Title	Version	Version date
AutoTester documentation	1.1	31.01.2015

AutoTester

Documentation

Petr Rašek

Document History and Version Overview:				
Version	Date	Author	Page(s)	Remarks
1.0	04.01.2015	Petr Rašek	48	Initial revision
1.1	31.01.2015	Petr Rašek	53	Added LOV functions, test data repository, maintenance

Table of Contents

1. Introduction 5
   1.1 Purpose 5
   1.2 Scope 5
   1.3 Download 5
2. Installation 6
   2.1 Operating system 6
   2.2 Java 6
   2.3 Application server 6
      2.3.1 Maintenance script 6
   2.4 Database 7
   2.5 Application 7
      2.5.1 Glassfish configuration 7
      2.5.2 Application configuration 8
   2.6 Deployment 8
3. User manual 10
   3.1 User interface 10
   3.2 Test case workflow 15
   3.3 Scheduled execution 15
4. Scripting 17
   4.1 Introduction 17
   4.2 Examples 17
      4.2.1 CLF 17
      4.2.2 DWH 24
      4.2.3 AMX 27
      4.2.4 CNH 28
      4.2.5 AP 30
      4.2.6 WS 31
   4.3 Functions 36
      4.3.1 Common 36
      4.3.2 CLF 37
      4.3.3 DWH 39
      4.3.4 AMX 40

4.3.5 AP 41
4.3.6 CNH 41
4.3.7 WS 41
5. System documentation 43
5.1 Architecture 43
5.1.1 Deployment 43
5.1.2 Test case administration 44
5.1.3 Test case execution 45
5.2 Class diagrams 46
5.2.1 Utilities 46
5.2.2 Beans 47
5.2.3 Database model 48
5.2.4 Customer structure 49
5.2.5 Services 50
5.2.6 Business services 50
5.2.7 LOV 52
5.2.8 Properties 52
5.3 Maintenance 53

1. Introduction

1.1 Purpose

AutoTester is application designed for preparation, execution and maintenance of automated test cases.

1.2 Scope

The scope of this document is to

- Describe installation of the AutoTester application
- Describe the application usage
- Describe scripting capabilities
- Describe system design

1.3 Download

The application is distributed under GNU GPL v2 license.

Source code is available on https://github.com/lynusbowman/AutoTester/tree/1.0

2. Installation

2.1 Operating system

The AutoTester application is implemented on J2EE technology so it can be deployed on any usual operating system (verified on Windows and Linux). Following manual describes detailed installation procedure for Linux (target OS in TMCZ).

2.2 Java

Install Java 7 development kit (i.e. JDK 1.7.0_80).

Extract the archive to folder `/usr/java`.

Set environment variables PATH and JAVA_HOME which point to Java `bin` folder.

```
export PATH=$PATH:/usr/java/jdk1.7.0_80/bin
```

```
export JAVA_HOME=/usr/java/jdk1.7.0_80/bin
```

2.3 Application server

Install Glassfish 4.1 application server.

Extract the archive to folder `/opt/glassfish`

Create maintenance script `/etc/init.d/glassfish` and set execute permission (`chmod a+x`).

Add the script to RC `update-rc.d glassfish defaults`

2.3.1 Maintenance script

The script is used to start, stop or restart application server and database.

```
#! /bin/sh

GLASSFISHPATH=/opt/glassfish/bin

case "$1" in

start)

echo "starting glassfish from $GLASSFISHPATH"
```

```
sudo $GLASSFISHPATH/asadmin start-domain domain1

sudo $GLASSFISHPATH/asadmin start-database

;;

stop)

echo "stopping glassfish from $GLASSFISHPATH"

sudo $GLASSFISHPATH/asadmin stop-database

sudo $GLASSFISHPATH/asadmin stop-domain domain1

;;

restart)

$0 stop

$0 start

;;

*)

echo $"usage: $0 {start|stop|restart}"

exit 3

;;

esac

:
```

2.4 Database

Glassfish contains embedded Derby database which is used as test case repository. Separate installation is not needed.

2.5 Application

2.5.1 Glassfish configuration

Configure JDBC connection pool for Derby database.

Configuration file is located in `/opt/glassfish/glassfish/domains/domain1/config/domain.xml`.

Add following text to element `resources`

```
    <jdbc-connection-pool datasource-classname="org.apache.derby.jdbc.ClientDataSource40"
name="TestPool" res-type="javax.sql.DataSource">

      <property name="TraceFileAppend" value="false"></property>

      <property name="SecurityMechanism" value="4"></property>
```

```
    <property name="ConnectionAttributes" value="create=true"></property>
    <property name="User" value="APP"></property>
    <property name="DatabaseName" value="TEST"></property>
    <property name="Ssl" value="off"></property>
    <property name="RetrieveMessageText" value="true"></property>
    <property name="LoginTimeout" value="0"></property>
    <property name="ServerName" value="localhost"></property>
    <property name="TraceLevel" value="-1"></property>
    <property name="PortNumber" value="1527"></property>
    <property name="Password" value="APP"></property>
  </jdbc-connection-pool>
  <jdbc-resource pool-name="TestPool" jndi-name="jdbc/TestDS"></jdbc-resource>
```

Add following test to element `servers`

```
<resource-ref ref="jdbc/TestDS"></resource-ref>
```

2.5.2 Application configuration

Configure application log folder in file `log4j2.xml`

```
<Property name="log-path">/home/lynus/private/autotester/log</Property>
```

Configure test case folder in file `config.properties`

```
testCasesPath=/home/lynus/private/autotester/tests/
```

2.6 Deployment

Make new build `AutoTester-1.0.war` with updated configuration files.

Start application server (default admin port 4848, default application port 8080) and database (default port 1527). Try to ping connection pool.

Copy war file `AutoTester-1.0.war` to `/opt/glassfish/glassfish/domains/domain1/autodeploy/AutoTester.war`

Open URL http://localhost:8080/AutoTester

Check `server.log` (located in `/opt/glassfish/glassfish/domains/domain1/logs`) and `app.log` (located in log4j2 folder) for exceptions.

Check database - connection string:`jdbc:derby://localhost:1527/TEST`, user `APP`, pass `APP`. Following tables are created `TEST_CASE_GROUP`, `TEST_CASE`, `TEST_RUN`, `TEST_RUN_LOG`.

3. User manual

3.1 User interface

Application has web graphical interface (see fig 1, 2).

Each user control is explained in tab 1.

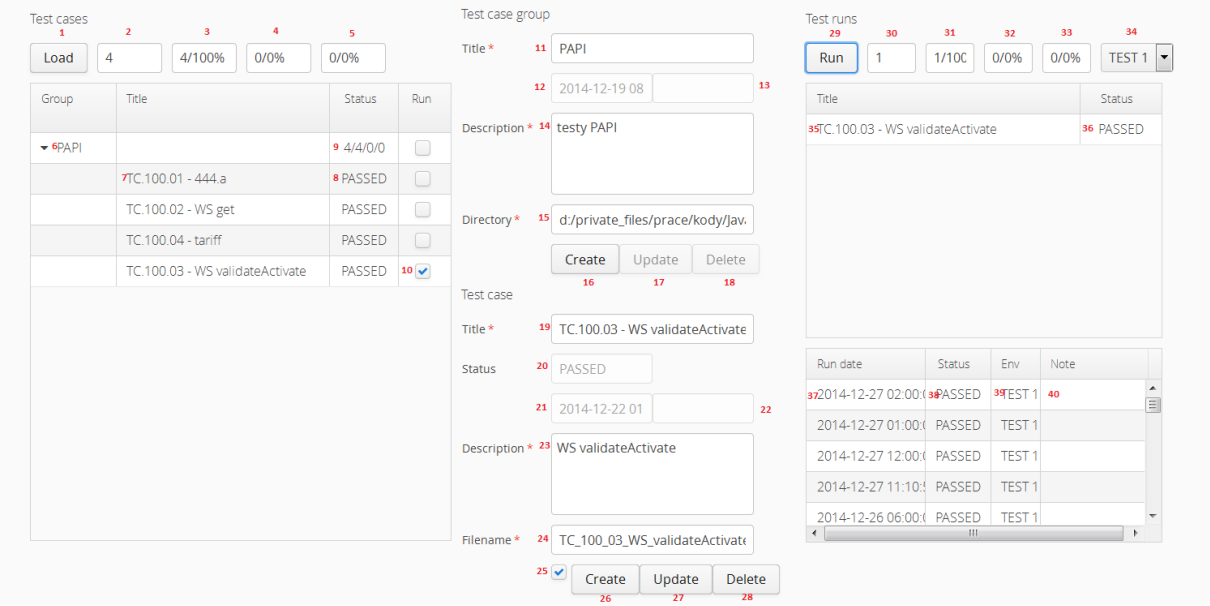

Figure 1: GUI Test cases

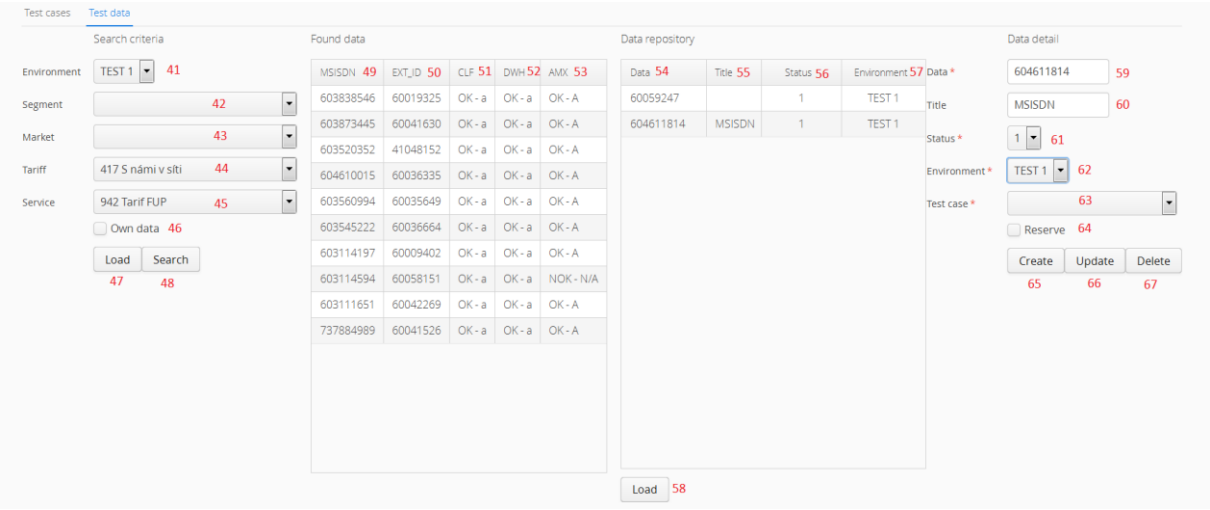

FIgure 2 - GUI Test data

Control ID	Control title	Description
1	Load test cases	• Load all test cases and groups from DB • Populate table and counters (controls 2-5) • Clear test runs
2	Total test cases	• Count of test cases in DB
3	Passed test cases	• Count of passed test cases in DB (including percentage)

