

Table of Contents

1 HB:Obtima Administration	1
1.1 Create a trial.....	1
1.2 Create a new ObTiMA user account.....	1
1.3 Assign trial chairman role.....	1
1.4 User administration.....	1
2 HB:Trial Chairman	2
2.1 Request ObTiMA accounts / Request a new trial.....	2
2.2 Designing a trial.....	2
2.3 Administration of users in a trial.....	2
2.4 Validation of clinical data collected in a trial.....	2
3 HB:Clinician	3
3.1 Enrollment of a patient to a trial.....	3
3.2 Collecting clinical data on CRFs.....	3
4 HB:Importing data	4
5 Why do I need to import databases directly to ACGT?	5
6 How to add my database - the import process	6
6.1 Adding a database with a default mapping.....	6
6.2 Adding a database with your own RDF mapping.....	7
7 What can I do with my data resources after adding it?	8
7.1 Simple actions.....	8
7.2 Accessing my data in workflows through SPARQL queries.....	9
8 HB:Storing data	10
9 Where and how is data stored in the ACGT environment?	11
10 Accessing data in the file system	12
11 Searching for files	13
12 Uploading a file to the ACGT file system	14
13 Deleting files	15
14 Manage file permissions	16
15 Manage metadata associated to your files	17
16 HB:Retrieving data	18
16.1 Introduction.....	18
16.2 Creating new queries to retrieve data.....	18
16.3 Creating basic queries.....	18
16.4 Using advanced features.....	22
16.5 Saving and submitting queries.....	23
17 registration of services	24
18 Where can services be found or registered in the ACGT system?	25
19 Registering an R-based service	26
20 Registering a tool	27
20.1 Adding a tool operation.....	27
20.2 Adding a tool location.....	27
21 Finding a tool/R-based service	29
22 registration of data types	30
23 Finding a data type	31
24 HB:Setting up workflows	32

Table of Contents

25	What is the Workflow Editor?	33
26	What you can do with the Workflow Editor	34
27	Editing / Designing workflows	35
28	What you can access and add to your workflows	36
28.1	Data Pools	36
28.2	Services	36
29	How to add data pools and services	37
29.1	Add a file from DMS	37
29.2	Adding a service	37
29.3	Adding a Biomoby service	37
29.4	Adding an R-Script	37
29.5	Adding a mediator Query	38
30	Save/Load and publish workflows	39
31	Executing a workflow	40
31.1	Execution from the service repository	40
31.2	Workflow execution output	40
31.3	Monitoring the execution of a workflow	40
32	Publishing a workflow	41
33	HB:Analyzing data with R	42
33.1	Connecting to ACGT GridR servers	42
33.2	Entering commands and scripts	42
33.3	Work with multiple GridR Sessions	43
33.4	Working with the File Explorer	43
33.5	GridR special commands	44
34	HB:ACGT Master Ontology	45
34.1	What is the ACGT Master Ontology?	45
34.2	Viewing the ontology	45
34.3	Updating the ontology	45
34.4	See also	45
35	PM:Integrating R scripts	46
36	Introduction	47
37	Login to the ACGT Portal	48
38	Open the Metadata Registration Page	49
39	Choose Functional Category	50
40	Open Registration Mask	51
40.1	Enter Description	51
40.2	Enter Parameters	51
40.3	Enter Code	51
40.4	Finalize the Registration	52
41	Checking the registered Script	53
42	Editing the registered Scripts	54
43	Using and executing a registered Script	55
44	PM:Developing a new service	56
45	Introduction	57
46	Preparing Hosting Environment (Service Container)	58
46.1	Creating a low privilege account	58
46.2	Setup CA certificates	58
46.3	Requesting service certificate	58
46.4	Downloading and installing Globus Toolkit 4.1.2	59

Table of Contents

46 Preparing Hosting Environment (Service Container)	
46.5 Downloading and installing Apache Tomcat	59
46.6 Deploying wsjava into Tomcat	59
46.7 Configuring Tomcat	60
46.8 Starting Tomcat	62
47 "Hello World!" Service and Client	63
47.1 Creating WSDL file describing the service interface	63
47.2 Implementing gsi-enabled Web Service	65
47.3 Implementing gsi-enabled Web Service Client	68
48 Globus Authorization Framework	73
48.1 GAS Plug-in	73
49 PM:Developing a new Proxy service	74
49.1 Introduction	74
49.2 Generic Guidelines for the Implementation	74
49.3 Examples	75
50 PM:ACGT Testbed Security Guide	77
50.1 Security	77
51 PM:ACGT Bug Reporting	82
52 ACGT Bug Reporting - General Guidelines	83
52.1 Introduction	83
52.2 Which types of bugs to report	83
52.3 How to report a bug	83
53 EX:ACGT End to End Scenario	84
54 Overview of the scenario	85
55 Prerequisites	86
56 Scenario	87
56.1 ObTiMA: Creation of a trial (CRF preparation)	87
56.2 ObTiMA: Running trial / Edition of patient information	87
56.3 Data mining: Creation of a mediator query	88
56.4 Data mining: Registration of an external database	88
56.5 Data mining: Registration of a GridR analytical service	89
56.6 Data mining: Creation and Registration of a workflow	89
56.7 Data mining: ACGT DMS and Visualization of results	91
56.8 Appendix: GridR script for data mining	91
57 EX:Integrated demonstrators	93
57.1 Hokkaido wf07	93
57.2 Hokkaido wf06	93
57.3 Hokkaido wf05	93
57.4 Hokkaido wf04	93
58 EX:Multi-centric multi-platform data analysis	94

1 HB:Obtima Administration

1.1 Create a trial

1.2 Create a new ObTiMA user account

1.3 Assign trial chairman role

1.4 User administration

2 HB:Trial Chairman

2.1 Request ObTiMA accounts / Request a new trial

A step by step introduction can be found in the [Obtima/Request_Access](#) page.

2.2 Designing a trial

2.3 Administration of users in a trial

2.4 Validation of clinical data collected in a trial

3 HB:Clinician

3.1 Enrollment of a patient to a trial

3.2 Collecting clinical data on CRFs

4 HB:Importing data

This section describes how data resources from outside of the ACGT system can be imported and accessed in ACGT Environment.

The **Data Access** area in the [ACGT Portal](#) allows end-users to connect databases to the ACGT Environment. Queries can be created over this databases and used in [workflows](#).

*For a detailed description of the **Data Access** area see [HB:Data_Access](#).*

*For a detailed description of the **Workflow Editor** area see [Workflow editor](#).*

6 How to add my database - the import process

Registering an SQL database can be done in the [Data Access](#) tab in the [ACGT Portal](#)

- Click on the *Add a new data resource* button to start the process:



Add new data resource

6.1 Adding a database with a default mapping

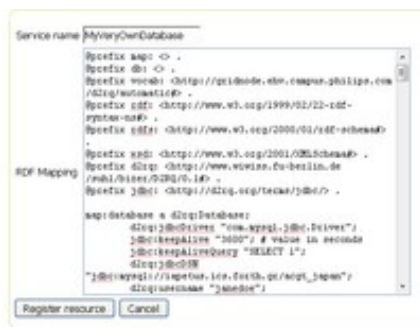


Example connection details

- Select the check box corresponding to *A default mapping*
- Provide the connection details for the database:
- Specify:
 - the **location** of the database - the JDBC connection string
 - the **username** required to access the database
 - the **password** required to access the database
 - the **dialect** of the database (*MySQL, PostgreSQL, Oracle*)

A test database with public data is available with the connection string:
`jdbc:mysql://iapedus.ics.forth.gr/acgt_transbig`
Another example of connection settings to a public database can be found [here](#).

- Click on *Next*



Example name and automatic RDF mapping

- Provide a name which will be used in the resource's actual name:

Example: If you name it XX, the resource will actually be accessible later with its full name:
`"Dynamic-XX-ogsadai-nnnnnnnnn"` (an automatic random unique identifier is added to avoid duplicates)

- A default **RDF mapping** describing the database is shown in an editable form.

If you want, you can modify it before finishing the import process (e.g. for optimizations).



Note: You can also save the RDF description separately for further reference.

- Click *Register resource* to finish the process

6.2 Adding a database with your own RDF mapping



Selecting your RDF mapping

- Select the check box corresponding to *My own mapping*



Note: The connection details are not requested at this step as they are part of the RDF mapping you must provide!

- Click on *Next*



Example name with **your** RDF description

- Provide a name which will be used in the resource's actual name:

Example: If you name it XX, the resource will actually be accessible later with its full name: "Dynamic-XX-ogsadai-nnnnnnnnnn" (an automatic random unique identifier is added to avoid duplicates)

- Provide the RDF (**Resource Description Framework**) mapping/description of the database.
- Click *Register resource* to finish the process

7 What can I do with my data resources after adding it?

After adding the data you can perform several [actions directly in the Data Access tab](#) or you can use it in the [Workflow editor](#).

