DOCUMENT

Software Validation Test Document

Prepared by	**SLE Java Team**
Reference	**SLES-SLE-SVTD-1001**
Issue	**2**
Revision	**0**
Date of Issue	**2018-12-11**
Status	**FINAL**
Document Type	**TP**
Distribution	

CHANGE LOG

Reason for change	Issue	Revision	Date
First draft 1	1	0	2016-05-02
sleapi_j#1	1	1	2016-11-28
SLEAPIJ-5	2	0	2018-12-11

CHANGE RECORD

Issue 2	Revision 0		
Reason for change	**Date**	**Pages**	**Paragraph(s)**
SLEAPIJ-5	2018-12-11		5.2, 5.3
small correction in TestRunner descriptions	2018-12-11		6.1, 6.2

Table of contents:

1 INTRODUCTION

1.1 Purpose and Scope

This document describes the testing procedures used to validate the SLE JAVA. It explains how to start and stop the application as provider and user. In addition it lists all the executed validation tests. The material presented in this document note is the result of WP2 of the project SLE API validation.

1.2 Document Overview

Section 1 – Introduction (this section) provides the purpose, scope and this document's overview.
Section 2 – Applicable and Reference Documents provides the list of reference documents.
Section 3 – Terms, Definitions and Abbreviated Terms provides a list of acronyms and terms used throughout this document.
Section 4 - Start-up and stop of the test harness provides the necessary steps of starting and stopping the test harness.
Section 5 – software validation testing lists all the executed tests for user and provider.
Section 6 - SLE API JAVA validation within the OSGi container describes how to test the library within the OSGi container

2 APPLICABLE AND REFERENCE DOCUMENTS

2.1 Applicable Documents

Ref.	Document Title	Issue and Revision, Date
[AD-1]		

2.2 Reference Documents

Ref.	Document Title	Issue and Revision, Date
[RD-1]	*SLE API Java Software Release Document*	Issue 1, March 2016

3 TERMS, DEFINITIONS AND ABBREVIATED TERMS

3.1 Acronyms

Acronyms	Description
API	Application Program Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CCSDS	Consultative Committee for Space Data Systems
CLTU	Communication Link Transmission Unit
ESA	European Space Agency
ESOC	European Space Operations Centre
ESTRACK	ESA Tacking Network
FSP	Forward Space Packet
GPL	GNU General Public License
IP	Internet Protocol
ISO	International Standardisation Organisation
LGPL	GNU Lesser General Public License
PDU	Protocol Data Unit
RAF	Return All Frames
RCF	Return Channel Frame
ROCF	Return Operational Control Fields
SE	Service Element
SHA	Secure Hash Algorithm
SI	Service Instance
SLE	Space Link Extension
SLES	SLE Services
TCP	Transmission Control Protocol

4 START-UP AND STOP OF THE TEST HARNESS

4.1 Introduction

This section explains in detail how to start and stop the test harness application. Figure 1 gives a general overview of the SLE Provider and SLE User test scenario.