4	Failed test cases	• Count of failed test cases in DB (including percentage)
5	Not run test cases	• Count of not run test cases in DB (including percentage)
6	Test case group title	• Test case group title • Form Test case group is pre-filled after cell click event
7	Test case title	• Test case title • Forms Test case group and Test case after pre-filled after cell click event
8	Test case status	• Test case status • Options PASSED, FAILED, NOTRUN
9	Group counters	• Test case status counters within group • Total/Passed/Failed/Not run
10	Run test case	• Add/Remove test case to/from test runs table • Checkbox in group row adds/removes all test cases within group
11	Group title	• Test case group title, mandatory, stored in column TITLE
12	Group create date	• Test case group create date, stored in column CREATE_DATE
13	Group modify date	• Test case group modify date, stored in column MODIFY_DATE
14	Group description	• Test case group description, mandatory, stored in column DESCRIPTION
15	Group directory	• Test case group directory, mandatory, stored in column DIRECTORY • The field is pre-filled with configured property testCasesPath
16	Create group	• Create test case group • Insert record to table TEST_CASE_GROUP and create directory on filesystem • Mandatory fields Title, Description, Directory
17	Update group	• Update test case group • Update record in table TEST_CASE_GROUP and directory on filesystem • Available fields Title, Description, Directory
18	Delete group	• Delete test case group • Delete record in table TEST_CASE_GROUP and directory on filesystem • Test cases within group are also deleted
19	Case title	• Test case title, mandatory, stored in column TITLE
20	Case status	• Test case status, stored in column STATUS • Options PASSED, FAILED, NOTRUN
21	Case create date	• Test case create date, stored in column CREATE_DATE
22	Case modify date	• Test case modify date, stored in colum MODIFY_DATE
23	Case description	• Test case description, mandatory, stored in

		column DESCRIPTION
24	Case filename	• Test case filename, mandatory, stored in column FILENAME • The field is pre-filled with configured property testCaseSuffix
25	Auto test case	• Test case will be executed automatically by scheduler, stored in column AUTO
26	Create case	• Create test case • Insert record to table TEST_CASE an create file on filesystem • Mandatory fields Title, Description, Filename, Test case group Title
27	Update case	• Update test case • Update record in table TEST_CASE and file on filesystem • Available fields Title, Description, Filename, Auto test case, Test case group
28	Delete case	• Delete test case • Delete record in table TEST_CASE and file on filesystem • Test case history is also deleted
29	Run test cases	• Run test cases in table on selected environment • Populate counters (controls 30-33)
30	Total test cases	• Count of test cases
31	Passed test cases	• Count of passed test cases
32	Failed test cases	• Count of failed test cases
33	Not run test cases	• Count of not run test cases
34	Environment	• Test environment • Options TEST1, TEST2
35	Case title	• Title of run test case
36	Case status	• Status of run test case
37	Case run date	• Date of test case run
38	Case run status	• Status of test case run
39	Case run environment	• Enviroment of test case run
40	Case run note	• Note of test case run • It is editable in DB only
41	Search environment	• Search data on given environment • Options TEST1, TEST2
42	Search segment	• Search data with given segment • Filled by Load button
43	Search market	• Search data with given market • Filled by Load button
44	Search tariff	• Search data with given tariff • Filled by Load button
45	Search service	• Search data with given service • Filled by Load button
46	Search own data	• Search only reserved data owned by app • If not checked search free data

47	Load Lovs	• Fill Segment, Market, Tariff, Service listboxes
48	Search data	• Search data in CLF according to set criteria • Data are checker also in DWH, AMX • Populate table Found data
49	MSISDN	• Data MSISDN
50	EXT_ID	• Data EXTID_SU
51	CLF status	• Data status in CLF • OK if active
52	DWH status	• Data status in DWH • OK if active, NOK if not active or missing
53	AMX status	• Data status in AMX • OK if active, NOK if not active or missing
54	Data	• Data ID, stored in column DATA_ID • Form Data detail pre-filled after cell click event
55	Data title	• Data title, stored in column TITLE
56	Data status	• Data status, stored in column STATUS • Options 1, 0 (valid, invalid)
57	Data environment	• Data environment, stored in column ENVIRONMENT • Options TEST1, TEST2
58	Load test data	• Load all test data from DB • Populate table Data repository • Fill Test case listbox
59	Data	• Data ID, mandatory, stored in column DATA_ID
60	Data title	• Data title, stored in column TITLE
61	Data status	• Data status, mandatory, stored in column STATUS • Options 1, 0
62	Data environment	• Data environment, mandatory, stored in column ENVIRONMENT • Options TEST1, TEST2
63	Data test case	• Data test case, mandatory, stored in column TEST_CASE_FK • Filled by Load button
64	Reserve data	• Reserve data in DB after creation
65	Create test data	• Create test data • Insert record to table TEST_DATA • Insert record to table DATA_RESERVATION (QAP DB)if checked • Mandatory fields Data, Status, Environment, Test case
66	Update test data	• Update test data • Update record in table TEST_DATA • Available fields Data, Title, Status, Environment, Test case
67	Delete test data	• Delete test data • Delete record in table TEST_DATA

Table 1 - User controls

3.2 Test case workflow

Typical application usage is displayed on fig 2.

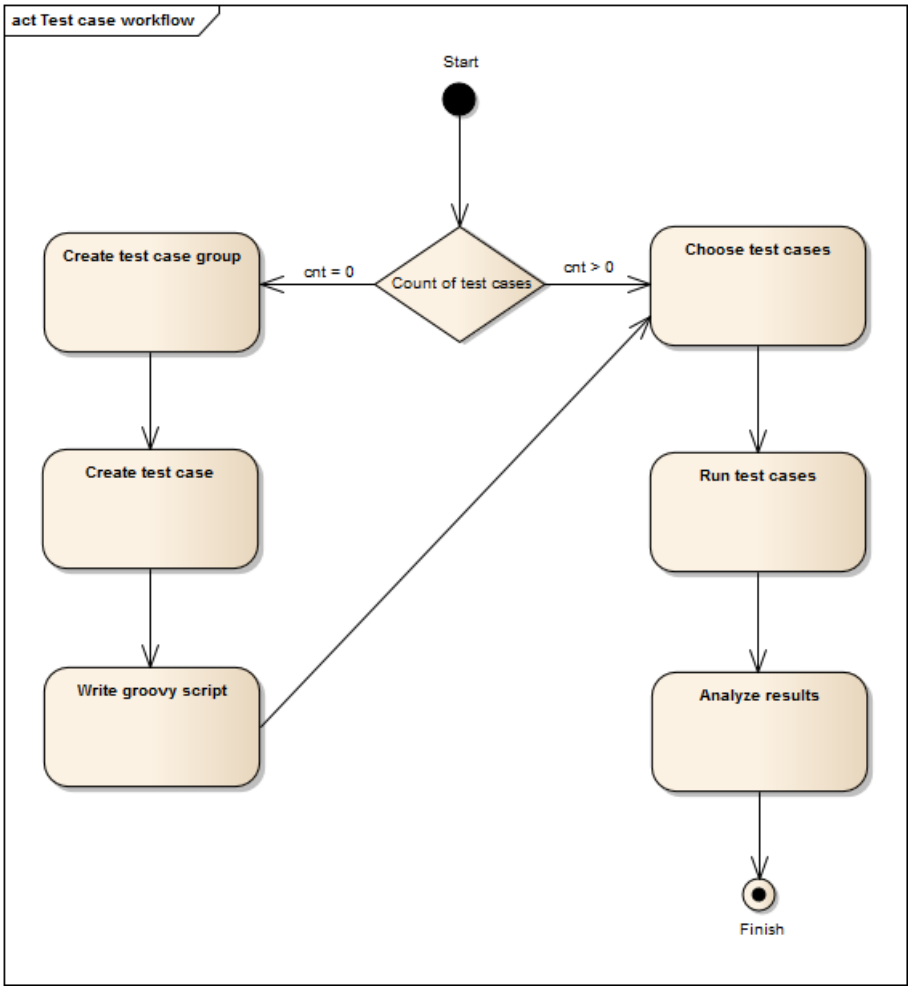

Figure 3 - Test case workflow

3.3 Scheduled execution

Test cases which are markered with checkbox Auto test case are executed automatically by scheduler (currently configured at 17:15).

The result is reported via email (currently sent to petr.rasek@t-mobile.cz), see fig 3.

AUTOTESTER report

autotester@t-mobile.cz

Sent: ne 28.12.2014 17:15

To: Rašek Petr

This is automatic report for executed test cases

Status report

Type	Count	Percentage
Total	4	
Passed	4	100%
Failed	0	0%
Not run	0	0%

Test case report

Group	Title	Status
PAPI	TC.100.01 - 444.a	PASSED
PAPI	TC.100.02 - WS get	PASSED
PAPI	TC.100.04 - tariff	PASSED
PAPI	TC.100.03 - WS validateActivate	PASSED

Figure 4 - Email report

4. Scripting

4.1 Introduction

Test scripts are written in Groovy language which runs on JVM so it can easily cooperate with Java. AutoTester application contains functions which facilitate testing of TMCZ applications.