7.1 Simple actions

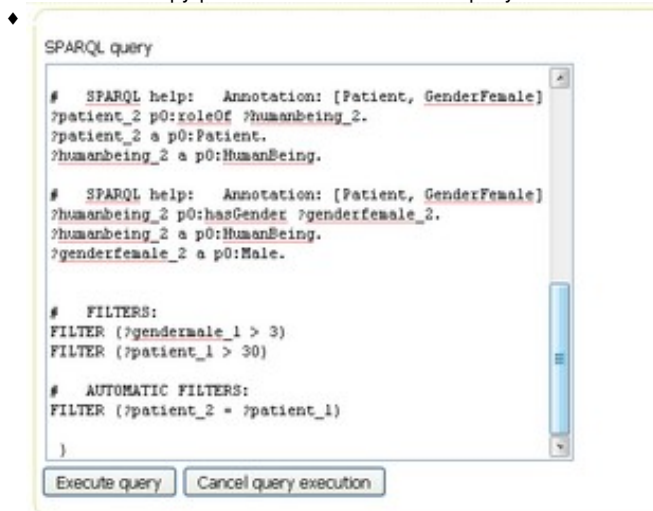
There are a few simple actions which can be done directly in this area:

- First, navigate in the database list to the database you wish to use - you can find it by the name you have given it in the importing process, see [the resource name format](#).
- You will find the actions in the fourth column of the [the resource name format](#).
- You can get the [RDF Schema](#) by clicking the *Get Schema* button.



Example of RDF Schema

- You will see either the default mapping that was created for you automatically or the schema you have supplied yourself in the import process.
- You can check and copy the RDF schema for further reuse.
- You can execute [SPARQL queries](#) on the data resource by clicking *Execute Query* button.



Example of SPARQL query

- Click the *Execute Query* button to see the results
- After reviewing the results of the [SPARQL](#) query you can save the results as [.csv files](#).

Note: Thus, you can use simple queries to test that the connection to the database is correct and that the import process was successful.

- You can **delete** the connection to the database by clicking the *Delete* button.

Caution: This will only delete the dynamic link and not the data resource itself. The data resource will no longer be available in the ACGT Environment unless it is imported again.

7.2 Accessing my data in workflows through SPARQL queries

In order to use your data in **complex scenarios**, you should embed queries on this data resource as part of **workflows**:

- Go to the [Workflow Editor](#).
- Add a generic SPARQL service that will be used to query your database. The ACGT environment has such a generic service that can be used to add arbitrary SPARQL queries to a workflow - called **Submit SPARQL**.
 - ◊ Locate the Services Panel - [left \(west\) frame of the editor](#)
 - ◊ Navigate in the tree and select **Services / Bioinformatics / Proxies / DynamicDAS / SubmitSPARQL**
 - ◊ Click on the "**SubmitSPARQL**" item
 - ◊ Drag-and-drop it on the workflow editor - you now have a box called "SubmitSPARQL" with four input (blue) dots and one output (red) dot.
- Configure the "**SubmitSPARQL**" service:
 - ◊ Connect the first blue dot in this box to an empty space to create the dynamic resource box. Double-click it and give it the name you wrote down after registering your dynamic database (ex:*Dynamic-Hokkaido-ogsadai-124777a375b*).
 - ◊ Connect the second blue dot of the "**SubmitSPARQL**" box to an empty space to create a SPARQL Query. Double-click this box and copy-paste your SPARQL Query.
 - ◊ Drag the third (from left to right) blue dot of box (called "DirName") to an empty space. A green box will be created. This will be the output directory that the service will use to store its results and any temporary files it might need.
 - ◊ Connect the fourth and last blue dot of the "**SubmitSPARQL**" box to an empty space. This **FileName** will be used to create a CSV file where it will store its results.
 - ◊ The single red dot contains the DMS FileID of the resulting CSV file. You can connect this dot to other components in your workflow (for example an R script using this file as input for analysis).

Hint: A working example is presented in [step 6 of the tutorial "ACGT End to End Scenario"](#).

8 HB:Storing data

This section is an introduction on how to use the ACGT distributed file system, how you can store, access and manipulate their own data.

*For a description of the user interface (called Data Management System - **DMS**) see [HB: Data_Management](#).*

9 Where and how is data stored in the ACGT environment?

In the ACGT system, you can store data in a [distributed file system](#). This file system allows you to access the files from different computers simultaneously providing an experience that is sufficiently similar to storing files on a single regular desktop computer.



Caution: Because having distributed access to files is technically *more complex* than having files on a single computer, working with this file system is sometimes *different*.

10 Accessing data in the file system

Each user has his own personal space (*folder*) in ACGT's Data Management System - **DMS**.

- See the [My Files tab](#) section for instructions on accessing this personal space through the [ACGT Portal](#).
- For a description of the user interface see [HB:Data_Management](#)

11 Searching for files

You can search for a file using various criteria in the **Search Mode** of the DMS interface.

12 Uploading a file to the ACGT file system

- **Step 1:** To access the ACGT File system (DMS)
 - ◆ Click on the **My Files** tab of the ACGT portal.
 - ◆ The DMS interface appears in **File-Directory Mode**.
 - ◆ You start from your personal workspace as your current working directory:

```
root/users/YOUR-ORGANIZATION/YOUR-USER-NAME
```

- - ◆ Use the left panel to navigate the directory structure of the DMS file system.



Note: This way you can use the other panel - the right panel - to display the contents of files or other information depending on the context.

- **Step 2 (optional)** - You can **create a directory** for your files:
 - ◆ In the *Action panel* make sure the **Add directory** button is selected. If it is not selected, click it.
 - ◆ Write the name of the directory to be created in the **Name** text field
 - ◆ Click **Execute**
 - ◆ The directory with the selected name should now be displayed in the *left panel*.
 - ◆ Click on its name to open it and see its contents.



Caution: Uploading a file will store it in the directory **currently open** in the left panel! You can always swap the left and right panels if you decide you want to upload something in the directory displayed in the right panel.

- **Step 3** - To upload a file from your computer to the DMS filesystem:
 - ◆ Navigate to the directory in which the file has to be stored by clicking **[..]** (in column 2) to go **up** one level in the tree **or** clicking on the Directory Name (in column 4 - *Name*) to go inside a directory.
 - ◆ In the *Action Panel* (below) select either the **Upload file** or **Upload file directly** (*see the description of the actions to see the difference*)
 - ◆ In the **Name text** field, write down the name of the file as you would like it to appear in the DMS
 - ◆ Click on the **Browse** button to select from your computer the file to upload.
 - ◆ Then click on **Execute**.



Note: The file is will then transferred to the ACGT DMS - and it shown as **IN_PROGRESS**

Once uploaded, the file contents can be visualized (in the right panel) by clicking on its name in the left panel.

13 Deleting files

Files and directories stored on the DMS can be **deleted**:

- Check the box which is located in the first column of the panel on the same row with the file or directory you want to delete
- You can check multiple files or directory to apply the same kind of action to all of them at once
- You can use the **Select all** / **Deselect all** buttons to select all the files at once
- Click the **Remove** button in the left side of the **Action Panel**.



Note: Directories must be emptied before they can be removed. First you need to click and enter the directory, delete all the files, then go up one level and delete the directory itself.

14 Manage file permissions


You can give and change permissions for **each** of the files and directories you have created. You can give permissions to each individual user or group available in the ACGT Environment.

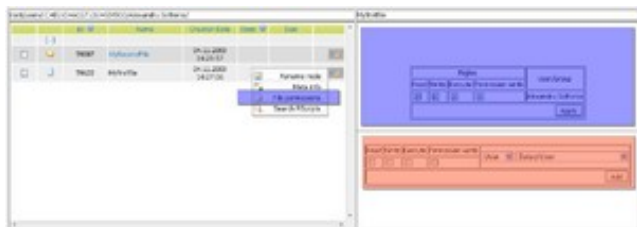
There are **4 different permissions**:

- **Read** - Previewing a file's contents
- **Write** - Changing the content of a file (overwrite)
- **Execute** - Using a file
- **Permission write** - Changing the permissions

Note: Note that the *read / write / Permission write* also **require** the *execute* permission.

To see or modify the permission for a file or directory that is already

- Click on the **Actions** image  in the last column of the panel (same row as the file or directory you want to change permissions)



Change file permissions

- Click the *File permissions* option from the *Actions menu* (left blue area).
- The permission management interface will be displayed in the *Right Panel*.
- A list of existing permissions will be displayed (right blue area).
- You can modify this list adding and changing permissions for each user or each group (right red area)
- Click the **Apply** button to save your changes modify the existing permissions.
- If you want to add a new permission which is not shown in the list select either *User* or *Group*
- Select the individual user or the group
- Check the permission(s) you want the user/group to have

Note: Note that the *Read / Write / Permission write* also **require** the *Execute* permission.