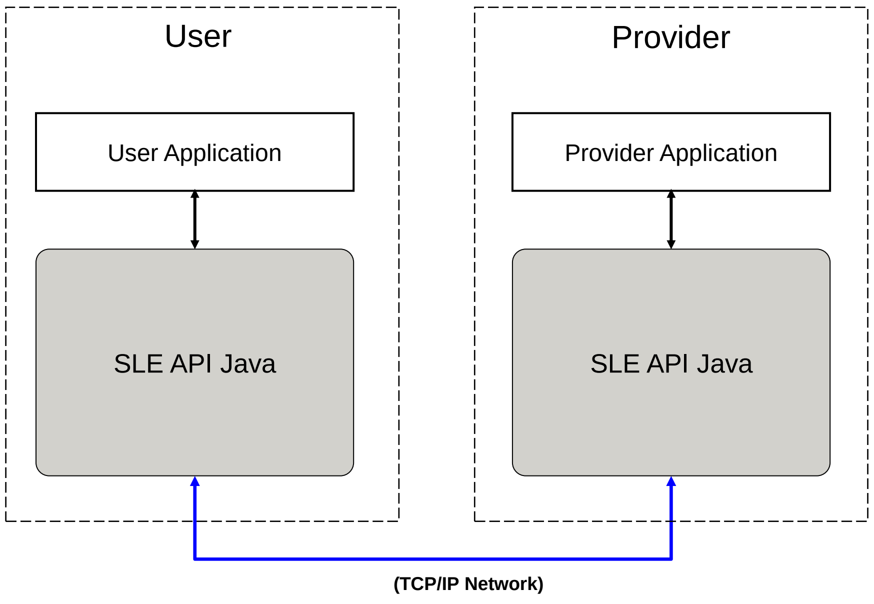

Figure 1 SLE Test scenario

Table **1** provides a list of notations to be used in the test activity.

Table 1: Notations and descriptions

Notation	Description
<Java 8 installation path>	Should be replaced with the installation path of the Java version 8. For example: “/SLEJAVA/Cots/jdk8”.
<delivery installation path>	Should be replaced with the installation path of the delivery.
<proxy database >	Should be replaced with the path to the configuration file of the proxy.
<service element database>	Should be replaced with the path to the configuration file of the service element database.
<command file >	Should be replaced with the path to the command file

Type the following command to export the JAVA_HOME environment variable:

```
$ export JAVA_HOME=<Java 8 installation path>
$ export PATH=$JAVA_HOME/bin:$PATH
```

4.2 Parameters used for running the test harness

In this section the parameters needed in order to start the test harness are described. As showed in Table **1**, there are different configuration file paths to set for user and provider.

The general form of the execution command is:

-u|-p [-e] [-q] [-T] -x <proxy database> -s <service element database> [-a <command file>] [-t <tracelevel>]

- -u|-p represents the options of running the application as user or provider.
- [-e] represents an optional flag for tracing the standard input stream values. Including this option in the run command it is possible to visualize the values from the standard input stream during the execution of the program.
- [-q] represents an optional flag for tracing the file input stream values. Including this option in the run command it is possible to visualize the values from the file input stream during the execution of the program.
- [-T] representing an optional flag for tracing the API functions. Including this option in the run command it is possible to visualize which function and parameters were called in a certain point of the execution.
- -x <proxy database> represents the configuration file of the database proxy provider or user.
- -s <service element database> represents the configuration file of the service element database for provider or user.
- [-a <command file>] represents a specific command file.
- [-t <tracelevel>] represents the trace level. There trace levels that can be selected are: low (0), medium (1), high (2), full (3)

4.3 Starting the SLE JAVA Provider

The following table lists the steps to be executed in order to start the provider application.

Step	Description
1	open a shell
2	navigate to <delivery installation path>/SLE JAVA/test_scripts
3	`export` SLE_JAR=<delivery installation path>/SLE JAVA/dist/jar/
4	`export` CLASSPATH=<delivery installation path>/SLE JAVA//extlib/
5	`./slecs.sh  -d <proxy database provider>` The communication server is started
6	open a shell
7	navigate to <delivery installation path>/SLE JAVA//test_scripts
8	`export` SLE_JAR=<delivery installation path>/SLE JAVA//dist/jar/
9	`export` CLASSPATH=<delivery installation path>/SLE JAVA//extlib/
10	`./sledfl.sh -d / <proxy database provider> -t 3` The default logger is started.
11	open a shell
12	navigate to <delivery installation path>/SLE JAVA//test_scripts
13	`export` SLE_JAR=<delivery installation path>/SLE JAVA/dist/jar/
14	`export` CLASSPATH=<delivery installation path>/SLE JAVA/extlib/
15	`./thapi.sh -p -e -T -x <proxy database provider>  -s < service element provider database> -a  <command file provider> -t 3` The provider application is started.

