



UsingSoapUIWithAgnis

Table of Contents

1 Overview	1
1.1 Getting Started	1
1.2 Adding a Web Service	1
1.3 Proxy Settings	2
1.4 SSL Settings	2
2 Creating a soapUI Project	4
2.1 New WSDL Project	4
2.2 Import WSDL	5
2.3 Web Service Interface	6
2.4 New SOAP Request	7
3 AGNIS Web Service	9
3.1 Ping	9
3.2 Submit	9
3.3 Publish	12
3.4 Retrieve & Acknowledge	13
4 caDSR 4.0 Web Service Interface	17
4.1 Search for Forms	18
4.2 Modules Within a Form	19
4.3 Questions Within a Module	20
4.4 DataElement	20
4.5 DataElementConcept	21
4.6 ValueDomain	21
4.7 ValueDomainPermissibleValue	22
4.8 PermissibleValue	23
4.9 ValueMeaning	23
5 Additional Information	25
User Interface	25
Web Service Testing	25
Groovy Scripting	25
Conclusion	25
Appendix A: Constructing a Java Key Store	26
A.1 OpenSSL	26
A.2 PKCS12Import	26
Appendix B: Other caDSR 4.0 Interfaces	28
B.1 Web Browser Interface	28
B.2 XML-HTTP Interface	28

1 Overview

Eviware's soapUI program is an excellent free tool for web service (SOAP) testing and debugging:

<http://www.soapui.org/>

The essential soapUI 2.0 software is free and open-source, and readily available for download from [soapui.org](http://www.soapui.org). A "Pro" edition, with professional support and extended features is also available.

Some notable capabilities offered by soapUI are:

- Inspecting Web Services
- Invoking Web Services
- Functional Testing of Web Services
- Load Testing of Web Services

This document offers basic information about soapUI, also and provides examples to illustrate usage of the AGNIS web service.

Trademark Notification: AGNIS and National Marrow Donor Program are registered trademarks of the National Marrow Donor Program. eviware and soapUI are trademarks of eviware. Other trademarks are the property of their respective owners.

1.1 Getting Started

Refer to the online *Getting Started with soapUI* guide for installation instructions:

<http://www.soapui.org/gettingstarted/>

The getting started guide also illustrates several typical procedures for interacting with web services.

1.2 Adding a Web Service

Most web services use Web Service Description Language (WSDL) to specify a formal description of their features. soapUI first processes the WSDL description of a web service, and then provides a graphical interface for interacting with the web service operations.

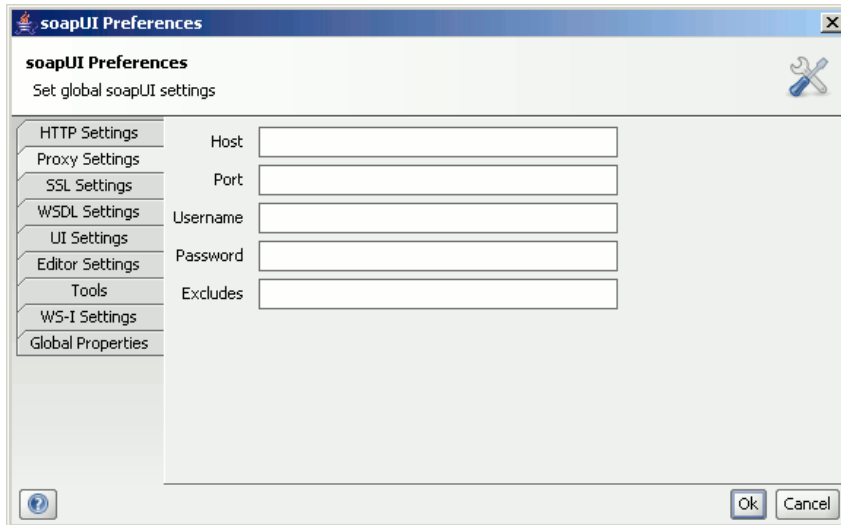
As described in the *Getting Started with soapUI* guide, the typical procedure for adding a web service to the soapUI workspace is:

1. Right-click on *Projects* and select *New WSDL Project*.
2. Input a project name and click *OK*.
3. Right-click on the project and select *Add WSDL from URL*.
4. Input the WSDL URL and click *OK*.
5. Click *No* in the *Create default requests for all operations* message box, then wait as soapUI downloads and processes the WSDL. (Clicking *Yes* here may lead to processing difficulties in some cases.)

1.3 Proxy Settings

Some computer systems are required to use a proxy server when accessing the internet. For those systems, it will be necessary to configure proxy settings within soapUI before loading the WSDL for an external web service:

1. Open the *soapUI Preferences* window (select *File*, then *Preferences*).
2. Select the *Proxy Settings* tab, input appropriate values, and click *Ok*.



1.4 SSL Settings

For a secure web service such as AGNIS, soapUI requires additional configuration.

The AGNIS web service uses Grid Security Infrastructure (GSI) security mechanisms, implemented by the Globus Toolkit, which depend on client-side X.509 certificates. The security chapter of the *Globus Toolkit 4 Programmer's Tutorial* offers an overview of certificate based security concepts, and a description of the GSI supported security mechanisms:

<http://gdp.globus.org/gt4-tutorial/multiplehtml/pt03.html>

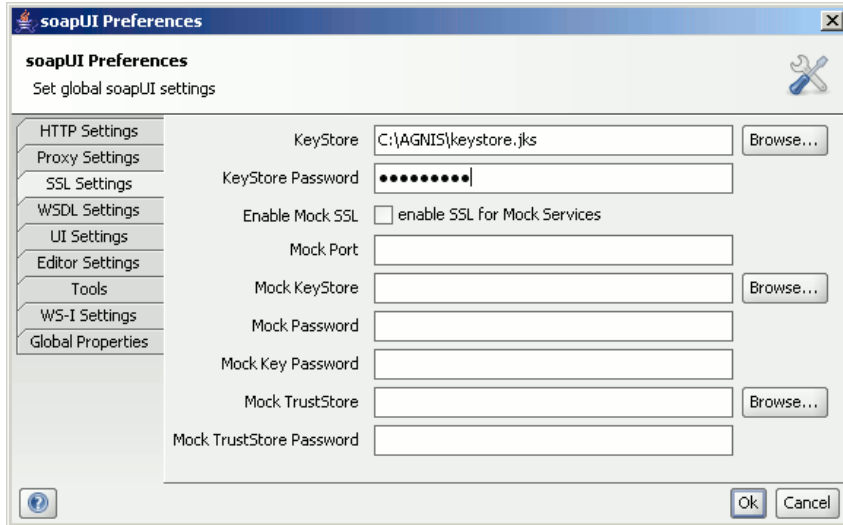
Of the GSI security mechanisms, AGNIS uses only the transport-level security scheme based on Transport Layer Security (TLS), and does not utilize the WS-Security or WS-SecureConversation message-level security schemes.

To access the secure AGNIS web services, as part of the TLS protocol, the client must authenticate using a client-side X.509 credential. For soapUI, the public key and private key components of the credential must be packaged as a Java keystore file.

See Appendix A for details of a method to convert the typical AGNIS user credential, comprised of files named `usercert.pem` and `userkey.pem`, to a Java keystore file compatible with soapUI.

After creating the keystore file, simply modify the soapUI SSL Settings to specify the keystore file location and password, as illustrated below:

1. Open the *soapUI Preferences* window (select *File*, then *Preferences*).
2. Select the *SSL Settings* tab.
3. Input the KeyStore file name and KeyStore Password, then click *Ok*.