- Click the **Add** button to add the rule to the existing permissions.

15 Manage metadata associated to your files

Property	Value
ID	302
Authority	ACGT
Language	English (US)
Owner	5961
Version	0.2
owner	genome
datatype	FASTQ
datadescription	Sample text file
provenance	Applied by user
content-type	text/plain



Move files and directories

- Go to the file you want to see or update the metadata info
- Click on the **Actions** image  and select the "Meta Info" option. In the right panel a form will appear allowing you to set some predefined properties to the selected file. The properties are defined by the ACGT Metadata Schema. After you have finished editing press the "Apply" button to save changes. To drop the changes and go back to browsing mode click the "Back to dir view" button.

16 HB:Retrieving data

This section describes how to create queries for retrieving data from trials and incorporate those queries to workflows.

*For a description of the tool used to create the queries see [HB:Query_Tool](#).
A step-by-step example can be seen here [EX:ACGT_Query_Tool](#)*

16.1 Introduction

A real-life challenge when having data from different clinical trials which are stored in **multiple** repositories is to access the data *as if* it were stored in a **single** repository. Accessing this data from a "global repository" requires a *semantic* integration of the repositories. Such an integration requires a special ontology to handle all the information that needs to be handled in the environment as the basis for a structured knowledge repository. The **ACGT Master Ontology** is the *ontology* that represents the domain of clinical trials on cancer (and in general, cancer research).

In the ACGT environment, data from different trials can be accessed *as if it was stored in a single global repository*, allowing the semantic integration of the different repositories. The schema of this "global repository" is the **HB:ACGT Master Ontology**, an ontology that defines the domain of clinical trials on cancer. A software module called the *mediator* is in charge of handling queries in terms of this global schema and translating them into subqueries for each of the specific trials. The language of these queries is **SPARQL**.

In order to hide the complexity of this language, the ACGT environment includes the **Query Tool**, a web-based tool designed to allow end-users to easily define new queries to access the data from registered trials. These queries can be tested and uploaded to a "query repository" for its subsequent use in the **Workflow Editor**.

16.2 Creating new queries to retrieve data

This section covers the creation of **SPARQL** queries using the **Query Tool**. There are several functions and features in the **Query Tool** that allow you to create queries to suit your needs. In most cases, only a subset of those features is actually used.

Please read the **interface section** first for a description of the different areas of the Query Tool. There are 6 separate sections of the Query Tool and 5 important areas that are used in the following sections:

- The section labeled **Creating basic queries** covers the most simple cases.
- The section labeled **Using advanced features** covers the remaining features that allow you to build more specific and complex queries.
- The section labeled **Saving and submitting queries** covers the actions that can be done with a query in the Query Tool: show and edit the SPARQL code, submit the query to see the results, save the query to a local database or upload it to the metadata repository.

16.3 Creating basic queries

16.3.1 Select trial

The first step is to select the trial from which you want to get the data.

- Locate the **Repository area**, the first area of the **Query Tool**.
- Check the list of trials and refresh the list (with the **Load Repositories** button) if you do not see the trial you want.
- Choose a trial from the list and click on the **Select** button.
- Check that the **Entries area** is no longer empty in **area 2** and the trial name is shown (colored with gray) above the entries list

(+) Query Description

Available variables to select **3** ? Selected variables

Created Filters **4** ?

Limit of the query result All **5** ?

Show Reset

UpLoad Query to metadata repository Submit Save Query

(+) Edit Query SPARQL

(+) Result

(+) Saved Queries

(+) Create Query

1 ? Repository TOP Clinical Trial database
MCMP simple C
(No description available for this mappin

Load Repositories

Repository DBId ObtimaResource

Entries of the repository MCMP simple C

2 ? Gender of Patient - Male
Gender of Patient - Female
Gender of Patient - Ambiguous
Gender of Patient
Value of Patient-hasName-Name-hasStringValue-strin

< Add Entry < Add All Entries

Selected entries in the query.

(+) Create Filters



Example - Repository selected

16.3.2 Select information from the trial

The second step is to specify what information you want the query to retrieve from the trial data. This is achieved by adding from the entry list the relevant information. The entries list shows the available entries i.e. "what things can be asked" in the selected trial. The list consists of lines of text in a *natural language*. E.g. an entry labeled *Patients and their names* allows questions regarding the correspondence between the patients of a trial and their names.

- Select (**one-by-one**) the entries that you think are needed for the query and add them using the **< Add Entry** button.
- Alternatively, you can add all the entries at once using the **< Add All Entries** button.
- Check that the entries are added in the Selected entries list below (still in area 2).

Note: The *available variables* list is no longer empty in *area 3*. For each entry, all the variables corresponding to that entry will be added to the list of available variables. E.g. for the entry *Gender of Patient-Male* two variables *Patient* and *GenderMale* will be added as available.

The screenshot shows the 'Query Description' panel on the left and the 'Saved Queries' panel on the right. The 'Query Description' panel includes sections for 'Available variables to select' (listing Patient, ClinicalTrialPatientNumberStr, NameString), 'Selected variables' (empty), 'Created Filters' (empty), and 'Limit of the query result' (set to All). The 'Saved Queries' panel shows a 'Repository' dropdown set to 'TOP Clinical Trial database', a 'Repository DBid' of 'OptimaResource', and a list of 'Entries of the repository' for 'MCMP simple C'. Two entries are selected: 'Value of Patient-hasName-Name-hasStringValue-strin' and 'Value of Patient-hasIdentifier-ClinicalTrialPatientNumb'. The 'Selected entries in the query' section shows these two variables.

© 2008 Biomedical Informatics Group



Example - entries added to the query

16.3.3 Choose the variables

The **third** (and last) **step** left to have a complete (basic) query is to choose the variables returned by the query from the list of available variables.

- Locate **available variables list** in **area 3** (top of the first column). This list of data fields/variables are **available** to be selected as columns of the result table of the query.

Note: By default, the data fields from the entries added to the query in the previous step (e.g. *patients*, *names*, etc) are *available* but not actually *selected* in the query. The difference comes from the fact that selected variables will be actual columns of the result table while the remaining available variables which are not selected will remain unused.

- To specify which data fields will form part of the result of our query (**and in what order**), select each of them and move them to the **Selected variables** area, selecting each of the desired data fields and clicking the >> button.

Caution: The entries in step 2 and the variables in step 3 are **not the same!**

Entries are **relationships** closer to ideas expressed in a natural language, while variables denote actual **data** seen as columns in a database.

E.g. Some entries such as *Gender of Patient - Male* have more than one corresponding variable: *Gender and Male*

- You can always change your mind and move selected variables back to the available variables list by clicking the << button.

(+) Query Description

Available variables to select

Patient

Selected variables

ClinicalTrialPatientNumberStr
NameString

Created Filters

Limit of the query result: All

(+) Edit QuerySPARQL

(+) Result

(+) Saved Queries

(+) Create Query

Repository: TOP Clinical Trial database

MCMP simple C
(No description available for this mappin

Repository DBid: OptimaResource

Entries of the repository: MCMP simple C

Selected entries in the query:

(+) Create Filters



Example - Data fields selected as results of the query.

16.3.4 Test the query

To see the results of the query, as it was defined in the previous steps:

- Click the **Submit** button.
- If no filters or limits are specified, the results of the query will contain a column for each variable selected and all the data.
- Please wait a few seconds for the **Result** section to be loaded with the results:

The screenshot displays the Query Tool interface with two main panels. The left panel, titled '(+) Query Description', '(+) Edit Query SPARQL', and '(+) Result', shows a 'Reset >' button and a list of query results. The results are in a table format with columns for patient ID and name. The right panel, titled '(+) Saved Queries' and '(+) Create Query', shows a 'Repository' dropdown set to 'TOP Clinical Trial database' with 'MCMP simple C' selected. Below this is a 'Load Repositories' button and a 'Repository DBId' field containing 'ObtimaResource'. The 'Entries of the repository' section shows a list of variables like 'Gender of Patient' and 'Value of Patient-hasName-Name-hasStringValue-strin'. There are '< Add Entry' and '< Add All Entries' buttons. The 'Selected entries in the query' section shows the selected variables. At the bottom of the right panel is a '(+) Create Filters' button.