Covered applications:

- Clarify
  - operates with QCLF DB
  - covers customer structure, services, PAPI, LOVs
- DWH
  - operates with QDTW DB
  - covers customer structure (in ODS), services (in ODS, MTX), LOVs (in ODS and replication to QAP)
- Billing
  - operates with QAMECU DB
  - covers customer structure, offers on SU level, LOVs
- DTS
  - operates with QAP DB
  - covers DTS tables designed for any consumer (VCC, CRM applications)
- Notifications
  - operates with QCNH DB
  - covers notifications triggered by any application which uses CNH
- Business services
  - operates with SA web service endpoint
  - covers operations: get, activate, modify, deactivate (including validation)

4.2 Examples

4.2.1 CLF

4.2.1.1 Customer structure

Script TC_EX_CLF_01_Customer_structure.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")
```

```
// setup
MSISDN = "603847346"
util.clf.connect(env)

// get customer hierarchy
hier = util.clf.getCustHierarchy(MSISDN)
CU = util.clf.getCU(MSISDN)
LE = util.clf.getLE(MSISDN)
OU = util.clf.getOU(MSISDN)
EU = util.clf.getEU(MSISDN)
BA = util.clf.getBA(MSISDN)
SU = util.clf.getSU(MSISDN)

// check hierarchy
assert util.check("CU", hier.getCU().getExtID(), CU) == 1
assert util.check("LE", hier.getLE().getExtID(), LE) == 1
assert util.check("OU", hier.getOU().getExtID(), OU) == 1
assert util.check("EU", hier.getEU().getExtID(), EU) == 1
assert util.check("BA", hier.getBA().getExtID(), BA) == 1
assert util.check("SU", hier.getSU().getExtID(), SU) == 1

// bilcycle
BC = util.clf.getBillCycle(MSISDN)
assert util.check("BC", BC.getID(), 52) == 1

// segment
segment = util.clf.getSegment(MSISDN)
assert util.check("Segment", segment.getTitle(), "LE") == 1

// market
market = util.clf.getMarket(MSISDN)
assert util.check("Market", market.getTitle(), "GSM") == 1

// tariff
tariff = util.clf.getTariff(MSISDN)
assert util.check("Tariff", tariff.getID(), 204) == 1

// contract duration
contr = util.clf.getContractDuration(MSISDN)
assert util.check("Contract duration", contr.getTitle(), "Indefinite") == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0
```

```
}
```

4.2.1.2 Services

Script TC_EX_CLF_02_Services.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.clf.connect(env)

// SU services
SU = util.clf.getSU(MSISDN)
suSrv = util.clf.getSUServices(SU)
suSrv.each { srv -> util.log(srv.getID()) }

// service 897
srv = util.clf.getSUService(SU, 897)
assert util.checkNok("Instance", srv.getInstanceID(), 0) == 1
assert util.check("Status", srv.getStatus(), "a") == 1
assert util.checkNok("Param 1531", srv.getParam(1531), "0") == 1
assert util.checkLike("Param 1534", srv.getParam(1534), "Z1-2") == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.1.3 PAPI get

Script TC_EX_CLF_03_PAPI_get.goovy

```
// initialization
```

```
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.clf.connect(env)

// PAPI 506, service deactive
util.log("PAPI 506")
SU = util.clf.getSU(MSISDN)
PAPI = util.clf.callPAPI("SU", SU, 506, "G")
assert util.check("Status", PAPI.getStatus(), "d") == 1

// PAPI 430, service active
util.log("PAPI 430")
PAPI = util.clf.callPAPI("SU", SU, 430, "G")
assert util.check("Status", PAPI.getStatus(), "a") == 1
assert util.check("Attr 1 reason_status", PAPI.getAttr(1), "1") == 1
assert util.check("Attr 1738 national_calls", PAPI.getAttr(1738),
"CZ=1000,") == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.1.4 PAPI activate

Script TC_EX_CLF_04_PAPI_activate.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {
```

```
// start
util.log("START")

// setup
MSISDN = "603847346"
util.clf.connect(env)

// get, service deactive
util.log("Get")
SU = util.clf.getSU(MSISDN)
PAPI = util.clf.callPAPI("SU", SU, 1, "G")
assert util.check("Status", PAPI.getStatus(), "d") == 1

// activation
util.log("Activation")
PAPI = util.clf.callPAPI("SU", SU, 1, "a", [1, 735, 736, 813], [21, "OBE",
"OBE", "lynus@seznam.cz"])

// get, service active
util.log("Get")
PAPI = util.clf.callPAPI("SU", SU, 1, "G")
assert util.check("Status", PAPI.getStatus(), "a") == 1
assert util.check("Attr 1 reason_status", PAPI.getAttr(1), "21") == 1
assert util.check("Attr 735 notification_type", PAPI.getAttr(735), "OBE")
== 1
assert util.check("Attr 736 message_type", PAPI.getAttr(735), "OBE") == 1
assert util.check("Attr 813 email_address", PAPI.getAttr(813),
"lynus@seznam.cz") == 1
assert util.check("Attr 1223 inkaso", PAPI.getAttr(1223), "N") == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.1.5 PAPI modify

Script TC_EX_CLF_05_PAPI_modify.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
```

```
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.clf.connect(env)

// get, service active
util.log("Get")
SU = util.clf.getSU(MSISDN)
PAPI = util.clf.callPAPI("SU", SU, 1, "G")
assert util.check("Status", PAPI.getStatus(), "a") == 1
assert util.check("Attr 1 reason_status", PAPI.getAttr(1), "21") == 1
assert util.check("Attr 735 notification_type", PAPI.getAttr(735), "OBE")
== 1
assert util.check("Attr 736 message_type", PAPI.getAttr(736), "OBE") == 1
assert util.check("Attr 813 email_address", PAPI.getAttr(813),
"lynus@seznam.cz") == 1
assert util.check("Attr 1223 inkaso", PAPI.getAttr(1223), "N") == 1

// modification
util.log("Modification")
PAPI = util.clf.callPAPI("SU", SU, 1, "m", [735, 736, 813, 1223],
["UHRADA", "EMAIL", "lynus@centrum.cz", "A"])

// get, service modified
util.log("Get")
PAPI = util.clf.callPAPI("SU", SU, 1, "G")
assert util.check("Status", PAPI.getStatus(), "a") == 1
assert util.check("Attr 735 notification_type", PAPI.getAttr(735),
"UHRADA") == 1
assert util.check("Attr 736 message_type", PAPI.getAttr(736), "EMAIL") == 1
assert util.check("Attr 813 email_address", PAPI.getAttr(813),
"lynus@centrum.cz") == 1
assert util.check("Attr 1223 inkaso", PAPI.getAttr(1223), "A") == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.1.6 PAPI deactivate

Script TC_EX_CLF_06_PAPI_deactivate.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.clf.connect(env)

// get, service active
util.log("Get")
SU = util.clf.getSU(MSISDN)
PAPI = util.clf.callPAPI("SU", SU, 1, "G")
assert util.check("Status", PAPI.getStatus(), "a") == 1

// deactivation
util.log("Dectivation")
PAPI = util.clf.callPAPI("SU", SU, 1, "d", [1], [22])

// get, service deactive
util.log("Get")
PAPI = util.clf.callPAPI("SU", SU, 1, "G")
assert util.check("Status", PAPI.getStatus(), "d") == 1
assert util.check("Attr 1 reason_status", PAPI.getAttr(1), "22") == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.1.7 Cases

Script TC_EX_CLF_07_Cases.groovy

```
// initialization
```

```
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.clf.connect(env)

// CASE C1878
cas = util.clf.getCase(MSISDN, "C1878")
assert cas != null
util.log(cas.getID())
assert util.check("Process", cas.getProcess(), "ZMĚNY ZÁKAZNICKÉHO
PROFILU") == 1
assert util.check("Subject", cas.getSubject(), "Tarif") == 1
assert util.check("Subsubject", cas.getSubsubject(), "Žádost zákazníka") ==
1
assert util.check("Step", cas.getStep(), "...") == 1
assert util.check("Status", cas.getStatus(), "OPEN") == 1
assert util.check("MFFT1 Predchozi tarif", cas.getMFFT()[0], "Profi na míru
1") == 1
assert util.check("MFFT2 Novy tarif", cas.getMFFT()[1], "Profi na míru 2")
== 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.clf.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.2 DWH

4.2.2.1 Customer structure

Script TC_EX_DWH_01_Customer_structure.groovy

```
// initialization
import com.bowman.autotester.Utilities
```

```
result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.dwh.connect(env)

// get customer hierarchy
hier = util.dwh.getCustHierarchy(MSISDN)
CU = util.dwh.getCU(MSISDN)
LE = util.dwh.getLE(MSISDN)
OU = util.dwh.getOU(MSISDN)
EU = util.dwh.getEU(MSISDN)
BA = util.dwh.getBA(MSISDN)
SU = util.dwh.getSU(MSISDN)

// check hierarchy
assert util.check("CU", hier.getCU().getExtID(), CU) == 1
assert util.check("LE", hier.getLE().getExtID(), LE) == 1
assert util.check("OU", hier.getOU().getExtID(), OU) == 1
assert util.check("EU", hier.getEU().getExtID(), EU) == 1
assert util.check("BA", hier.getBA().getExtID(), BA) == 1
assert util.check("SU", hier.getSU().getExtID(), SU) == 1

// bilcycle
BC = util.dwh.getBillCycle(MSISDN)
assert util.check("BC", BC.getID(), 52) == 1

// segment
segment = util.dwh.getSegment(MSISDN)
assert util.check("Segment", segment.getTitle(), "LE") == 1

// market
market = util.dwh.getMarket(MSISDN)
assert util.check("Market", market.getTitle(), "GSM") == 1

// tariff
tariff = util.dwh.getTariff(MSISDN)
assert util.check("Tariff", tariff.getID(), 204) == 1

// teardown
util.clf.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.dwh.disconnect()