4.4 Starting the SLE JAVA User

This section describes the necessary steps for starting the user application.
Once the communication server, the default logger and the provider are running, the following steps must be executed:

Step	Description
1	Open a shell
2	navigate to <delivery installation path>/SLE JAVA/test_scripts
3	`export` SLE_JAR=/<delivery installation path>/SLE JAVA/dist/jar/
4	`export` CLASSPATH=<delivery installation path>/SLE JAVA/extlib/
5	`./thapi.sh -u -e -T -x <proxy database user> -s <service element user database> -a <command file user> -t 3`

4.5 Using the Multi-Instance Test Harness

The SLE API supports the instantiation of multiple instances. In order to test this feature, a simple evolution of the Test Harness has been implemented, called Multi-Instance Test Harness (MTH). This tool supports concurrent instantiation and run of several SLE API test procedures in parallel, using different configuration.

For instance, the following set-up has been successfully tested:

- A first MTH instance, loading a provider role running the raf3-provider control procedure on configuration A and the raf3-user control procedure on configuration B;
- A second MTH instance, loading a provider role running the raf3-provider control procedure on configuration B and the raf3-user control procedure on configuration A.

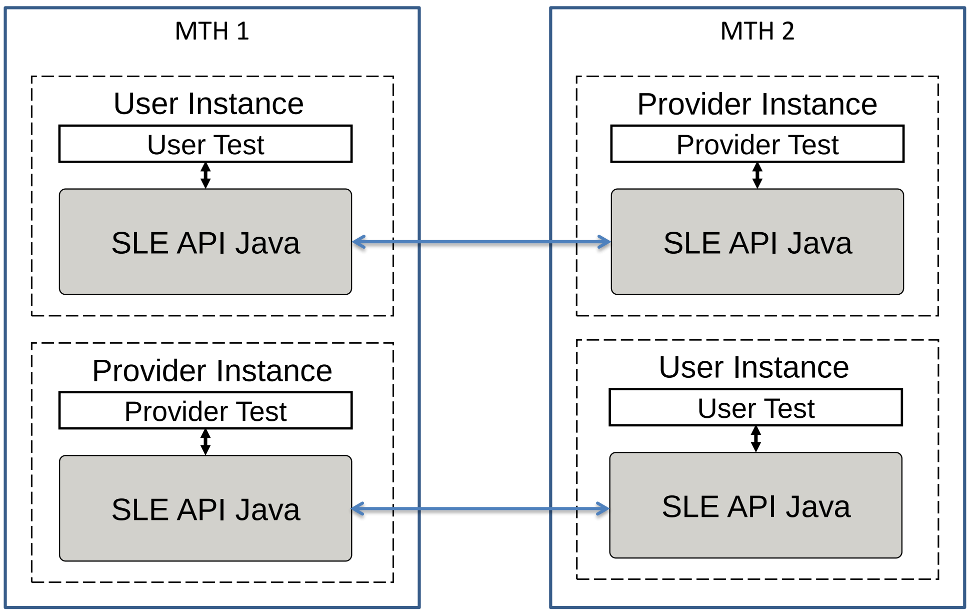

Figure 2 SLE Multi instance test scenario

The Multi-Instance Test Harness can be launched using the script mthapi.sh located inside the test_script folder. The syntax is the following:

```
./mthapi.sh [-p|-u <path to proxy file> <path to SE file> <path to test
procedure file> <delay time>]+
```

The block delimited by square brackets can be repeated one or more times. The arguments are

- -u|-p represents the options of running the specific instance as user or provider.
- <path to proxy file> is the path to the SLE API proxy configuration file to be used for the specific instance;
- <path to SE file> is the path to the SLE API SE configuration file to be used for the specific instance;
- <path to test procedure file> is the path to the test procedure file to be run by the specific instance;
- <delay time> is the time in millisecond that the MTH will wait before starting the test procedure.

