Integrating the Healthcare Enterprise

IHE/MESA XDS-I Tests User Guide

École de technologie supérieure,
University of Quebec
1100 Notre-Dame Street West
Montreal, QC H3W 1K3
Canada
514.396.8800 (Voice)
514.396.8950 (Fax)

Table of Contents

1 Introduction

1.1 Contact

All technical comments or problems like bugs or unsolvable problems may be reported to
Renaud Berube <renaud.berube.1@ens.etsmtl.ca>.

1.2 Installation

Each test case requires different components from the XDSI-I test distribution. The installation of each component is described separately in the installation guide. The description for each test case contains a listing of the requirements.

1.3 Logging

During a test execution all created logs and evidence files go to a directory owned by the execution (see <1.8 Test Execution>). If there is no explicitly started test execution, an implicitly started execution is used for logging purposes having the id 'default'.

1.4 Choosing profile

Like XDS, XDS-I toolkit may support newer web standards in the future. For now, it follows XDS.a profile standards like SOAP 1.1 and ebXML 2.1. XDS.b profile will be supported in the future.

By default, XDS-I uses profile a.

To choose in which profile you want to operate, simply open the file 'XDSITEST_HOME/xdsitest/xdsitest.properties' and edit value of property ‘xdsitest.profile’.

Make sure to use the URLs that correspond to the profile (see NIST IHE web site).

You must shutdown and restart the Tomcat server after you’ve edited values in this file.

1.5 Loading Test Data

The following procedure only needs to be performed if a test requires already submitted DICOM manifests. This is the case for consumer tests and freestyle tests originating from your own needs.

The test distribution includes a tool that enables one to share almost every DICOM instance. A set of prepared samples is packaged with the test distribution. The samples can be found in 'XDSITEST_HOME/data/sample_submission'. Each subdirectory corresponds to one sample.

1. Start the ImageCtn (see installation guide).
2. On the command line (use 'setenv' before) issue the command
   java ca.etsmtl.ihe.xdsitest.docsource.SimplePublisher <sample directory>

   OR

   java ca.etsmtl.ihe.xdsitest.docsource.SimplePublisher <dcm SOPinstance>
   This step may be repeated.
3. Stop the ImageCtn.

By using the ‘-k’ switch with SimplePublisher and MultiPublisher, the submission request won’t be sent to the repository. The gererated sreport and submission_request will be available for use in test 3202 and 3203.

Each execution of step 2 submits a manifest referencing the sample DICOM instance. Among the files

created in the sample directory there are 'document_entry.xml' and 'submission_set.xml'. These two files may be consulted in order to formulate a registry query using the information from the submission.

If you want to submit other DICOM instances, these must be in the database of the ImageCtn (see installation guide). Create a file 'publication.properties' analogeous to the files with the same name for the samples and perform the above procedure. For this, you can issue the command

java ca.etsmtl.ihe.xdsitest.docsource.PublicationCreator <DICOM instance>

It should create the properties file with the required properties. It should work with the DICOM manifest and put each referenced SOP instance into the child folders (numbers), and get the AETitle for the instances. The publication.properties in the DICOM name folder (parent) cannot be retrieve as the manifest isn't in the imagectn database.

Though, the easiest way to populate the registry now is by taking either taking a .dcm SOPinstance and passing it to SimplePublisher (read above) or sending a batch of .dcm via MultiPublisher

java ca.etsmtl.ihe.xdsitest.docsource.MultiPublisher <DCMs SopInstance directory>

It will take all .dcm and generate a default KOS for each one, one at a time and send them to the repository. You'll have to edit <XDSIHOME>/xdsitest/xml/xdsi/docsource/constants.xsl as described in the file (Creating patientID on NIST).

1.6 Configuration Test Provider

- For each tested application the test server has to know some parameters. Copy the file 'XDSITEST_HOME/data/testserver/sample_user_XXXX.conf' to a file '<some user name>.conf' in the same directory and adjust its content to the appropriate values. Each parameter is preceded by a comment that explains its purpose and enables the user to choose an appropriate value.
  This manipulation does not require a restart of the test server. The parameters are read at the beginning of each test execution.