// error, test failed
```

```
util.err("ERR")
result = 0

}
```

4.2.2.2 Services

Script TC_EX_DWH_02_Services.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.dwh.connect(env)

// SU services
SU = util.dwh.getSU(MSISDN)
suSrv = util.dwh.getSUServices(SU)
suSrv.each { srv -> util.log(srv.getID()) }

// service 897
srv = util.dwh.getSUService(SU, 897)
assert util.checkNok("Instance", srv.getInstanceID(), 0) == 1
assert util.check("Status", srv.getStatus(), "a") == 1
assert util.checkNok("Param 1531", srv.getParam(1531), "0") == 1
assert util.checkLike("Param 1534", srv.getParam(1534), "Z1-2") == 1

// service 501 in MTX
srv = util.dwh.getMTXService(SU, 501)
assert util.check("Status", srv.getStatus(), "d") == 1

// teardown
util.dwh.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.dwh.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.3 AMX

4.2.3.1 Customer structure

Script TC_EX_AMX_01_Customer_structure.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.amx.connect(env)

// SU
SU = util.amx.getSU(MSISDN)
assert util.check("SU", SU, 41999702)

// CU
CU = util.amx.getCU(MSISDN)
assert util.check("CU", CU, 100005344)

// teardown
util.amx.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.amx.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.3.2 Offers

Script TC_EX_AMX_02_Offers.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
```

```
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
MSISDN = "603847346"
util.amx.connect(env)

// SU offers
SU = util.amx.getSU(MSISDN)
suOffer = util.amx.getSUOffers(SU)
suOffer.each { offer -> util.log(offer.getID() + ":" + offer.getStatus()) }

// offer 457028096
offer = util.amx.getSUOffer(SU, 457028096)
assert util.check("Status", offer.getStatus(), "A") == 1
assert util.check("Title", offer.getTitle(), "Tariff Profi (60+1)") == 1

// offer 983
offer = util.amx.getSUOffer(SU, 983)
assert util.check("Status", offer.getStatus(), "C") == 1
assert util.check("Title", offer.getTitle(), "FUOM Bez hranic na miru") ==
1

// teardown
util.amx.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.amx.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.4 CNH

4.2.4.1 Notifications

Script TC_EX_CNH_01_Notifications.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)
```

```
try {

// start
util.log("START")

// setup
util.cnh.connect(env)

// SMS
util.log("SMS")
MSISDN = "603713546"
notif = util.cnh.getNotification(MSISDN)
assert util.check("Template", notif.getTemplate(),
"pactum2_insurance_deactivate_sms") == 1
assert util.check("Status", notif.getStatus(), 108) == 1

params = notif.getParameters()
params.each {key, value -> util.log(key + ":" + value)}

assert util.check("SERVICE", notif.getParam("SERVICE"), "T-Mobile pro
jistotu") == 1

// EMAIL
util.log("EMAIL")
EMAIL = "martin.zima@t-mobile.cz"
notif = util.cnh.getNotification(EMAIL)
assert util.check("Template", notif.getTemplate(),
"pactum2_insurance_deactivate_email") == 1
assert util.check("Status", notif.getStatus(), 103) == 1

params = notif.getParameters()
params.each {key, value -> util.log(key + ":" + value)}

assert util.checkNok("CODE", notif.getParam("CODE"), "0") == 1

// teardown
util.cnh.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.cnh.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.5 AP

4.2.5.1 DTS for VCC

Script TC_EX_AP_01_DTS_for_VCC.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
SU = 41999702
util.ap.connect(env)

// RC_SWITCH
util.log("RC_SWITCH")
dts = util.ap.callDTS("SU", SU, "RC_SWITCH", ["NEW_TARIFF"], ["433"])
assert util.check("Result", dts.getResult(), "RC_T8") == 1
assert util.check("Action 1",
dts.getAction("WARN_ABOUT_UNUSED_FUOMS_LOST")[0],
"WARN_ABOUT_UNUSED_FUOMS_LOST") == 1
assert util.check("Action 2",
dts.getAction("ROAMING_DAILY_PASS_ACTIVATE")[0],
"ROAMING_DAILY_PASS_ACTIVATE") == 1

params = dts.getParameters()
params.each { key, value -> util.log(key + ":" + value) }

// teardown
util.ap.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.ap.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.5.2 DTS for CCM

Script TC_EX_AP_02_DTS_for_CCM.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
SU = 41999702
util.ap.connect(env)

// TO_SU
util.log("TO_SU")
dts = util.ap.callDTS("SU", SU, "TO_SU")
assert util.check("Result", dts.getResult(), "2") == 1
assert util.checkLike("Warning", dts.getWarnings()[0], "vod je mo")

params = dts.getParameters()
params.each { key, value -> util.log(key + ":" + value) }

// teardown
util.ap.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.ap.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.6 WS

4.2.6.1 GetStatus

Script TC_EX_WS_01_GetStatus.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
```

```
util.log("START")

// setup
SU = 41999702
util.ws.connect(env)

// MissedCallsRegister - service deactive
WS = util.ws.callWS("SU", SU, "MissedCallsRegister", "getStatus")
assert util.check("Status", WS.getStatus(), "d") == 1
assert util.checkNok("CorrelationId", WS.getCorrelationId(), "") == 1

// allowed action - MissedCallsRegister.ACTIVATE
util.log("AA MissedCallsRegister.ACTIVATE")
AA = WS.getAllowedAction("MissedCallsRegister.ACTIVATE")
assert util.check("Allowed", AA.getAllowed(), true) == 1

// allowed action - MissedCallsRegister.DEACTIVATE
util.log("AA MissedCallsRegister.DEACTIVATE")
AA = WS.getAllowedAction("MissedCallsRegister.DEACTIVATE")
assert util.check("Allowed", [AA.getAllowed(), AA.getBreData().get(0)],
[false, "SERVICE_NOT_ACTIVE"]) == 1

// teardown
util.ws.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.ws.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.6.2 Activate

Script TC_EX_WS_02_Activate.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
SU = 41999702
```

```
util.ws.connect(env)

// getStatus - service deactive
util.log("GetStatus");
WS = util.ws.callWS("SU", SU, "MyBill", "getStatus")
assert util.check("Status", WS.getStatus(), "d") == 1

// validateActivate
util.log("ValidateActivate");
WS = util.ws.callWS("SU", SU, "MyBill", "validateActivate",
                    "service", ["notificationType", "messageType",
"emailAddress", "directDebit"], ["OBE", "OBE", "lynus@seznam.cz", "A"],
                    "action", ["reason_status"], [21])
assert util.check("OrderStatus", WS.getOrderStatus(), "VALIDATED") == 1

// activate
util.log("activate");
WS = util.ws.callWS("SU", SU, "MyBill", "activate",
                    "service", ["notificationType", "messageType",
"emailAddress", "directDebit"], ["OBE", "OBE", "lynus@seznam.cz", "A"],
                    "action", ["reason_status"], [21])
assert util.check("OrderStatus", WS.getOrderStatus(), "COMPLETED") == 1

// getStatus - service active
util.log("GetStatus");
WS = util.ws.callWS("SU", SU, "MyBill", "getStatus")
assert util.check("Status", WS.getStatus(), "a") == 1
assert util.check("notificationType",
WS.getServiceParam("notificationType"), "OBE") == 1
assert util.check("messageType", WS.getServiceParam("messageType"), "OBE")
== 1
assert util.check("emailAddress", WS.getServiceParam("emailAddress"),
"lynus@seznam.cz") == 1
assert util.check("directDebit", WS.getServiceParam("directDebit"), "A") ==
1
assert util.check("reason_status", WS.getActionParam("reason_status"),
"21") == 1

// teardown
util.ws.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.ws.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.6.3 Modify

Script TC_EX_WS_03_Modify.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
SU = 41999702
util.ws.connect(env)

// getStatus - service active
util.log("GetStatus");
WS = util.ws.callWS("SU", SU, "MyBill", "getStatus")
assert util.check("Status", WS.getStatus(), "a") == 1
assert util.check("notificationType",
WS.getServiceParam("notificationType"), "OBE") == 1
assert util.check("messageType", WS.getServiceParam("messageType"), "OBE")
== 1
assert util.check("emailAddress", WS.getServiceParam("emailAddress"),
"lynus@seznam.cz") == 1
assert util.check("directDebit", WS.getServiceParam("directDebit"), "A") ==
1

// validateModify
util.log("ValidateModify");
WS = util.ws.callWS("SU", SU, "MyBill", "validateModify",
                    "service", ["notificationType", "messageType",
"emailAddress", "directDebit"], ["UHRADA", "EMAIL", "lynus@centrum.cz",
"N"])
assert util.check("OrderStatus", WS.getOrderStatus(), "VALIDATED") == 1

// modify
util.log("modify");
WS = util.ws.callWS("SU", SU, "MyBill", "modify",
                    "service", ["notificationType", "messageType",
"emailAddress", "directDebit"], ["UHRADA", "EMAIL", "lynus@centrum.cz",
"N"])
assert util.check("OrderStatus", WS.getOrderStatus(), "COMPLETED") == 1

// getStatus - service modified
util.log("GetStatus");
WS = util.ws.callWS("SU", SU, "MyBill", "getStatus")
assert util.check("Status", WS.getStatus(), "a") == 1
assert util.check("notificationType",
WS.getServiceParam("notificationType"), "UHRADA") == 1
assert util.check("messageType", WS.getServiceParam("messageType"),
"EMAIL") == 1
assert util.check("emailAddress", WS.getServiceParam("emailAddress"),
"lynus@centrum.cz") == 1
```

```
assert util.check("directDebit", WS.getServiceParam("directDebit"), "N") ==
1

// teardown
util.ws.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.ws.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.2.6.4 Deactivate

Script TC_EX_WS_04_Deactivate.groovy

```
// initialization
import com.bowman.autotester.Utilities

result = 0
util = new Utilities(logID)

try {

// start
util.log("START")

// setup
SU = 41999702
util.ws.connect(env)

// getStatus - service active
util.log("GetStatus");
WS = util.ws.callWS("SU", SU, "MyBill", "getStatus")
assert util.check("Status", WS.getStatus(), "a") == 1

// validateDeactivate
util.log("ValidateDeactivate");
WS = util.ws.callWS("SU", SU, "MyBill", "validateDeactivate",
                    "action", ["reason_status"], ["22"])
assert util.check("OrderStatus", WS.getOrderStatus(), "VALIDATED") == 1

// modify
util.log("deactivate");
WS = util.ws.callWS("SU", SU, "MyBill", "deactivate",
                    "action", ["reason_status"], ["22"])
assert util.check("OrderStatus", WS.getOrderStatus(), "COMPLETED") == 1
```

```
// getStatus - service deactivated
util.log("GetStatus");
WS = util.ws.callWS("SU", SU, "MyBill", "getStatus")
assert util.check("Status", WS.getStatus(), "d") == 1

// teardown
util.ws.disconnect()

// finish, test passed
util.log("FINISH")
result = 1

}
catch (AssertionError e) {

// teardown
util.ws.disconnect()

