Using the e-Science Central REST API

API Client version 3.0

Using the API

The esc API is broken into a number of sections, each of which allow you to access a specific piece of system functionality. The following APIs are currently provided:

1. Storage: allows access to data files stored within the system
2. Workflow: allows execution of workflows stored in esc

Identifying objects within e-Science Central

All documents and records of data are stored by e-Science Central within a relational database and all records have an id field which is used internally by e-Science Central. This field is referred to as the "e-Science Central database id" in this document and is the mechanism used by the various APIs to locate pieces of information within e-Science Central. Whenever an API method contains an `id` parameter, it is this e-Science Central database id that is being referred to.

The Storage API

Data stored on the system is represented as Files stored within Folders. The storage API has been designed to reflect this fact and the methods provided allow standard file operations to be performed on this data.

There are three ways of accessing these storage functions:

1. Using the java REST client
2. Using a suitable automatic client generation tool to process the SOAP WSDL file produced by the storage web service
3. Posting and retrieving JSON data directly from the REST API URL.

Each of these three methods provides access to the same underlying data regardless of the access pattern. This data is stored according to the following model:

All pieces of data within e-Science Central (e-SC) are represented as document objects (`EscDocument`), which are stored with folders (`EscFolder`). When a file is uploaded a new `EscDocument` must be created in order to store a record of this file. `EscDocuments` only contain a description of the data stored - they do not actually contain any data themselves. Because each file within e-SC can contain multiple versions, each `EscDocument` has a number of distinct versions associated with it (`EscDocumentVersion`). Each time a file is uploaded against an `EscDocument` a new `EscDocumentVersion` is created. These `EscDocumentVersions` can then be used to retrieve the actual file data.

The process for uploading a new file is therefore:

1. Locate an `EscFolder` in which to store the file. This can be the home folder or one of its subfolders.
2. Create a new `EscDocument` in this folder and give it a name. This is typically the same name as the file being uploaded, but this is not essential.
3. Post the actual contents of the file to the upload URL. This will return a new *EscDocumentVersion* representing a record of the contents of the file that has just been uploaded.

All documents within e-Science Central can have metadata associated with them. This data is used within the website to search for documents based on a metadata query. These pieces of metadata are represented in the e-Science Central API as `EscMetadataItem` objects. The API provides methods to list all of the `EscMetadataItems` associated with a given `EscDocument` and also to attach new `EscMetadataItems` to `EscDocuments`.

Common Storage API tasks

Setting up the Java client within a project

In order to make use of the e-Science Central API Java client, the relevant .jar file needs to be included within your project. The exact process for doing this depends upon the IDE that use for development, but broadly the process involves:

- Downloading the correct api-model.jar file
- Placing this in the libraries for your project
- Including this api-model.jar file in any libraries you distribute with your code

If you make use of Apache Maven to build your software, the API client library can be downloaded and referenced by your project automatically by including the following items in the pom.xml file for your project:

Add the following to the dependencies section:

```
<dependency>
  <groupId>com.connexience</groupId>
  <artifactId>api-model</artifactId>
  <version>3.0</version>
</dependency>
```

Add the following to the repositories section

```
<repositories>
…
  <repository>
    <id>central</id>
    <name>libs-release</name>
    <url>http://esciencecentral.co.uk/artifactory/libs-release-local</url>
  </repository>
</repositories>
```

Once this has been done, it should be possible to import the relevant API client packages into your code:

```
import com.connexience.api.model.*;
import com.connexience.api.*;
```

Connecting to the API with the Java client

To connect the Java API client to an e-Science Central server, the following steps must be performed:

1. Locate a suitable e-Science Central server
2. Make sure you have an account on the server
3. Join the StorageAPI group on the server
4. Import the correct packages into your Java code
5. Create and authenticate a StorageClient object

The Java code to connect to an e-Science Central server is show below. In this code you should replace:

- **USERNAME** with your e-Science Central logon name.
- **PASSWORD** with your e-Science Central password.
- **HOSTNAME** with the name of the machine running the e-Science Central server (www.esciencecentral.co.uk for the public e-Science Central installation).
- **PORT** with the port number the server is running on.
- **SECURE** with a true / false value depending on whether you have enabled https on the e-Science Central server.

```
import com.connexience.api.*;
import com.connexience.api.model.*;

public class ClientTest {
  public static void main(String[] args) {
    try {
      // Create a new storage client with username and password
      StorageClient client = new StorageClient(HOSTNAME,PORT,SECURE, USERNAME,PASSWORD);

      // Check that the current user can be accessed
      EscUser currentUser = client.currentUser();
      System.out.println(currentUser.getName());
    } catch (Exception e){
      e.printStackTrace();
    }
  }
}
```

The code above creates a connection an e-Science Central server, authenticates a user and then prints the full name of that user to the System.out window. If the code is successful, the output should be a single line with the full name of the authenticated user on it.

If this step succeeds, then you have a valid e-Science Central connection that you can use to upload, download and manage data within the server.

Manipulating e-Science Central Folders

Because all documents, data and workflows within e-Science Central must be contained within a logical container (`EscFolder`), the e-Science Central Storage API contains a number of methods for accessing, modifying and creating folders.