1.7 Configuration Test User

In order to perform the transactions required by the individual tests, the tested application has to know some parameters. These are listed below:

- RAD-54 – 'Provide and Register Imaging Document Set'
  The SOAP messages are posted to
  http://<test server host>:8080/xdsitest/services/xdsadaptorA/adaptor
  or
  http://<test server host>:8080/xdsitest/services/xdsadaptorA/adaptor?check
  The second alternative differs from the first one by not registering the provided documents. The attachments and metadata are only analyzed. There are two reasons to use this alternative (if allowed by the test case): metadata containing the same uniqueIds may be sent more than once; the component 'ebxmlrr Registry' is not needed. However, by not trying to register the metadata, some unresolved references from an ebXML perspective are not detected.
- ITI-16 – 'Query Registry'
  The SOAP messages are posted to
  http://<test server host>:8080/xdsitest/services/xdsadaptor/adaptor
- RAD-55 – 'WADO Retrieve'
  The RetrieveAETitle 'XDSI_IMAGECTN' maps to

http://<test server host>:8080/xdsitest/wado

- RAD-16, RAD-17, RAD-27, RAD-31, RAD-45 – 'Retrieve Some DICOM Instance'
  The application entity 'XDSI_IMAGECTN' has the presentation address
  <test server host>:10101

1.8 Test Execution

All tests are performed using an identical procedure.

- Make sure the test server is started and get in your web browser
  http://<test server>:8080/xdsitest/control
  Select a test case and a test user configuration and start the test execution.
- Go through the different stages and perform the indicated actions.
  After having advanced one stage look at the 'Stage' information on the status page. Proceed only if there is no error indication after the stage number.
- All created logs and evidence files go to the directory 'XDSITEST_HOME/data/testserver/<test id>'. In case of problems look at these files.
- Evaluation is performed automatically at the end of the test execution. The result is displayed on the status page and contained in the logs.
  In order to submit a result, archive the above mentioned directory and send it in compressed form to your project manager.
  This procedure may also be chosen if there are problems or unsolvable error situations during the test execution.

2 Test Cases

2.1 Test Case 3201: Imaging Document Consumer – Retrieve shared DICOM instances

The Imaging Document Consumer retrieves DICOM instances referenced in a DICOM manifest.

2.1.1 References

ITI TF-2: 3.16, 3.17; RAD TF-2: 4.16, 4.17, 4.27, 4.31, 4.45, 4.55

2.1.2 Actors Tested

Imaging Document Consumer

2.1.3 Requirements

- XDS-I Test Web Application
- XDS-I Test Command Line Applications
- ImageCtn
- <1.5 Loading Test Data>

2.1.4 Description

The document consumer queries the registry in order to obtain the URI of published DICOM manifests. It gets at least one of these manifests from the repository and retrieves at least one referenced DICOM

instance using the information in the manifest along with its configuration. The retrieval may be done by either C-MOVE or WADO.

In order to pass the test, the document consumer must retrieve at least one DICOM instance that is referenced in a retrieved manifest. The URL of the later one must be obtained from a registry query.

2.1.5 Instructions

Execute test 3201 as described in <1.8 Test Execution>.

2.2 Test Case 3202: Imaging Document Source – Share DICOM instances available via C-MOVE

The Imaging Document Source provides a DICOM manifest that contains references to DICOM instances available for retrieval via C-MOVE.

2.2.1 References

RAD TF-2: 4.16, 4.17, 4.27, 4.31, 4.45, 4.54

2.2.2 Actors Tested

Imaging Document Source

2.2.3 Requirements

- XDS-I Test Web Application

2.2.4 Description

The document source issues a 'Provide and Register Imaging Document Set' transaction (RAD-54) containing at least one DICOM manifest.

A basic check is performed for the sent metadata (validation against the ebXML schemas version 2.1 – the official version for XDS, structural semantics specified by XDS). The contained DICOM manifest is analyzed and the referenced DICOM instances are retrieved via C-MOVE by the test server.

In order to pass the test, the document source must provide at least one valid DICOM manifest to the repository. The test server must succeed in retrieving all DICOM instances referenced in the provided manifest(s).

2.2.5 Instructions

Execute test 3202 as described in <1.8 Test Execution>. The test may be performed by using the registry adaptor without upstream registry (see <1.7 Configuration Test User>).

2.3 Test Case 3203: Imaging Document Source – Share DICOM instances available via WADO

The Imaging Document Source provides a DICOM manifest that contains references to DICOM instances available for retrieval via WADO.

2.3.1 References

RAD TF-2: 4.54, 4.55

2.3.2 Actors Tested

Imaging Document Source

2.3.3 Requirements

- XDS-I Test Web Application

2.3.4 Description

The document source issues a 'Provide and Register Imaging Document Set' transaction (RAD-54) containing at least one DICOM manifest.

A basic check is performed for the sent metadata (validation against the ebXML schemas version 2.1 – the official version for XDS, structural semantics specified by XDS). The contained DICOM manifest is analyzed and the referenced DICOM instances are retrieved via WADO by the test server.

In order to pass the test, the document source must provide at least one valid DICOM manifest to the repository. The test server must succeed in retrieving all DICOM instances referenced in the provided manifest(s).

2.3.5 Instructions

Execute test 3203 as described in <1.8 Test Execution>. The test may be performed by using the registry adaptor without upstream registry (see <1.7 Configuration Test User>).

