait(Jaxmpp from, Jaxmpp to)`
  - `testSubscribeAndWait(Jaxmpp from, Jaxmpp to)`
- utility and configuration
  - `getDomain()` - returns random domain from the list of available domains from the list of available domains (see *Test Configuration*)
  - `getDomain(int i)` - returns i-th domain from the list of available domains from the list of available domains (see *Test Configuration*)
  - `getInstanceHostname()` - returns random machine hostname from the list of available hostnames from the list of available domains (see *Test Configuration*)
  - `getInstanceHostnames()` - returns i-th machine hostname from the list of available hostnames from the list of available domains (see *Test Configuration*)
  - `getApiKey()` - returns configured HTTP-API key

  - `getHttpPort()` - returns configured HTTP-API port
  - `getBoshURI()` - returns BOSH URI based on configuration in *Test Configuration*
  - `getWebSocketURI()` - returns WebSocket URI based on configuration in *Test Configuration*

In addition `tigase.tests.utils.AccountBuilder` class allows:

- `setUsername(String username)` - set username/local part name of particular account
- `setDomain(String domain)` - set domain name of particular account
- `setPassword(String password)` - set password of particular account
- `setEmail(String email)` - set e-mail address of particular account
- `setLogPrefix(String logPrefix)` - allows to customize log prefix for log entries for this particular account
- `setRegister(boolean register)` - specify whether account should be registered automatically

For the purpose of testing delayed response `tigase.tests.Mutex` can be used: * `waitFor(long timeout, String... items)` - instruct Mutex to wait for the particular items during configured `timeout` * `notify(String... itemName)` - upon receiving desired response notify Mutex about it * `isItemNotified(String item)` - can be used to verify whether particular item was received (useful in asserts)

4.2 Creating simple test

As an example we will use `src/test/java/tigase/tests/ExampleJaxmppTest.java` test case. Followings steps should be taken:

1. extend `AbstractTest` class:

```
public class ExampleJaxmppTest extends AbstractTest {}
```

2. create test method and annote it with `@Test`. In addition specify test group and provide short description

```
@Test(groups = { "examples" }, description = "Simple test verifying logging in by␣
↪the user")
public void SimpleLoginTest() {}
```

3. create an Account object, configure it, later build Jaxmpp object from it and connect to the server

```
Account userAccount = createAccount().setLogPrefix("test_user" ).build();
Jaxmpp jaxmpp = userAccount.createJaxmpp().build();
jaxmpp.login( true );
```

4. check whether the connection was successful

```
assertTrue(createJaxmpp.isConnected(), "contact was not connected" );
```

4.3 Adding test to suite

As described in *TTS-NF Configuration*, each test case must be included in Test Suite configuration.

1. create new xml file under `src/test/resources/`, for example `example.xml`
2. in the created xml file add new test case as follows, creating new Suite (specifying name) with a list of tests (specifying names), and each test can contain multiple classes (for details please refer to TestNG documentation)

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name="Tigase Various Tests" verbose="1">
    <test name="Example Tests">
        <classes>
            <class name="tigase.tests.ExampleJaxmppTest" />
        </classes>
    </test>
</suite>
```

3. include created xml file in the Test Suite (see *TTS-NF Configuration*)