Each user (`EscUser`) within e-Science Central is assigned a "Home Folder", which is created when the user registers. This is the base folder for all data stored by that user when they are not operating within a project (N.B. Each project has it's own EscFolder assigned to it which is used to store project data). This home folder can be accessed using a configured and authenticated Java API client (See example above) as follows:

```
EscFolder homeFolder = client.homeFolder();
```

This will return a reference to the users home folder that can be used as a base for subsequent API calls. It is possible to then list either the `EscDocuments` contained in this folder:

```
EscDocument[] docs = client.folderDocuments(homeFolder.getId());
```

> *Note: the use of `homeFolder.getId()` to obtain the e-Science Central database id of the folder to list the contents of.*

Or to obtain a list of subfolders within this folder:

```
EscFolder[] folders = client.listChildFolders(homeFolder.getId());
```

The EscFolders returned can then be used to descend through the directory hierarchy:

```
EscFolder[] subFolders = client.listChildFolders(folders[0].getId());
EscDocument[] subDocs = client.folderDocuments(folders[0].getId());
```

which will list the subfolders and document contents of the first folder within the home directory.

In addition to listing documents, the Storage API provides mechanisms for creating and manipulating folders. To create a child folder called "New Folder" within a users home folder:

```
EscFolder homeFolder = client.homeFolder();
EscFolder child = client.createChildFolder(homeFolder.getId(), "New Folder");
```

> *Note: If the parent folder already contains a folder with the same name, that folder will be returned instead of a new folder.*

`EscFolder` properties can also be changed and then saved back to the e-Science Central server. This can be used, for example to rename folders or to move folders around the folder hierarchy by changing their `containerId` property to reflect a new parent folder. For example, the following code renames the first child folder contained within a user's home folder to "Renamed Folder":

```
EscFolder homeFolder = client.homeFolder();
```

```
EscFolder children = client.listChildFolders(homeFolder.getId());
EscFolder folder = children[0];
folder.setName("Renamed Folder");
EscFolder updatedFolder = client.updateFolder(folder);
```

Note: The updateFolder method returns a copy of the folder object containing the original `EscFolder` *with any changes contained in the original folder object. This can be used to verify that the desired changes have actually been reflected in the underlying database.*

Folders can also be deleted from the system using the `deleteFolder` method. For example, the following code deletes the first folder contained within a user's home folder:

```
EscFolder homeFolder = client.homeFolder();
EscFolder children = client.listChildFolders(homeFolder.getId());
EscFolder folder = children[0];
client.deleteFolder(folder);
```

Note: This method removes all of the child folders and all of the documents and data contained within the folder and any folder contained within the hierarchy beneath the folder targeted for deletion. Caution should therefore be exercised when deleting folders using the Storage API.

Uploading data to e-Science Central

One of the key uses for the Storage API is to upload data files into the e-Science Central system. Data can be uploaded directly by POSTing data to the server, or using the Java API client which provides a number of convenience methods to simplify the upload process. The simplest upload method uploads a Java `File` object directly into the system creating the corresponding `EscDocument` and `EscDocumentVersion` objects. For example, the following code uploads the file called "text.txt" from the users local home directory into their home directory on the e-Science Central server.

```
java.io.File fileToUpload = new java.io.File("/home/user/test.txt");
EscFolder folder = client.homeFolder();
EscDocumentVersion version = client.upload(folder, fileToUpload);
EscDocument uploadedDocument = client.getDocument(version.getDocumentId());
```

If the process described above is unsuitable, it is possible to exercise more control over the upload procedure by carrying out the following tasks:

1. Locate a folder in which to store the file
2. Create a new `EscDocument` in that folder which will act as a placeholder for the new data
3. Use the client to upload the contents of an `InputStream` to the placeholder document.

```
EscFolder folder = client.homeFolder();
EscDocument doc = client.createDocumentInFolder(folder.getId, "Name");
java.io.File fileToUpload = new File("/home/user/text.txt");
long length = fileToUpload.length();
java.io.FileInputStream stream = new java.io.FileInputStream(fileToUpload);
EscDocumentVersion version = client.upload(doc, stream, length);
```

Note: It is also possible to POST data directly to the e-Science Central using the uploadPath specified in an `EscDocument`.

Downloading data from e-Science Central

The Storage API also provides a mechanism for downloading files from e-Science Central and storing them on a local machine. Broadly speaking, the procedure for downloading a file is:

1. Locate the document to download.
2. Identify the version of the contents data desired (or download the latest version).
3. Create a local file to hold the data.
4. Perform the download.

The Java API client provides convenience methods to perform downloads of documents or versions of documents to either a local file or a local java OutputStream.

The simplest form of download is to download the latest version of an `EscDocument` to a file stored on the local filesystem:

```
EscDocument doc = client.getDocument("1234");
java.io.File localFile = new java.io.File("/home/Users/file.txt");
client.download(doc, localFile);
```

To access a specific version of a file, there is a similar download method that takes an `EscDocumentVersion` as an argument and downloads the contents of that to a local file. For example, the code below downloads the second version of the same document.

```
EscDocument doc = client.getDocument("1234");
EscDocumentVersion[] versions = client.listDocumentVersions(doc.getId());
java.io.File localFile = new java.io.File("/home/Users/file.txt");
client.download(versions[2], localFile);
```

In addition to downloading to a file, the Java storage API client provides similar methods for downloading the contents of files to java.io.OutputStreams:

```
client.download(doc, outputStream);
```

or

```
client.download(version, outputStream);
```

Using document Metadata within e-Science Central

All `EscDocuments` within the e-Science Central system can have a number of pieces of metadata attached to them. This metadata can then used to locate documents by applying more sophisticated search criteria than merely searching for a document with a specific name.