// error, test failed
util.err("ERR")
result = 0

}
```

4.3 Functions

4.3.1 Common

4.3.1.1 Utilities

- **public Utilities (int iLogID)**

4.3.1.2 check

- **public int check (String sLog, Object oValue, Object oCheck)**
- **public int check (String sLog, List<Object> oValues, List<Object> oChecks)**
- **public int check (String sLog, HashMap<Object, Object> oValues, List<Object> oCheckedKeys, List<Object> oCheckedValues)**

4.3.1.3 checkLike

- **public int checkLike (String sLog, Object oValue, Object oCheck)**

4.3.1.4 checkNok

- **public int checkNok (String sLog, Object oValue, Object oCheck)**

4.3.1.5 connect

- **public int connect (String sEnvironment)**

4.3.1.6 disconnect

- **public int disconnect ()**

4.3.1.7 err

- **public void err (Object oErr)**

4.3.1.8 getTestData

- **public int getTestData (String sEnvironment)**

4.3.1.9 log

- **public void log (Object oLog)**

4.3.1.10 pause

- **public void pause (float fDuration)**

4.3.2 CLF

4.3.2.1 callPAPI

- **public PAPI callPAPI (String sLevel, int iExtID, int iPropertyID, String sAction)**
- **public PAPI callPAPI (String sLevel, int iExtID, int iPropertyID, String sAction, List<Integer> iAttrIDs, List<String> sAttrValues)**

4.3.2.2 connect

- **public int connect (String sEnvironment)**

4.3.2.3 disconnect

- **public int disconnect ()**

4.3.2.4 getBA

- **public int getBA (String sMSISDN)**

4.3.2.5 getBADetail

- **public BADetail getBADetail (int iExtID)**

4.3.2.6 getBAService

- **public Service getBAService (int iExtID, int iServiceID)**

4.3.2.7 getBAServices

- **public List<Service> getBAServices (int iExtID)**

4.3.2.8 getBillCycle

- **public BillCycle getBillCycle (String sMSISDN)**

4.3.2.9 getCU

- **public int getCU (String sMSISDN)**

4.3.2.10 getCase

- **public Case getCase (String sMSISDN, String sCaseType)**

4.3.2.11 getContractDuration

- **public ContractDuration getContractDuration (String sMSISDN)**

4.3.2.12 getCustHierarchy

- **public CustHierarchy getCustHierarchy (String sMSISDN)**

4.3.2.13 getEU

- **public int getEU (String sMSISDN)**

4.3.2.14 getLE

- **public int getLE (String sMSISDN)**

4.3.2.15 getLov

- **public Lov getLE (String sLovType, Object oID)**
- **covered LOVs: TARIFF, NON_PUBLIC_OFFER, TARIFF_PROMO, RETENTION_OFFER, SERVICE (details, parameters, resource, offers), PAPI (actions, attributes)**

4.3.2.16 getMarket