The instances are started in parallel and the control procedures are executed in parallel. The execution of a test procedure can be deferred by specifying a delay time. If no delay is required, the delay time shall be set to 0.

Example of execution

Terminal 1

cd <delivery installation path>/SLE JAVA/test_scripts

./mthapi.sh
-p
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBLocalProxyProvider2.txt
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBSEProvider.txt
<delivery installation path>/SLEJAVA/test_procedures/control-v2/raf3-provider.ctl
0
-u
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBProxyUser1.txt
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBSEUser.txt
<delivery installation path>/SLEJAVA/test_procedures/control-v2/raf3-user.ctl
30000

Terminal 2

cd <delivery installation path>/SLE JAVA/test_scripts

./mthapi.sh
-p
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBLocalProxyProvider1.txt
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBSEProvider.txt
<delivery installation path>/SLEJAVA/test_procedures/control-v2/raf3-provider.ctl
0
-u
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBProxyUser2.txt
<delivery installation path>/SLEJAVA/test_procedures/database/multi/DBSEUser.txt
<delivery installation path>/SLEJAVA/test_procedures/control-v2/raf3-user.ctl
10000

5 SOFTWARE VALIDATION TESTING

5.1 Test set-up

During the validation activity a test has been considered successful not only if the last UNBIND operation invoked by the user is returned with positive result from the provider, but also if all the other operations during the test have been as expected from the command files.
The command files listed in the following sections can be found in directories <delivery installation path>/SLE JAVA/test_procedures/control-v<1-5>. Before running the validation test, each file should be edited to update the provision period.

5.2 User validation

In this chapter, the results of the testing activity related to the first scenario are shown.

5.2.1 *RAF*

The following table provides the complete list of validating tests run to validate the RAF service.

Version	User command file	Provider command file	Result
1	raf1-user	raf1-provider	ok
1	raf2-user	raf2-provider	ok
1	raf3-user	raf3-provider	ok
1	raf4-user	raf4-provider	ok
1	raf5-user	raf5-provider	ok
2	raf1-user	raf1-provider	ok
2	raf2-user	raf2-provider	ok
2	raf3-user	raf3-provider	ok
2	raf4-user	raf4-provider	ok
2	raf5-user	raf5-provider	ok
3	raf1-user	raf1-provider	ok
3	raf2-user	raf2-provider	ok
3	raf3-user	raf3-provider	ok
3	raf4-user	raf4-provider	ok
3	raf5-user	raf5-provider	ok
4	raf1-user	raf1-provider	ok
4	raf2-user	raf2-provider	ok
4	raf3-user	raf3-provider	ok
4	raf4-user	raf4-provider	ok
4	raf5-user	raf5-provider	ok
5	raf1-user	raf1-provider	ok
5	raf2-user	raf2-provider	ok
5	raf3-user	raf3-provider	ok
5	raf4-user	raf4-provider	ok
5	raf5-user	raf5-provider	ok

5.2.2 *RCF*

The following table provides the complete list of validating tests run to validate the RCF service.

Version	User command file	Provider command file	Result
1	rcf1-user	rcf1-provider	ok
1	rcf2-user	rcf2-provider	ok
1	rcf3-user	rcf3-provider	ok
1	rcf4-user	rcf4-provider	ok
1	rcf5-user	rcf5-provider	ok
2	rcf1-user	rcf1-provider	ok

2	rcf2-user	rcf2-provider	ok
2	rcf3-user	rcf3-provider	ok
2	rcf4-user	rcf4-provider	ok
2	rcf5-user	rcf5-provider	ok
3	rcf1-user	rcf1-provider	ok
3	rcf2-user	rcf2-provider	ok
3	rcf3-user	rcf3-provider	ok
3	rcf4-user	rcf4-provider	ok
3	rcf5-user	rcf5-provider	ok
4	rcf1-user	rcf1-provider	ok
4	rcf4-user	rcf4-provider	ok
5	rcf1-user	rcf1-provider	ok
5	rcf2-user	rcf2-provider	ok
5	rcf3-user	rcf3-provider	ok
5	rcf4-user	rcf4-provider	ok
5	rcf5-user	rcf5-provider	ok

5.2.3 *CLTU*

The following table provides the complete list of validating tests run to validate the CLTU services.