Metadata (`EscMetadataItem`) attached to documents must at least have a name and a value set. Optionally, pieces of metadata can also be assigned a category value in order to group related pieces of information. Metadata values can be selected from the following types:

- Text: A simple piece of textual metadata
- Date: A value representing a single point in time
- Numerical: An arbitrary numerical value
- Boolean: A true/false value

The following codes list the metadata associated with a specific `EscDocument` in the form of an array of `EscMetadataItems`:

```
EscMetadataItem[] metadata = client.getDocumentMetadata("1234");
```

New metadata can be attached to documents using the following code:

```
EscMetadataItem md = new EscMetadataItem();
md.setCategory("Category");
md.setName("SomeMetadata");
md.setMetadataType(METADATA_TYPE.NUMERICAL);
EscMetadataItem savedMetadata = client.addMetadataToDocument("1234", md);
```

The Workflow API

The Workflow API allows external programs to execute and monitor e-Science Central workflows and use the e-Science Central server as a calculation engine for a variety of data processing tasks.

Typically, a developer will create and test a data processing workflow within e-Science Central and then either make this workflow public or share it with a user or group of users. This workflow can then be called using the API.

Workflows can be run without setting any parameters, in which case they will run using their standard settings. More usefully, workflows can either be applied to a specific `EscDocument` or can be executed with an `EscWorkflowParameterList` in which case any of the block settings contained within the workflow can be modified using the API.

Regardless of the execution pattern, whenever the API runs a workflow an `EscWorkflowInvocation` object is returned to the API. This invocation object represents a record of the workflow run within the e-Science Central system. It contains data regarding the status of the workflow and the folder in which all of the workflow results will be stored. It is important to note that the execution status data contained within an `EscWorkflowInvocation` object represents the state at the time the API call was made. Any subsequent execution status changes will not be reflected in this object until the `EscWorkflowInvocation` object is refreshed by making another API call. Therefore, in order to monitor the progress of a workflow run, the user of the API needs to periodically refresh the `EscWorkflowInvocation` object returned when the workflow is first started in order to identify execution status changes.

Common Workflow API Tasks

Running workflows

The two most common patterns for executing workflows are:

1. Running a workflow using a specified file as the input
2. Running a workflow using a set of block property overrides

To execute a workflow using a specified input file, the workflow must first be configured to take a document reference as an input parameter. The process of doing this is outlined in the training course document. Once the workflow has been configured correctly, the following code will execute the latest version of the workflow with the e-Science Central database id of "1234" using the `EscDocument` with the e-Science Central database id of "324" as an input.

```
EscWorkflowInvocation invocation = client.executeWorkflowOnDocument("1234", "324");
```

To execute a workflow with a set of block property overrides, the process is similar, however a suitable `EscWorkflowParameterList` object must be constructed and passed as an argument to the API execute workflow call. For example, the following code performs the same execution as the code above, but uses an `EscWorkflowParameter` to specify the input file and workflow block to pass it to:

```
EscWorkflowParameterList p = new EscWorkflowParameterList();
p.add("blockName", "Source", "1234");
EscWorkflowInvocation i = client.executeWorkflowWithParameters("1234", p);
```

> *Note, that in order to monitor the progress of the workflow, the user is required to periodically check the execution status using the API. The procedure for this is outlined in the next example.*

Monitoring workflow progress

Whenever an e-Science Central workflow is executed, the execution request is placed in a queue and then serviced by one of the workflow execution engines whenever a workflow execution slot becomes available. This process is key to the scalability of e-Science Central, however it does mean that whenever the API schedules a workflow for execution there is no guarantee that the workflow will execute before the API call returns. In fact, it is a virtual certainty that the workflow execution will not have started before the `EscWorkflowInvocation` object is returned to the users code.

Because of this, if the user needs to be informed of workflow progress, their code needs to periodically poll the Workflow API to determine the status of an `EscWorkflow`.

> *Note: This should be avoided wherever possible as the process of polling workflow status does place a slight additional load on the system. Ideally, code using the API should not depend upon the completion of a workflow in order to function.*

If, however, notification of workflow progress is required, the following code can be used:

```
// Submit the workflow for execution
EscWorkflowInvocation i = client.executeWorkflow("1234");

// Wait until the status does not indicate the workflow is either waiting
// or running
while(i.getStatus().equals("Queued") || i.getStatus.equals("Running") ||
      i.getStatus().equals("Debugging")){
      Thread.sleep(5000); // Wait a while before checking

      // Re-fetch the invocation object to get updated state
      i = client.getInvocation(i.getId());
}

// Display the final status
System.out.println("Workflow finished with status: " + i.getStatus());
```

Terminating workflows

Once a workflow run has been initiated if it later becomes apparent that the execution is no longer required, it is possible to terminate the invocation using the Workflow API:

```
EscWorkflowInvocation i = client.executeWorkflow("1234");
client.terminateInvocation(i.getId());
```

This code will either terminate the workflow immediately if it is currently executing or cancel the execution request if the workflow is currently in the queue of tasks to run.