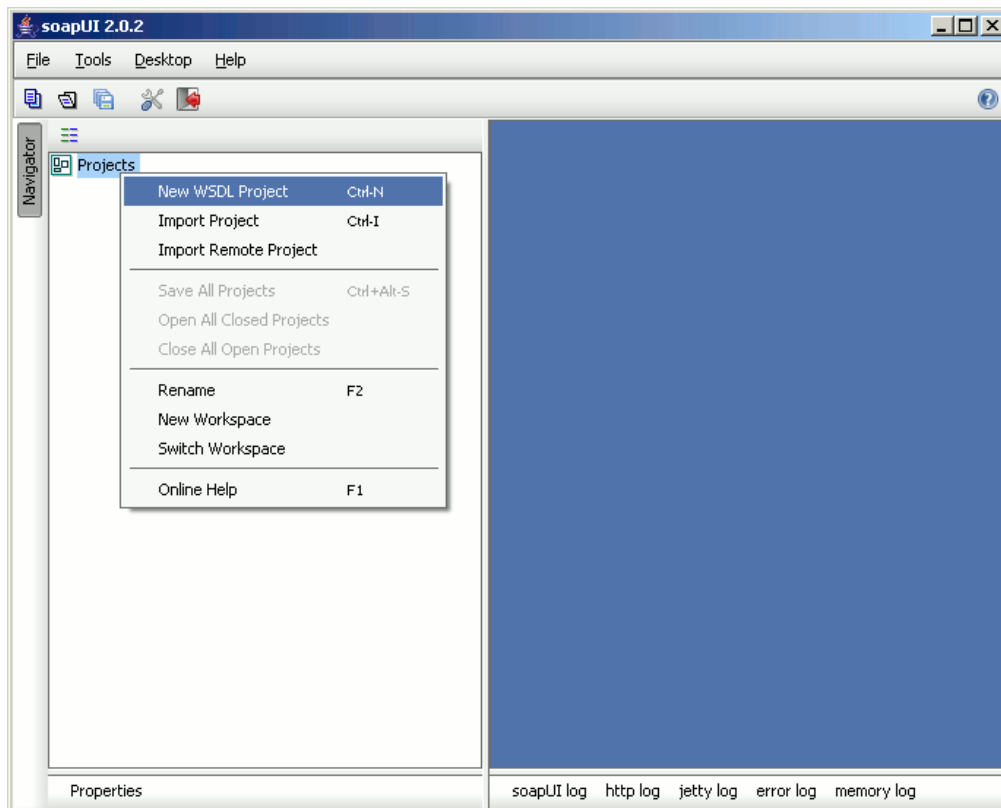
With this change to the SSL Settings, soapUI becomes capable of communicating with secure AGNIS services, using the client-side credential from the keystore file.

2 Creating a soapUI Project

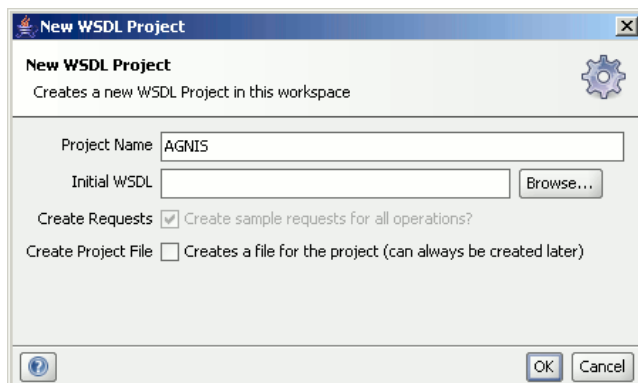
This section describes how to connect soapUI with an AGNIS web service. The procedure given here matches instructions found in the soapUI documentation, with the addition of screen shots pertinent to AGNIS. The same general procedure can be used to connect with other web services which offer a WSDL description, such as the caDSR service.

2.1 New WSDL Project

Start soapUI, then right-click on *Projects* and select *New WSDL Project*.

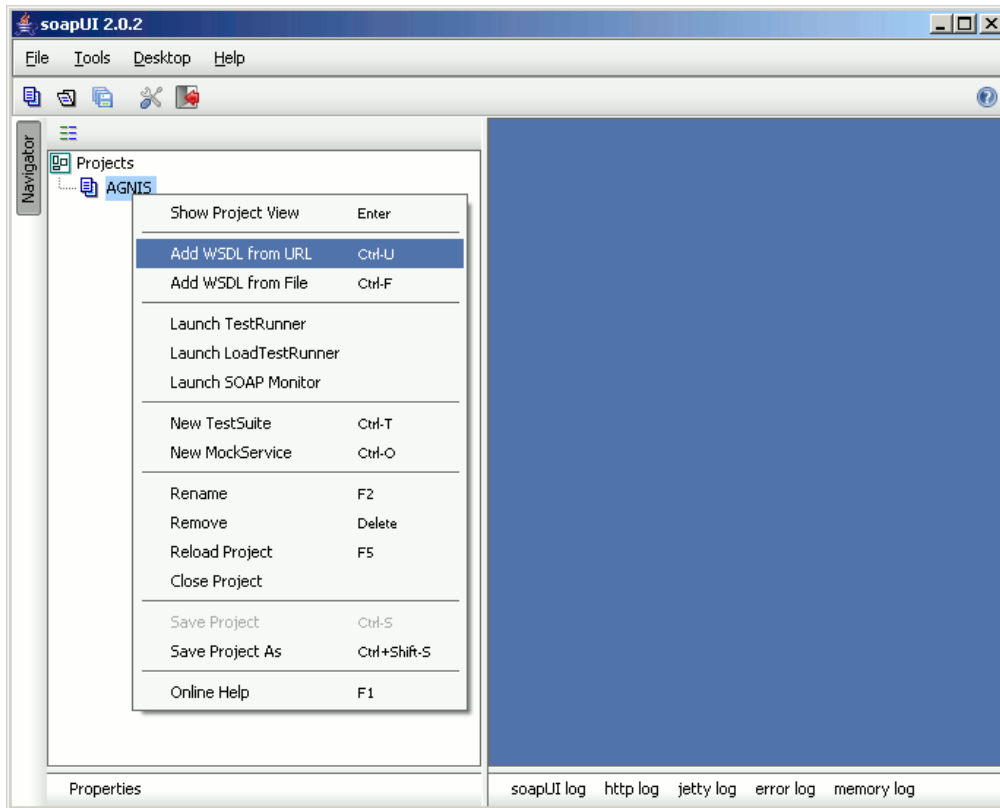


In the *New WSDL Project* dialog window, input a project name (e.g. "AGNIS") and click *OK*.



2.2 Import WSDL

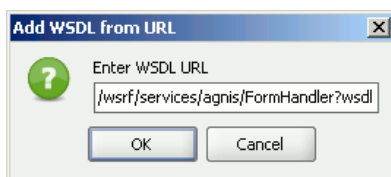
Right-click on the project and select *Add WSDL from URL*.



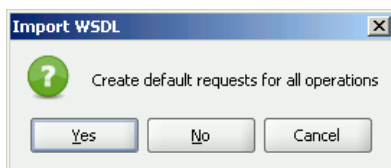
The WSDL URL for the external development and testing instance of the current AGNIS web service is:

<https://agnis-dev.nmdp.org:18443/agnis/services/FormHandler?wsdl>

In the *Add WSDL from URL* dialog window, input the appropriate WSDL URL and click *OK*.

