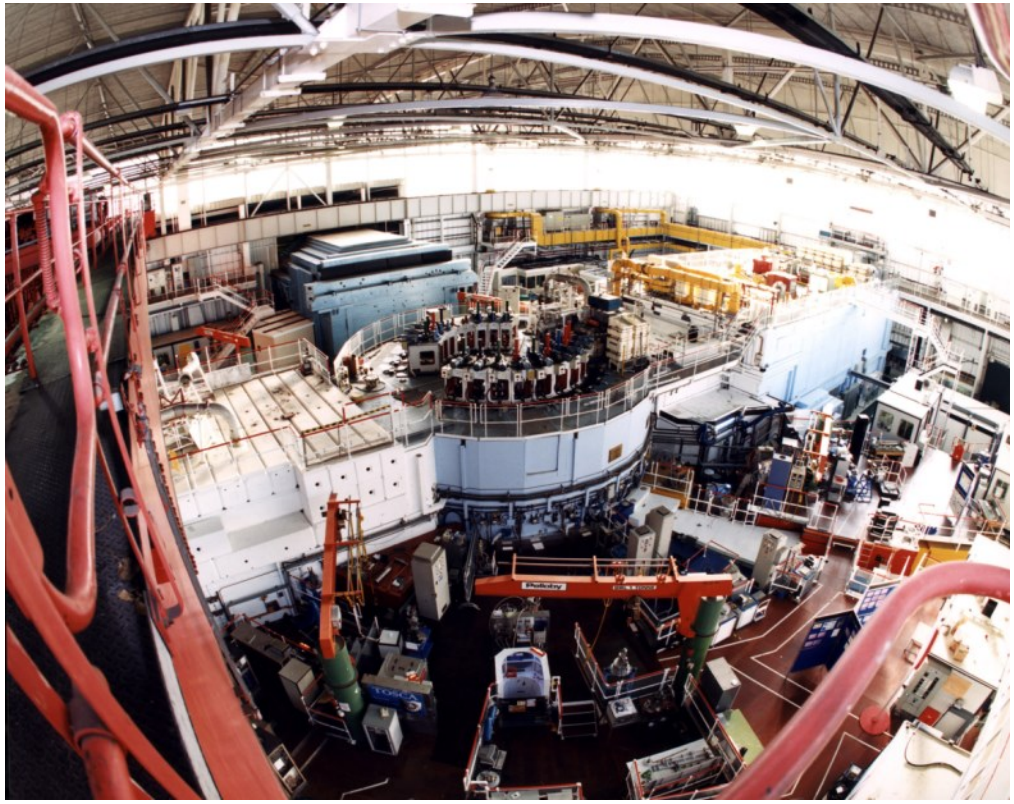
Using the ICAT API



Tom Griffin, ISIS Facility

Elettra, Trieste

November 2013

tom.griffin@stfc.ac.uk

Science & Technology Facilities Council
ISIS

Contents

- The available documentation
- WSDL or client?
- Finding things
- Creating things
- What's cascaded and when – Create and update
- Authorization – Rules, PublicSteps, PublicTables etc
- Authentication plugin – how I do it.
- Lucene
- Logging
- Notifications

The documentation

- http://www.icatproject.org/mvn/site/icat/4.3.0/index.html
- Updated per release
- 'ICAT Java Client User Manual'
  - Short examples of most operations
  - http://www.icatproject.org/mvn/site/icat/4.3.0/icat.client/manual.html
- 'ICAT Schema'
  - A constant reference
  - http://www.icatproject.org/mvn/site/icat/4.3.0/icat.core/schema.html
- Slides, code examples, etc

Generating a client

- WSDL or client?
- I use: Java, C# and C++ (gsoap)
- I don't use the provided client jar
- WSDL can cause "[][]" (.net)
- Java requires the WSDL to be 'hacked'
  - See the maven job

Logging in

```
Credentials credentials = new Credentials();
List<Entry> entries = credentials.getEntry();
Entry e;

e = new Entry();
e.setKey("username");
e.setValue("root");
entries.add(e);
e = new Entry();
e.setKey("password");
e.setValue("secret");
entries.add(e);

String sessionId = icat.login("db", credentials);
```