The e-Science Central Storage API Java Client Interface

The Java client library provides the following interface to the storage functions within e-Science Central. The methods exposed by this interface are described in more detail in this section of the manual.

```
package com.connexience.api.model;
/**
 * This interface defines the standard storage service. It is implemented by
 * both the REST and SOAP services, and a REST client is provided.
 * @author hugo
 */
public interface StorageInterface {
    EscFolder createChildFolder(String id, String name) throws Exception;
    EscDocument createDocumentInFolder(String id, String name) throws Exception;
    EscUser currentUser() throws Exception;
    EscDocument[] folderDocuments(String id) throws Exception;
    EscDocument getDocument(String id) throws Exception;
    EscFolder getFolder(String id) throws Exception;
    EscFolder homeFolder() throws Exception;
    EscFolder[] listChildFolders(String id) throws Exception;
    EscDocumentVersion[] listDocumentVersions(String id) throws Exception;
    EscDocumentVersion getDocumentVersion(String id) throws Exception;
    EscDocument updateDocument(EscDocument document) throws Exception;
    EscFolder updateFolder(EscFolder folder) throws Exception;
    EscDocumentVersion getLatestDocumentVersion(String documentId) throws Exception;
    void deleteDocument(String documentId) throws Exception;
    void deleteFolder(String folderId) throws Exception;
    EscMetadataItem[] getDocumentMetadata(String id) throws Exception;
    EscMetadataItem addMetadataToDocument(String id, EscMetadataItem metadataItem)
                      throws Exception;
}
```

createChildFolder(id, name);

Description

Creates a new subfolder within an existing folder. This can be used, for example, to create new folders in a user's home folder in which to upload data to.

Parameters

id: The e-Science Central database id of the parent folder. (i.e. the folder that the subfolder will be created in.

name: The name for the new subfolder.

Return values

EscFolder: A folder object representing the newly created folder.

Example

```
EscFolder f = c.createChildFolder(topFolder.getId(), "New Folder");
```

createDocumentInFolder(id, name);

Description

Creates a new empty `EscDocument` within a folder on the e-Science Central server. This method should be used to create a placeholder before uploading a document to e-Science Central. Once created, new data can be uploaded against this `EscDocument`.

Parameters

id: The e-Science Central database id of the parent folder. (i.e. the folder that the new document will be created in.

name: The name for the new document.

Return values

EscDocument: The newly created document. N.B. If a document with the same name already existed within the parent folder, then the `EscDocument` returned will be that document and not a new one.

Example

```
EscDocument d=c.createDocmentInFolder(topFolder.getId(),"Name");
```

currentUser()

Description

Returns an `EscUser` object representing the user that is currently authenticated for a specific instance of an e-Science Central Java client.

Parameters

This method has no call parameters.

Return values

EscUser: An `EscUser` object representing the current user.

Example

```
EscUser u = c.currentUser();
```

folderDocuments(id);

Description

Returns a list of all of the documents directly contained within a folder. This does not return the documents contained in any child folders – these must be examined individually.

Parameters

id: The e-Science Central database id of the folder that is being examined.

Return values

EscDocument[]: An array containing all of the `EscDocument` objects contained within the specified folder.

Example

```
EscDocument[] contents = c.folderDocuments(folder.getId());
```

getDocument(id);

Description

Returns an `EscDocument` object using the e-Science Central database id of the document that is *t*o be retrieved.

Parameters

id: The e-Science Central database id of the document required.

Return values

EscDocument: The document object corresponding to the supplied e-Science Central database id.

Example

```
EscDocument doc = c.getDocument("567");
```

getFolder(id);

Description

Returns an `EscFolder` object using given an e-Science Central database id.

Parameters

id: The e-Science Central database id of the folder object required.

Return values

EscFolder: The folder object corresponding to the supplied e-Science Central database id.

Example

```
EscFolder folder f = c.getFolder("6546");
```

homeFolder();

Description

Return an `EscFolder` object corresponding to the default home folder of the authenticated API client user.

Parameters

This method has no parameters.

Return values

EscFolder: The `EscFolder` object corresponding to the authenticated users home folder.

Example

```
EscFolder home = c.homeFolder();
```

listChildFolders(id);

Description

Returns a list of `EscFolders` that are contained directly beneath a folder specified using its e-Science Central database id.

Parameters

id: e-Science Central database id of the folder being examined.

Return values

EscFolder[]: An array of `EscFolders` that are direct children of the specified folder.

Example

```
EscFolder[] children = c.listChildFolders(c.homeFolder().getId());
```

listDocumentVersions(id);

Description

Returns a list of document versions stored within e-Science Central for an `EscDocument` (specified using its e-Science Central database id).

Parameters

id: e-Science Central id of the document from which to obtain the list of versions.

Return values

EscDocumentVersion[]: An array of `EscDocumentVersion` objects corresponding to the versions of the file data held for the specified document.

Example

```
EscDocumentVersion[] versions = c.listDocumentVersions(doc.getId());
```

getDocumentVersion(id);

Description

Returns a specific `EscDocumentVersion` object given it's e-Science Central database id.

Parameters

id: The e-Science Central database id of the document version record to retrieve.

Return values

EscDocumentVersion: Single `EscDocumentVersion` object corresponding to the supplied e-Science Central database id.

Example

```
EscDocumentVersion version = c.getDocumentVersion("1234");
```

updateDocument(doc);

Description

Saves changes made to an `EscDocument` object. This method is used to change attributes such as the name, parent folder or description of a document within the e-Science Central system.

Parameters

doc: An `EscDocument` object containing modified parameters.

Return values

EscDocument: The `EscDocument` containing the server held data + the parameters in the modified doc object sent as the call parameter.