- **public Market getMarket (String sMSISDN)**

4.3.2.17 getOU

- **public int getOU (String sMSISDN)**

4.3.2.18 getOUDetail

- **public OUDetail getOUDetail (int iExtID)**

4.3.2.19 getOUService

- **public Service getOUService (int iExtID, int iServiceIS)**

4.3.2.20 getOUServices

- **public List<Service> getOUServices (int iExtID)**

4.3.2.21 getSU

- **public int getSU (String sMSISDN)**

4.3.2.22 getSUDetail

- **public SUDetail getSUDetail (int iExtID)**

4.3.2.23 getSUService

- **public Service getSUService (int iExtID, int iServiceID)**

4.3.2.24 getSUServices

- **public List<Service> getSUServices (int iExtID)**

4.3.2.25 getSegment

- **public Segmet getSegment (String sMSISDN)**

4.3.2.26 getStatus

- **public String getStatus (String sMSISDN)**

4.3.2.27 getTariff

- **public Tariff getTariff (String sMSISDN)**

4.3.3 DWH

4.3.3.1 connect

- **public int connect (String sEnvironment)**

4.3.3.2 disconnect

- **public int disconnect ()**

4.3.3.3 getBA

- **public int getBA (String sMSISDN)**

4.3.3.4 getBAService

- **public Service getBAService (int iExtID, int iServiceID)**

4.3.3.5 getBAServices

- **public List<Service> getBAServices (int iExtID)**

4.3.3.6 getBillCycle

- **public BillCycle getBillCycle (String sMSISDN)**

4.3.3.7 getCU

- **public int getCU (String sMSISDN)**

4.3.3.8 getContractDuration

- **public ContractDuration getContractDuration (String sMSISDN)**

4.3.3.9 getCustHierarchy

- **public CustHierarchy getCustHierarchy (String sMSISDN)**

4.3.3.10 getEU

- **public int getEU (String sMSISDN)**

4.3.3.11 getLE

- **public int getLE (String sMSISDN)**

4.3.3.12 getLov

- **public Lov getLov (String sLovType, Object oID)**
- **covered LOVs: TARIFF, NON_PUBLIC_OFFER, TARIFF_PROMO, DISCOUNT_PROPERTY, RETENTION_OFFER, SERVICE (parameters)**

4.3.3.13 getMTXService

- **public Service getMTXService (int iExtID, int iServiceID)**

4.3.3.14 getMarket

- **public Market getMarket (String sMSISDN)**

4.3.3.15 getOU

- **public int getOU (String sMSISDN)**

4.3.3.16 getOUService

- public Service getOUService (int iExtID, int iServiceID)

4.3.3.17 getOUServices

- public List<Service> getOUServices (int iExtID)

4.3.3.18 getSU

- public int getSU (String sMSISDN)

4.3.3.19 getSUService

- public Service getSUService (int iExtID, int iServiceID)

4.3.3.20 getSUServices

- public List<Service> getSUServices (int iExtID)