(+) Query Description
(+) Edit Query SPARQL
(+) Result

Reset >

```
clinicaltrialpatientnumberstring_1, namestring_2
"OXFT_114", "Theodora Fig"
"OXFU_535", "Megara Pimento"
"OXFT_669", "Yolanda Calubura"
"OXFT_736", "Charissa Tamarillo"
"OXFU_104", "Candace Wampi"
"OXFU_320", "Teresa Bacac"
"OXFU_573", "Eunice Feijoa"
"OXFT_380", "Circe Rambutan"
"OXFU_662", "Voleta Sesban"
"OXFU_138", "Maeva Bignai"
"OXFT_729", "Odessa Engkala"
"OXFT_209", "Melita Naranjilla"
"OXFU_281", "Eirene Carob"
"OXFT_680", "Alyssa Tangelo"
"OXFT_705", "Kassia Habbel"
"OXFT_3597", "Melaina Guanabana"
"OXFU_531", "Nerita Titoki"
"OXFU_557", "Stella Pedalai"
"OXFT_595", "Zanita Guava"
"OXFT_876", "Melody Sourtop"
"OXFT_619", "Nicolette Sapucaia"
"OXFT_427", "Adenia Comatura"
```

(+) Saved Queries
(+) Create Query

Repository: TOP Clinical Trial database
MCMP simple C
(No description available for this mapping)

Load Repositories

Repository DBId: ObtimaResource

Entries of the repository: MCMP simple C

Gender of Patient
Value of Patient-hasName-Name-hasStringValue-strin
Value of Patient-hasIdentifier-ClinicalTrialPatientNumb
Value of Patient-hasBirthDate-date
Value of Patient-undergoes-Recruitment-hasDate-date

< Add Entry < Add All Entries

Selected entries in the query:

Value of Patient-hasIdentifier-ClinicalTrialPatientNumb
Value of Patient-hasName-Name-hasStringValue-strin

(+) Create Filters



Example - submit the query and see the results that it generates.

16.4 Using advanced features

The Query Tool has two features to allow designing more specific queries, accessible in [area 4](#) and [area 5](#) of the Query Tool interface.

16.4.1 Filtering the data

Filters are restrictions on the data fields (or variables) included in the query. Adding a filter will allow you to restrict the results of your query to some numerical constraints, creating rules such as *include only patients older than 30 years* or whose name matches a specific value.

Note: Creating filters on the variables requires that the necessary entries were added (step 2 above) so that the corresponding variables are in the **available variables** list.

A filter can be created:

- Locate the **Create Filters** section in [area 4](#) (the last section in the right column) of the Query Tool
- Select an available variable as the *first parameter*
- Select an operator (<, >, =) to specify the numerical constraint (*less than, equal, greater than*)
- Enter a numerical value as the *second parameter*
- Click the **Create Filter** button to add the filter in the created filters list.

For example, to select patients older than 30 years, you must restrict the data field *Age* to be greater than 30. Select the field *Age* as the *First parameter*, select the operator *'>'* and write 30 in the *Second parameter* box. Click the **Create Filter** button. You will see the filter named *Age must be greater than 30* appear in the created filters list.

16.4.2 Limiting the results of the query

It is possible to limit the results of the query by specifying the maximum amount of results that a query must generate:

- Locate [area 5](#) of the Query Tool.

- Select from the drop-down list labeled *Limit of the query result* a maximum limit for the results of the query (All, 10, 50, 100 or 500).

Note: The default option is **All** - all possible results will be retrieved after executing the query

16.5 Saving and submitting queries

Saving, submitting and editing queries can be done through the [main action buttons](#) in the Query Tool.

- You can see the SPARQL code, clicking the **Show** button. The code can be copied to the clipboard and then to a file or the [Edit QuerySPARQL section](#)
- You can save a Query to the local database, clicking the **Save Query** button. The query can be loaded at a later time in the Query Tool using the **Saved Queries** section in the right column
- You can submit the query by clicking the Submit button. The actual results will be loaded in the **Result** section(last section, left column)
- You can **upload** the query to the [Metadata Repository](#) so you can use it in the [Workflow Editor](#).

17 registration of services

This section describes how services can be registered in the ACGT system.

18 Where can services be found or registered in the ACGT system?

Browsing the existing services as well as registering new services is done through the **Metadata Registration** tab of the portal, selecting the **Tools** sub-tab.

- The tree on the left hand side can be used to browse the services classified in various functional categories.
- Services can be searched using keywords in the **search** tool in the upper-right side of the portlet.

Registering a new service (or creating a new functional (sub)category) in a specific functional category can be done by clicking on the category in the tree on the left, then by selecting the specific action to be done in the lower right part of the interface.

20 Registering a tool

Clicking on the "Add Tool" button opens a new interface which can be used to register a tool. A number of tool-description fields can then be filled. The mandatory fields are the Name which is then used to identify the service in other ACGT components, the Version which is used to distinguish different versions of the same tool and the WSDL. However it is recommended to fill the other fields as well as they are the source of information when searching for services.



Registering a generic tool

After the tool has been created, one can access it from the service tree. Once selected, the tool operations and tool locations will be displayed, and the user can add/modify the attached operations or the tool locations associated. The user can also modify its service-description attributes or delete the tool.

20.1 Adding a tool operation

When viewing a tool, the tool operation list is displayed. To add an operation, the user must click on the "Add Operation" link, at the bottom of the operation list. This opens a new interface which can be used to add a tool operation. A number of operation-description fields can then be filled. The only mandatory field to register is the Name which is then used to identify the operation in the tool's operation list. The parameters are passed to/from the operation through two lists, one for input parameters, and one for output parameters.



Adding an operation to a tool

After the operation has been created, one can select it from the tool's operation list for modifications of the attributes, or for deleting it.

20.2 Adding a tool location

When viewing a tool, the tool location list is displayed. To add a tool location the user must click on the "Add Tool Location" link, on the bottom of the tool location list. This opens a new interface which can be used to add a tool location. A number of tool location-description fields can then be filled. Note that only one type of tool locations can be associated with a tool.

Add tool location

Tool location attributes

Tool name

Host

Tool location type

Tool location status

Is main tool location

Parameter name	Parameter value	
<input type="text" value="my attribute"/>	<input type="text" value="my value"/>	Delete

[Add attribute](#)



Adding a location for a tool

After the location has been created, one can select it from the tool's location list and modify its attributes, or delete it.

21 Finding a tool/R-based service

The tool interface provides an easy to use search engine for tools/R-based services. To use this engine, the user must fill in the search string and provide the search rule:

1. a combination of all the words contained in the search string
2. at least one of the words contained in the search string
3. a regular expression



Searching for services

Note: The search can be case sensitive, or case insensitive, depending on the user's selection.

22 registration of data types

Some or all of the information on this page is inconsistent, irrelevant or confusing.

Please help clean it up if you are able.

Browsing and registration of data types is based on the same principles as for services, by selecting the "DataTypes" subtab of the "Metadata Registration" tab of the portal.

Clicking on the "Add new data type" link opens a new interface which can be used to register a data type. A number of data type-description fields can then be filled. If the user wants to create a non primitive data type, he can use the existing data types to compose it's new data type by clicking "Add data type component" link and selecting a relation type, and a data type the relation refers to.

Note that when creating a sub data-type (by selecting a data type and then clicking the "Add new data type" button), an "Extended" relation is automatically created between the parent data type and the newly created data type. After the data type has been created, one can select it from the data type tree and modify it's attributes, or delete it.

23 Finding a data type

The data type interface provides an easy to use search engine for data types. To use this engine, the user must fill in the search string and provide the search rule:

1. a combination of all the words contained in the search string
2. at least one of the words contained in the search string
3. a regular expression

Note that the search can be case sensitive, or case insensitive, depending on the user's selection.

24 HB:Setting up workflows

This section describes how to create, edit, execute and publish workflows in the ACGT environment.

For a description of the user interface of the Workflow Editor see [HB:Workflow_Editor](#).

25 What is the Workflow Editor?

Main article: [HB:Workflow_Editor](#)

The **Workflow Editor** aims to assist physicians in their scientific research by supporting the composition of different data access, knowledge extraction and analytical services into **complex workflows**. This way one can extend and enrich the functionality of the ACGT system by reusing *existing* ACGT compliant services and producing "added value" *composite services*.

The **Workflow Editor** and its "environment" attempt to bridge the usability gap between **grid infrastructure** and **clinicians**. Furthermore, the Workflow Editor takes advantage of the computational and storage power of a grid. This is essential for **scientific workflows** which tend to operate on large amounts of data with complex algorithms. The Workflow Environment allows the users to share finished workflows, thus to *exchange information* so that users can benefit from each other's research.

The **Workflow Editor** allows you to:

- **create** your own workflows from scratch in a private workspace
- access and **use** your files and own data resources
- **use** community-shared services (including GridR scripts) and Mediator Queries
- save, **run** and redesign your workflows
- **publish** your workflow in the community shared workspace

26 What you can do with the Workflow Editor

- You can *search* and *browse* the available *services* and *data sources*
- You can create workflows combining available services and data sources through an intuitive and user friendly interface.
- You can *store* the workflows in your private area and later *retrieve* and *edit* them so that new versions can be produced.
- You can *publish* and *share* your workflows so that other users can use them in their research.
- You can *execute* the workflows and *monitor* their enactment status.

