# How to Build and Deploy OpenClinicaCRF Data Service Version 1.2

## 1 Overview

This document describes the process to follow for setting up the OpenClinicaCRF Data Service. The OpenClinicaCRF Data Service provides grid-enabled access to clinical data stored in OpenClinica through the information model shown in Figure 1. The information model corresponds to a subset of the OpenClinica database schema and describes the relationship of studies, subjects, and case report form information. The OpenClinicaCRF Data Service allows users to navigate this model by subject, study, or type of case report form data.

## 2 Required Software

| Name | Version |
|---|---|
| Java | 1.5 (Note: 1.6 will not work) |
| Ant | 1.6.5 |
| Tomcat | 5.0.28 |
| PostgreSQL | 8.1.11 |
| Subversion client | 1.4.2 |
| OpenClinica | 2.0 |
| caGrid | 1.2 |
| caCORE SDK | 4.0 (modified; see section 3.1) |

## 3 Building the OpenClinicaCRF Data Service

### 3.1 Building and Deploying the OpenClinica caCORE SDK Application

The OpenClinica caCORE SDK application is built with caCORE 4.0, modified to support the PostgreSQL database system – data in OpenClinica is stored in a PostgreSQL database, but caCORE does not it support it by default. Please contact the CVRG team to obtain the modified caCORE SDK instance. The modifications are configuration changes to enable a caCORE SDK generated application to access a PostgreSQL database backend. No modifications have been done to the core caCORE SDK system.

In this document, $CACORE_SDK_HOME refers to the path of the caCORE SDK application directory.

Before building the caCORE SDK application, you must perform two preparation steps:

1. Install PostgreSQL and create the OpenClinica database. Instructions for doing this can be found on the OpenClinica web site: http://www.openclinica.org/.

2. Install a Tomcat container.

    a. This Tomcat container instance will host the OpenClinica caCORE SDK application.

b. The location of this Tomcat instance will be referred to as $CACORE_SDK_CATALINA_HOME in this document.
c. Instructions for installing a non-secure Tomcat container via the caGrid installer can be found at: http://wiki.cagrid.org/display/caGrid12/caGrid+Installer+User%27s+Guide#caGridInstallerUser%27sGuide-ConfigureContainer.

3. You must configure the caCORE SDK properties file to properly build the application.
   a. Open the file `$CACORE_SDK_HOME/conf/deploy.properties`.
   b. Find the section labeled APPLICATION SERVER PROPERTIES.
   c. Set SERVER_TYPE to "other", and set SERVER_URL to the path to the Tomcat instance that will contain the application. For example, if Tomcat is running on localhost on port 8081, the two lines in this section would read as follows:

```
SERVER_TYPE=other
SERVER_URL=http://localhost:8081/${PROJECT_NAME}
```