Logging in (2)

```
Credentials credentials = new Credentials();
List<Entry> entries = credentials.getEntry();
Entry e;

e = new Entry();
e.setKey("session");
e.setValue("abc123-def456-acb123-def456");
entries.add(e);

String sessionId = icat.login("sso", credentials);
```

Finding Stuff

- Two syntaxes to choose from
  - ‘SEQL’ (Steve’s Easy Query Language)

    ```
    Investigation <-> InvestigationUser <-> User [:user]
    ```

  - JPQL inspired/extended

    ```
    SELECT i FROM Investigation i JOIN i.investigationUser iu
    JOIN iu.user u WHERE u = :user
    ```

- Can also use lucene
- Authorization rules are applied to the results.
  - ```
    List<Object> results = icat.searchText(sessionId, “copper",
    50, "Investigation");
    ```

Science & Technology Facilities Council
ISIS
    ```

Finding Stuff

- You can back the first thing you ask for
- Use INCLUDE if you want other objects
- Can use COUNT, DISTINCT etc

SEQL

- SEQL (Steve's Easy Query Language?)

```
Investigation <-> InvestigationUser <-> User [name = :user]
```

- <-> = JOIN
- [ ] = WHERE
- INCLUDE = `INCLUDE Datafile,DatafileParameter`
- Start, count

```
5,5 Investigation <-> InvestigationUser <-> User [name = :user]
```

- Limits
  - Loops
  - Single direction only
- Top Tips
  - 'Include 1'
  - It is translated to JPQL – check the logs

JPQL

```
SELECT i FROM Investigation i JOIN i.investigationUser iu JOIN
iu.user u WHERE u = :user
```

- Pretty much normal JPQL – well documented!
- INCLUDE only sees the first 'thing'

```
SELECT i FROM Investigation i WHERE i.name = 'abc' INCLUDE
i.datasets AS ds, ds.parameters
```

- LIMIT 100 100 (Skip, count)

- Limits
  - Can't do: WHERE i.investigationInstruments.instrument.name = 'GEM'
  - Must make explict JOINS

Science & Technology Facilities Council
ISIS

JPQL

```
SELECT COUNT(i) FROM Investigation i JOIN
i.investigationInstruments ii JOIN ii.instrument ins
WHERE ins.name = 'HRPD'
…
SELECT i FROM Investigation i JOIN
i.investigationInstruments ii JOIN ii.instrument ins
WHERE ins.name = 'HRPD'
INCLUDE i.investigationInstruments AS invInst,
invInst.instrument AS inst, i.samples as s , s.type,
i.investigationUsers AS iu, iu.user, i.type,
i.parameters AS params, params.type, i.keywords,
i.publications
LIMIT 0,10
```

Investigation

An investigation or experiment

Uniqueness constraint facility, name, visitId

Relationships

Card	Class	Field	Cascaded	D
0,*	StudyInvestigation	studyInvestigations	Yes	
0,*	Sample	samples	Yes	
1,1	InvestigationType	type		
0,*	Publication	publications	Yes	
0,*	Shift	shifts	Yes	
0,*	InvestigationParameter	parameters	Yes	
0,*	InvestigationUser	investigationUsers	Yes	
0,*	Dataset	datasets	Yes	
0,*	Keyword	keywords	Yes	
0,*	InvestigationInstrument	investigationInstruments	Yes	
1,1	Facility	facility		

```
SELECT i FROM Investigation
i JOIN i.investigationUser
iu JOIN iu.user u WHERE u =
:user
```

Use the schema as a handy reference

Science & Technology Facilities Council
ISIS

Creating Stuff

- ```
  Dataset ds = new Dataset();
  ds.setName(dsName);
  ds.setType(type);
  Datafile datafile = new Datafile();
  datafile.setDatafileFormat(format);
  datafile.setName(dfName);
  ds.getDatafiles().add(datafile);
  ds.id = icat.create(sessionId, ds);
  ```
- Dataset id is updated, datafiles will have null ids

- Create cascades one to many (Lists)
- Must create in order
- Facility -> ****Types -> Investigation -> ….
  ```