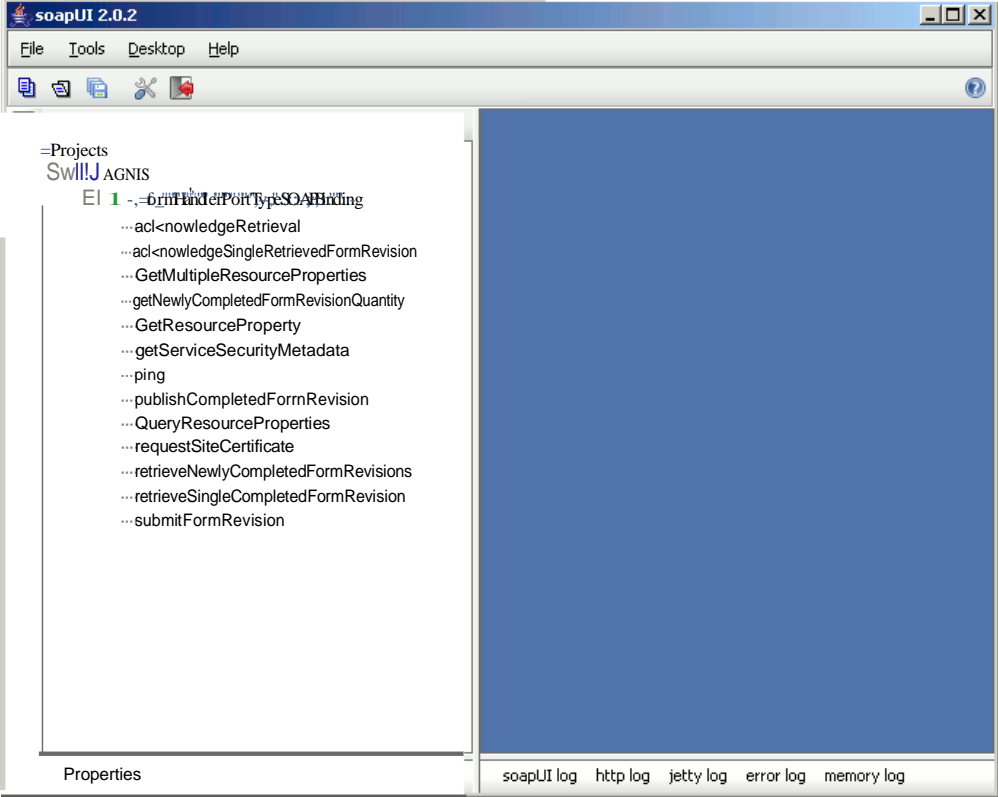


In the *Import WSDL* message box, click *No* (or *Yes*, as preferred), then wait as soapUI loads and processes the WSDL.



2.3 Web Service Interface

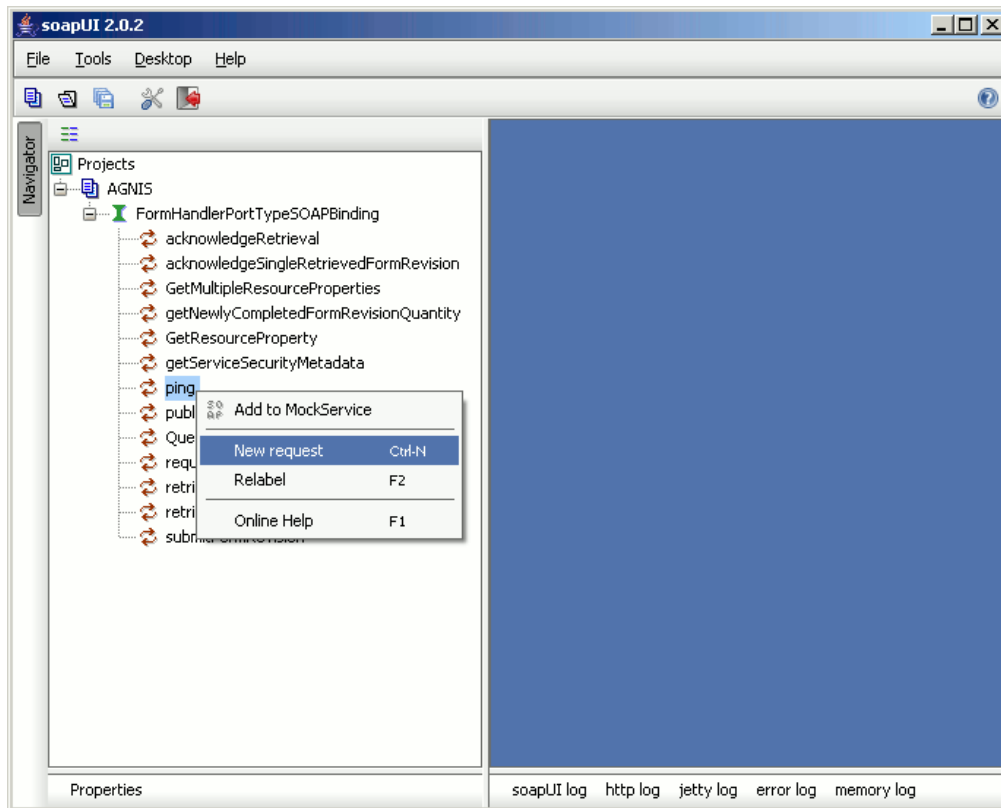
After processing the WSDL, soapUI displays a *FormHandlerPortTypeSOAPBinding* interface, with web service operations listed below it



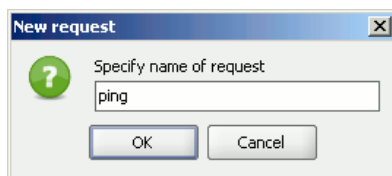
2.4 New SOAP Request

To interact with one of the web service operations, create a request for that operation. The example below illustrates creation of a request for the ping operation.

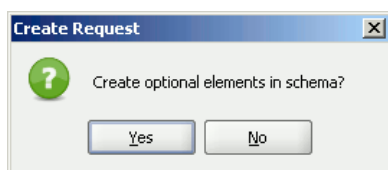
Right-click on the ping operation and select *New request*.