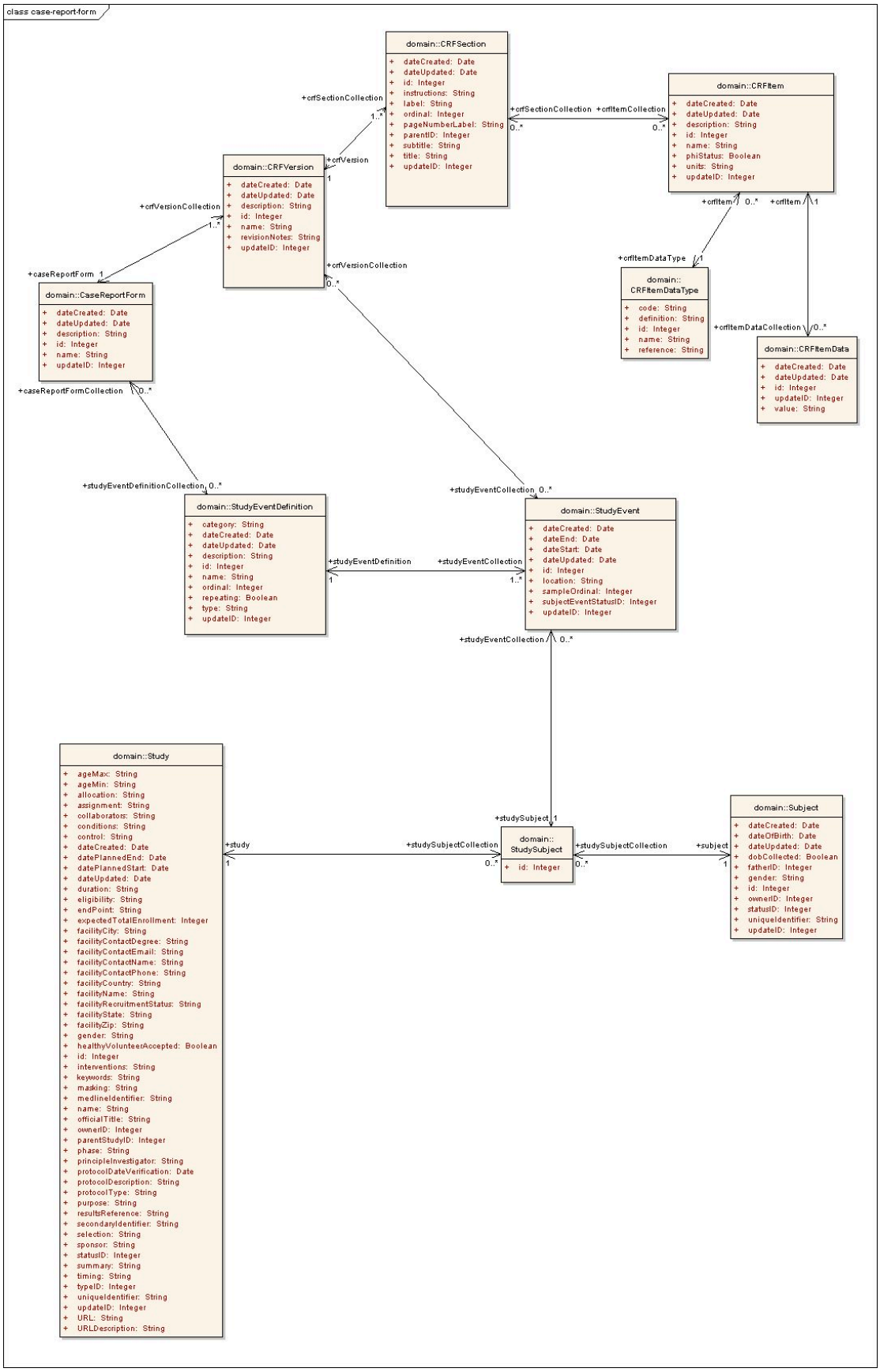



**Figure 1. OpenClinica information model**

4. Find the section labeled MODEL PROPERTIES, and set the MODEL_FILE and MODEL_FILE_TYPE variables to match the properties of your model file.
    a. For example, if your model XMI file is called openclinica-model.xmi and it was generated by Enterprise Architect, the variables would be set as follows:

```
MODEL_FILE=openclinica-model.xmi
MODEL_FILE_TYPE=EA
```

**Note: The model file must be stored in $CACORE_SDK_HOME/models.**

5. Find the section labeled DATABASE CONNECTION PROPERTIES and modify it to reflect your PostgreSQL database settings. For example:

```
# A database named "openclinica" running on localhost
DB_CONNECTION_URL=jdbc:postgresql://localhost/openclinica

# Database user that owns the openclinica database
DB_USERNAME=clinica

# No password needed for the clinica user to access the database
DB_PASSWORD=

# JDBC PostgreSQL driver
DB_DRIVER=org.postgresql.Driver

# Hibernate dialect value
DB_DIALECT=org.hibernate.dialect.PostgreSQLDialect
```

6. Now you are ready to build the OpenClinica caCORE SDK application. Do so using Ant:

```
cd $CACORE_SDK_HOME
ant build-system
```

   The result of this build process is a file called openclinica.war in $CACORE_SDK_HOME/output/openclinica/package/webapp. Copy this file to your Tomcat instance:

```
cp $CACORE_SDK_HOME/output/openclinica/package/webapp/openclinica.war
$CACORE_SDK_CATALINA_HOME/webapps/
```

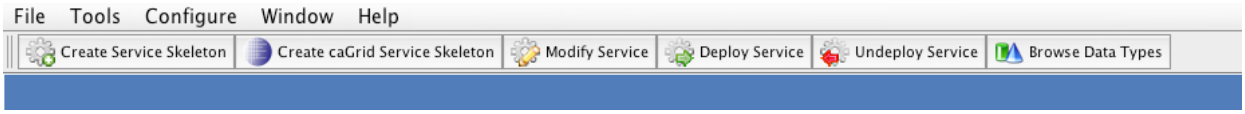
7. To verify that the build and deployment were successful, navigate to the URL of your application in a Web browser.
    a. Using the example settings defined earlier, the URL would be: http://localhost:8081/openclinica.

### 3.2 Generating and Deploying the OpenClinicaCRF Data Service

The OpenClinicaCRF data service is generated using the caGrid Introduce toolkit. To begin, navigate to $CAGRID_LOCATION and load Introduce.

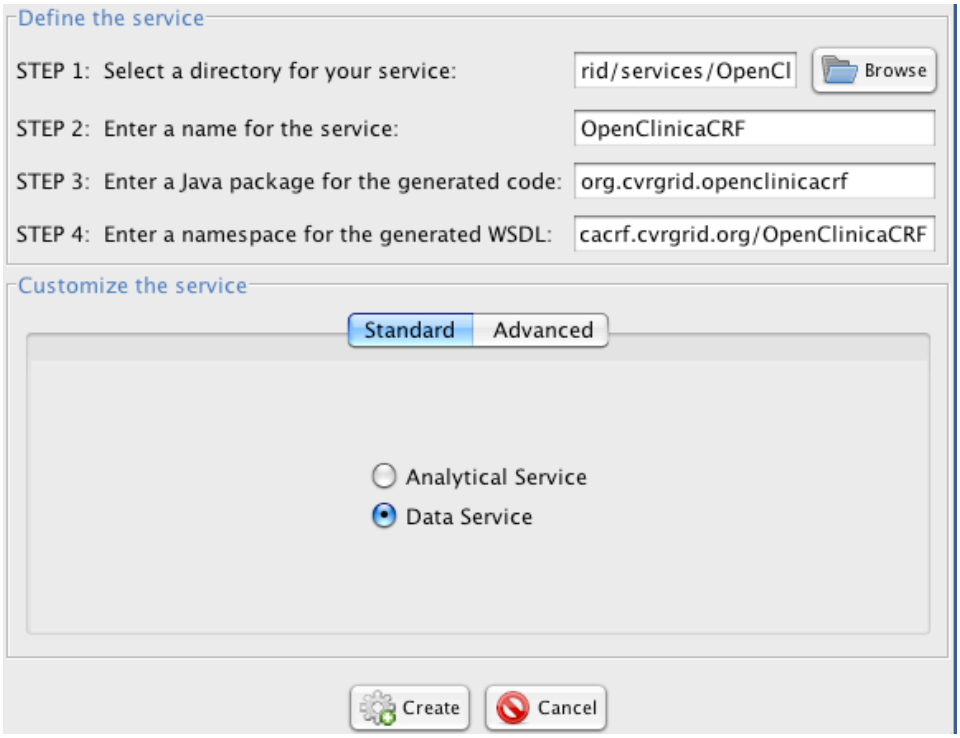
```
cd $CAGRID_LOCATION
ant introduce
```

When the main Introduce screen comes up, click Create caGrid Service Skeleton.