Investigation

An investigation or experiment

Uniqueness constraint facility, name, visitId

Relationships

Card	Class	Field	Cascadec
0,*	StudyInvestigation	studyInvestigations	Yes
0,*	Sample	samples	Yes
1,1	InvestigationType	type	
0,*	Publication	publications	Yes
0,*	Shift	shifts	Yes
0,*	InvestigationParameter	parameters	Yes
0,*	InvestigationUser	investigationUsers	Yes
0,*	Dataset	datasets	Yes
0,*	Keyword	keywords	Yes
0,*	InvestigationInstrument	investigationInstruments	Yes
1,1	Facility	facility	

ParameterType

A parameter type with unique name and units

Uniqueness constraint facility, name, units

Relationships

Card	Class	Field	Cascaded
0,*	SampleParameter	sampleParameters	Yes
0,*	PermissibleStringValue	permissibleStringValues	Yes
0,*	DataCollectionParameter	dataCollectionParameters	Yes
0,*	DatafileParameter	datafileParameters	Yes
1,1	Facility	facility	
0,*	InvestigationParameter	investigationParameters	Yes
0,*	DatasetParameter	datasetParameters	Yes

Cascades - Update

- Update updates "many to one" relationships and any simple field values.
- Essential to get the existing values for any "many to one" relationships.
- This is most reliably achieved by the notation INCLUDE 1
- ```
  Dataset ds = (Dataset) icat.get(sessionId, "Dataset INCLUDE 1", dsid);
  ds.setInvestigation(anotherInvestigation);
  icat.update(sessionId, ds);
  ```
- "One to many" relationships are ignored by the update mechanism so you need to start at the correct end of the relationship to have the desired effect.
- Database Exception on cascaded constraint validation
  ```

Cascade - Delete

- All cascaded "one to many" related objects will also be deleted. **In the extreme case, if you delete a facility, you lose everything associated with that facility.**
- This privilege should not be given to many

Creating things

If I could offer you only one tip for the future, createMany would be it.

The performance benefits of createMany have been proved by scientists whereas the rest of my advice has no basis more reliable than my own meandering experience

Creating things

- CreateMany

Authorisation

- Rules – by default everything is closed.
- ISIS data policy requires >100 rules…..not finished
- Can be specified in either syntax (text is stored, flags are used)
- Are (now) sanity checked on creation
- Required at every level for **direct** access (see PublicSteps)

- Can be checked: `boolean isAccessAllowed(String sessionId, EntityBaseBean bean, AccessType accessType)`

Authorisation

Example: Admin access

```
List<String> allTables = port.getEntityNames();
for (String table : allTables)
{
      Rule rule = new Rule();
      rule.grouping = facilityAdmins;
      rule.crudFlags = "CRUD";
      rule.what = table;
      port.create(sessionId, rule);
}
```

Authorisation

- Example: Instrument Scientist to Investigation

```
Rule isInv = new Rule();
isInv.crudFlags = "CRU";
isInv.what = "SELECT i FROM Investigation i JOIN
i.investigationInstruments ii JOIN ii.instrument
inst JOIN inst.instrumentScientists instSci JOIN
instSci.user u WHERE u.name = :user";
isInv.grouping = null;
port.create(sessionId, isInv);
```

Authorisation

- Example: Principal Investigator to Investigation

```
Rule piInv = new Rule();
piInv.crudFlags = "CRU";
piInv.what = "SELECT i FROM Investigation i JOIN
i.investigationuser iu JOIN iu.user u WHERE u.name =
:user AND iu.role = 'Principal Investigator' ";
piInv.grouping = null;
port.create(sessionId, piInv);
```

Authorisation

- Example: Public Datafiles after embargo

```
Rule ueDatafile = new Rule();
ueDatafile.crudFlags = "R";
ueDatafile.what = "SELECT df FROM Datafile df JOIN
df.dataset ds JOIN ds.investigation i WHERE
i.releaseDate < CURRENT_TIMESTAMP AND ds.name =
'RAW' ";
port.create(sessionId, ueDatafile);
```

Authorisation

- PublicSteps
  - Allow access to a related object (attribute)
  - ONLY used in INCLUDE processing.
  - Offer a significant speedup on INCLUDE

- PublicTables
  - For read only access to open tables
  - Cached
  - Instrument, Application, DatasetType, InvestigationType etc
  - Can be used for link tables : DataCollectionDatafile, InvestigationUser

Authorisation