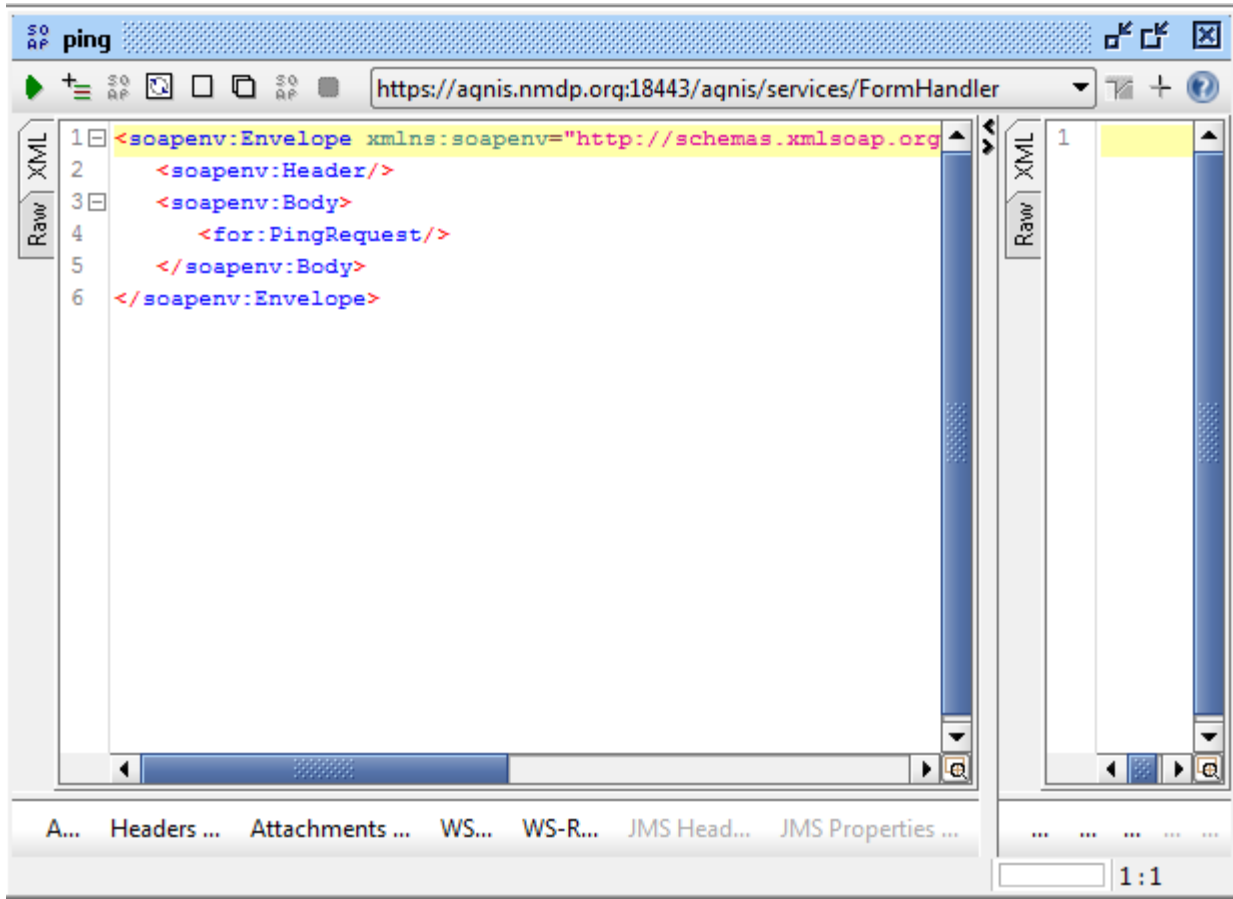
In the *New request* dialog window, input a name for the new request.



In the *Create Request* message box, click *Yes* (or *No*, as preferred).



soapUI performs some processing and then opens a request window for the newly created request. The text area on the left side of the request window holds the SOAP request, and the text area on the right side holds the response from the server.



To submit the request to the web service, click the green arrow button in the upper left. Refer to the soapUI documentation for details regarding other features of the request window.

3 AGNIS Web Service

The examples below show SOAP messages that can be submitted to the AGNIS web service via soapUI, depicting usage of AGNIS web service operations. To submit a message, first create a soapUI request, then input the SOAP message into the request text area and click the green arrow button (submit request).

3.1 Ping

The *ping* operation provides a simple way to check the status of the AGNIS web service. When processing a *ping* request, the service tests connectivity with other services upon which it depends, such as the AGNIS repository database and the Grid Grouper authorization service. If a problem is encountered, it returns a SOAP fault to the client; otherwise, it returns successfully.

The example below shows a *ping* request which could be submitted to the AGNIS web service using soapUI.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler">
  <soapenv:Header/>
  <soapenv:Body>
    <for:PingRequest/>
  </soapenv:Body>
</soapenv:Envelope>
```

The *ping* operation has no input parameters or return value, and simply generates a SOAP fault if it encounters a problem.

3.2 Submit

The *submitFormRevision* operation provides a way to electronically transmit forms data to a back-end forms processing system via the AGNIS web service. The NMDP's AGNIS service passes submitted forms data to another NMDP system called FormsNet.

The example below shows a *submitFormRevision* request which could be submitted to the AGNIS web service using soapUI. The *submitFormRevision* request takes a single parameter of type FormRevision.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler"
  xmlns:net="gme://forms.AGNIS/1.0/net.agnis.forms">
  <soapenv:Header/>
  <soapenv:Body>
    <for:SubmitFormRevisionRequest>
      <for:formRevision>
        <net:FormRevision>
          <net:form instanceId="112233" publicId="2630454" version="4.0">
            <net:originator uniqueName="cibmtr_center_number:99999"/>
          </net:form>
        </net:FormRevision>
      </for:formRevision>
    </for:SubmitFormRevisionRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

```

<net:questionCollection dataElementPublicId="2527889"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>112233</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2527895"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>99999</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2527897"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>445566</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2180389"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>male</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2527909"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>7/25/2008</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2527911"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>5/2/2000</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2730901"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>Allogeneic, unrelated</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2730901"
  dataElementVersion="1.0" repeatSequenceNumber="2">
  <net:value>Allogeneic, related</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2730901"
  dataElementVersion="1.0" repeatSequenceNumber="3">
  <net:value>Syngeneic (identical twin)</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2730905"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>Yes</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2730905"
  dataElementVersion="1.0" repeatSequenceNumber="2">
  <net:value>No</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2730905"
  dataElementVersion="1.0" repeatSequenceNumber="3">
  <net:value>No</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2693568"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>Reactive</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2734635"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <net:value>01/01/2001</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2179589"
  dataElementVersion="2.0" repeatSequenceNumber="1">
  <net:value>JOHN</net:value>
</net:questionCollection>

```