This will start the caGrid Service Skeleton creation wizard. The first window asks you to define the service; there are 4 steps to doing so:

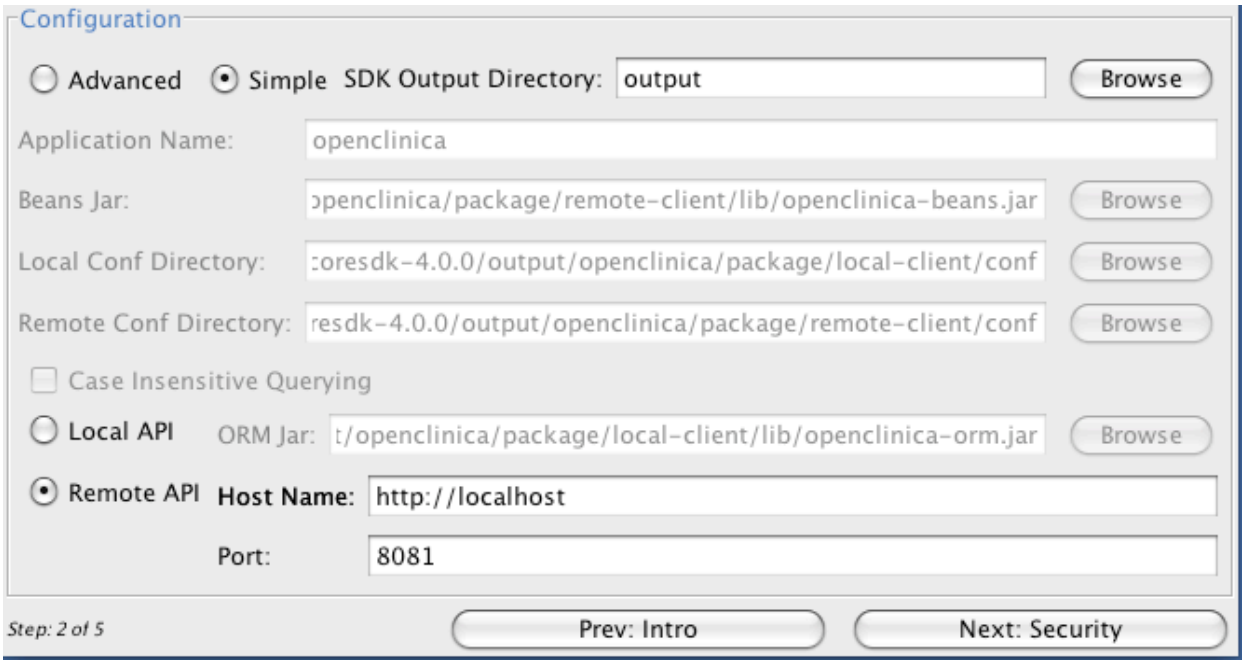
**Step 1:** Select a directory to your service. This can be whatever you like.
**Step 2:** Name your service. It should be named: OpenClinicaCRF.
**Step 3:** Specify a Java package for the generated code. This can also be whatever you like, but something like `org.cvrgrid.openclinicacrf` works well.
**Step 4:** Specify the namespace for the WSDL. This field is automatically filled in based on the previous steps.




Under “Customize the Service”, select Data Service. Click Create.

Since this is a data service, the Data Service Configuration wizard appears. It first asks you select a Service Style. Select “caCORE SDK v 4.0” and click OK. This will start the Create caCORE SDK Backend caGrid Data Service wizard. The first window will display a description of the wizard. Click “Next: Configuration”.

In the configuration step, select Simple and click Browse to select the SDK Output Directory. In the file chooser dialog, navigate into the caCORE SDK directory and select the `output` directory. This will automatically fill in several other fields.



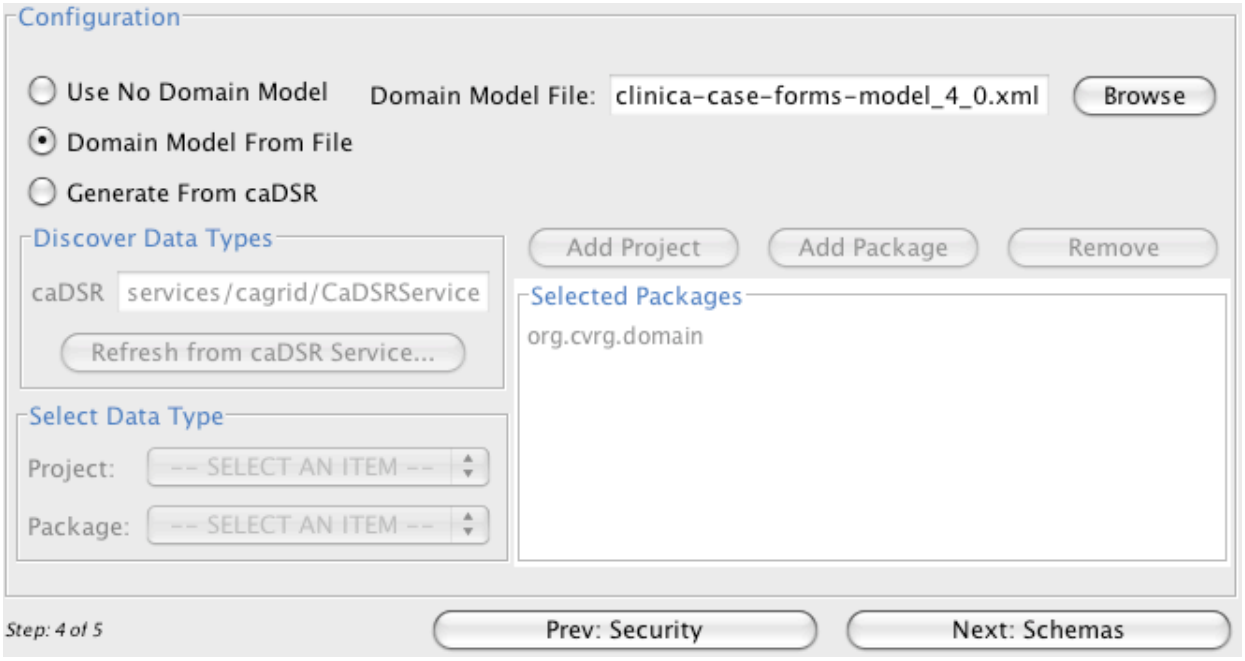

For the API, select Remote API. This will require you to enter the host name and port of where you deployed the OpenClinica caCORE application to.