27 Editing / Designing workflows

For a description of the user interface of the Workflow Editor see [HB:Workflow_Editor](#).

By default, the workflow editor shows an empty area in the center - a canvas on which a workflow can be designed. A new workflow is created by picking **New** in the *File* menu.

Workflows are constructed by combining various services with data pools, describing the flow of information by creating the appropriate connections between workflow elements. You can connect as many services as you want, add constants and input/output variables to design a workflow that allows you to reach your research goal(s). The workflow design environment allows you to move, connect, delete objects from the central canvas.

At any time you can save your workflow to the ACGT servers and continue working on it a later time, accessing the ACGT Portal from anywhere.

28 What you can access and add to your workflows

28.1 Data Pools

Data pools which can be used in workflows:

- **Constants** (created by selecting an input slot and dragging it to the background of the workflow central panel)
- **Files** (specified by dragging the appropriate file icon from the "My Files" browser on the left to the workflow edition panel)
- Registered **mediator queries** (using the Icon with an "M" in the toolbar below the menu bar)
- **BASE microarray database queries** (created using the "BASE" icon in the tool bar; this component requires has some input parameters allowing the selection of the appropriate data from the BASE database)
- SQL and SPARQL **database queries** allowing the query of **dynamically connected databases**. Note that these components also require additional inputs specifying the external database and the actual query to perform.

Note: SQL and SPARQL database queries can be added to the workflow by dragging the corresponding object from the **Services** tab in the left (west) area:



Services -> Bioinformatics -> Proxies -> [Dynamic]DAS -> submitSPARQL and submitSQL

28.2 Services

Services which can be used in workflows:

- **GridR registered services** - which can be added to the workflow by clicking on the "R" icon from the tool bar
- **Non-GridR ACGT services** - which can be selected from the **Services** tree in the left (West) frame
- External services - currently only **BioMoby services** which can be dragged and dropped from the *Biomoby* tab on the left (West) frame

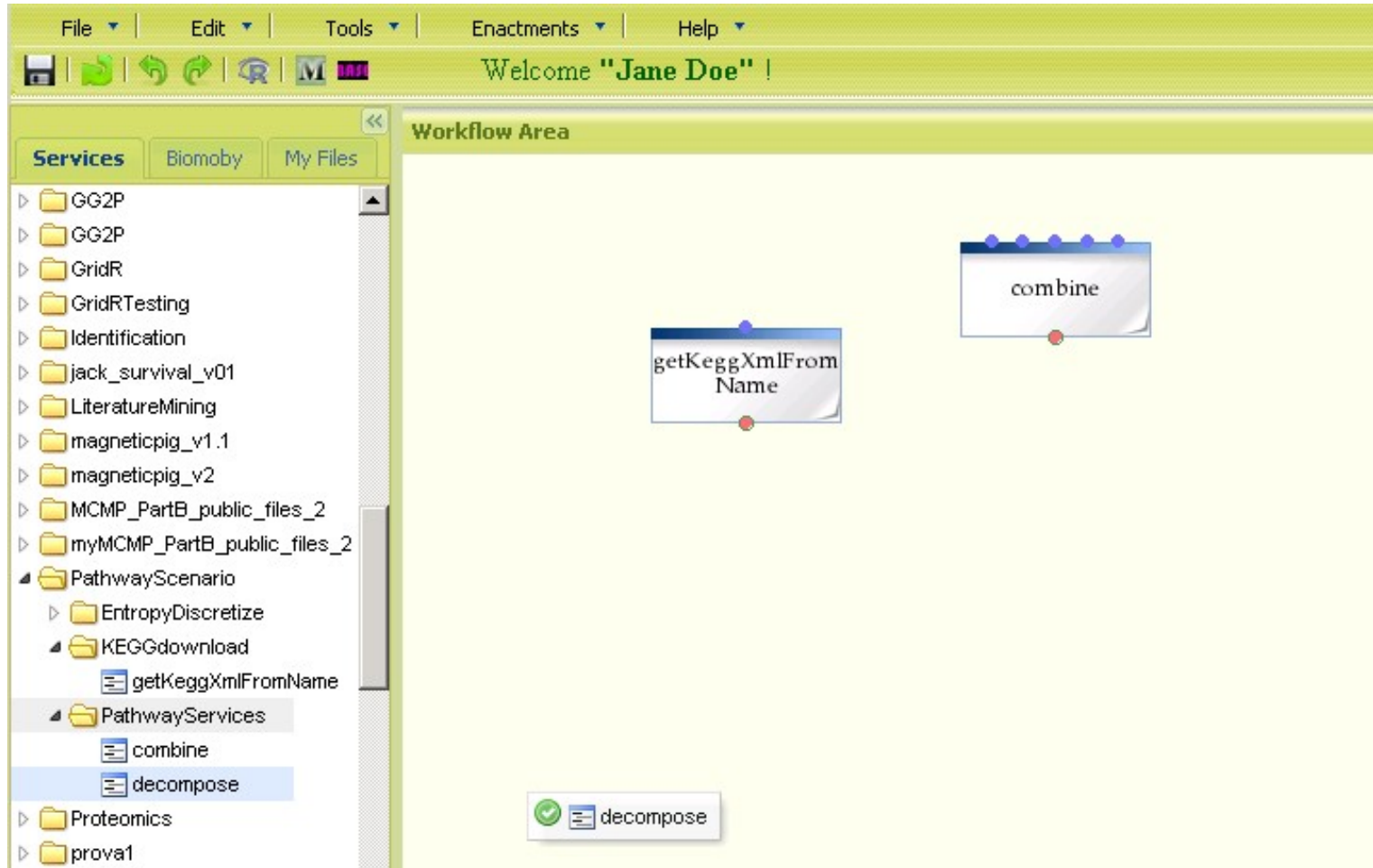
29 How to add data pools and services

29.1 Add a file from DMS

You can add your files as data input, from ACGT's [distributed file system](#) account using the **My Files** tab in the left (West) frame of the Editor. The tree will show only the available files in **your** folder in the [DMS](#).

29.2 Adding a service

Regular services are inserted in the workflow by drag-and-dropping them to the work area or by double clicking the specific service from the **Services** tab in the left (West) panel.




Adding a service in the workflow editing area by drag&drop

29.3 Adding a Biomoby service

Biomoby services can be seen by clicking the *Biomoby* view of the left (west) frame. You can search and double click on the Biomoby service you to add, and finally drag and drop it to the workflow area.

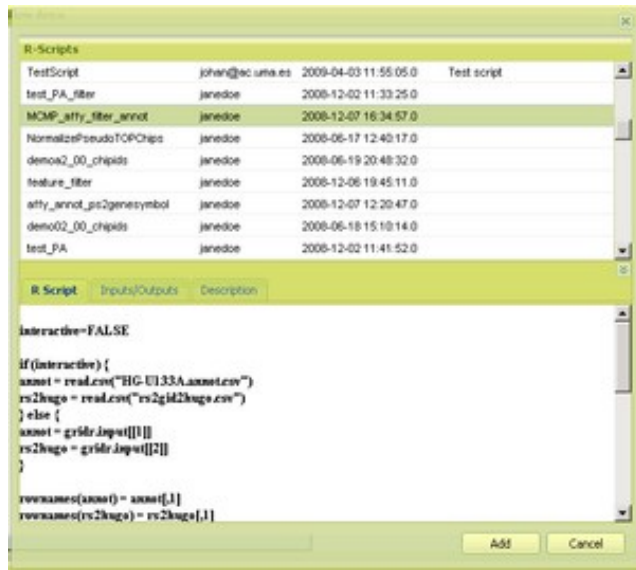
29.4 Adding an R-Script

To insert a GridR service click on the corresponding icon from the upper toolbar: 

 **Note:** GridR-based services are internally treated differently from regular services; the same is true for Mediator queries and access to BASE databases


A window appears which displays all the R-scripts that are available and can be added to the workflow. As we can see in figure below a special window appears (called R-scripts window) which display all the available r-scripts to the user (available r-scripts are scripts that he has create and scripts that

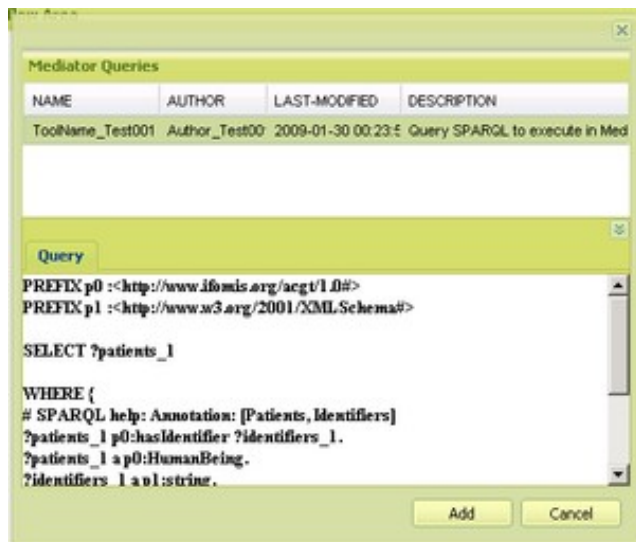
are public to the ACGT community). From the R-scripts window the user can select one r-script and view the R code, view the inputs/outputs of the r-script (R-scripts in ACGT are considered to be services that have inputs and outputs and run over the Grid-R web service) and the description of the script. The R-scripts are also visible as services in the service tree (left panel services area).