```

<net:questionCollection dataElementPublicId="2179591"
  dataElementVersion="2.0" repeatSequenceNumber="1">
  <net:value>SMITH</net:value>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2695021"
  dataElementVersion="1.0" repeatSequenceNumber="1"
  delete="true">
  <net:value/>
</net:questionCollection>
<net:questionCollection dataElementPublicId="2734639"
  dataElementVersion="1.0" repeatSequenceNumber="1"
  delete="true">
  <net:value/>
</net:questionCollection>
</net:FormRevision>
</for:formRevision>
</for:SubmitFormRevisionRequest>
</soapenv:Body>
</soapenv:Envelope>

```

The above example illustrates several important concepts related to data submission via AGNIS.

The originator *uniqueName* is tied to the AGNIS permissions system. Appropriate permissions must be granted before an AGNIS user is allowed to submit forms with a specific originator *uniqueName*.

To precisely define the data being transmitted, AGNIS uses metadata defined in the National Cancer Institute Center for Bioinformatics (NCICB) Cancer Data Standards Repository (caDSR). The caDSR defines Common Data Elements (CDEs), which provide detailed and unambiguous descriptions of data, and thereby facilitate semantic interoperability between disparate systems.

Many elements of the caDSR metadata model inherit important attributes from a superclass named *AdministeredComponent*. Each *AdministeredComponent* can be uniquely identified by the combination of two fields named *publicID* and *version*. Because the caDSR *DataElement* class which defines a CDE is an *AdministeredComponent*, each CDE defined in the caDSR can be uniquely identified by its *publicID* and *version*.

In the XML shown above, each *questionCollection* element contains *dataElementPublicId* and *dataElementVersion* attributes which identify a CDE in the caDSR. The CDE, in turn, describes the enclosed value. Details on a specific CDE can be viewed using the NCICB's web based CDE Browser tool:

<https://cdebrowser.nci.nih.gov/CDEBrowser/>

The CDE Browser displays details related to a CDE, including its value domain, workflow status, alternate names, and formal definition using terms from a controlled vocabulary.

In addition to CDEs, the caDSR also defines Case Report Forms (CRFs), which specify grouping of related CDEs that describe data entered on a paper form. Because the caDSR *Form* class which defines a CRF is an *AdministeredComponent*, each CRF defined in the caDSR can also be uniquely identified by its *publicID* and *version*.

In the example, the *form* element contains *publicID* and *version* attributes which identify a CRF in the caDSR. The CRF describes the usage of each CDE within a particular type of form. Details on a specific CRF can be viewed using the NCICB's web based Form Builder tool:

<https://formbuilder.nci.nih.gov/FormBuilder/formSearchAction.do>

The Form Builder displays information about form questions, grouped into modules, and may also include reference documents such as a PDF rendition of the form.

Another tool for exploring caDSR metadata is the caDSR API service:

<http://cadsrapi.nci.nih.gov/cadsrapi40/>

The caDSR API service includes a web browser interface capable of, for example, retrieving details about CRF modules not displayed by the Form Builder tool.

Some of the information within the *form* element is also repeated within individual *questionCollection* elements. In the above, the *form instanceId* matches the value for CDE 2527889 (*version* 1.0), and the numeric portion of the *originator uniqueName* matches the value for CDE 2527895 (*version* 1.0).

Most of the *questionCollection* values shown above correspond directly to questions from the form. However, the values for CDE 2730901 (*version* 1.0) and CDE 2730905 (*version* 1.0) use a slightly different pattern known as normalized curation, in which a set of CDEs are tied together by their *repeatSequenceNumber* values, and one of those CDEs (2730901 in this case) indicates which question from the paper form is being referenced.

The above example also includes two *questionCollection* elements which depict usage of the *delete* attribute. Two others have *dataElementVersion* set to 2.0, showing that a *dataElementVersion* other than 1.0 is possible, depending on the specific CDE.

3.3 Publish

The AGNIS service maintains a repository of forms data, intended to hold forms which have reached a "completed" status in the forms processing system. Forms data in the AGNIS repository is available for retrieval by AGNIS users with appropriate permissions. The forms processing system uses the *publishCompletedFormRevision* operation to transmit a completed form to AGNIS.

The example below shows a *publishCompletedFormRevision* request which could be submitted to the AGNIS web service using soapUI. The *publishCompletedFormRevision* request takes a single parameter of type FormRevision.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler"
  xmlns:net="gme://forms.AGNIS/1.0/net.agnis.forms">
  <soapenv:Header/>
  <soapenv:Body>
    <for:PublishCompletedFormRevisionRequest>
      <for:formRevision>
        <net:FormRevision>
```

```

<net:form instanceId="112233" publicId="2630454" version="4.0">
  <net:publisher uniqueName="NMDP FormsNet"/>
  <net:originator uniqueName="cibmtr_center_number:99999"/>
</net:form>
<ns1:questionCollection dataElementPublicId="2695096"
  dataElementVersion="1.0" repeatSequenceNumber="1">
  <ns1:errorCollection
    message="23 WNV-NAT testing must be answered "/>
  </ns1:questionCollection>
</net:FormRevision>
</for:formRevision>
</for:PublishCompletedFormRevisionRequest>
</soapenv:Body>
</soapenv:Envelope>

```

Each form in the AGNIS repository is uniquely identified by the combination of its form *instanceId* and publisher *uniqueName*. The AGNIS repository is able to hold multiple revisions of the same form, but the AGNIS service normally only supports retrieval of the most recent revision of a particular form.

In addition to serving as the partial identifier of a form, the publisher *uniqueName* is also tied to the AGNIS permissions system. Appropriate permissions must be granted before an AGNIS user will be allowed to publish forms with a specific publisher *uniqueName*.

The originator *uniqueName* is also tied to the AGNIS permissions system. An AGNIS user authorized to submit forms for a specific originator, such as "cibmtr_center_number:99999", is also permitted to retrieve forms associated with that same originator.

3.4 Retrieve & Acknowledge

The procedure for retrieving data from the AGNIS repository consists of two steps:

1. Retrieve the data.
2. Acknowledge the retrieved data.

The acknowledgement step is important, because it tells AGNIS to mark the data as retrieved, so the same data is not returned in response to a later retrieval request.

The example below shows a *retrieveNewlyCompletedFormRevisions* request which could be submitted to the AGNIS web service using soapUI.

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler">
  <soapenv:Header/>
  <soapenv:Body>
    <for:RetrieveNewlyCompletedFormRevisionsRequest>
      <for:subscriberUniqueName>cibmtr_center_number:99999</for:subscriberUni
<queName>
      <for:maximumFormQuantityRetrieved>100</for:maximumFormQuantityRetrieved
<>
    </for:RetrieveNewlyCompletedFormRevisionsRequest>

```

```
</soapenv:Body>
</soapenv:Envelope>
```

In response to the *retrieveNewlyCompletedFormRevisions* request, the AGNIS service queries its repository and returns a Retrieval object, which will contain a collection of newly completed FormRevision objects, if any were found.

The *subscriberUniqueName* parameter identifies the organization for which the retrieval is being performed. Before processing a *retrieveNewlyCompletedFormRevisions* request, the AGNIS service first queries its permissions system, to verify that the user who submitted the request is authorized to retrieve forms for the specified *subscriberUniqueName*. As shown, a typical center retrieving forms from AGNIS should pass a *subscriberUniqueName* of the form "cibmtr_center_number:99999", where "99999" is the center number.

The *maximumFormQuantityRetrieved* parameter limits the number of forms returned by a single *retrieveNewlyCompletedFormRevisions* request. This enables the client to retrieve, process, and acknowledge reasonably sized sets of data, instead of potentially retrieving the entire contents of the AGNIS repository in a single request, which could be problematic.

The example below shows an *acknowledgeRetrieval* request which could be submitted to the AGNIS web service using soapUI.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler">
  <soapenv:Header/>
  <soapenv:Body>
    <for:AcknowledgeRetrievalRequest>
      <for:subscriberUniqueName>cibmtr_center_number:99999</for:subscriberUni
<queName>
      <for:retrievalSequenceNumber>1</for:retrievalSequenceNumber>
    </for:AcknowledgeRetrievalRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

During processing of an *acknowledgeRetrieval* request, the AGNIS service sets the status of the specified retrieval to "ACKNOWLEDGED", so subsequent *retrieveNewlyCompletedFormRevisions* requests will not return form revisions associated with that retrieval.

The *subscriberUniqueName* parameter identifies the organization for which the retrieval was performed.

The *retrievalSequenceNumber* parameter comes from the Retrieval object, and identifies the specific set of form revisions which are being acknowledged.

The example below shows a *retrieveSingleCompletedFormRevision* request which could be submitted to the AGNIS web service using soapUI.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler">
  <soapenv:Header/>
  <soapenv:Body>
    <for:RetrieveSingleCompletedFormRevisionRequest>
      <for:subscriberUniqueName>cibmtr_center_number:99999</for:subscriberUni
«queName>
      <for:publisherUniqueName>NMDP FormsNet</for:publisherUniqueName>
      <for:formInstanceId>112233</for:formInstanceId>
    </for:RetrieveSingleCompletedFormRevisionRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

In response to the *retrieveSingleCompletedFormRevision* request, the AGNIS service queries its repository and, if found, returns the most recent revision of the specified form.

As with *retrieveNewlyCompletedFormRevisions*, the *subscriberUniqueName* parameter identifies the organization for which the retrieval is being performed, and a typical center retrieving forms from AGNIS should pass a *subscriberUniqueName* of the form "cibmtr_center_number:99999", where "99999" is the center number.

The *publisherUniqueName* and *formInstanceId* parameters together identify the form to be retrieved from the AGNIS repository.

The example below shows an *acknowledgeSingleRetrievedFormRevision* request which could be submitted to the AGNIS web service using soapUI.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler">
  <soapenv:Header/>
  <soapenv:Body>
    <for:AcknowledgeSingleRetrievedFormRevisionRequest>
      <for:subscriberUniqueName>cibmtr_center_number:99999</for:subscriberUni
«queName>
      <for:publisherUniqueName>NMDP FormsNet</for:publisherUniqueName>
      <for:formInstanceId>112233</for:formInstanceId>
      <for:formRevisionSequenceNumber>1</for:formRevisionSequenceNumber>
    </for:AcknowledgeSingleRetrievedFormRevisionRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

During processing of an *acknowledgeSingleRetrievedFormRevision* request, the AGNIS service creates a retrieval record for the specified form revision and immediately sets the status of the retrieval to "ACKNOWLEDGED", so later *retrieveNewlyCompletedFormRevisions* requests will not return that form revision.

The *subscriberUniqueName* parameter identifies the organization for which the retrieval was performed.

The *publisherUniqueName*, *formInstanceId*, and *formRevisionSequenceNumber* parameters together identify the form revision to be acknowledged.

The example below shows a *getNewlyCompletedFormRevisionQuantity* request which could be submitted to the AGNIS web service using soapUI.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:for="http://grid.agnis.net/FormHandler">
  <soapenv:Header/>
  <soapenv:Body>
    <for:GetNewlyCompletedFormRevisionQuantityRequest>
      <for:subscriberUniqueName>cibmtr_center_number:99999</for:subscriberUni
    <queName>
      </for:GetNewlyCompletedFormRevisionQuantityRequest>
    </soapenv:Body>
  </soapenv:Envelope>
```