**Note: The hostname *must* begin with "http://".**

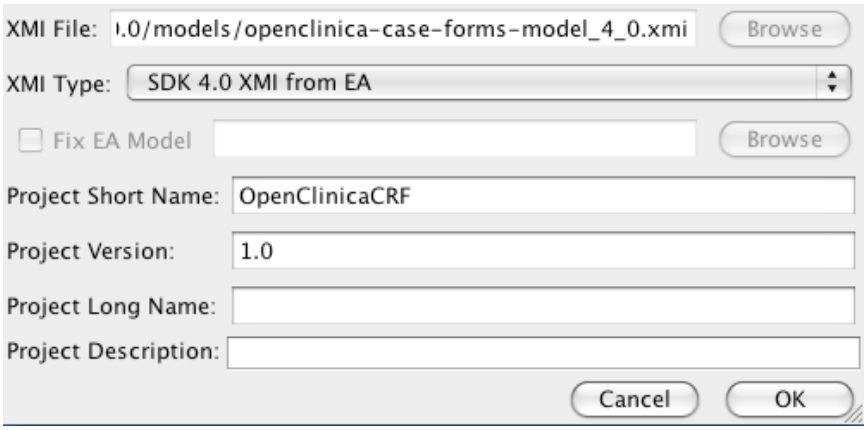
The next window allows you to specify whether or not the caCORE SDK application uses security. Leave the "Use Security" box unchecked.

Step 4 is to specify the domain model. Select the "Domain Model From File" option, then click the Browse button to select the domain model file. In the file selection dialog, navigate to the OpenClinica domain model XMI file.

**Note: Be sure to change the file selection filter to XML Metadata Interchange Files (*.xmi).**

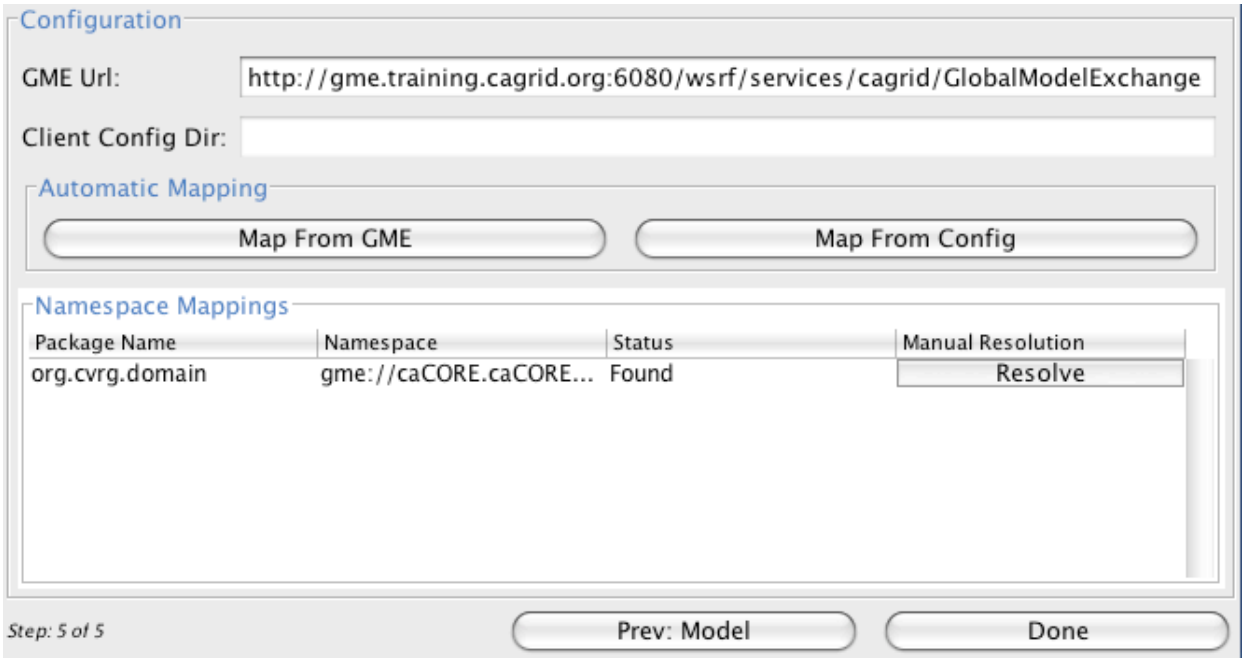


This brings up the Generate Domain Model window, with the XMI File field prefilled. For XMI Type, select SDK 4.0 XMI from EA. For Project Short Name and Project Version, enter OpenClinicaCRF and 1.0, respectively, and click OK.




Once control returns to the Domain Model Selection window, click Next: Schemas.