Version	User command file	Provider command file	Result
1	cltu1-user	cltu1-provider	ok
1	cltu2-user	cltu2-provider	ok
1	cltu3-user	cltu3-provider	ok
2	cltu1-user	cltu1-provider	ok
2	cltu2-user	cltu2-provider	ok
2	cltu3-user	cltu3-provider	ok
3	cltu1-user	cltu1-provider	ok
3	cltu2-user	cltu2-provider	ok
3	cltu3-user	cltu3-provider	ok
4	cltu1-user	cltu1-provider	ok
4	cltu2-user	cltu2-provider	ok
4	cltu3-user	cltu3-provider	ok
4	cltu5-user	cltu5-provider	ok
5	cltu1-user	cltu1-provider	ok
5	cltu2-user	cltu2-provider	ok
5	cltu3-user	cltu3-provider	ok

5.2.4 *ROCF*

The following table provides the complete list of validating tests run to validate the ROCF services.

Version	User command file	Provider command file	Result
1	rocf1-user	rocf1-provider	ok
1	rocf2-user	rocf2-provider	ok
1	rocf3-user	rocf3-provider	ok
1	rocf4-user	rocf4-provider	ok
1	rocf5-user	rocf5-provider	ok
2	rocf1-user	rocf1-provider	ok
2	rocf2-user	rocf2-provider	ok
2	rocf3-user	rocf3-provider	ok
2	rocf4-user	rocf4-provider	ok
2	rocf5-user	rocf5-provider	ok

5	rocf1-user	rocf1-provider	ok
5	rocf2-user	rocf2-provider	ok
5	rocf3-user	rocf3-provider	ok
5	rocf4-user	rocf4-provider	ok
5	rocf5-user	rocf5-provider	ok

5.2.5 *FSP*

The following table provides the complete list of validating tests run to validate the FSP services.

Version	User command file	Provider command file	Result
1	fsp1-user	fsp1-provider	ok
1	fsp2-user	fsp2-provider	ok
1	fsp3-user	fsp3-provider	ok
1	fsp4-user	fsp4-provider	ok
2	fsp1-user	fsp1-provider	ok
2	fsp2-user	fsp2-provider	ok
2	fsp3-user	fsp3-provider	ok
2	fsp4-user	fsp4-provider	ok
5	fsp1-user	fsp1-provider	ok
5	fsp2-user	fsp2-provider	ok
5	fsp3-user	fsp3-provider	ok
5	fsp4-user	fsp4-provider	ok

5.3 Provider validation

In this chapter, the results of the testing activity related to the second scenario are shown.

5.3.1 *RAF*

The following table provides the complete list of validating tests run to validate the RAF services.

Version	User command file	Provider command file	Result
1	raf1-user	raf1-provider	ok
1	raf2-user	raf2-provider	ok
1	raf3-user	raf3-provider	ok
1	raf4-user	raf4-provider	ok
1	raf5-user	raf5-provider	ok
2	raf1-user	raf1-provider	ok
2	raf2-user	raf2-provider	ok
2	raf3-user	raf3-provider	ok
2	raf4-user	raf4-provider	ok
2	raf5-user	raf5-provider	ok
3	raf1-user	raf1-provider	ok
3	raf2-user	raf2-provider	ok
3	raf3-user	raf3-provider	ok
3	raf4-user	raf4-provider	ok
3	raf5-user	raf5-provider	ok
4	raf1-user	raf1-provider	ok
4	raf2-user	raf2-provider	ok
4	raf3-user	raf3-provider	ok
4	raf4-user	raf4-provider	ok
4	raf5-user	raf5-provider	ok
5	raf1-user	raf1-provider	ok
5	raf2-user	raf2-provider	ok

5	raf3-user	raf3-provider	ok
5	raf4-user	raf4-provider	ok
5	raf5-user	raf5-provider	ok

5.3.2 *RCF*

The following table provides the complete list of validating tests run to validate the RCF services.