4.3.3.21 getSegment

- public Segment getSegment (String sMSISDN)

4.3.3.22 getStatus

- public String getStatus (String sMSISDN)

4.3.3.23 getTariff

- public Tariff getTariff (String sMSISDN)

4.3.4 AMX

4.3.4.1 connect

- public int connect (String sEnvironment)

4.3.4.2 disconnect

- public int disconnect ()

4.3.4.3 getCU

- public int getCU (String sMSISDN)

4.3.4.4 getLov

- public Lov getLov (String sLovType, Object oID)
- covered LOVs: OFFER (details)

4.3.4.5 getStatus

- public String getStatus (String sMSISDN)

4.3.4.6 getSU

- public int getSU (String sMSISDN)

4.3.4.7 getSUOffer

- public Offer getSUOffer (int iExtID, int iOfferID)

4.3.4.8 getSUOffers

- **public List<Offer> getSUOffers (int iExtID)**

4.3.5 AP

4.3.5.1 callDTS

- **public DTS callDTS (String sLevel, int iExtID, String sTitle)**
- **public DTS callDTS (String sLevel, int iExtID, String sTitle, List<String> sParamNames, List<String> sParamValues)**

4.3.5.2 connect

- **public int connect (String sEnvironment)**

4.3.5.3 disconnect

- **public int disconnect ()**

4.3.5.4 getLov

- **public Lov getLov (String sLovType, Object oID)**
- **covered LOVs: TARIFF, NON_PUBLIC_OFFER, TARIFF_PROMO, DISCOUNT_PROPERTY, RETENTION_OFFER, SERVICE (parameters)**

4.3.6 CNH

4.3.6.1 connect

- **public int connect (String sEnvironment)**

4.3.6.2 disconnect

- **public int disconnect ()**

4.3.6.3 getNotification

- **public Notification getNotification (String sMSISDN)**

4.3.7 WS

4.3.7.1 callWS

- **public WS callWS (String sLevel, int iExtID, String sService, String sAction)**
- **public WS callWS (String sLevel, int iExtID, String sService, String sAction, String sParamsType, List<String> sParamNames, List<String> sParamValues)**
- **public WS callWS (String sLevel, int iExtID, String sService, String sAction, String sParamsType1, List<String> sParamNames1, List<String> sParamValues1, String sParamsType2, List<String> sParamNames2, List<String> sParamValues2)**

4.3.7.2 connect

- **public int connect (String sEnvironment)**

4.3.7.3 disconnect

- **public int disconnect ()**

5. System documentation

5.1 Architecture

5.1.1 Deployment

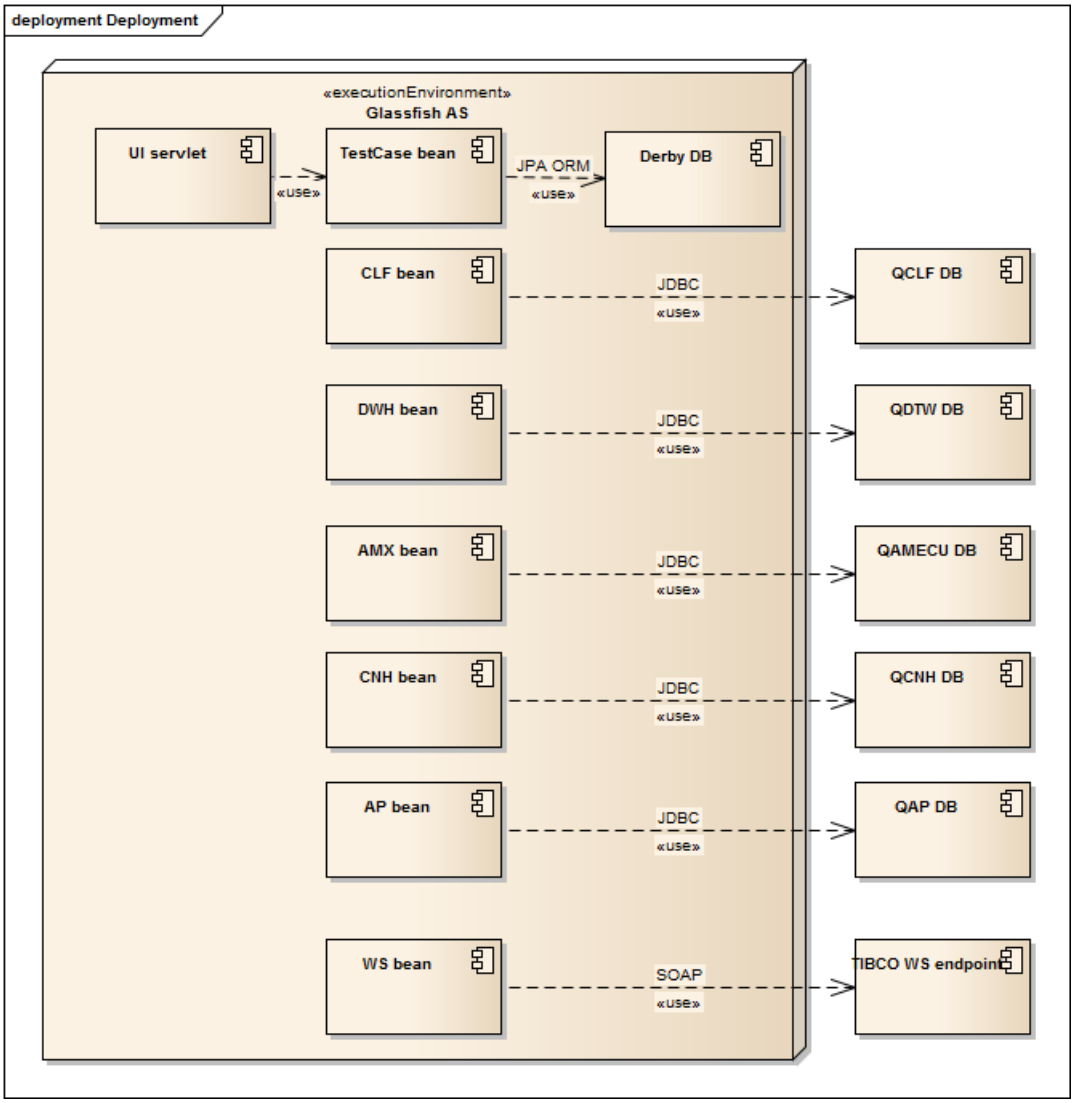

Figure 5 - Deployment

5.1.2 Test case administration

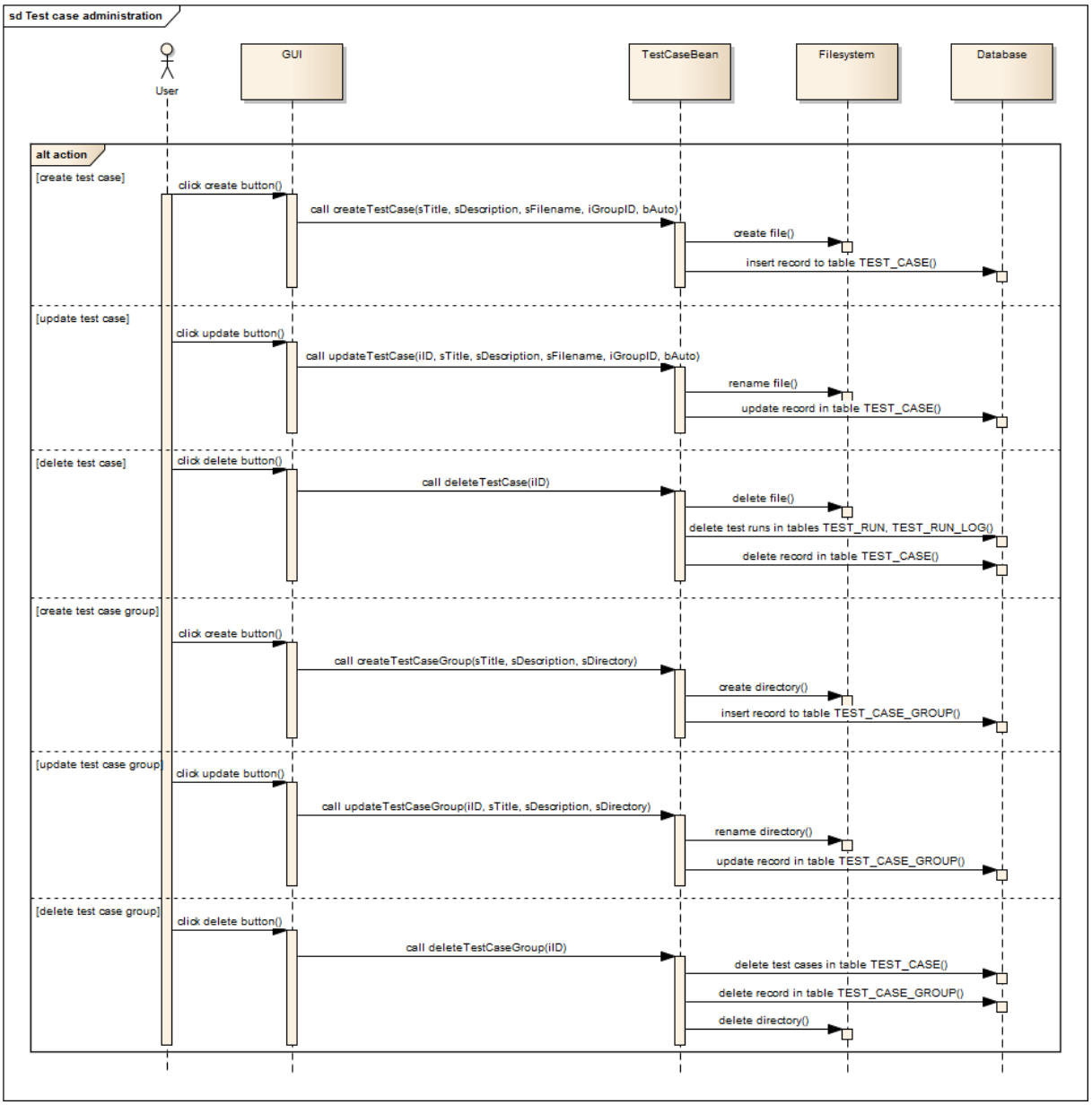

Figure 6 - Test case administration

5.1.3 Test case execution