```
String[] parameterBasedPublicSteps = new String[]{"Investigation",
"Dataset", "Datafile", "Sample"};
            for (String table : parameterBasedPublicSteps) {
                PublicStep paramPublicStep = new PublicStep();
                paramPublicStep.origin = table;
                paramPublicStep.field = "parameters";
                publicSteps.add(paramPublicStep);
            }

            String[] publicStepsFromInvestigation = new String[]{"samples",
"publications", "shifts", "investigationUsers", "keywords",
"investigationInstruments"};
            for (String step : publicStepsFromInvestigation) {
                PublicStep invToSomething = new PublicStep();
                invToSomething.origin = "Investigation";
                invToSomething.field = step;
                publicSteps.add(invToSomething);
            }
```

Authorisation

```
List<String> publicTables = new ArrayList<>();

            publicTables.add("Application");
            publicTables.add("DatafileFormat");
            publicTables.add("DatasetType");
            publicTables.add("Facility");
            publicTables.add("FacilityCycle");
            publicTables.add("Instrument");
            publicTables.add("InstrumentScientist");
            publicTables.add("InvestigationType");
            publicTables.add("ParameterType");
            publicTables.add("PermissibleStringValue");
            publicTables.add("Publication");
            publicTables.add("Shift");
            publicTables.add("User");
```

Authorisation

```
//Link tables that can be public for speed reasons
publicTables.add("DataCollectionDatafile");
publicTables.add("DataCollectionDataset");
publicTables.add("InvestigationUser");
publicTables.add("StudyInvestigation");

List<EntityBaseBean> publicRules = new ArrayList<>();
for (String publicTableName : publicTables) {
    Rule publicRule = new Rule();
    publicRule.what = publicTableName;
    publicRule.crudFlags = "R";
    publicRules.add(publicRule);
}
port.createMany(sessionId, publicRules);
```

Authentication

- How I do it
- Link to the STFC Facilities User office via their (custom) API
- UOWS API takes: username + password or existing sessionId

- Plugin: accepts username + password or UO Session Id

- Will allow *web browser based* access from Umbrella

- Hide LDAP behind the UOWS
  - Plugin decides what to do, not the client.

- Happy to share if useful

Authentication

- How I do it
- Link to the STFC Facilities User office via their (custom) API
- UOWS API takes: username + password or existing sessionId

- Plugin: accepts username + password or UO Session Id

- Will allow *web browser based* access from Umbrella

- Hide LDAP behind the UOWS
  - Plugin decides what to do, not the client.

- Happy to share if useful

Logging

- 3 main areas of interest
- icat.log – via icat.properties & a log4j config file
- Server.log – how to get SQL logs
- Query logging – table and/or file
- Identify slow common queries
- Usage logging
- (Session ids are cleared out frequently)

- icat.properties

```
log.list = table
log.file = SRW
log.table = SRW
```

Science & Technology Facilities Council
ISIS

Notifications

- The JMS messages - PubSub rather than point to point.
- Topic of "jms/ICAT/Topic".
- Properties
  - "entity" which is the type of the entity such as Dataset
  - "operation" which is one of the letters: C, U and D
- Body of the message is an object which holds the entity id as a Long.
- For the "xxxMany" calls multiple notifications will be generated.
- A receiver typically implemented as an MDB (Message Driven Bean) should filter the messages it processes by using the properties of the message.
- Does not leak information

Lucene

- Text indexing – quick ‘google’ search
- Normally keeps up to date
- If you need to re-populate
  - Turn logging down
  - Do it on a Friday evening
  - Asynchronous call

```
lucenePopulate(sessionId, String entityName)

List<String> luceneGetPopulating(sessionId)
```

Science & Technology Facilities Council
ISIS

Questions...

Science & Technology Facilities Council
ISIS