3 Test Execution

Every new TestExecution created by the web interface will start by creating it own unique logging folder. In that folder, the test will log the activities of the different actors and keep the submissions for evaluation at the last step of the test execution. The folder will keep every request messages sent to it (xxxx_request.xml), every response it sends (xxxx_response.xml) and all request attachment file (xxxx_attachment.bin). In addition, since the repository is now located elsewhere, every retrieved file (KOS) will be saved in the TestExecution work directory (xxxx_retrievedKOS.content).

For each step, the web interface will tell what the tested application (user) should do with the test server. It is important that you do the action at that point of the test as if you complete the task before or after the step, it won't be evaluated correctly. The evaluation checks for the right task at the right step.

3.1 Default Test

Every transaction is logged on the test server, and if no specific test has been started, the default test will be used to log those transactions. All information will be dumped into the default folder of the testserver (<XDSIHOME>/data/testserver). No evaluation phase is included in that "test".

3.2 Consumer Actor Test (Test 3201)

This instance will test an Imaging Document Consumer actor, and provides the Document Registry and the Document Repository. It will, at some point, start up an Imaging Document Source actor.

3.2.1 Step 1:

Test Server operations:

The server starts by instantiating an ImageCtn object. The test then builds up the database for the source ImageCtn with all the DICOM instance of the <XDSITEST_HOME>/data/imagectn directory. It logs all instances, and writes the index into the "db_" folder of the test execution. It then builds up the configuration file for the ImageCtn process to be started.

Note that some information is provided by the "xdsitest.properties" file. The aeTitles and ports shouldn't be changed. For connection errors, you should check the information in the xdsitest.properties and user.conf files and start a new test. Do NOT modify the imagectn configuration file as for each test, the server reloads the content from the properties file.

Finally, if no exception has occurred, it starts up an ImageCTN process, with the configuration (.cfg) created before. It executes the test to check its success and stops it. This procedure is done mainly in order to detect problems like a used presentation address early. In this case, the most frequent error is a TCP initialization error. Either another application uses the ports or another instance of imagectn is running.

User Actions:

For the tester, at that stage, it should query the registry. The user should send a query (SQL or stored query) to the XDS adaptor (XDS-I Test Web Application, see **xdsitest.properties** for URLs). The adaptor will forward the query to the registry after validation and send the registry response to the user. Bypassing it will make the evaluation to fail later since it won't have seen the transaction.

A quick example of an SQL (profile ‘a’ only) AdhocQueryRequest that will suffice for testing:

```
<?xml version="1.0"?>
<AdhocQueryRequest xmlns="urn:oasis:names:tc:ebxml-regrep:query:xsd:2.1">
    <ResponseOption returnType = "LeafClass" returnComposedObjects="false"/>
    <SQLQuery>
    SELECT eo.id FROM ExtrinsicObject eo
        WHERE
            eo.id = 'urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
    </SQLQuery>
</AdhocQueryRequest>
```

The most important thing to know is that the root element must have the right namespace. You can use RegistryClient to send your request.

3.2.2 Step 2:

Test Server operations:

In this step, the server does nothing except logging the users operations. Wait for the user to do its action before passing to the next step.

User Actions:

From the registry response file, find the URL for the queried DICOM manifest of interest. You can retrieve the manifest with a simple web browser, pasting the URL. Every retrieved manifest will be logged by the server. You must use the XDS Adaptor URL to retrieve the manifest (http://< test server host >:8080/xdsitest/documents/xxxxx.kos). If you use the URL from the manifest directly, the test will fail at the end because it won’t have seen the transaction.

3.2.3 Step 3:

Test Server operations:

Restarts an imagectn process with the configuration created in step 1 but leaves it running so we can test retrieving DICOM instance from the server (by C-MOVE or WADO). Every output from imagectn is written into stdout_X.log file. The output will be use in step 4 for review.

User Actions:

With the manifest, retrieve the DICOM SOP instance from the source actor. Fetch the REFERENCED "StudyInstanceUID", "SeriesInstanceUID" and "SOPInstanceUID" parameters from the manifest. That information will be enough to retrieve the image.

The tested consumer must use C-MOVE of WADO to retrieve the instance. Though, by default, only a localhost consumer will be able to download it. With the complete build, the WadoServlet retrieves the DICOM manifest via C-MOVE. Since the provided WADO server is authorise to fetch from the source, using it might be the easiest way to understand the test server.

If the consumer use C-MOVE directly to the source (not using the WADO server), verify that the aeTitle, host and port number of the consumer are all written into the “sample_user_3201.conf” of the test server BEFORE starting step 1 of the test.

The user should retrieve at least one DICOM SOP instance from the retrieved KOS manifest of step 2.