Example

```
EscDocument doc = c.getDocument("1234");

doc.setName("A new name");

EscDocument updated = c.updateDocument(doc);
```

updateFolder(folder);

Description

Saves changes made to an `EscFolder` object. This method is used to change attributes such as the name, parent folder or description of a folder within the e-Science Central system

Parameters

folder: An `EscFolder` object containing modified parameters.

Return values

EscFolder: The `EscFolder` object containing the server held data + the parameters in the modified folder object sent as the call parameter.

Example

```
EscFolder folder = c.getFolder("1234");

folder.setName("A new name");

EscFolder updated = c.updateFolder(folder);
```

getLatestDocumentVersion(id);

Description

Returns an `EscDocumentVersion` object corresponding to the latest version of the data held within the e-Science Central system for the specified document.

Parameters

id: e-Science Central database id of the document to retrieve information on the latest version for.

Return values

EscDocumentVersion: `EscDocumentVersion` object representing the latest version of the document contents data held on the e-Science Central server.

Example

```
EscDocumentVersion latest = c.getLatestDocumentVersion("1234");
```

deleteDocument(id);

Description

Removes a document and all it's versions and associated contents data from the e-Science Central system.

Parameters

id: e-Science Central database id of the document to delete.

Return values

This method has no return values.

Example

```
c.deleteDocument("1234");
```

deleteFolder(id);

Description

Removes a folder and all of it's contents from the e-Science Central system. This method deletes the specified folder, all of the documents contained within it and all of the child folders and their associated documents.

Parameters

id: e-Science Central database id of the folder to delete.

Return values

This method has no return values.

Example

```
c.deleteFolder("1234");
```

getDocumentMetadata(id);

Description

Returns a list of all of the `EscMetadataItems` associated with the document specified by its e-Science Central database id.

Parameters

id: The e-Science Central database id of the document to fetch metadata for.

Return values

EscMetadataItem[]: An array of `EscMetadataItems` representing all of the metadata stored within e-Science Central for the document specified by the id parameter.

Example

```
EscMetadataItem[] metadataItems = c .getDocumentMetadata("1234");
```

addMetadataToDocument(id, metadataItem);

Description

Attach a new `EscMetadataItem` to an `EscDocument` stored within the e-Science Central system.

Parameters

id: The e-Science Central database id of the `EscDocument` to attach the new `EscMetadataItem` to.

metadataItem: The configured `EscMetadataItem` to upload and attach to the `EscDocument` specified by the id parameter.

Return values

EscMetadataItem: A copy of the attached `EscMetadataItem` with the e-Science Central database id set to the actual id value assigned within the database.

Example

```
EscMetadatatItem md = new EscMetadataItem();

md.setCategory(“Category”);

md.setName(“Name”);

md.setValue(“Some text metadata”);

md.setMetadataType(EscMetadataItem.METADATA_TYPE.TEXT);

EscMetadataItem savedMetadata = c.addMetadataToDocument(“1234”, md);
```

The e-Science Central Workflow API Java client interface

The Java client library provides the following interface to the workflow execution functions within e-Science Central. The methods exposed by this interface are described in more detail in this section of the manual.

```
package com.connexience.api.model;
/**
 * This interface describes the functionality of the externally accessible
 * workflow management API.
 * @author hugo
 */
public interface WorkflowInterface {
 EscWorkflow[] listWorkflows() throws Exception;
 EscWorkflow getWorkflow(String workflowId) throws Exception;
 void deleteWorkflow(String workflowId) throws Exception;
 EscWorkflowInvocation executeWorkflow(String id) throws Exception;
 EscWorkflowInvocation executeWorkflow(String id, String versionId) throws Exception;
 EscWorkflowInvocation executeWorkflowOnDocument(String id, String docId)
                         throws Exception;
 EscWorkflowInvocation executeWorkflowOnDocument(String id, String vId, String docId)
                         throws Exception;
 EscWorkflowInvocation executeWorkflowWithParameters(String id,
                         EscWorkflowParameterList params) throws Exception;
 EscWorkflowInvocation executeWorkflowWithParameters(String workflowId, String versionId,
                         EscWorkflowParameterList parameters) throws Exception;
 EscWorkflowInvocation[] listInvocationsOfWorkflow(String workflowId) throws Exception;
 EscWorkflowInvocation getInvocation(String invocationId) throws Exception;
 EscWorkflowInvocation terminateInvocation(String invocationId) throws Exception;
}
```

listWorkflows();

Description

Returns a list of all of the `EscWorkflow` objects that the authenticated user owns. *Note: This method does not return a list of shared workflow objects.*

Parameters

This method has no call parameters.

Return values

EscWorkflow[]: An array of `EscWorkflow` objects representing the workflows that the authenticated user owns.

Example

```
EscWorkflow[] workflows = c.listWorkflows();
```

getWorkflow(id);

Description

Returns an `EscWorkflow` object given an e-Science Central database id. This method can be used to retrieve a shared workflow object if it's e-Science Central database id is known.

Parameters

id: The e-Science Central database id of the `EscWorkflow` object to return.

Return values

EscWorkflow: the `EscWorkflow` object associated with the specified e-Science Central database id.

Example

```
EscWorkflow workflow = c.getWorkflow("1234");
```

deleteWorkflow(id);

Description

Deletes a workflow object from the e-Science Central system.

Parameters

id: The e-Science Central database id of the workflow to delete.

Return values

This method has no return values.