Version	**User command file**	**Provider command file**	**Result**
1	rcf1-user	rcf1-provider	ok
1	rcf2-user	rcf2-provider	ok
1	rcf3-user	rcf3-provider	ok
1	rcf4-user	rcf4-provider	ok
1	rcf5-user	rcf5-provider	ok
2	rcf1-user	rcf1-provider	ok
2	rcf2-user	rcf2-provider	ok
2	rcf3-user	rcf3-provider	ok
2	rcf4-user	rcf4-provider	ok
2	rcf5-user	rcf5-provider	ok
3	rcf1-user	rcf1-provider	ok
3	rcf2-user	rcf2-provider	ok
3	rcf3-user	rcf3-provider	ok
3	rcf4-user	rcf4-provider	ok
3	rcf5-user	rcf5-provider	ok
4	rcf1-user	rcf1-provider	ok
4	rcf4-user	rcf4-provider	ok
5	rcf1-user	rcf1-provider	ok
5	rcf2-user	rcf2-provider	ok
5	rcf3-user	rcf3-provider	ok
5	rcf4-user	rcf4-provider	ok
5	rcf5-user	rcf5-provider	ok

5.3.3 *CLTU*

The following table provides the complete list of validating tests run to validate the CLTU services.

Version	**User command file**	**Provider command file**	**Result**
1	cltu1-user	cltu1-provider	ok
1	cltu2-user	cltu2-provider	ok
1	cltu3-user	cltu3-provider	ok
2	cltu1-user	cltu1-provider	ok
2	cltu2-user	cltu2-provider	ok
2	cltu3-user	cltu3-provider	ok
3	cltu1-user	cltu1-provider	ok
3	cltu2-user	cltu2-provider	ok
3	cltu3-user	cltu3-provider	ok
4	cltu1-user	cltu1-provider	ok
4	cltu2-user	cltu2-provider	ok
4	cltu3-user	cltu3-provider	ok
4	cltu5-user	cltu5-provider	ok
5	cltu1-user	cltu1-provider	ok
5	cltu2-user	cltu2-provider	ok
5	cltu3-user	cltu3-provider	ok

5.3.4 *ROCF*

The following table provides the complete list of validating tests run to validate the ROCF services**.**

Version	User command file	Provider command file	Result
1	rocf1-user	rocf1-provider	ok
1	rocf2-user	rocf2-provider	ok
1	rocf3-user	rocf3-provider	ok
1	rocf4-user	rocf4-provider	ok
1	rocf5-user	rocf5-provider	ok
2	rocf1-user	rocf1-provider	ok
2	rocf2-user	rocf2-provider	ok
2	rocf3-user	rocf3-provider	ok
2	rocf4-user	rocf4-provider	ok
2	rocf5-user	rocf5-provider	ok
5	rocf1-user	rocf1-provider	ok
5	rocf2-user	rocf2-provider	ok
5	rocf3-user	rocf3-provider	ok
5	rocf4-user	rocf4-provider	ok
5	rocf5-user	rocf5-provider	ok

5.3.5 *FSP*

The following table provides the complete list of validating tests run to validate the FSP services**.**

