DataGRID

EDG-VOMS-ADMIN TESTING PLAN

Document identifier:	**edg-voms-admin-testplan**
EDMS id:	
Date:	January 14, 2004
Work package:	**WP07: Security**
Partner(s):	**CERN, ELTE**
Lead Partner:	**CERN**
Document status:	**WORKING DRAFT**
Author(s):	?kos Frohner
File:	**edg-voms-admin-testplan**

Abstract: A document to describe the planned and executed tests on this package.

CONTENTS

Document Log

Issue	Date	Comment	Author
0-1	2003-08-04	First draft	?kos Frohner

1. OVERVIEW

There are four major components of the edg-voms-admin package:

- database abstraction classes
- service implementation classes for the SOAP interface
- web servlets for the web interface
- client side applications

These components (except the database abstraction classes) have to be tested for correct functionality, reliability and secure operation.

2. GENERAL CONFIGURATION

Most of these tests require a running and properly configured database back-end and installed software. There are *targets* in the build configuration (build.xml) to automate these procedures:

test.install Install the software into a test environment.

test.configure Configure the software.

test.unconfigure Remove the test configuration.

test.remove Remove the installed software.

These targets can be also used independently: if there is an already installed system, then one can skip *test.install* (and *test.remove*) and only configure the system for testing.

Since edg-voms-admin is prepared for multi-instance operation the testing cannot even disturb a production system, given the name of the testing Virtual Organisation (VO) is different from others.

2.1. CONFIGURATION FILE

The parameters of the test targets are controlled by the test.properties file. The build process looks for the file at the following locations (in this order):

1. test.properties (in the current/top-level directory)
2. build/test.properties (must be generated here!)
3. config/tests/test.properties (defaults)

The configuration file can define the following values (with defaults):

```
test.vo TestVO
test.voalias test
test.port 8443
test.dba.user root
test.dba.pwd root
test.prefix build/tests/edg
```

2.2. INSTALL AND REMOVE

The *test.install* target creates some essential directories and calls *install* to do the rest of the job. It has the same effect as if the software was installed from RPM, although the prefix directory will be most likely different: test.prefix.

2.3. CONFIGURATION

The *test.configure* target calls the edg-voms-admin-configure script to generate the configuration files and create a schema in the database.

The *test.unconfigure* removes all the changes made by the previous option: it removes the files and also the created database!

It would be quite inconvenient in case a test fails, because it would prevent the detailed investigation. To prevent such an accident the *test.unconfigure* target fails to execute if the *test.failed* property is set in the build process.

3. DB ABSTRACTION AND SERVICE CLASSES

Unit tests for the database abstraction layer and for the services are created along the implementation classes.

For example the VOMSAdmin interface is implemented as VOMSAdminSoapBindingImpl class and the corresponding unit test is in VOMSAdminSoapBindingImplTest[1]

These unit tests are using the JUnit unit testing framework for Java, which can be easily integrated into the build system.

3.1. AUTHORISATION TESTS

The main difficulty in these tests is the proper configuration of the environment, including the client's identity. These settings are not passed as parameters to the functions, but taken from the SecurityContext object[2].

To simplify the testing of the pure functionality we allow bypassing of the authorisation checks in special circumstances. These special parameters can be simply configured by the InitSecurityContext.initBypassSC() method.

To test authorisation we can also configure the context to artificial client identities with a code like to this:

```
SecurityContext sc = new SecurityContext ();
SecurityContext.setCurrentContext (sc);

sc.setAuthorizedAttributes (Arrays.asList (attributes));

sc.setClientName (clientName);
sc.setIssuerName (clientIssuer);
```

[1]look for *Test.java classes in the org.edg.security.voms.service package!
[2]thread local context; see more in the edg-java-security package

3.2. Running the Tests

To run the test suite execute the following command:

```
ant service-test
```

After running the tests one can generate the detailed description of the unit tests and the corresponding reports by

```
ant doc.testdoc
```

the result is placed in the dist/doc/tests directory.

4. SOAP Interface

SOAP interface over the service classes is provided by *Axis*.

The Axis framework provides the generation of WSDL from Java interfaces, Java wrapper classes from WSDL and the invocation service embedded in Tomcat.

Since we did not modify this layer we will not test this functionality separately.

5. Web Interface, Servlets

Not implemented yet.

The web interface is provided by the org.edg.security.voms.webui.* packages by servlets, which have their own tests alongside[3]

The web interface is built on top of the service classes, so the tests are focused on the interpretation of parameters and rendering the results, rather than on the functionality itself.

There are two main groups of these tests: with and without the servlet container.

5.1. Unit Tests

The interpretation of the parameters and rendering of the results can be tested without the servlet container by simulating the *requests* and analysing the *responses*. In this work a great deal of the routine work is provided by the *MockObjects*[4] and HttpUnit[5] frameworks.

5.2. Integration Tests

Not implemented yet.

Of course certain tests cannot be run without the configuring the servlet container and processing requests through the whole system. Such a test simulates a deployment and configuration and tests the various components in their production environment.

For these tests we use the Cactus[6] framework and the Tomcat servlet container.

Test cases cover:

[3]look for *Test.java classes in the org.edg.security.voms.webui package!
[4]http://www.mockobjects.org
[5]http://httpunit.sourceforge.net
[6]http://jakarta.apache.org/cactus/

- basic configuration tests (static pages, Axis and web UI servlets)
- security settings (CA, CRL and service certificate)
- authentication and authorisation (client certificate)

The test cases doesn't use real certificates, but ones generated only for test purposes.

6. CLIENT SIDE APPLICATIONS

Not implemented yet.

The final stage of the testing is via client applications. Since the web interface's functionality is covered by the integration tests we only test the command line clients. These tests also use the generated test certificates.

These tests can be configured on a production system as well, so this test suit packaged into RPM to provide a deployment test suite.