3.2.4 Step 4:

Stops imagectn, and analyse the results from log.txt and imagectn (stdout_X.log). The class EvalImageCtn will parse the output file to create a list of information (UIDs). It will write in log.xml every UID extracted by the parser into a <sopInstance> child element. After creating the xml file, it analyse the logs.

NOTE:log.xml won't be a valid XML file as the root element is never close. It is use only for internal analysis.

A typical entry looks like: (For log.xml)

```
<imagectn_moved xxxx_stage="4">
    <sopInstance>2.16.840.1.113662.2.1.12345.19950126.112629.1900</sopInstance>
</imagectn_moved>
```

Evaluation:

A typical XML tree looks like: (For xxxx_analyse_log.txt)

```
<evaluation result="PASSED">
    <fileResponse value=
      "/home/stephan/xdsihome/data/testserver/20060412152607437e228/0002_response.x
ml">
        <uriKosd value=
      "http://localhost:8080/xdsitest/documents/185667299261772">
            <fileKosd value=
      "/home/stephan/xdsihome/data/fileserver/185667299261772.content">
                <sopInstance>
                    <retrieveAeTitle>XDSI_IMAGECTN</retrieveAeTitle>
                    <sopInstanceUid
      >2.16.840.1.113662.2.1.12345.19950126.112629.1900<
      /sopInstanceUid>
                    <seriesUid
      >2.16.840.1.113662.2.1.53544936282433.12345.336.1665.9990<
      /seriesUid>
                    <studyUid
      >2.16.840.1.113662.2.1.53544936282433.12345.336.16650<
      /studyUid>
                </sopInstance>
            </fileKosd>
        </uriKosd>
        <uriKosd value=
      "http://localhost:8080/xdsitest/documents/746237424336078"/>
    </fileResponse>
</evaluation>
```

Every response to a registry query request during stage 1 generates a `<fileResponse />` element.

Every 'ExtrinsicObject' in a response having mimeType 'application/dicom' and a slot named 'URI' generates a `<uriKosd />` element.

These are left joined with the activity information for the file server during stage 2. If a file with an URI from 2. is retrieved, a `<fileKosd />` element is generated.

The KOSD files are analyzed and the referenced SOP instances information is extracted. If a file is no valid KOSD, the item is dropped.

These are left joined with the activity information for the ImageCtn during stage 3. A not retrieved item is dropped.

3.3 Source Actor Test (Test 3202/3203)

The only difference between test 3202 and 3203 is the way the test server will try to retrieve the DICOM instance from the source you want to test. Test 3202 will try to retrieve via C-MOVE, and 3203 will retrieve with a WADO server.

With this test build, 3203 is set by default to use the WADO server of the build. The servlet then use C-Move to retrieve the DICOM instance in a similar manner than test 3202. It should be a good idea for the tester to provide its own WADO server (and source) for the test to check its workability.

3.3.1 Step 1:

Test Server operations:

The server only sets up the way its going to retrieve the file from the source in the last step.

User Actions:

The user should send a Submission Request with the DICOM manifest to the XDS adaptor. To test a source actor installed on a remote machine, we need to duplicate (or edit) the user file associated to the test, and change the host address (3202) or wadoUrl (3203).

AeTable warning!

Be sure that the imagectn.cfg file that the source will be using has those 2 peers authorised:

(XDSI_TEST_RCV, <HOST NAME from client, user file>, 10102), (XDSI_WADO_RCV, localhost, 10103)

There must not be any duplicated aeTitle.

3.3.2 Step 2:

The server tries to retrieve the all submitted DICOM instance from the source depending on the test selected. If at least one instance got retrieve successfully, the test is completed with success.

In the evaluation process, it will fetch all SubmissionRequest from step 1, convert all associated manifest (fileKosd) to XML files and check every sopInstance. In the evaluation file, it will write the AETitle, the UIDs and if the instance got retrieved for every submission. Here's an example of an analyse_log.txt file (for test 3202):

```
<<<< 20060713 14:48:15.324 (MoveSCU.retrieve) <<<<
>>>> 20060713 14:48:17.688 (MoveSCU.retrieve) OK >>>>
<evaluation result="PASSED">
    <fileKosd
value="c:\XDSI\xdsihome\data\testserver\20060713144653977b939\0000_attachment.bin">
        <sopInstance retrieveStatus="ok">
            <retrieveAeTitle>XDSI_IMAGECTN</retrieveAeTitle>

<sopInstanceUid>2.16.840.1.113662.2.1.12345.19950126.112629.1900</sopInstanceUid>

<seriesUid>2.16.840.1.113662.2.1.53544936282433.12345.336.1665.9990</seriesUid>

<studyUid>2.16.840.1.113662.2.1.53544936282433.12345.336.16650</studyUid>
        </sopInstance>
    </fileKosd>
</evaluation>
Evaluation result: PASSED
```