User can also add an r-script from the services area same way as a service (by drag and drop or double click). In the services tree area the r-scripts have the R icon to distinguish from the ACGT services.

29.5 Adding a mediator Query

Mediator queries can be inserted workflow area using the ACGT services tree (drag and drop or double click operation) or by selecting the  icon from the menu buttons. When the user selects the mediator icons from the menu buttons a window which displays all the Mediator Queries that are available and can be added to the workflow appears. The available queries (available mediator queries are queries that the user has created and all the public queries from other users to the ACGT community) are shown in a special window like R-scripts. The user can view the SPARQL code of a query and add it in the workflow as a service.



User can also add a mediator query from the services area same way as a service (by drag and drop or double click). In the services tree area the mediator queries have the Mediator icon to distinguish from the ACGT services.

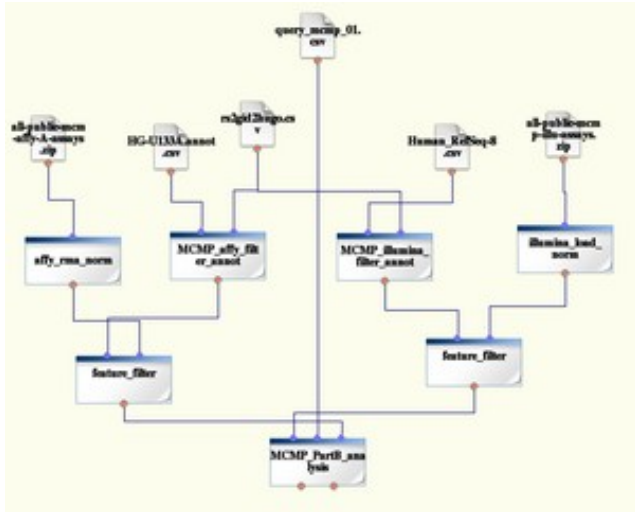
30 Save/Load and publish workflows

While you design your workflow, you can at any time save it to the ACGT servers. At a later time, you can load the workflow and continue the design process. To load a workflow, select **File** and then **Load** from the main menu of the editor.

The saved workflows belong to the user that created them and are initially visible and available only to him/her. To share them with the community, the author of the workflow must publish it using the **Publish** button in the *Workflow* menu.

⚠ Caution: A published workflow is "frozen", no one can edit or redesign it anymore, not even the original author.

Note: The figure below shows the MCMP scenario workflow (based on files) that can be seen from the *janedoe* account, by opening the *MCMP_PartB_public_files_2* public workflow. In this example, the data analysis has been split in modules, some of which are reusable in the context of different analysis.



31 Executing a workflow

The workflow can be executed directly from the workflow editor, after it has been "deployed". Internally, deployment is the translation of the workflow topology as constructed in the editor into a set of instructions that can be executed by the so-called workflow enactor.) A workflow must be saved before it can be deployed. Deployment is done through the "Make runnable" item of the *Workflow* menu of the workflow editor. A dialog then appears indicating if the deployment was successful or not.

After a successful deployment the workflow can be executed by selecting the "Run" item of the *Workflow* menu. Once the "Run" action is triggered a form appears, letting the user fill in the input fields for the workflow, which execution is actually started by clicking on the "Run" button at the bottom. Figure below shows the execution window for a workflow containing only one parameter called DirName with data type String. The user has to type the input he/she wants for this parameter and then select "Run" to start the execution of the workflow.



31.1 Execution from the service repository

A predefined workflow can be executed directly from the service repository: Select the workflow from the service tree, then click on the "Execute" action in the "Available operations" section of the right panel. (If the selected service is not a workflow other options are available on the right panel; such services are usually not runnable in standalone mode.)

Once the "Execute" action is triggered a form appears, letting the user fill in the input fields for the workflow, which execution is actually started by clicking on the "Execute operation?" button at the bottom.

31.2 Workflow execution output

Each GridR component in a workflow creates its own working directory in the DMS, in which temporary files are created. The results of a GridR-based workflow execution can thus be found in the GridR directories associated to the scripts of interest.

31.3 Monitoring the execution of a workflow

The execution of the workflow is monitored in the "Status" panel at the bottom of the workflow editor. Additionally, active workflow elements ("services") can be seen blinking in the editor.

Monitoring starts automatically when a workflow is executed. The various services under execution are blinking in the workflow edition area. All workflows are running in "background" which means that the user can log out and close the browser without stopping their execution. After logging in again the user can check the list of the workflows currently running for him. The status of a specific running workflow can be watched by selecting the corresponding execution "process" from the list that pops-up after selecting "Open" in the "Enactments" menu.

32 Publishing a workflow

The user can also make available to the ACGT community a workflow that he/she has created. The workflow to be published must be saved and deployed (option "Make runnable" in *Workflow* menu). Then the user has to select the option "Publish" in *Workflow* menu. As in figure below, a window appears with fields for the workflow and the author (Name, Description, Author, and Authority) and with description for every input/output of the workflow. The user has to fill in the fields and then select "Publish" button in the bottom of the window. Then the workflow editor sends the data to the ACGT repository and a new record is created to the repository. Now the workflow is available to every user in the ACGT community and anyone can execute it. Please note that other users can only view and execute the workflow but cannot edit it.

Publish Workflow.

General Description.

Name:

Service Type Uri:

Short Description:

Long Description:

Author:

Authority:

Description(s) for I/O Parameters.

DirName:

33 HB:Analyzing data with R

This page contains all the necessary information to allow you to analyze data with R in the ACGT environment. A special tool called *GridR Session* allows you to create and manage sessions where you can run scripts written in the R in the [grid environment](#) in order to analyze data that is accessible within the ACGT environment.

For a detailed description of the interface of the GridR Session tool see [HB:GridR_Session](#).

If you have developed an R script and you wish to use it in the [ACGT workflow environment](#) see [PM:Integrating_R_scripts](#).

33.1 Connecting to ACGT GridR servers

33.1.1 Connect to an ACGT R server

- First, [log on to ACGT](#).
- Open the [GridR Session tab](#).
- Click on the **"Connect"** button in the [Top Menu](#) to connect to an ACGT R server and start a GridR session.

There is a default ACGT R server already selected. You can change the R server by clicking the **Setup** button in the *Top Menu* and changing the server connection details. The default R server is in the [list of ACGT network connections](#).

Connecting may take some time during which the "Connect" button will remain pressed and no other buttons or areas can be used! Once you are logged in, the other buttons in the Top Menu (e.g. **Logout**) will become available.

33.1.2 Logging out from a GridR Session

Clicking on **"Logout"** button in the [Top Menu](#) will terminate the connection to the ACGT R server.

Note that GridR sessions live **as long as they are not explicitly terminated**. If you do not click the **"Logout"** button from the GridR Session [Top Menu](#), your GridR Session will not be terminated. In this case, if you logout from the ACGT Portal, login again at a later time and access the [GridR Session tab](#), you will see the GridR Session is still active and you may continue right where you left.

33.2 Entering commands and scripts

33.2.1 R commands and R scripts

You can send (single-line) R commands to the GridR session:

- Type the R command in the form in the middle of the [HB:GridR_Session#Right_Area](#).
- Press **Enter** and you will see the command appear in the console output window above.
- The output of the command will appear also in the console output window. This can take a while in case of a long running command.

Note: You can enter an empty command / just press **Enter** to refresh the console output window

If you want to use an R script which has several lines:

- Click on the **"Script"** button in the middle of the [HB:GridR_Session#Right_Area](#).
- A new form will popup
- Copy your multi-line R script and paste it in this form.

33.2.2 Re-using entered commands

- Click the **"Display History"** button in the [Top Menu](#)
- You will see the list of R commands that have been entered in the current session.
- You can copy these R commands in a local text editor to create an R script.