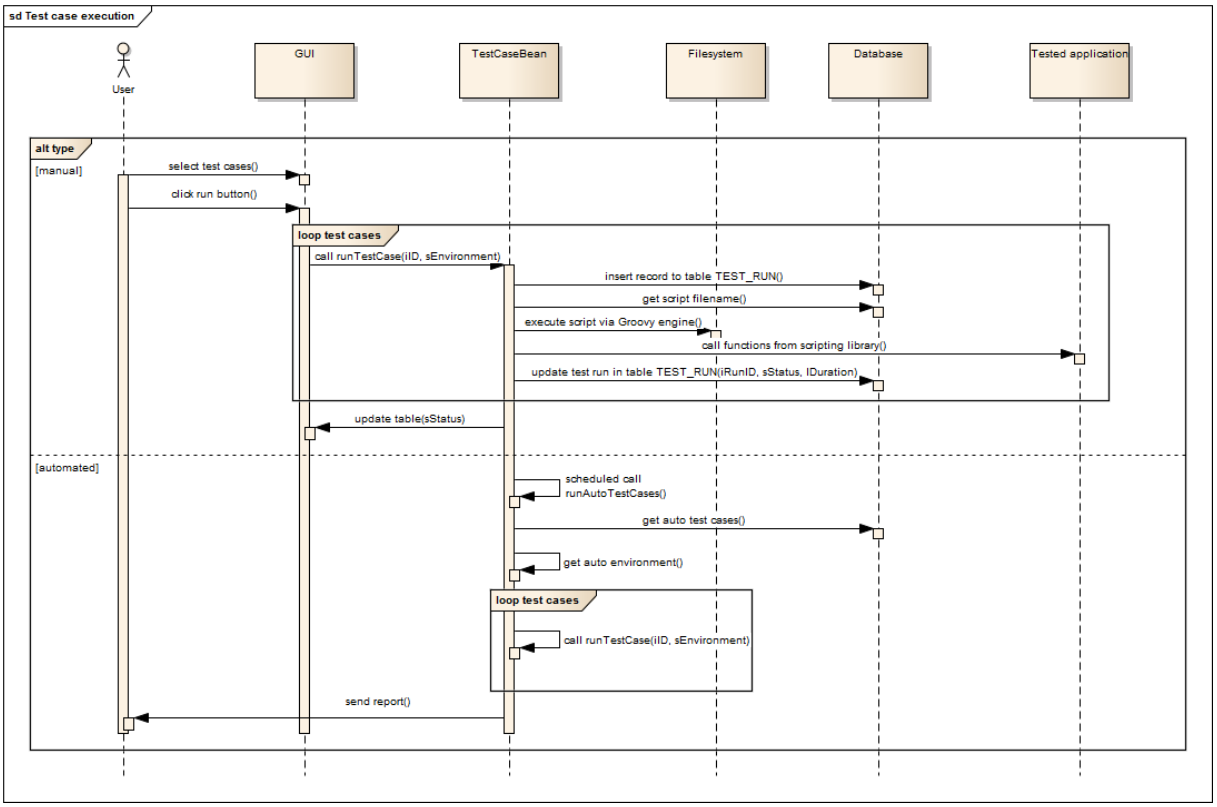

Figure 7 - Test case execution

5.2 Class diagrams

5.2.1 Utilities

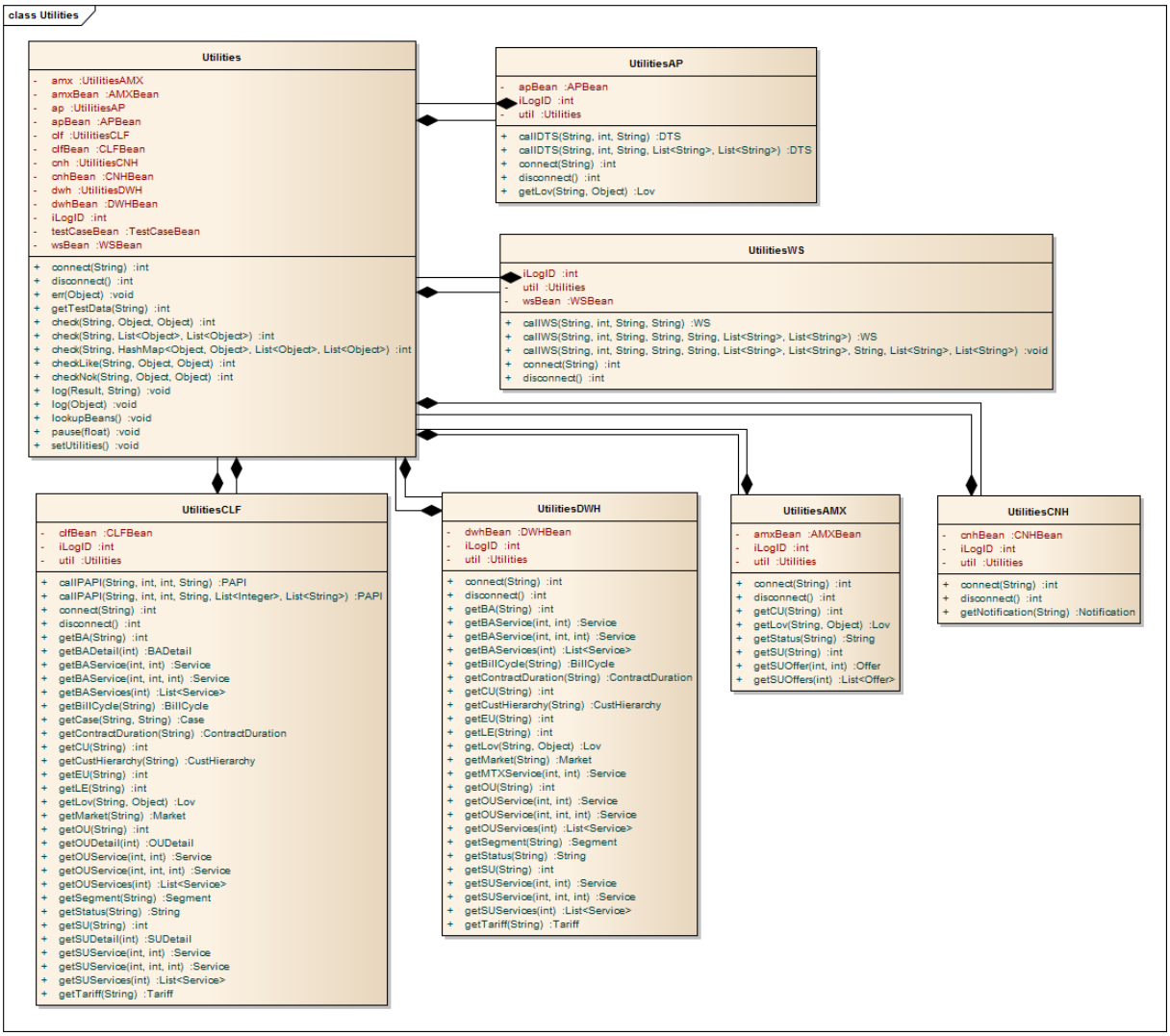

Figure 8 - Utilities

5.2.2 Beans

Figure 9 - Beans

5.2.3 Database model

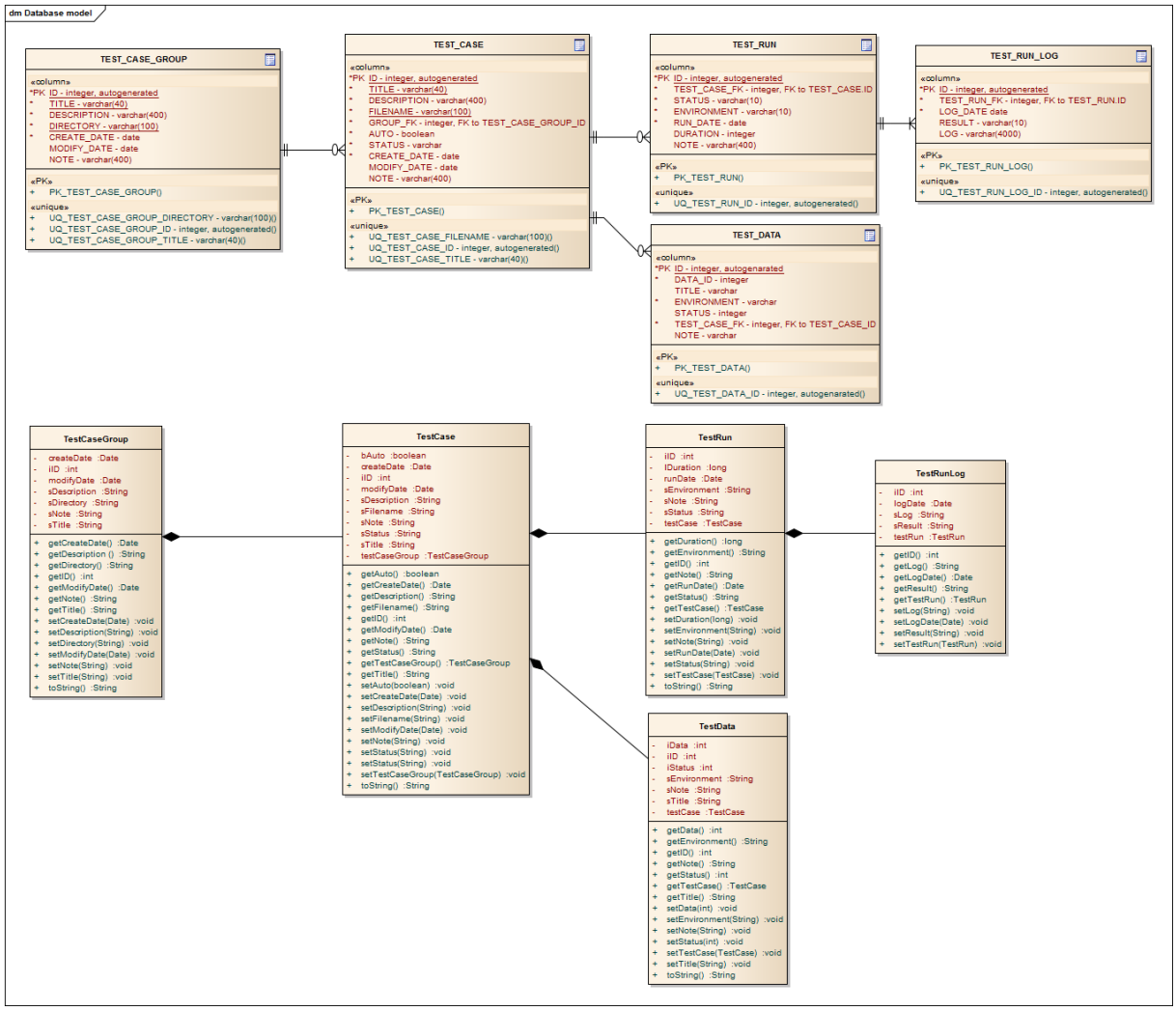

Figure 10 - Database model

5.2.4 Customer structure

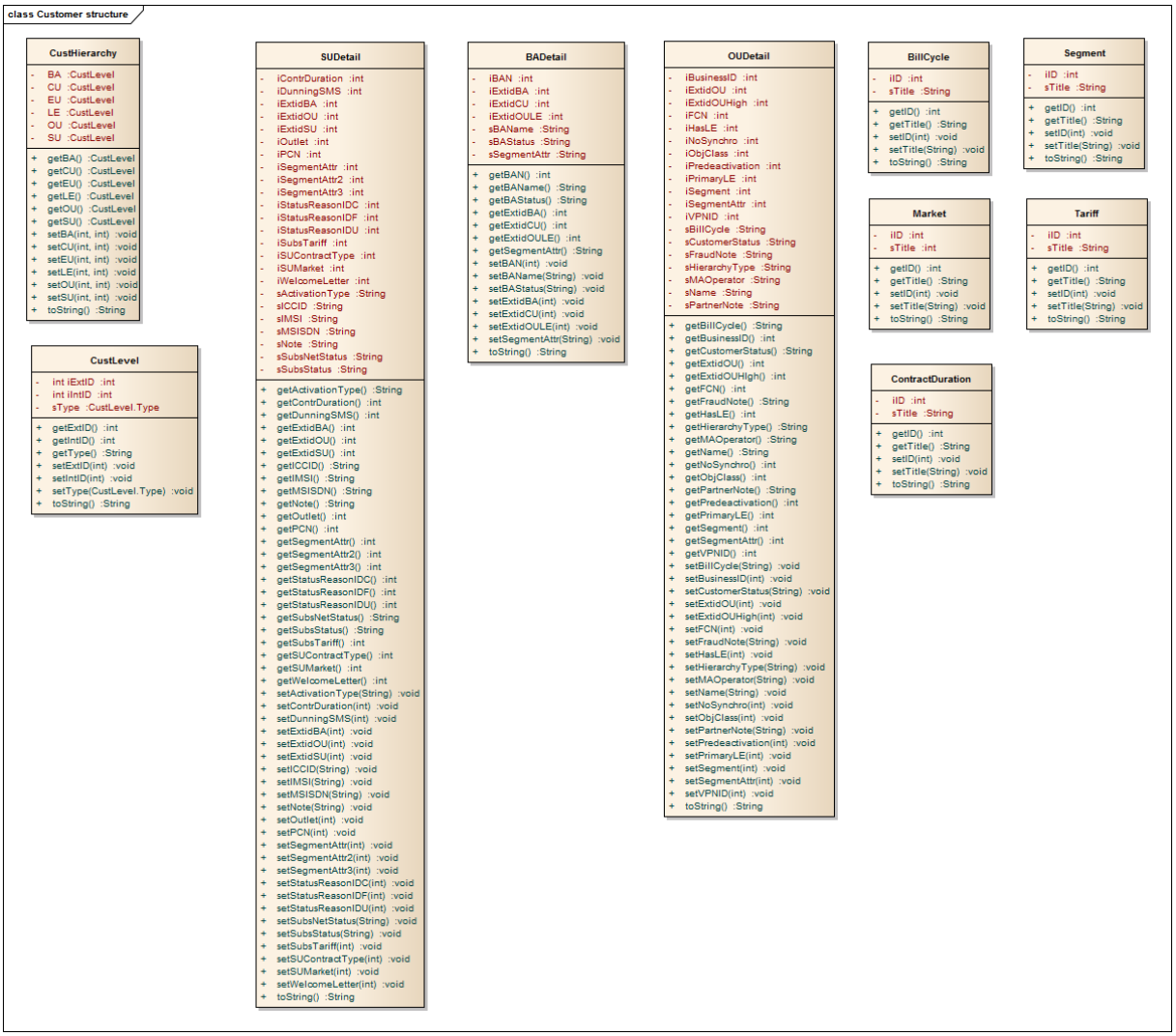

Figure 11 - Customer structure

5.2.5 Services

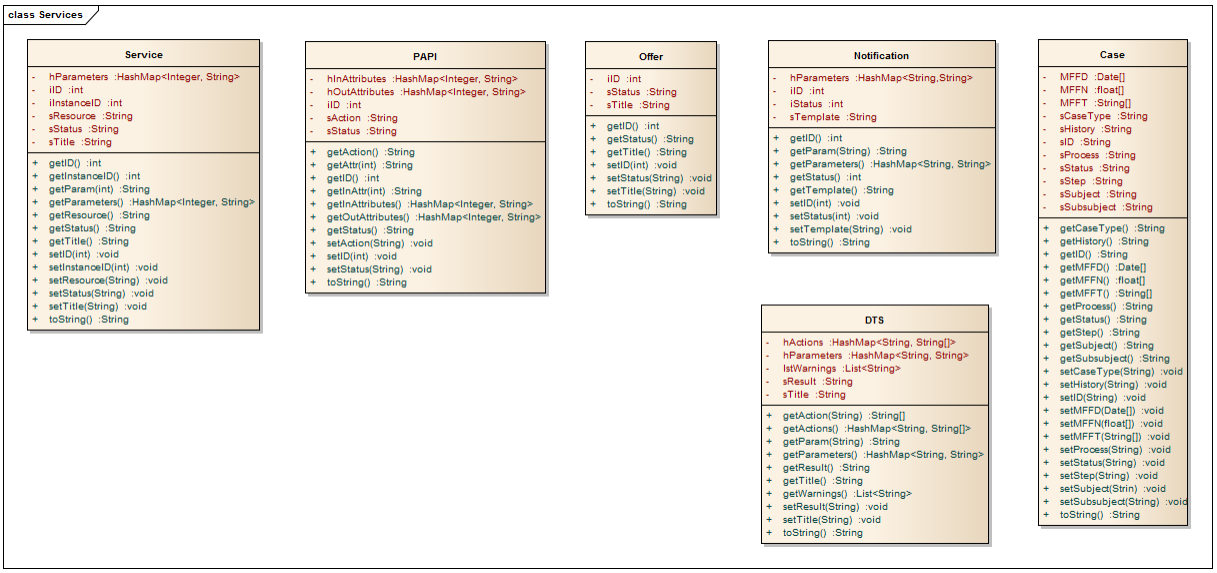

Figure 12 - Services

5.2.6 Business services

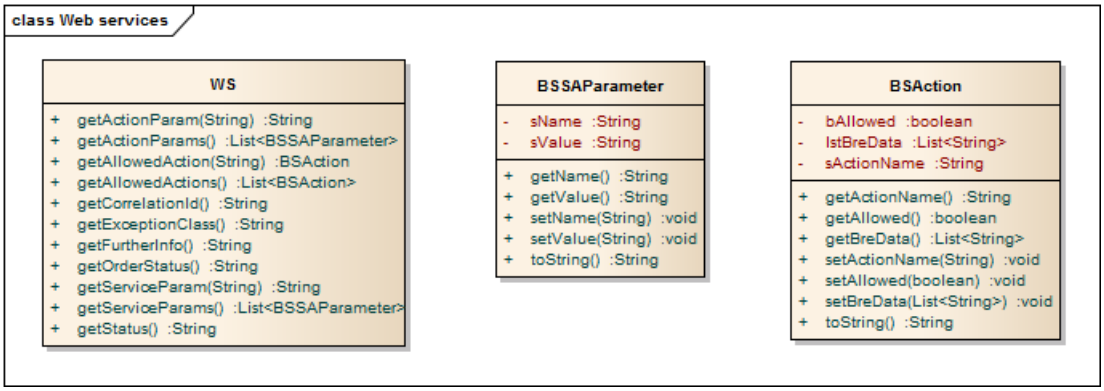

Figure 13 - Web services

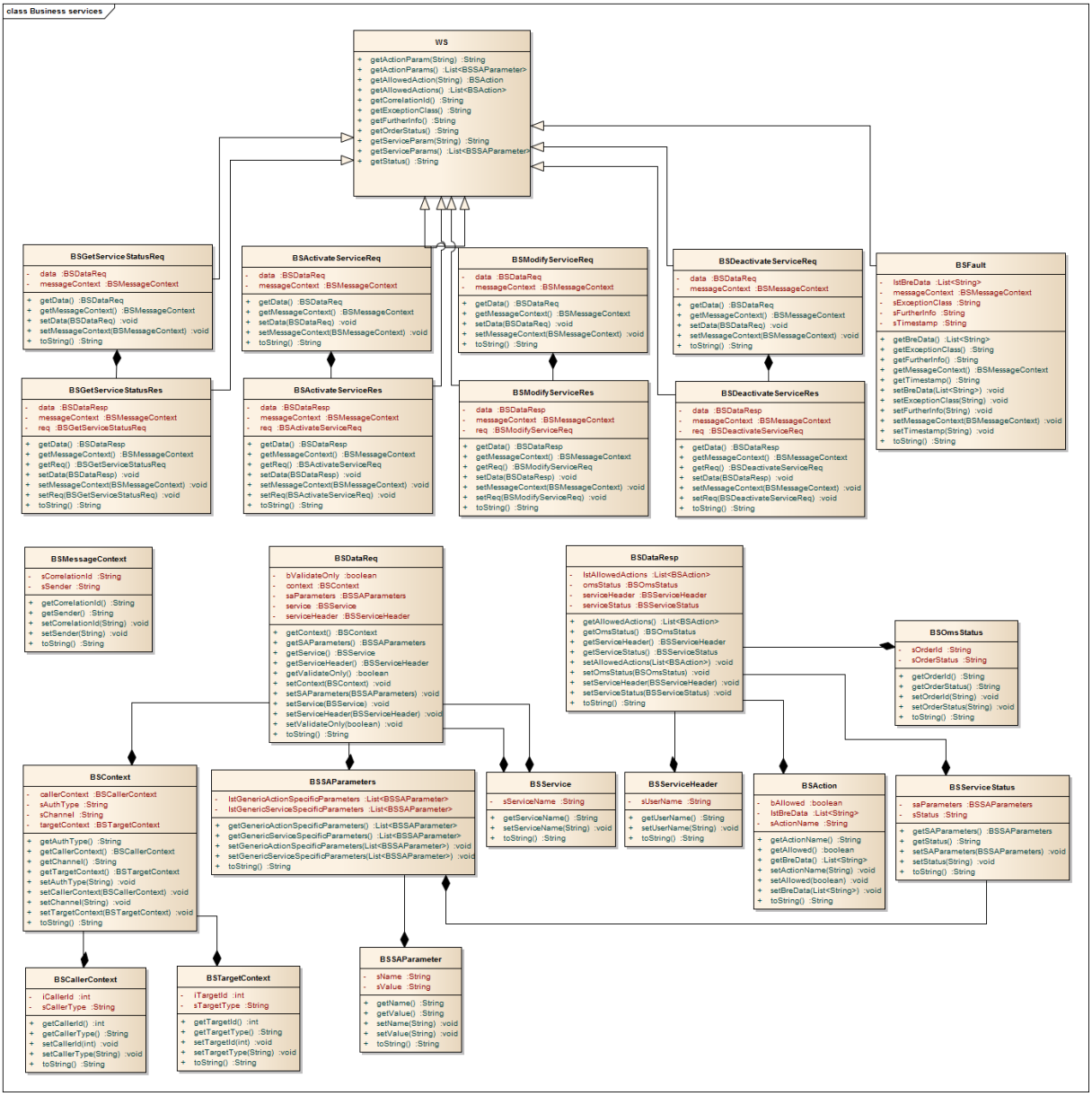

Figure 14 - Business services

5.2.7 LOV

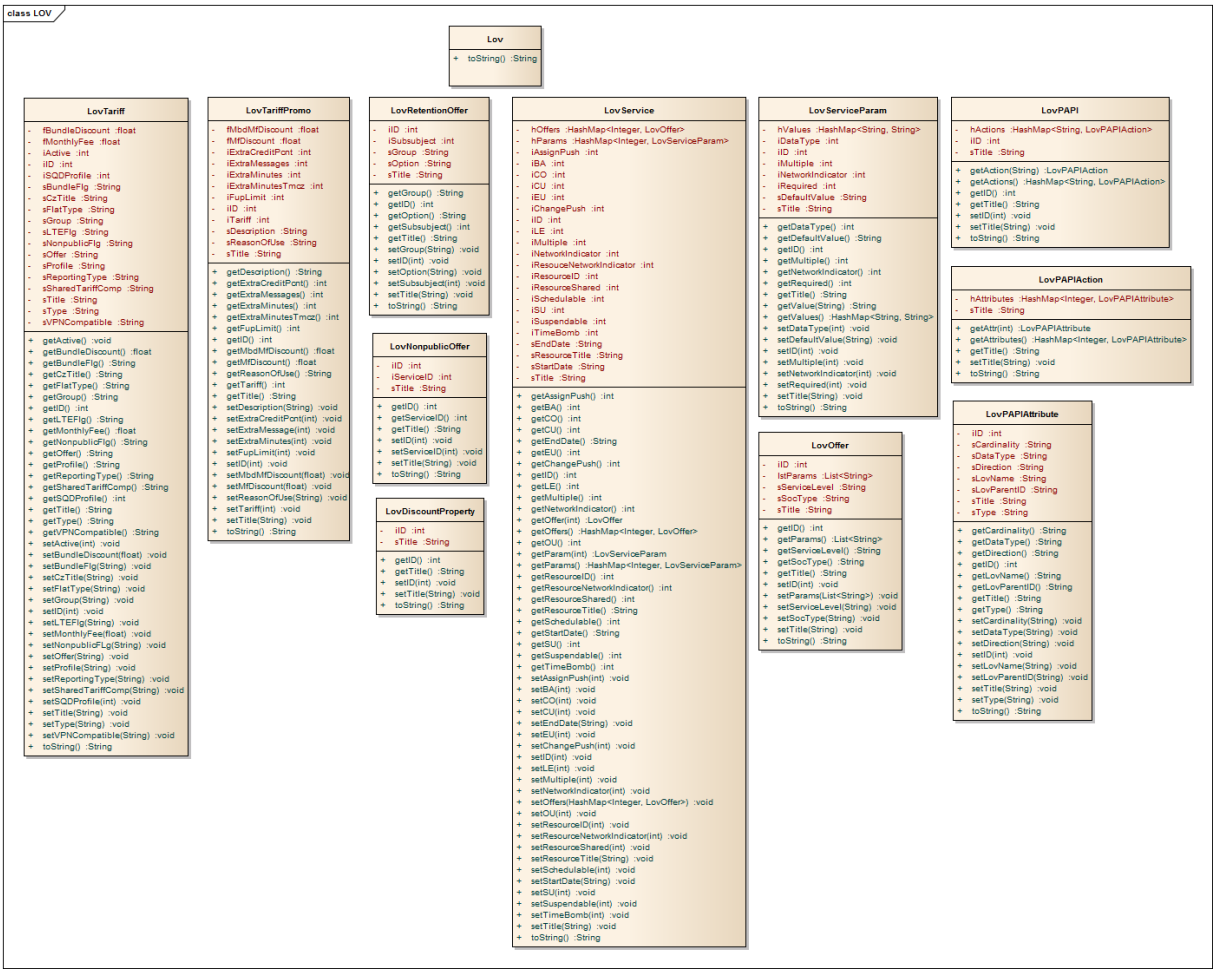

Figure 15 - LOV

5.2.8 Properties

Name	Value
testCasesPath	/home/lynus/private/autotester/tests/
testCaseSuffix	.groovy
autoEnvironment	TEST 1
test1	TEST 1
test1CLFUrl	jdbc:oracle:thin:@rztclt13.cz.tmo:1526/QCLF12.world
test1CLFUser	APPTEST
test1CLFPass	ittc
test1DWHUrl	jdbc:oracle:thin:@rztdtwt.cz.tmo:1580/QDTW2.world
test1DWHUser	APPTEST
test1DWHPass	ittc
test1APUrl	jdbc:oracle:thin:@rztapt.cz.tmo:1584/QAP01.world
test1APUser	TIBUSR
test1APPass	tibco
test1CNHUrl	jdbc:oracle:thin:@hkpow104.cz.tmo:1529/QCNH01.world
test1CNHUser	APPTEST
test1CNHPass	ittc
test1AMXUrl	jdbc:oracle:thin:@hkpowm02.cz.tmo:1520/QAMECU01.world

test1AMXUser	TMCDB2
test1AMXPass	TMCDB2
test1TIBCO	http://hktibt1.cz.tmo:9990
testWSEndpoint	/WebServices/SA/SA.serviceagent/SaPortTypeEndpoint
test2	TEST 2
test2CLFUrl	jdbc:oracle:thin:@rztclt14.cz.tmo:1626/Q2CLF12.world
test2CLFUser	APPTEST
test2CLFPass	ittc
test2DWHUrl	jdbc:oracle:thin:@hkpowm03.cz.tmo:1680/Q2DTW2.world
test2DWHUser	APPTEST
test2DWHPass	ittc
test2APUrl	jdbc:oracle:thin:@hkpowm03.cz.tmo:1684/Q2AP01.world
test2APUser	TIBUSR
test2APPass	tibco
test2CNHUrl	jdbc:oracle:thin:@hkpow104.cz.tmo:1529/QCNH01.world
test2CNHUser	APPTEST
test2CNHPass	ittc
test2AMXUrl	jdbc:oracle:thin:@hkpowm02.cz.tmo:1520/QAMECU01.world
test2AMXUser	TMCDB3
test2AMXPass	TMCDB3
test2TIBCO	http://hktibt2.cz.tmo:9990
reserveDBUrl	jdbc:oracle:thin:@rztapt.cz.tmo:1584/QAP01.world
reserveDBUser	BOLINAM
reserveDBPass	paegas

Table 2 – Properties

5.3 Maintenance

Logfiles older than 30 days are deleted. This is implemented using log4j configuration.

Records in tables TEST_RUN and TEST_RUN_LOG older than 1 month are deleted. Function execMaintenance is scheduled to 23:15.