Example

```
c.deleteWorkflow("1234");
```

executeWorkflow(id);

Description

Execute the latest version of the specified workflow in the e-Science Central system without passing any configuration parameters.

> *Note: the workflow to be executed is placed on a queue within the e-Science Central system and therefore users code should periodically check the status of the workflow to identify when it has actually finished executing.*

Parameters

id: The e-Science Central database id of the workflow to execute.

Return values

EscWorkflowInvocation: An invocation object representing the workflow execution within the e-Science Central system.

Example

```
EscWorkflowInvocation invocation = c.executeWorkflow("1234");
```

executeWorkflow(id, versionId);

Description

Execute a specific version of a workflow in the e-Science Central system without passing any configuration parameters.

> *Note: the workflow to be executed is placed on a queue within the e-Science Central system and therefore users code should periodically check the status of the workflow to identify when it has actually finished executing.*

Parameters

id: The e-Science Central database id of the workflow to execute.

versionId: The e-Science Central database id of the `EscDocumentVersion` containing the version of the workflow to execute.

Return values

EscWorkflowInvocation: An invocation object representing the workflow execution within the e-Science Central system.

Example

```
EscWorkflowInvocation invocation = c.executeWorkflow(“1234”, “345653”);
```

executeWorkflowOnDocument(id, docId);

Description

Executes a workflow on a specified document within the e-Science Central system. When a workflow is configured correctly, it can be setup to use an `EscDocument` as an input parameter.

> *Note: the workflow to be executed is placed on a queue within the e-Science Central system and therefore users code should periodically check the status of the workflow to identify when it has actually finished executing.*

Parameters

id: e-Science Central database id of the workflow to execute.

docId: e-Science Central database id of the `EscDocument` to apply this workflow to.

Return values.

EscWorkflowInvocation: An invocation object representing the workflow execution within the e-Science Central system.

Example

```
EscWorkflowInvocation invocation = c.executeWorkflowOnDocument("1234", "124");
```

executeWorkflowOnDocument(id, vId, docId);

Description

Executes a specific version of a workflow on a document within the e-Science Central system. When a workflow is configured correctly, it can be setup to use an `EscDocument` as an input parameter.

> *Note: the workflow to be executed is placed on a queue within the e-Science Central system and therefore users code should periodically check the status of the workflow to identify when it has actually finished executing.*

Parameters

id: e-Science Central database id of the workflow to execute.

vId: The e-Science Central database id of the `EscDocumentVersion` containing the version of the workflow to execute.

docId: The e-Science Central database id of the `EscDocument` to apply this workflow to.

Return values

EscWorkflowInvocation: An invocation object representing the workflow execution within the e-Science Central system.

Example

```
EscWorkflowInvocation invocation = c.executeWorkflowOnDocument(“1234”, “34”, “124”);
```

executeWorkflowWithParameters(id, params);

Description

Executes the latest version of the specified workflow using a set of workflow parameters to override a subset of the workflow settings.

> *Note: the workflow to be executed is placed on a queue within the e-Science Central system and therefore users code should periodically check the status of the workflow to identify when it has actually finished executing.*

Parameters

id: The e-Science Central database id of the workflow to execute.

params: An `EscWorkflowParameterList` object containing the parameters to override.

Return values

EscWorkflowInvocation: An invocation object representing the workflow execution within the e-Science Central system.

Example

```
EscWorkflowParameterList params = new WorkflowParameterList();

params.addParameter("input", "Source", "78778");

params.addParameter("model", "Iterations", 45);

EscWorkflowInvocation i = c.executeWorkflowWithParameters("134", params);
```

executeWorkflowWithParameters(id, versionId, params);

Description

Executes a specific version of a workflow using a set of workflow parameters to override a subset of the workflow settings.

> *Note: the workflow to be executed is placed on a queue within the e-Science Central system and therefore users code should periodically check the status of the workflow to identify when it has actually finished executing.*

Parameters

id: The e-Science Central database id of the workflow to execute.

versionId: The e-Science Central database id of the `EscDocumentVersion` containing the version of the workflow to execute.

params: An `EscWorkflowParameterList` object containing the parameters to override.

Return values

EscWorkflowInvocation: An invocation object representing the workflow execution within the e-Science Central system.

Example

```
EscWorkflowParameterList p = new WorkflowParameterList();

p.addParameter(“input”, “Source”, “78778”);

p.addParameter(“model”, “Iterations”, 45);

EscWorkflowInvocation i = c.executeWorkflowWithParameters(“134”, “4545”, p);
```

listInvocationsOfWorkflow(id);

Description

Returns a list of all of the invocations of a specified workflow that are owned by the authenticated user.

> *Note: This method will return invocations of any status. i.e Running, Finished etc.*

Parameters

id: The e-Science Central database id of the workflow to list the invocations of.

Return values

EscWorkflowInvocation[]: An array of `EscWorkflowInvocation` objects containing all of the invocations of the specified workflow.

Example

```
EscWorkflowInvocation[] invocations = c.listInvocationsOfWorkflow("1234");
```

getInvocation(id);

Description

Returns a specific invocation (as specified by invocationId) of a workflow.

> *Note: The status value of this invocation will reflect the status of the workflow within e-Science Central at the time that this method was invoked. Repeatedly invoking this function is a mechanism for identifying when a particular workflow invocation has finished.*

Parameters

id: The e-Science Central database id of the `EscWorkflowInvocation` object to retrieve.