33.2.3 Plotting

Plotting **cannot** be done interactively within the GridR Session applet. Instead, you can create a file that contains the plot in the GridR Session working directory. You can preview this plot and transfer it to your local computer or the [ACGT distributed file manager](#) for further use.

To create a plot as a PNG image file:

- Enter the respective plot R commands inside the R session.
- Click on **"Create Plot"** button in the top menu

- Refresh the **File Explorer** area and you will see a new file that contains the image of the plot.
- Selecting this file in the **File Explorer** area will show you a preview of the plot in the display area (bottom-right area)

To change the type of the file (PNG, PS, PDF) :

- Click on "**Setup**" button in the top menu
- Change the image filetype. Available options are PNG, PS and PDF.
- Click **Save** to enable these settings for future plots.

33.3 Work with multiple GridR Sessions

Multiple Grid R sessions come in handy if you want to start working in a different session without interrupting your current session, e.g. if you are running a long script in the current Grid R session which will take time to finish, and you would like to input other R commands without interrupting this.

You can use the **Session Explorer** in the left area to manage different R sessions. The **Session Explorer** area contains a list of R sessions. The active session is shown as "selected". This is the session you can access (enter R commands, see output) in the right area of the Grid R tool.

You can open, close and switch between sessions in the **Session Explorer** area. Note that each GridR session has its own working directory that can be used to store and retrieve files. The **File Explorer** area in the bottom of the left area shows each working directory and provides access to the files of each session.

33.3.1 Open a new session

- Right-click on the **Session Explorer** area
- Select "**New Session**" from the menu and a new session (with a unique generated name) will be created.

33.3.2 Switch between sessions

You can switch between the sessions listed in the **Session Explorer** area simply by selecting them (left-click). Note that the active Grid R session is always the one that is selected.

33.3.3 Rename sessions

- Right-click on a session listed in the **Session Explorer** area
- Select "**Rename Session**" from the menu
- A new form will pop up where you can enter a new name for the session

33.3.4 Delete sessions

- Right-click on a session listed in the **Session Explorer** area
- Select "**Delete Session**" from the menu and the session will be terminated and removed from the list.

33.4 Working with the File Explorer

Each GridR session has its own working directory that can be used to store and retrieve files. You can use the **File Explorer** to manage these files.

The **File Explorer** window in the left area contains a tree-view of the working directory of the R sessions. To refresh the tree-view right-click on the **File Explorer** area and select "**Refresh**" from the menu.

You can download files from the GridR session working directory directly to your local computer or upload your file directly to the GridR session working directory from the **File Explorer** area.

33.4.1 Download files to your local computer

- Select a file in the **File Explorer** area
- Right-click on the file in the **File Explorer** area
- Select "**Download**" from the menu.
- A window will popup asking you where to place the file on your local computer.

33.4.2 Upload files from your local computer

- Select a GridR session in the **File Explorer** area
- Right-click on the session in the **File Explorer** area
- Select "**Upload**" from the menu.

- A window will popup where you can select the file on your local computer you want to copy in the GridR Session working directory.

33.4.3 Copy images

Image files that are shown in the preview can be copied directly to your clipboard so you can use them in your local image editors:

- Right-click on the file in the **File Explorer** area
- Select "**Clipboard**" from the menu.
- Paste your image (*CTRL+V*) in your favourite local image editor.

33.5 GridR special commands

Each GridR session has its own working directory that can be used to store and retrieve files. To interface with other ACGT components (such as the [distributed file manager](#)) you must use the GridR R package, which is automatically loaded at session startup. The same forms which allow you to enter single-line and multi-line R commands can be used for GridR commands.

The following basic GridR commands will allow you to move files from the temporary GridR session working directories to your permanent storage area in the [ACGT distributed file manager\(DMS\)](#).

33.5.1 Upload files to the DMS

You can upload files from the GridR session working directory to the [DMS](#) using the following code:

```
grid.init()
id=grid.AcgtUploadToDms("path/toFile", "urn:eu-acgt.org:datatype:csv")
print(id)
```

Note: The second argument to the function *gridr.AcgtUploadToDms* specifies the type that has to be associated to the file (e.g.CSV) as meta-data. This parameter is optional and can be left out if the file type is not important.

33.5.2 Download files from DMS

You can download files from the [DMS](#) to the GridR session working directory using the following code:

```
grid.init()
content= grid.AcgtReadFromDms(id, "/path/to/local/file")
```

Some or all of the information on this page is inconsistent, irrelevant or confusing.

Please help clean it up if you are able.

The command *gridr.AcgtReadFromDms* copies the file which has the ID "*id*" from the DMS to the local directory, the file is simultaneously read, internally using the command *read.csv()*, and the contents is returned by the function (TODO, update this).

34 HB:ACGT Master Ontology

ACGT includes an [ontology](#) - a formal representation of concept and relations in the domain of cancer - to structure and describe clinical data.

The **ACGT Master Ontology** captures a large part of the important concepts of interest to medical practitioners and biology researchers (such as diseases, genes patent records and such like) as well as their properties and relationships allowing the various ACGT services to process and make use of the semantics of these concepts.

The ACGT Master Ontology is used in the background by ACGT services so for end-users it may remain *invisible*. In some situations, however, it may be necessary to have a closer look at the ontology or suggest updates to the ontology.

More background information about the ACGT ontology can be found [here](#)

34.1 What is the ACGT Master Ontology?

The **ACGT Master Ontology (MO)** is an [ontology](#) that represents the domain of clinical trials on cancer (and in general, cancer research). This ontology collects all the information related to cancer that needs to be handled within the ACGT environment, offering the conceptual basis for a structured knowledge repository. The intention of the *ACGT Master Ontology* is to represent the domain of cancer research and management in a computationally tractable manner.

The *ACGT MO* is built using the [Protégé-OWL](#) free open-source ontology editor, Version 4 . It is written in [OWL-DL](#) and presented as an *.owl* file. The ACGT MO is re-using [Basic Formal Ontology \(BFO\)](#) as *upper level* and the [OBO Relation Ontology](#).

34.2 Viewing the ontology

Main article [ACGT Ontology Viewer](#)

The **ACGT Ontology Viewer (OV)** is a user interface to the **Master ACGT Ontology**. The **Ontology viewer** can be opened by clicking on the *Ontology Viewer* subtab of the [Ontology tab](#) of the ACGT portal. It can be used by [registered ACGT users](#) as well as non-registered visitors which access the site directly (although in this case some of the functionalities it offers are not available for visitors).

The *Ontology Viewer* allows the visualization of the tree-like structure of the **ACGT Master Ontology**. User-specific views on the ontology can be created. Details on the selected node in the ontology are displayed on the right panel. A search tool allows the user to locate a ontology term in the ontology tree without knowing the hierarchical semantic path leading to it.

A step-by-step tutorial is available [on the ACGT Portal](#)

34.3 Updating the ontology

Main article [ACGT Ontology Submission Tool](#)

As an ACGT user, you **cannot** update the ACGT ontology yourself. However, if you are missing something in the ontology, you can contact the maintainers of the ontology and submit suggestions for updates. Advanced users with *Ontology Curation Rights* can use the **Ontology Submission Tool** to update the ontology.

34.4 See also

- [ACGT Ontology website](#)
- [ACGT Master Ontology Wiki \(external\)](#)
- [ACGT Ontology Viewer \(external\)](#)

35 PM: Integrating R scripts

36 Introduction

Scripts for the statistical software [R](#) can be integrated into the ACGT system as part of a workflow. This allows to easily combine the functionality of R with the advantages of the ACGT environment. In order to do so, each R script has to be registered as a service in ACGT.

This manual will guide you through the integration process. You will need to have your R script ready. If you are interested in developing an R script instead, you can get info in the manual about [analyzing data with R](#).

37 Login to the ACGT Portal

Login to the [ACGT portal](#) with your username and password.



Note: For details on the registration and login process, please see [Getting Started](#)

38 Open the Metadata Registration Page

You can browse existing services as well as register new services at the **Metadata Registration** page in the ACGT portal:



Metadata Registration/Tools

1. Click on **Metadata Registration** in the top menu
2. Click on **Tools** in the sub-menu

You can see the page for browsing and registering services now. You can use the tree on the left hand side to browse the services which are classified in various functional categories. In addition, you can search for services by keywords using the search tool in the upper-right side of the page.

39 Choose Functional Category



Add Functional Category

Select a functional category under which you want the new R script to appear by clicking on a category of the tree.

The name and the unique identifier (ID) of the functional category will be displayed in the right part of the page.

You can create new functional categories by selecting the **Add functional category** action in the lower right part of the page.

40 Open Registration Mask

Please click on the "Add RScript" action for opening a new page on which you can register an R-based service. The page that opens up is split into different parts for entering description data, information on the input and output parameters and the script code.