In response to a *getNewlyCompletedFormRevisionQuantity* request, the AGNIS service queries its repository to determine the number of unretrieved form revisions available for the specified *subscriberUniqueName*, and returns that number.

As with *retrieveNewlyCompletedFormRevisions* and *retrieveSingleCompletedFormRevision*, the *subscriberUniqueName* parameter identifies the organization for which the retrieval is being performed, and a typical center retrieving forms from AGNIS should pass a *subscriberUniqueName* of the form "cibmtr_center_number:99999", where "99999" is the center number.

4 caDSR 4.1 Web Service Interface

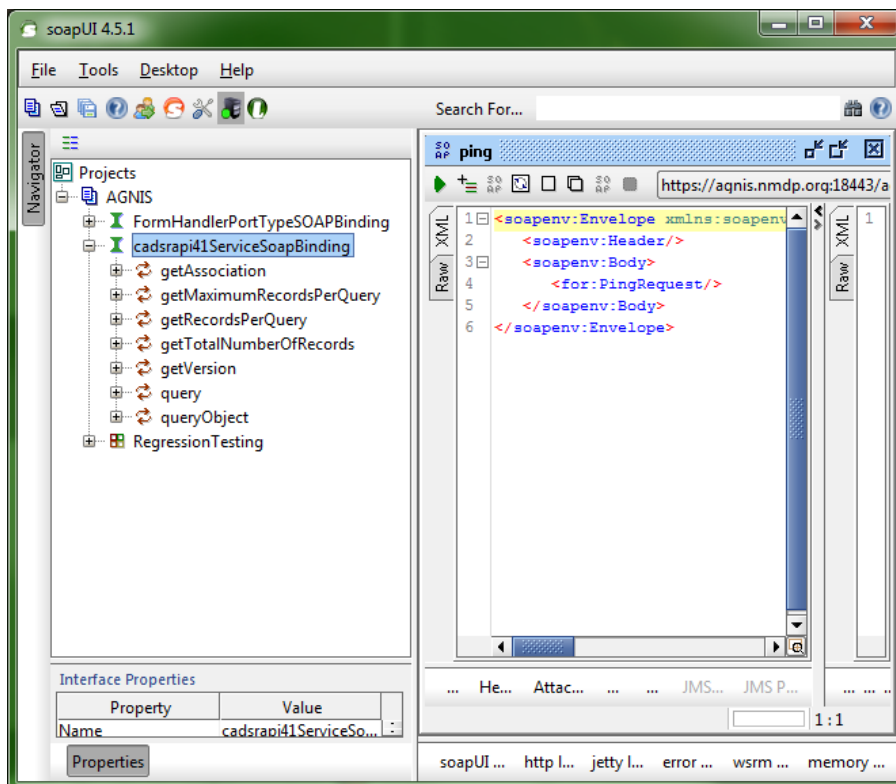
The caDSR metadata repository holds detailed information about forms and data elements which are transmitted via AGNIS. This metadata can be viewed with a web browser using tools such as the CDE Browser, FormBuilder, and the caDSR API web interface:

<https://cdebrowser.nci.nih.gov/CDEBrowser/>
<https://formbuilder.nci.nih.gov/FormBuilder/formSearchAction.do>
<http://cadsrapi.nci.nih.gov/cadsrapi41/>

FormBuilder, the CDE Browser, and the caDSR API web interface all access the same underlying caDSR metadata repository. Using soapUI, it's possible to also access the caDSR metadata via the publicly accessible caDSR 4.1 web service. The WSDL URL for the caDSR 4.1 web service is:

<http://cadsrapi.nci.nih.gov/cadsrapi41/services/cadsrapi41Service?wsdl>

To add the caDSR 4.1 web service to the soapUI workspace, follow the typical procedure for adding a WSDL project, described in section 2 of this document. After processing the WSDL for the caCORE web service, soapUI displays a `cadsrapi41ServiceSoapBinding` interface, with web service operations listed under it, as illustrated below.



The caDSR 4.1 web service is a public web service and, unlike the AGNIS web service, does not require a client-side X.509 credential, or any modification to the soapUI SSL settings.

The Common Data Element (CDE), represented in the caDSR 4.1 web service by the *DataElement* class, is the basic unit of caDSR metadata. Some important related classes are *DataElementConcept*, *ValueDomain*, and *PermissibleValue*.

Another caDSR artifact important to AGNIS is the Case Report Form (CRF), represented in the caCORE web service by the *Form* class. A *Form* contains *Module* objects, and a *Module* contains *Question* objects. Each *Question* is associated with a *DataElement* (CDE). The *Form* data structure defines a form which might be transmitted via AGNIS.

The caDSR 4.1 technical guide offers an overview of caDSR metadata, and includes details regarding specific components of the metadata model which can be accessed via the caDSR 4.1 web service:

<https://wiki.nci.nih.gov/display/caDSR/caDSR+4.x+Technical+Guide>

The caDSR 4.1 technical guide also includes Java code examples which demonstrate use of the caDSR 4.1 web service to retrieve CRF metadata from the caDSR. The following sections of this document show similar examples of soapUI requests to retrieve CRF metadata. These examples use the *queryObject* operation of the caCORE web service.

The *caDSR for Application Developers* web pages contain background information which may also be of interest:

http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/cadst/app_developer

4.1 Search for Forms

Within the caDSR metadata, a Case Report Form (CRF) is represented by an object of class *Form*.

The example below shows a *queryObject* request to retrieve the latest version of all CRFs within the NHLBI context.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.Form</targetClassName>
      <criteria xsi:type="ns1:Form"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <context>
          <name>NHLBI</name>
        </context>
        <latestVersionIndicator>Yes</latestVersionIndicator>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>
```

The above specifies a *targetClassName* of *Form*, and as *criteria* passes a *Form* with its *context name* set to "NHLBI" and *latestVersionIndicator* set to "Yes", to limit the query results to the latest version of CRFs within the NHLBI context. An alternate version of this query limits the query results to only those CRFs with a workflow status of RELEASED.

The example below shows a *queryObject* request to retrieve CRFs within the NHLBI context which have a workflow status of RELEASED.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.Form</targetClassName>
      <criteria xsi:type="ns1:Form"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <context>
          <name>NHLBI</name>
        </context>
        <workflowStatusName>RELEASED</workflowStatusName>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>
```

The above specifies a *Form workflowStatusName* value of "RELEASED", as a query condition.