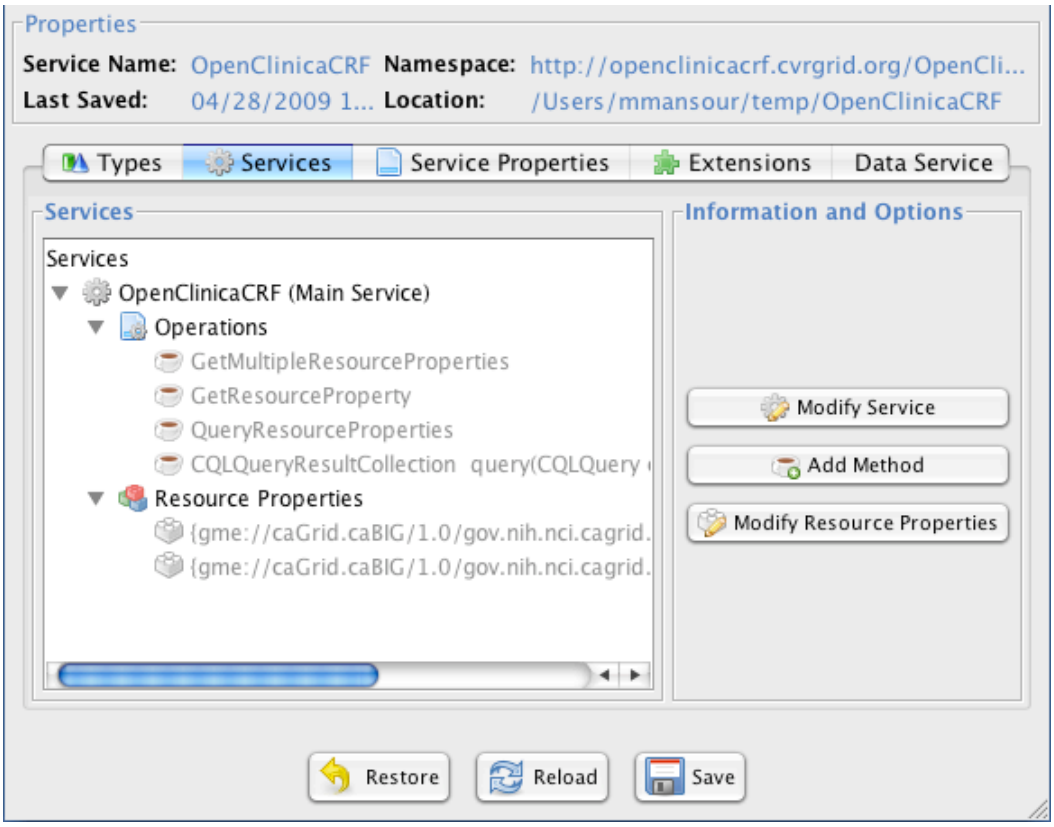
In the Package to Schema Mapping window, click Map From Config, then click Done. Introduce will now complete the creation process of the service.



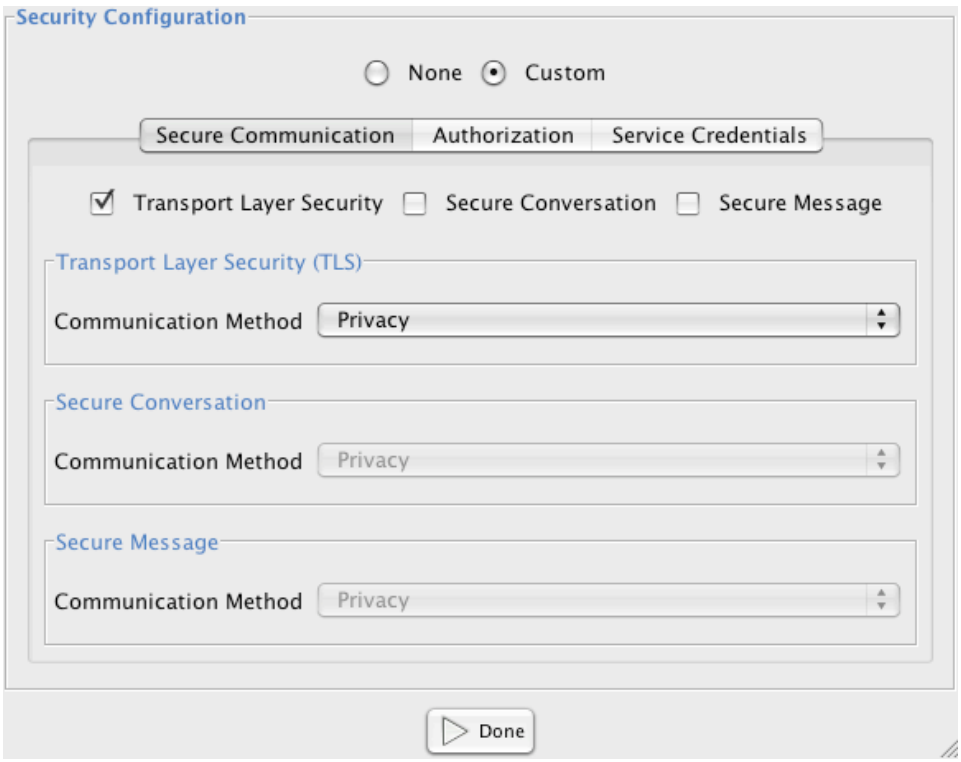

The Modify Service Interface appears.

### 3.3 Securing the Service

Select the Services tab. Select OpenClinicaCRF (Main Service).