Return values

EscWorkflowInvocation: An `EscWorkflowInvocation` object representing the state of the specified invocation at the time the call was made.

Example

```
EscWorkflowInvocation invocation = c.getInvocation("35445");
```

terminateInvocation(id);

Description

Terminates a running invocation or removes a queued invocation from the system.

Parameters

id: The e-Science Central database id of the `EscWorkflowInvocation` to terminate.

Return values

EscWorkflowInvocation: An invocation object representing the state of the invocation at the time the call was made.

Example

```
EscWorkflowInvocation invocation = c.terminateInvocation("35445");
```

e-Science Central API object types

EscUser

The `EscUser` object represents a record of a user within e-Science Central. When signed in using the API, all operations will be performed as the user that has been authenticated for the session. Details of this user are represented by an `EscUser` object which can be obtained from the client. `EscUser` objects are transmitted to and from the server using the following JSON representation:

```
{
  "id": "8a808282328c961d01328c9a99130000",
  "name": "Hugo Hiden",
  "surname": "Hiden",
  "firstName": "Hugo"
}
```

The Java client library provides a Java representation of an `EscUser` with the following attributes:

Property Name	Description
id	The e-SC database id of the user. This is used internally by e-Science Central to locate and manage the user.
name	The full name of the user. This is composed of the firstName + surname properties and is the name displayed in the e-Science Central user interface for the user.
firstName	The given name of the user.
surname	The family name of the user.

EscDocument

All pieces of data within esc are represented as document objects (`EscDocument`) which are stored with folders (`EscFolder`). `EscDocuments` only contain a description of the data stored - they do not actually contain any data themselves. Because each file within e-SC can contain multiple versions, each `EscDocument` has a number of distinct versions associated with it (`EscDocumentVersion`). Each time a file is uploaded against an `EscDocument` a new `EscDocumentVersion` is created. `EscDocuments` are transmitted to and from the server using the following JSON representation:

```
{
  "id": "1257",
  "name": "mixed.csv",
  "creatorId": "8a808282328c961d01328c9a99130000",
  "downloadPath": "/data/1257/latest",
  "uploadPath": "/data/1257",
  "currentVersionNumber": 3,
  "containerId": "8a808282328c961d01328c9a9dfc0003",
  "currentVersionSize": 32
}
```

The Java client library provides the following `EscDocument` object:

Property Name	Description
id	The e-SC database id of the document.
name	The filename of the document.
creatorId	e-SC database id of the user (`EscUser`) that created the document.
downloadPath	Relative URL of the contents of the latest version of the document within the API website.
uploadPath	Location where the content of new versions of the document should be POSTed to.
currentVersionNumber	The version number of the latest version of the document.
currentVersionSize	The size (in bytes) of the contents of the latest version of the document.
containerId	Id of the folder (`EscFolder`) containing this document.

EscDocumentVersion

Uploading data against an `EscDocument` creates a new `EscDocumentVersion` object. An `EscDocument` can have multiple `EscDocumentVersions` associated with it representing the history of contents of that document. `EscDocumentVersions` are represented and transmitted using the following JSON representation:

```
{
  "timestamp": 1333027217828,
  "id": "8a88100c",
  "userId": "8a8082823",
  "downloadPath": "/data/8a88100c36/8a8810f7d1a406c8",
  "documentRecordId": "8a88100c3218c06c6",
  "versionNumber": 1,
  "comments": "File from: Support Script Examples",
  "size": 14
}
```

The Java client library provides an `EscDocumentVersion` object which contains the following attributes:

Property Name	Description
id	e-SC database id of the document version within e-SC.
timestamp	The time that this document version was created (represented as the difference, measured in milliseconds, between the current time and midnight, January 1, 1970 UTC.).
userId	e-SC database id of the user (`EscUser`) that uploaded this version of the document.
downloadPath	Relative URL within the API web server from which to download the actual contents of this document version.
documentRecordId	e-SC database id of the document (`EscDocument`) that this `EscDocumentVersion` is a version of.
versionNumber	The integer number of this version.
comments	Contains any comments that have been attached to this specific version of the `EscDocument`.
size	The size (in bytes) of the actual contents of this version of the `EscDocument`.

EscFolder

An `EscFolder` represents a logical folder within the e-Science Central filesystem. Folders can hold multiple `EscDocuments` and multiple child folders (`EscFolders`). Each user within the system has a default "home" `EscFolder` which is used to store all of their data. This home folder can contain an arbitrary number of `EscDocument` objects and also `EscFolder` child folders. `EscFolders` are represented and transmitted to and from the API using the following JSON representation:

```
{
  "id": "8a808282328c",
  "description": "Home folder for Test User",
  "name": "Test User Home Folder",
  "creatorId": "8a808282328c9",
  "containerId": "8a808282328"
}
```

The Java client library provides an object representation of an `EscFolder` with the following attributes:

Property Name	Description
id	The e-SC database id of the folder. This is used internally by e-Science Central to locate and manage the folder.
name	The name of the folder. This is the name that is displayed in the Data browser.
description	Text describing this folder.
containerId	e-SC database id of the folder containing this folder. This is effectively the parent folder of this folder.
creatorId	e-SC database id of the `EscUser` that initially created (and now owns) this folder.

