Gateway Verification and Validation Plan

Service/Component description and version

Description is available in Gateway SRC

This test plan is updated for the component's version: 6.4.0

Gateway can be installed on a separate machine or on the same machine where one or more back-end servers are installed.

Functionality tests

[TO BE DONE - Add remaining cases, the current list is complete only for the initial release in EMI-1. Note that most of those test cases marked as not implemented are in fact nearly available - only some details are missing.]

Resilience against invalid requests (implemented: gw_wrongHdr)

Gateway must deny requests with invalid SOAP formatting. Especially SOAP headers with invalid data must not be accepted. Correct but too big (to be kept in memory) headers must be denied too.

Dynamic site registration (implemented: gw_dynReg)

Test whether gateway allows for dynamic registration of new backend sites. The registration using HTTP POST method to the VSITE_REGISTRATION_REQUEST resource at the gateway should succeed if and only if the dynamic registration is enabled in configuration.

Forwarding of GET requests (implemented: gw_fwGetReq)

Gateway must properly forward GET requests to the backend site. This must be tested with a simple backend service and HTTP client. The original request must be forwarded and response produced by the service returned to the client. The test is passed only if the response received by the client is unmodified.

Forwarding of POST/SOAP requests (implemented: gw_fwPostReq)

Gateway must properly forward POST requests to the backend site. This must be tested with a simple backend HTTP service and HTTP client. The original SOAP request body must be forwarded and response produced by the service returned to the client. The test is passed only if:

- the SOAP Body of the client request is unmodified at the backend server side.
- all SOAP Headers of the client request are present and are unmodified at the backend server side (but new headers may be present too).
- the response received by the client the same as what the backend server produced.

Security of dynamic site registration (implemented: gw_secDyn)

Verify that the dynamic registration filters configured with registration.deny and registration.allow work correctly:

- host matching allow rule should be accepted when there is no deny rule
- host matching deny rule should be denied even if are entered in accept rule

Authentication (NOT implemented)

Test whether gateway properly extracts client's identity from SSL and inserts it as an SAML assertion in SOAP header.

Integration tests

[TO BE DONE]

Performance tests

1. Test performance of request processing by invoking 10000 operations with do-nothing implementations of the back-end service. PASS criteria: at least 1000 exchanges per second on a commodity hardware. [implemented]
2. Measure data transfer speed by streaming 100MB of data through gateway back and forth. **TBD: define pass criteria [NOT implemented]**

Scalability tests

1. Test gateway by 100 of concurrent clients that should submit 1000 requests each to do-nothing service behind the gateway. Note that for this test maximum number of concurrent connections per service should be increased to 100 (or different services should be used). **TBD: define pass criteria [NOT implemented]**
2. Invoke in parallel 10 instances of the transfer speed performance test and measure decreasing of the added transfer speed. **TBD: define pass criteria [NOT implemented]**

Standard Compliance/Conformance tests

N/A

Regression tests and unit tests

Unit tests coverage must be included in the test report. All bugs reported should have an automated regression test attached if it is possible. Otherwise manual bug checking procedure should be added to this section. Note that this applies to bugs reported from the 1.11.2010.

Regression tests to be performed manually:

- N/A

Deployment scenarios

Gateway is deployed as an administrative domain-central service which might serve one or more domain's servers.

1. Install from the binary package.
2. Add a backend site to the configuration (to the 'connection.properties' file).
3. Start service using system init script by root.
4. There should be no errors/warnings in the log file.
5. Server's process uid should be 'unicore'.
6. A request from the client to the backend service must be performed and succeed (along with passing a response back).
7. There should be no errors/warnings in the log file.

8. Check also service status with =/etc/init.d/unicore-gateway status.
9. Eventually check for PID files.

This topic: EMI > UNICOREGatewaySVVP
Topic revision: r8 - 2011-08-08 - KarolStasiakExternal