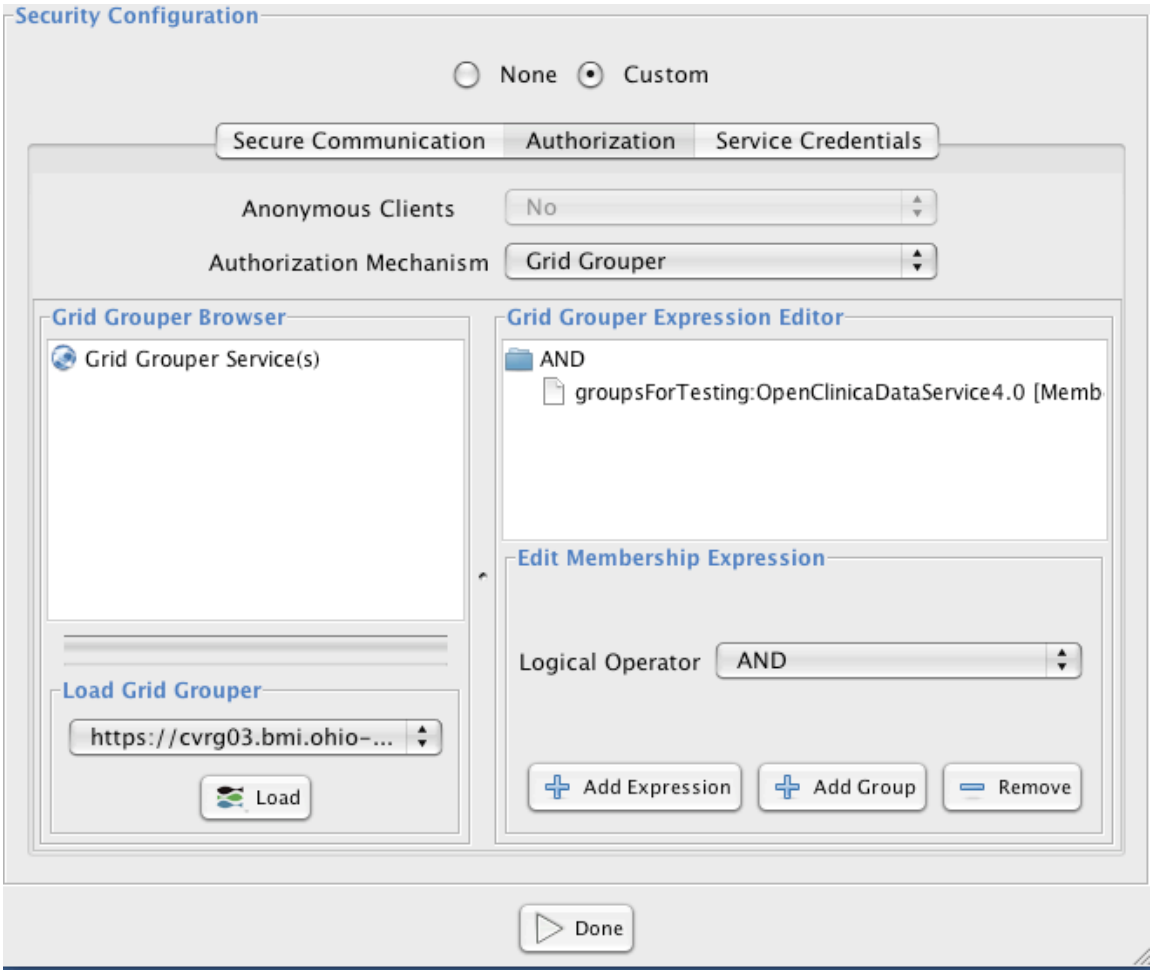



Click Modify Service. The Modify Service Context window comes up.



Within the Secure Communication tab, under Security Configuration, select the Custom radio button then check the Transport Layer Security box (leave the Secure Conversation and Secure Message boxes unchecked). Leave the TLS Communication Method set to Privacy.

Next, select the Authorization tab. Keep Anonymous Clients set to No, but set Authorization Mechanism to Grid Grouper. A two-pane interface now appears.




In the Grid Grouper Browser pane, click Load. This will load the available groups for the target grid. Find the group you want to grant service access to and select it. In the Grid Grouper Expression Editor pane, select the AND expression, and click the Add Group button. Then click Done to return to the Modify Service Interface window. Click Save to save your changes.

### 3.4 Deploying the Service

From a terminal, navigate to the directory of the OpenClinicaCRF service you just created. Set the `$CATALINA_HOME` environment variable to the location of your secure Tomcat instance that will host the service. (You can find instructions for installing a secure Tomcat container using the caGrid Installer at: http://wiki.cagrid.org/display/caGrid12/caGrid+Installer+User%27s+Guide#caGridInstallerUser%27sGuide-ConfigureContainer). For example, in bash,

```
export CATALINA_HOME=~/tomcat-secure/jakarta-tomcat-5.0.28
```

Run the Ant task to deploy the service to the Tomcat instance specified by `$CATALINA_HOME`:

```
ant deployTomcat
```

The service is now deployed. You can verify this by navigating to the service URL in a Web browser. For example, if the secure Tomcat container is running on localhost on port 8443, the URL would be: https://localhost:8443/wsrf/services/cagrid/OpenClinicaCRF.

# 4 OpenClinicaCRF Client API

The service created in Section 3 is a standard caGrid data service. It provides the standard query interface. The information managed by the service can be queried using CQL and the OpenClinica information model.

We have developed a set of client-side helper classes for the implementation of the OpenClinica data service that is distributed with the CVRG software stack. These helper classes enable a developer to perform common queries without dealing with the specifics of CQL (although this functionality is still available). Note that these helper classes are not automatically generated, when the data service is created as described in Section 3.

## 4.1 Helper Classes for Navigating the OpenClinica Information Model

The following helper classes and methods are included in the CVRG implementation of the OpenClinica data service:

- **SubjectQueryHelper**
  - Facilitates navigation of the OpenClinica information model from a particular subject to the studies it is associated with.
  - **Methods**
    - **Constructor**
      - `public SubjectQueryHelper(Subject subject, OpenClinicaCRFClient client)`
        - Creates a new `SubjectQueryHelper` using the given `Subject` as the navigation source and the given `OpenClinicaCRFClient` as the client to invoke the service
        - **Parameters**
          - `subject` – the `Subject` that will serve as the source of navigation
          - `client` – the `OpenClinicaCRFClient` that is invoking the service

    - `public List<Study> getStudies()`
      - Retrieves the studies associated with the specified subject
      - **Parameters:** None
      - **Return:** a list of `Study` objects associated with the specified `Subject`
      - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **StudyQueryHelper**
  - Facilitates navigation of the OpenClinica information model from a particular study to the subjects, study event definitions, and case report forms associated with it.
  - **Methods**
    - **Constructor**
      - `public StudyQueryHelper(Study study, OpenClinicaCRFClient client)`
        - Creates a new `StudyQueryHelper` using the given `Study` as the navigation source and the given `OpenClinicaCRFClient` as the client to invoke the service
        - **Parameters**
          - `study` – the `Study` that will serve as the source of navigation
          - `client` – the `OpenClinicaCRFClient` that is invoking the service