EscMetadataItem

Documents within e-Science Central can have metadata attached to them which is used to describe the contents of the document in a way that can be used in a search term later on. Documents can have an arbitrary number of metadata items attached to them of the following types:

- Text: A simple piece of textual metadata
- Date: A value representing a single point in time
- Numerical: An arbitrary numerical value
- Boolean: A true/false value

These metadata items can be given names and organised into categories which allows moderately sophisticated search metadata to be attached to documents within e-Science Central. `EscMetadataItems` are represented and transmitted to and from the API using the following JSON representation:

```
{
  "id": 13651,
  "category": "",
  "objectId": "7013659a7ee7e0657",
  "name": "SubjectID",
  "stringValue": "0x1.c8p8",
  "type": "text"
}
```

The Java client library provides an object representation of an `EscMetadataItem` with the following attributes:

Property Name	Description
id	The e-Science Central database id of the `EscMetadataItem`.
category	The category name in for this `EscMetadataItem`.
objectId	The id of the `EscDocument` that this `EscMetadataItem` is attached to.
name	The name of this `EscMetadataItem`.
stringValue	The contents of this `EscMetadataItem` represented as text. This text is automatically converted into the correct format when it is uploaded.
type	Data type of this `EscMetadataItem`. This can be `boolean`, `date`, `numerical` or `text`.

EscWorkflow

An `EscWorkflow` represents a record of a workflow within the e-Science Central system. As workflows are stored in the same way as data, workflows have a number of `EscDocumentVersions` associated with them that contain the actual workflow data for each version. `EscWorkflows` are represented and transmitted to and from the API using the following JSON structure:

```
{
  "id": "3311",
  "description": "Workflow",
  "name": "Reference",
  "creatorId": "786875",
  "currentVersionNumber": 2,
  "containerId": "56565",
  "currentVersionSize": 10581
}
```

Because of the similarity to `EscDocuments`, `EscWorkflow` objects share many common attributes with `EscDocuments`. The Java client library provides an object representation of an `EscWorkflow` with the following attributes:

Property Name	Description
id	The e-SC database id of the workflow.
name	The filename of the workflow.
creatorId	e-SC id of the user (`EscUser`) that created the workflow.
currentVersionNumber	The version number of the latest version of the workflow.
currentVersionSize	The size (in bytes) of the contents of the latest version of the workflow.
containerId	e-SC id of the folder (`EscFolder`) containing this workflow.
description	Text describing this workflow.

EscWorkflowInvocation

Whenever a workflow is executed in e-Science Central, a workflow invocation object is created. Internally, workflow invocations are treated as folders with some additional attributes and, therefore `EscWorkflowInvocations` share many of the same properties as `EscFolders`. `EscWorkflowInvocations` are represented and transmitted by the API using the following JSON structure:

```
{
  "id":"3965",
  "workflowVersionId":"3327",
  "status":"Finished",
  "workflowId":"3325",
  "description":"Workflow data folder",
  "name":"TestBasicServices run #499",
  "creatorId":"786786",
  "containerId":"3462"
}
```

The Java client library provides the following object representation of an `EscWorkflowInvocation`:

Property Name	Description
id	The e-SC database id of the workflow invocation.
name	The name of the invocation.
description	Text describing this invocation.
containerId	The e-SC database id of the folder containing this invocation.
creatorId	The e-SC database id of the `EscUser` that initially created (and now owns) this invocation.
workflowId	The e-SC database id of the `EscWorkflow` that this is an invocation of.
workflowVersionId	The e-SC database id of the `EscDocumentVersion` containing the data representing the structure of this workflow.
status	Text describing the execution status of this workflow. (`Queued`, `Running`, `Debugging`, `Finished`, `ExecutionError`).

EscWorkflowParameter

When workflows are executed via the API, there is a facility that allows external code to configure various properties of workflows that are executed. The API allows a set of parameters to be passed during the execution code that can overwrite any of the block properties when the workflow is executed. These parameters are represented as a collection of `EscWorkflowParameter` obejcts which contain the new parameter values to set in the workflow when it is executed. `EscWorkflowParameter` objects are represented by the API using the following JSON structure:

```
{
  "name": "Interations",
  "blockName": "MyBlock",
  "value": "55"
}
```

The Java API client provides an object representation of an `EscWorkflowParameter` object with the following attributes:

Property Name	Description
name	The name of the block property to overwrite.
blockName	The name of the block containing the property to overwrite.
value	The new value to set the property to. This is specified as a String, but the workflow engine will convert this to the correct type to match the actual type of the parameter being modified.

EscWorkflowParameterList

When executing workflows using a list of parameters via the API, these parameters are passed in using an `EscWorkflowParameterList` which groups these parameters into a single object. `EscWorkflowParameterLists` are represented and transmitted by the API using the following JSON representation:

```
{"values": [
  {
    "name": "Iterations",
    "blockName": "Block1",
    "value": "45"
  },
  {
    "name": "Neurons",
    "blockName": "Block1",
    "value": "3"
  },
  {
    "name": "FileName",
    "blockName": "Block3",
    "value": "data.csv"
  }
]}
```

The e-Science Central Java API client provides an object representation of the `EscWorkflowParameterList` with the following methods:

Property Name	Description
addParameter	Adds a new `EscWorkflowParameter` to the list. This parameter is added using a name, blockname, value call with the following structure: `list.addParameter("paramName", "blockName", "value);`
getValues	Returns an array of `EscWorkflowParameter` objects containing the current values of the parameters stored in the list.
setValues	Sets all of the parameter values in one call by passing in an array of `EscWorkflowParameter` objects.