4.2 Modules Within a Form

Module objects contain information about the structure of the *Form*. Because *Form* is a subclass of *AdministeredComponent*, an individual *Form* is identified by the combination of its *publicID* and *version* attributes.

The example below shows a *queryObject* request to retrieve all modules within the CRF identified by public id 2481311 and version 1.0.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.Module</targetClassName>
      <criteria xsi:type="ns1:Form"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <publicID>2481311</publicID>
        <version>1.0</version>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
```

```
</soapenv:Envelope>
```

The above specifies a *targetClassName* of *Module*, and as *criteria* passes a *Form* with its *publicID* and *version* attributes set, to limit the query results to *Module* objects associated with that *Form*.

4.3 Questions Within a Module

Question objects hold details about the contents of a *Module*. Because *Module* is a subclass of *AdministeredComponent*, an individual *Module* is identified by the combination of its *publicID* and *version* attributes.

The example below shows a *queryObject* request to retrieve all questions within the module identified by public id 2778232 and version 1.0.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.Question</targetClassName>
      <criteria xsi:type="ns1:Module"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <publicID>2778232</publicID>
        <version>1.0</version>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>
```

The above specifies a *targetClassName* of *Question*, and as *criteria* passes a *Module* with its *publicID* and *version* attributes set, to limit the query results to *Question* objects associated with that *Module*.

4.4 DataElement

A *DataElement* object defines the Common Data Element (CDE) associated with a *Question*. Because *Question* is a subclass of *AdministeredComponent*, an individual *Question* is identified by the combination of its *publicID* and *version* attributes.

The example below shows a *queryObject* request to retrieve the data element associated with the question identified by public id 2778234 and version 1.0.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
```

```

<web:queryObject>
  <targetClassName>gov.nih.nci.cadsr.domain.DataElement</targetClassName>
  <criteria xsi:type="ns1:Question"
    xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
    <publicID>2778234</publicID>
    <version>1.0</version>
  </criteria>
</web:queryObject>
</soapenv:Body>
</soapenv:Envelope>

```

The above specifies a *targetClassName* of *DataElement*, and as *criteria* passes a *Question* with its *publicID* and *version* attributes set, to retrieve the *DataElement* associated with that *Question*.

4.5 DataElementConcept

A *DataElementConcept* defines the underlying concept associated with a *DataElement*, but does not define a specific value domain. Because *DataElement* is a subclass of *AdministeredComponent*, an individual *DataElement* is identified by the combination of its *publicID* and *version* attributes.

The example below shows a *queryObject* request to retrieve the data element concept for the data element identified by public id 2730901 and version 1.0.

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.DataElementConcept</targetCla
<ssName>
      <criteria xsi:type="ns1:DataElement"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <publicID>2730901</publicID>
        <version>1.0</version>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>

```

The above specifies a *targetClassName* of *DataElementConcept*, and as *criteria* passes a *DataElement* with its *publicID* and *version* attributes set, to retrieve the *DataElementConcept* associated with that *DataElement*.

4.6 ValueDomain

A *ValueDomain* object defines the specific representation, formatting, etc. of a *DataElement*. Because *DataElement* is a subclass of *AdministeredComponent*, an individual *DataElement* is identified by the combination of its *publicID* and *version* attributes.

The example below shows a *queryObject* request to retrieve the value domain for the data element identified by public id 2730901 and version 1.0.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.ValueDomain</targetClassName>
      <criteria xsi:type="ns1:DataElement"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <publicID>2730901</publicID>
        <version>1.0</version>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>
```

The above specifies a *targetClassName* of *ValueDomain*, and as *criteria* passes a *DataElement* with its *publicID* and *version* attributes set, to retrieve the *ValueDomain* associated with that *DataElement*.

4.7 ValueDomainPermissibleValue

A *ValueDomainPermissibleValue* object links a *ValueDomain* of type *EnumeratedValueDomain* to a specific *PermissibleValue*. Because *EnumeratedValueDomain* is a subclass of *AdministeredComponent*, an individual *EnumeratedValueDomain* is identified by the combination of its *publicID* and *version* attributes.

The example below shows a *queryObject* request to retrieve the value domain permissible values for the enumerated value domain identified by public id 2730897 and version 1.0.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.ValueDomainPermissibleValue</
<targetClassName>
      <criteria xsi:type="ns1:EnumeratedValueDomain"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <publicID>2730897</publicID>
        <version>1.0</version>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>
```


The above specifies a *targetClassName* of *ValueDomainPermissibleValue*, and as *criteria* passes an *EnumeratedValueDomain* with its *publicID* and *version* attributes set, to limit the query results to *ValueDomainPermissibleValue* objects associated with that *EnumeratedValueDomain*.

4.8 PermissibleValue

A *PermissibleValue* object defines the representation of the specific data value associated with a *ValueDomainPermissibleValue*. *ValueDomainPermissibleValue* is not a subclass of *AdministeredComponent*, but an individual *ValueDomainPermissibleValue* can be identified by its *id* attribute.

The example below shows a *queryObject* request to retrieve the permissible value for the value domain permissible value identified by id 46ADC95C-E10A-452D-E044-0003BA3F9857.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
  <soapenv:Body>
    <web:queryObject>
      <targetClassName>gov.nih.nci.cadsr.domain.PermissibleValue</targetClass
<Name>
      <criteria xsi:type="ns1:ValueDomainPermissibleValue"
        xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
        <id>46ADC95C-E10A-452D-E044-0003BA3F9857</id>
      </criteria>
    </web:queryObject>
  </soapenv:Body>
</soapenv:Envelope>
```

The above specifies a *targetClassName* of *PermissibleValue*, and as *criteria* passes a *ValueDomainPermissibleValue* with its *id* attribute set, to retrieve the *PermissibleValue* associated with that *PermissibleValue*.

4.9 ValueMeaning

A *ValueMeaning* object specifies the meaning associated with a *PermissibleValue*. As of caDSR 4.0, *ValueMeaning* is an administered component, and each *ValueMeaning* instance can be uniquely identified by the combination of its *publicID* and *version* attributes. *PermissibleValue* is not a subclass of *AdministeredComponent*, but an individual *PermissibleValue* can be identified by its *id* attribute.

The example below shows a *queryObject* request to retrieve the value meaning for the permissible value identified by id 46ADC95C-E101-452D-E044-0003BA3F9857.

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:web="http://webservice.system.nci.nih.gov"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header/>
```

```
<soapenv:Body>
  <web:queryObject>
    <targetClassName>gov.nih.nci.cadsr.domain.ValueMeaning</targetClass
<Name>
  <criteria xsi:type="ns1:PermissibleValue"
    xmlns:ns1="urn:domain.cadsr.nci.nih.gov">
    <id>46ADC95C-E101-452D-E044-0003BA3F9857</id>
  </criteria>
</web:queryObject>
</soapenv:Body>
</soapenv:Envelope>
```

The above specifies a *targetClassName* of *ValueMeaning*, and as *criteria* passes a *PermissibleValue* with its *id* attribute set, to retrieve the *ValueMeaning* associated with that *PermissibleValue*.