    - `public List<Subject> getSubjects()`
      - Retrieves the subjects belonging to the specified study
      - **Parameters**: None
      - **Return**: a list of `Subject` objects associated with the given `Study`
      - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public List<CaseReportForm> getCaseReportForms()`
      - Retrieves the case report forms related to the given study
      - **Parameters:** None
      - **Return:** a list of `CaseReportForm` objects associated with the given `Study`
      - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public List<StudyEventDefinition> getStudyEventDefinitions()`
      - Retrieves the study event definitions that include the given study
      - **Parameters:** None
      - **Return:** a list of `StudyEventDefinition` objects associated with the given `Study`
      - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **StudyEventQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular study event to the study event definition, case report form versions, subjects, and study associated with it.
  - **Methods**
    - **Constructor**
      - `public StudyEventQueryHelper(StudyEvent studyEvent, OpenClinicaCRFClient client)`

        - Creates a new `StudyEventQueryHelper` using the given `StudyEvent` as the navigation source and the given OpenClinicaCRFClient as the client to invoke the service
        - **Parameters**
            - `studyEvent` – the `StudyEvent` that will serve as the source of navigation
            - `client` – the `OpenClinicaCRFClient` that is invoking the service

    - `public StudyEventDefinition getStudyEventDefinition()`
        - Retrieves the study event definition that defines the specified study event
        - **Parameters:** None
        - **Return:** the `StudyEventDefinition` associated with the given `StudyEvent`
        - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public List<CRFVersion> getCRFVersions()`
        - Retrieves the case report form version data associated with the given study event
        - **Parameters:** None
        - **Return:** a list of `CRFVersion` objects associated with the given `StudyEvent`
        - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public Subject getSubject()`
        - Retrieves the subject associated with the given study event
        - **Parameters:** None
        - **Return:** the `Subject` associated with the given `StudyEvent`
        - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public Study getStudy()`
        - Retrieves the study associated with the given study event
        - **Parameters:** None
        - **Return:** the `Study` associated with the given `StudyEvent`
        - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **StudyEventDefinitionQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular study event definition to the study events and case report forms associated with it.
    - **Methods**
        - **Constructor**
            - `public StudyEventDefinitionQueryHelper(StudyEventDefinition studyEventDefinition, OpenClinicaCRFClient client)`
                - Creates a new `StudyEventDefinitionQueryHelper` using the given study event definition as the navigation

source and the given `OpenClinicaCRFClient` as the client to invoke the service

- **Parameters**
  - `studyEventDefinition` – the `StudyEventDefinition` that will serve as the source of navigation
  - `client` – the `OpenClinicaCRFClient` that is invoking the service

- `public List<StudyEvent>getStudyEvents()`
  - Retrieves the study events belonging to the given study event definition
  - **Parameters:** None
  - **Return:** A list of `StudyEvent` objects associated with the given `StudyEventDefinition`
  - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- `public List<CaseReportForm> getCaseReportForms()`
  - Retrieves the case report forms associated with a given study event
  - **Parameters:** None
  - **Return:** A list of `CaseReportForm` objects associated with the given `StudyEventDefinition`
  - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **CaseReportFormQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular case report form to the study event definitions and case report form versions associated with it.
  - **Methods**
    - **Constructor**
      - `public CaseReportFormQueryHelper(CaseReportForm crf, OpenClinicaCRFClient client)`
        - Creates a new `CaseReportFormQueryHelper` using the given `CaseReportForm` as the navigation source and the given `OpenClinicaCRFClient` as the client to invoke the service
        - **Parameters**
          - `crf` – the `CaseReportForm` that will serve as the source of navigation
          - `client` – the `OpenClinicaCRFClient` that is invoking the service

    - `public List<StudyEventDefinition> getStudyEventDefinitions()`
      - Retrieves the study event definitions that a given case report form belongs to
      - **Parameters:** None
      - **Return:** A list of `StudyEventDefinition` objects associated with the given `CaseReportForm`

        - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public List<CRFVersion> getCRFVersions()`
        - Retrieves the version information for a given case report form
        - **Parameters:** None
        - **Return:** A list of `CRVersion` objects associated with the given `CaseReportForm`
        - **Throws**: `java.rmi.RemoteException` if there is a problem contacting the service

- **CRFVersionQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular case report form version to the case report form, case report form sections, and study events associated with it.
    - **Methods**
        - **Constructor**
            - `public CRFVersionQueryHelper(CRFVersion version, OpenClinicaCRFClient client)`
                - Creates a new `CRFVersionQueryHelpler` using the given `CRFVersion` as the navigation source and the given `OpenClinicaCRFClient` as the client to invoke the service
                - **Parameters**
                    - `version` – the `CRFVersion` that will serve as the source of navigation
                    - `client` – the `OpenClinicaCRFClient` that is invoking the service

        - `public List<CRFSection> getCRFSections()`
            - Retrieves the section information associated with a given case report form version
            - **Parameters:** None
            - **Return:** a list of `CRFSection` objects associated with the given `CRFVersion`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

        - `public CaseReportForm getCaseReportForm()`
            - Retrieves the case report form that a given version belongs to
            - **Parameters:** None
            - **Return:** the `CaseReportForm` associated with the given `CRFVersion`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

        - `public List<StudyEvent> getStudyEvents()`
            - Retrieves the study events associated with a given case report form version
            - **Parameters:** None
            - **Return:** a list of `StudyEvent` objects associated with the given `CRFVersion`

- **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **CRFSectionQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular case report form section to the case report form items and case report from version associated with it.
  - **Methods**
    - **Constructor**
      - `public CRFSectionQueryHelper(CRFSection section, OpenClinicaCRFClient client)`
        - Creates a new `CRFSectionQueryHelper` using the given `CRFSection` as the source of navigation and the given `OpenClinicaCRFClient` as the client to invoke the service
        - **Parameters**
          - `section` – the `CRFSection` that will serve as the source of navigation
          - `client` – the `OpenClinicaCRFClient` that is invoking the service