Version	User command file	Provider command file	Result
1	fsp1-user	fsp1-provider	ok
1	fsp2-user	fsp2-provider	ok
1	fsp3-user	fsp3-provider	ok
1	fsp4-user	fsp4-provider	ok
2	fsp1-user	fsp1-provider	ok
2	fsp2-user	fsp2-provider	ok
2	fsp3-user	fsp3-provider	ok
2	fsp4-user	fsp4-provider	ok
5	fsp1-user	fsp1-provider	ok
5	fsp2-user	fsp2-provider	ok
5	fsp3-user	fsp3-provider	ok
5	fsp4-user	fsp4-provider	ok

6 SLE API JAVA VALIDATION WITHIN THE OSGI CONTAINER

The SLE JAVA API within the OSGi container can be validated using Pax Exam and the applications contained in the <delivery installation path>/TestOsgiUser and <delivery installation path>/TestOsgiProvider projects.
In the following description it is assumed that all the projects have already been built typing the ant command at highest level, as explained in [RD-1].
The monitored SICF folder is the <delivery installation path>/TestRunner/watched and the <delivery installation path>/TestRunner/watched/SICF.txt file contains the configuration information of the Service Instance that will be used for validation:

```
BEGIN_GROUP = R-AF-TS-P$1;
   service-instance-id = "sagr=SAGR.spack=SPACK.rsl-fg=RSL-FG.raf=onlt1";
   service-instance-start-time = 2018-265T10:00;
   service-instance-stop-time = 2019-265T10:00;
   initiator-id = "tester1";
   responder-id = "tester1";
   responder-port-id = "Harness_Port_1";
   return-timeout-period = 60;
   delivery-mode = TIMELY_ONLINE;
   initiator = USER;
   permitted-frame-quality = "allFrames.erredFramesOnly.goodFramesOnly";
   latency-limit = 1;
   transfer-buffer-size = 240;
END_GROUP = R-AF-TS-P$1;
```

6.1 User validation

The <delivery installation path>/TestOsgiUser project contains a test application that registers to the Service Instance with identifier `sagr=SAGR.spack=SPACK.rsl-fg=RSL-FG.raf=onlt1` and executes the following actions:

- sends a BIND operation,
- waits for a BIND return,
- sends an UNBIND operation,
- waits for the UNBIND return.

In order to validate the User side, the Test Harness for the Provider, the Communication Server and the Default Logger should be started following the steps listed in section 4.3.
For this test the following path definitions are used:

`<proxy database provider>`	<delivery installation path>/SLEJAVA/test_procedures/database/DBProxyProvider
`<service element provider database>`	<delivery installation path>/SLEJAVA/test_procedures/database/DBSEProvider
`<command file provider>`	<delivery installation path>/SLEJAVA/test_procedures/control-v2/raf1-provider-osgi.ctl

Once these processes have been started, enter the TestRunner project folder and type the command:

```
ant –f ./build_custom.xml user
```

The test has been run and considered successful since the sequence of the operations is as expected and no errors are generated.

6.2 Provider validation

The <delivery installation path>/TestOsgiProvider project contains a test application that registers to the Service Instance with identifier `sagr=SAGR.spack=SPACK.rsl-fg=RSL-FG.raf=onlt1` and executes the following actions:

- waits for a BIND invocation,
- sends a positive BIND return,
- waits for an UNBIND invocation,
- sends a positive UNBIND return.

In order to validate the Provider, the Communication Server and the Default Logger should be started before, following the steps listed in section 4.3.
For this test the following path definitions are used:

`<proxy database provider>`	<delivery installation path>/SLEJAVA/test_procedures/database/DBProxyProvider
`<service element provider database>`	<delivery installation path>/SLEJAVA/test_procedures/database/DBSEProvider
`<command file provider>`	<delivery installation path>/SLEJAVA/test_procedures/control-v2/raf1-user-osgi.ctl

Once these processes have been started, enter the TestRunner project folder and type the command:
```
ant -f ./build_custom.xml provider
```

Then start the Test Harness User following the indications provided in section 4.4.
The test has been run and considered successful since the sequence of the operation is as expected and no errors are generated.