5 Additional Information

User Interface

soapUI adheres to established user interface concepts for Integrated Development Environments (IDEs). Many of its components offer keyboard shortcuts and right-click menus, support drag-and-drop, and provide "inspectors", which are tabs that can be shown and hidden by selecting them.

Web Service Testing

soapUI is primarily designed as a web service testing tool, and supports a variety of testing approaches, including functional testing, load testing, and mock web services.

Tests in soapUI are organized into test suites, which contain test cases. Each test case is comprised of test steps, such as a SOAP request, which may have assertions such as XPath queries or Groovy scripts attached.

Groovy Scripting

Groovy is a scripting language offering features such as Java-like syntax, powerful XML parsing capabilities, and seamless interoperability with Java code libraries. To supplement its rich set of built-in testing functions, soapUI also provides support for scripts written using Groovy, which can be integrated into the testing framework at any level.

Groovy scripts are capable of interfacing with any Java code library, including that of soapUI itself. Javadoc documentation for the soapUI code is available here:

<http://www.soapui.org/apidocs/>

Conclusion

soapUI can be helpful in exploring how web services work, and a valuable tool for testing and troubleshooting purposes. For AGNIS purposes, soapUI can be readily configured to connect with both the secure AGNIS service and the public caDSR API service.

Appendix A: Constructing a Java Key Store

Before constructing a Java keystore file, it is necessary to acquire a valid client-side X.509 certificate, as described in the *AGNIS Quick Start* document. The `usercert.pem` and `userkey.pem` files obtained by following the *AGNIS Quick Start* instructions can then be used to create a Java keystore file for soapUI.

A.1 OpenSSL

The `userkey.pem` file contains the user's private key, and the `usercert.pem` file contains a copy of the the user's public key which was signed by the AGNIS certificate authority. These two files need to be converted to a single PKCS#12 formatted file. The conversion can be performed using the open-source cryptography software known as OpenSSL:

<http://www.openssl.org/>

Most unix-like operating systems either include OpenSSL as part of the system, or offer it as package which can be readily installed. For Windows operating systems, a compiled version of OpenSSL is available from Shining Light Productions:

<http://www.slproweb.com/products/Win32OpenSSL.html>

Use OpenSSL to create a PKCS#12 file named `keystore.p12` from the `usercert.pem` and `userkey.pem` files. Example:

```
> openssl pkcs12 -export -out keystore.p12 -in usercert.pem -inkey userkey.pem
Loading 'screen' into random state - done
Enter pass phrase for userkey.pem:
Enter Export Password:
Verifying - Enter Export Password:
```

Before creating the output file, OpenSSL prompts the user for the `userkey.pem` pass phrase, and then for an "Export Password" to protect the `keystore.p12` file.

Note: Access to the `keystore.p12` file should be closely guarded. An untrusted party who obtains a copy of that file could potentially use the private key it contains to impersonate the the key's owner within AGNIS. Similarly, access to the `userkey.pem` file should also be restricted.

A.2 PKCS12Import

The Java development kit (JDK) includes a program called `keytool` for managing keystore files. However, `keytool` does not offer a method for importing an existing key pair, such as that contained in the `keystore.p12` file, which was generated externally. Fortunately, there is a small open-source Java program named `PKCS12Import` which can create a Java keystore file from a key pair stored in a PKCS#12 file.

The `PKCS12Import.java` code originates from Jetty, the pure Java web server:

<http://jetty.mortbay.org/>

The `PKCS12Import.zip` file is available for download from the AGNIS Google Site, under the Tools section, under Available Documentation, for the AGNIS mailing list. You will need to unzip the file first. Then you will be able to access the `pkcs12import.jar` file which is executable, and also includes the `PKCS12Import.java` source code. Follow the link below for the download.

<https://sites.google.com/site/agnisdevelopment/available-documentation/tools>

The example below illustrates usage of the `PKCS12Import.jar` file, but this step could also be accomplished by other methods; for example, download the Jetty source code, then compile and execute `PKCS12Import` by hand.

After downloading the `PKCS12Import.jar` file, execute the `PKCS12Import` program to construct a Java keystore file named `keystore.jks` from the PKCS#12 formatted `keystore.p12` file containing the AGNIS user key and certificate. Example:

```
> java -jar PKCS12Import.jar keystore.p12 keystore.jks
Enter input keystore passphrase:
Enter output keystore passphrase:
Alias 0: 1
Adding key for alias 1
```

Before creating the output file, `PKCS12Import` prompts the user for the `keystore.p12` passphrase, and then for a passphrase to protect the `keystore.jks` file.

Note: Access to the `keystore.jks` file should be closely guarded. An untrusted party who obtains a copy of that file could potentially use the private key it contains to impersonate the the key's owner within AGNIS. Similarly, access to the `keystore.p12` file should also be restricted.

Optionally, use `keytool` to change the key alias from the default value of "1" to something more meaningful, such as "agnis_user". To do this, first clone the existing key, then delete it. Example:

```
> keytool -keyclone -alias 1 -dest agnis_user -keystore keystore.jks
Enter keystore password:
Enter key password for
    (RETURN if same as for )
> keytool -delete -alias 1 -keystore keystore.jks
Enter keystore password:
```

For further information on `keytool`, refer to the JDK documentation.

Appendix B: Other caDSR 4.0 Interfaces

In addition to the familiar CDE Browser and Form Builder tools, and the web service (SOAP) interface shown above in section 4, the caDSR 4.0 API service also offers a web browser interface and a RESTful XML-HTTP interface. Chapter 5 of the caCORE SDK 4.0 Developer's Guide has additional information about these interfaces:

https://gforge.nci.nih.gov/docman/view.php/148/8650/caCORE%20SDK%204.0%20Developer%27s%20Guide_101007.pdf

The caDSR 4.0 Technical Guide contains related details about caDSR data structures:

https://gforge.nci.nih.gov/docman/view.php/59/15452/caDSR_v40_TechnicalGuide.pdf

B.1 Web Browser Interface

The web browser interface can be a convenient tool for navigating and querying metadata stored in the caDSR. For example, to use the caDSR 4.0 web browser interface, first open the caDSR API home page:

<http://cadsrapi.nci.nih.gov/cadsrapi40/>

Then, under Select Criteria, click the Continue button. On the domain class browser page, select *gov.nih.nci.cadsr.domain.Form* in the selection tree. Enter publicID 2481311 and version 1.0, and click the Submit button. This brings up a web page with an URL similar to the following.

[http://cadsrapi.nci.nih.gov/cadsrapi40/GetHTML?query=gov.nih.nci.cadsr.domain.Form&gov.nih.nci.cadsr.domain.Form\[@publicID=2481311\]\[@version=1.0\]](http://cadsrapi.nci.nih.gov/cadsrapi40/GetHTML?query=gov.nih.nci.cadsr.domain.Form&gov.nih.nci.cadsr.domain.Form[@publicID=2481311][@version=1.0])

The web page shows Form metadata, with links to related information such as the Form's moduleCollection, etc.

B.2 XML-HTTP Interface

The caDSR 4.0 API service also offers a RESTful XML-HTTP interface. For example, substituting "GetXML" for "GetHTML" in the above URL causes the service to display an XML rendition of the Form metadata:

[http://cadsrapi.nci.nih.gov/cadsrapi40/GetXML?query=gov.nih.nci.cadsr.domain.Form&gov.nih.nci.cadsr.domain.Form\[@publicID=2481311\]\[@version=1.0\]](http://cadsrapi.nci.nih.gov/cadsrapi40/GetXML?query=gov.nih.nci.cadsr.domain.Form&gov.nih.nci.cadsr.domain.Form[@publicID=2481311][@version=1.0])