    - `public List<CRFItem> getCRFItems()`
      - Retrieves the items associated with a given case report form section
      - **Parameters:** None
      - **Return:** a list of `CRFItem` objects associated with the given `CRFSection`
      - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

    - `public CRFVersion getCRFVersion()`
      - Retrieves the version information associated with a given case report form section
      - **Parameters:** None
      - **Return:** the `CRFVersion` associated with the given `CRFSection`
      - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **CRFItemQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular case report form item to the item data, item data type and case report form section associated with it.
  - **Methods**
    - **Constructor**
      - `public CRFItemQueryHelper(CRFItem item, OpenClinicaCRFClient client)`
        - Creates a new `CRFItemQueryHelper` using the given `CRFItem` as the navigation source and the given `OpenClinicaCRFClient` as the client to invoke the service
        - **Parameters**
          - `item` – the `CRFItem` that will serve as the source of navigation

            - `client` – the `OpenClinicaCRFClient` that is invoking the service

        - `public CRFItemDataType getCRFItemDataType()`
            - Retrieves the data type of a given case report form item
            - **Parameters:** None
            - **Return:** the `CRFItemDataType` associated with the given `CRFItem`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

        - `public List<CRFItemData> getCRFItemDatas`
            - Retrieves the data values for a given case report form item
            - **Parameters:** None
            - **Return:** a list of `CRFItemData` objects associated with this `CRFItem`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

        - `public List<CRFSection> getCRFSections()`
            - Retrieves the case report form sections where a given item appears
            - **Parameters:** None
            - **Return:** a list of `CRFSection` objects associated with the given `CRFItem`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **CRFItemDataQueryHelper:** Facilitates navigation of the OpenClinica information model from a particular case report form item data record to the item associated with it.
    - **Methods**
        - **Constructor**
            - `public CRFItemDataQueryHelper(CRFItemData itemData, OpenClinicaCRFClient client)`
                - Creates a new `CRFItemDataQueryHelper` using the given `CRFItemData` as the navigation source and the given `OpenClinicaCRFClient` as the client invoking the service
                - **Parameters**
                    - `itemData` – the `CRFItemData` that will serve as the source of navigation
                    - `client` – the `OpenClinicaCRFClient` that is invoking the service

        - `public CRFItem getCRFItem()`
            - Retrieves the CRF item that a given data value belongs to
            - **Parameters:** None
            - **Return:** the `CRFItem` associated with the given `CRFItemData`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

- **CRFItemDataTypeQueryHelper:** Facilitates navigation of the OpenClinica data model from a particular case report form item data type to the items associated with it.
    - **Methods**
        - **Contstructor**
            - `public CRFItemDataTypeQueryHelper(CRFItemDataType itemDataType, OpenClinicaCRFClient client)`
                - Creates a new `CRFItemDataTypeQueryHelper` using the given `CRFItemDataType` as the navigation source and the given `OpenClinicaCRFClient` as the client to invoke the service
                - **Parameters**
                    - `itemDataType` – the `CRFItemDataType` that will serve as the source of navigation
                    - `client` – the `OpenClinicaCRFClient` that is invoking the service

        - `public List<CRFItem> getCRFItems()`
            - Retrieves the case report form items that are of a given data type
            - **Parameters:** None
            - **Return:** a list of `CRFItem` objects that are of the given `CRFItemDataType`
            - **Throws:** `java.rmi.RemoteException` if there is a problem contacting the service

### 4.2 Example Query

In this example, the user wishes to retrieve all subjects in the database whose case report forms contain an item with a certain name ("LVEDV"). Note that a CQL query must be built and submitted to obtain the initial list of CRFItems that will be iterated through, but that all navigation between objects in the data model is handled through the helper classes.

```
CQLQuery query = new CQLQuery();
gov.nih.nci.cagrid.cqlquery.Object target = new
gov.nih.nci.cagrid.cqlquery.Object();

target.setName(org.cvrg.domain.CRFItem.class.getName());

Attribute itemNameAttr = new Attribute("name", Predicate.EQUAL_TO, "LVEDV");
target.setAttribute(itemNameAttr);

query.setTarget(target);
CQLQueryResults results = client.query(query);
InputStream istream = OpenClinicaCRFClient.class.getResourceAsStream("client-
config.wsdd");

// subjects with the requested CRFItem
List<Subject> subjects = new ArrayList<Subject>();

Iterator<CRFItem> itemIter = new CQLQueryResultsIterator(results, istream);
while (itemIter.hasNext()) {
      CRFItem item = itemIter.next();
      System.out.println("Item name: " + item.getName());
      CRFItemQueryHelper crfiqh = new CRFItemQueryHelper(item, client);
      List<CRFSection> sections = crfiqh.getCRFSections();
```

```
        for (CRFSection section : sections) {
              System.out.println("Section ID: " + section.getId());
              CRFSectionQueryHelper crfsqh = new CRFSectionQueryHelper(section,
              client);
              CRFVersion version = crfsqh.getCRFVersion();
              System.out.println("Version ID: " + version.getId());
              CRFVersionQueryHelper crfvqh = new CRFVersionQueryHelper(version,
              client);
              List<StudyEvent> studyEvents = crfvqh.getStudyEvents();
              System.out.println("# Study events: " + studyEvents.size());
              for (StudyEvent studyEvent : studyEvents) {
                    System.out.println("StudyEvent ID: " + studyEvent.getId());
                    StudyEventQueryHelper seqh = new
                    StudyEventQueryHelper(studyEvent, client);
                    System.out.println("\tStudy ID: " +
                    seqh.getStudy().getId());
                    System.out.println("\tSubject ID: " +
                    seqh.getSubject().getId());
              }
        }
  }
```