40.1 Enter Description

Please fill the service-description fields. The only mandatory field to register the script is the Name which is later used to identify the service in other ACGT components. However, it is recommended to fill the other fields as well as they are the source of information when searching for services.

40.2 Enter Parameters

The interface of the script is defined by describing the various inputs and outputs (parameters) of the R script. At the beginning, the list of parameters is empty. Parameters are distinguished as input- and output parameters. Each of these parameters refers to an input or an output file.

You can add new parameters by clicking on the "Add Parameter" button. Please fill the following information for each parameter:

- Parameter name: the name of the parameter
- Parameter type: the type of the parameter. You can choose between **simple** for normal parameters (e.g. the ID of a file) and **array** for array parameters (e.g. a list of file IDs)
- Parameter mime type: the data type format of the files that will be accepted referring to the mime standard
- Parameter is input: specifies whether the parameter is an input or output.
- Parameter description: a description of the parameter to provide more info for the users

In case a new data type is needed for the script, please refer to [Adding Data Types](#)

Advanced user note: Internally all exchanges between the R script and the other ACGT components are made through files, with each parameter being associated to one file. The contents of the files are handled through the `gridr.input` and `gridr.output` special R objects (lists) described in the next section.

Example (matching to the code example presented in one of the next paragraphs):

input parameter: input.csv | Simple | text/csv | CSV file to be converted

output parameter: appended.csv | Simple | text/csv | The output CSV file

40.3 Enter Code

Please enter the code of the R script itself at the "Script data" section on the registration page. Notice that you have to stick to the following convention of specifying the inputs and outputs of the script:

The parameters are passed to/from the R script from other services through two R lists which are created when the ACGT environment executes the script. The two lists are named **gridr.input** and **gridr.output**. The entries in these lists refer to the numbers of the parameters, e.g.

```
gridr.input[[1]]
```

refers to the input parameter 1.

For some mime types there is a special input handling.

- application/zip input files are automatically extracted. The `gridr.input` list contains a String with the filename of the input file.
- for text/csv input files, each element of `gridr.input` is the result of the R command `read.csv()` executed on the corresponding file, in the order in which they have been registered (see above), i.e., with a loose notation,

```
gridr.input[[i]]=read.csv(parameter[i])
```

In other cases, the `gridr.input` field contains a String with the name of the input file.

For outputs, the list `gridr.output` contains R Strings with the name of the parameter as specified during registration. Thus, for instance, writing data to a file can be done in the script through the R command:

```
write.table(my.data, file=gridr.output[[1]])
```

Files created with this method are written in a DMS directory by the workflow execution engine (see [Workflow Environment](#)).

Example:

```
table=gridr.input[[1]]
table[length(table)+1]=0
write.csv(table,file=gridr.output[[1]])
```

In case of array parameters, the gridr.input and gridr.output lists hold arrays containing the data or the filenames.

40.4 Finalize the Registration

Please click on the "Add RScript" button for storing the script in the ACGT repository. The registration is completed now.

41 Checking the registered Script

You can check the registered data for the scripts as follows: Browse the tree on the left hand side or use the search tool in the upper-right side. Click on a script at the tree or at the result list of the search in order to select the script. The registration mask with the registered data will open up.

42 Editing the registered Scripts

You can edit the script just by changing the content of the description fields, the parameters or the script code area and clicking the "Save RScript" button.

If you want to delete a script, click on the "Delete RScript" button.

43 Using and executing a registered Script

There are different ways of using and executing an R script. One way is to click on the "Execute" button at the registration page of a selected script. A new mask will open which contains fields for providing the input parameters. In detail, these input parameters are the IDs of the files stored at the DMS which are taken as inputs. By clicking on the "Execute" button, the execution of the script is launched.

Another way of executing the script is to use it inside a workflow (see [Workflow Environment](#))

44 PM:Developing a new service

45 Introduction

This page contains all the necessary information to setup a secure ACGT web service from scratch based on the Globus Toolkit (version 4.1.2).

For any issue concerning the guide please contact the ACGT technical support using this email address: acgt-rt@ics.forth.gr

46 Preparing Hosting Environment (Service Container)

The hosting environment for the axis based web service is *tomcat* as a servlet container and *axis* as a SOAP engine. The simplest way to create the hosting environment for gsi-enabled web services is to deploy the java ws-core (which is part of Globus Toolkit 4) into tomcat and then to remove or not unnecessary globus core WSRF services leaving only part of globus responsible for ws-security.

46.1 Creating a low privilege account

The first [optional] step is to create a low privilege user for the service. On most linux/unix systems the following commands should work.

```
groupadd services
useradd -g services -d /home/hello -c "GSI-enabled Hello World Web Service" hello
```

46.2 Setup CA certificates

This step defines set of Certificate Authorities which the service will trust.

1. Download the ACGT CA certificates: [Acgt-ca.tar](#)
2. Extract the certificates and place them in the correct directory
 - ◆ locally: **\$HOME/.globus** - to be "visible" only for the created user - recommended when there is no Globus installation
 - ◆ globally: **/etc/grid-security** - used by whole system - recommended with Globus installation

46.3 Requesting service certificate

Users who have access to full version of Globus Toolkit can use **grid-cert-request** command to generate certificate request otherwise the request can be created using CoG library.

46.3.1 Requesting service certificate using Globus tools

Use the **grid-cert-request** command to create a certificate request for the service. The **grid-cert-request** command is part of Globus toolkit.

- if you deployed CA certificates globally (in /etc/grid-security)

```
grid-cert-request -service hello -host druid-bis.man.poznan.pl -ca e622f687 -dir .
```

- if you deployed CA certificates locally (in you \$HOME/.globus location)

```
export X509_CERT_DIR=/home/piontek/.globus/certificates/
grid-cert-request -service hello -host druid-bis.man.poznan.pl -ca e622f687 -dir .
```

46.3.2 Requesting service certificate using CoG Kit

1. Deploy CA certificates locally
2. Make sure that there is SUN Java JDK or JRE installed in your system and that **java** binary is on your default **PATH**
3. Download and install COG Kit for java - COG Kit is available at: <http://wiki.cogkit.org>
 1. download a binary distribution of **Java CoG Kit 4.1.x**
 2. unzip or untar the bundle
4. for generating certificate request it is not required to have full COG Kit configuration but some other tool may require it. For the details concernin COG installation and configuration please look at: http://wiki.cogkit.org/wiki/Java_CoG_Kit_Installation_Guide
5. use **bin/grid-cert-request** from COG Kit directory to generate service's private key and certificate request:

```
cog-4_1_x/bin/grid-cert-request -service <your service name> -host <domain name of machine> -dir .
```

Here is a short example of the steps described above:

```
mkdir -p ~/.globus/
cd ~/.globus
wget https://acgt.custodix.com/Acgt-ca.tar
tar xfv acgt-ca-2008.tar
cd ~
wget http://www.cogkit.org/release/4_1_5/cog-4_1_5-bin.tar.gz
tar xzf cog-4_1_5-bin.tar.gz
cog-4_1_5/bin/grid-cert-request -service hello -host druid-bis.man.poznan.pl -dir .
```

46.3.3 Signing certificate

Send the certificate request (*hellocert_request.pem* file) to the **ACGT CA** and wait for the signed certificate.

IMPORTANT: Detailed information concerning security can be found in separate [ACGT Testbed Security Guide](#).

46.4 Downloading and installing Globus Toolkit 4.1.2

Some initial steps can be skipped by downloading [the tomcat with the wscore already deployed into it](#). Then please follow the guide starting from the "Configuring Tomcat" chapter.

This section describes how to build the wscore-4.1.2 (part of globus 4.1.2). Alternatively [the compiled version](#) can be downloaded.

IMPORTANT: For unknown reasons, Axis, which is part of hosting environment, doesn't work properly (is not able to find configuration file) if the globus was compiled with different version of java than 1.4. Please configure your environment to use java 1.4 before you start compiling globus. After deploying the compiled globus into tomcat the hosting environment can be ran using java 5 or 6.

You can skip this point if the GT 4.1.2 is already installed.

```
wget http://www-unix.globus.org/ftppub/gt4/4.1.2/installers/src/gt4.1.2-all-source-installer.tar.gz
tar xzf gt4.1.2-all-source-installer.tar.gz
mkdir gt-4.1.2
cd gt4.1.2-all-source-installer
```

Use `--prefix` option to choose destination directory.

```
./configure --prefix=/home/piontek/tmp/ACGT/globus/gt-4.1.2
```

Globus Toolkit 4.1.2 can be also [downloaded](#) from alternative site.

Install Globus limiting installation only to **wsjava** package (aka. ws-core).

```
make wsjava install
```

If the installation fails with following of similar error:

```
piontek@druoid-bis ~/tmp/ACGT/globus/gt4.1.2-all-source-installer $ make wsjava install
cd gpt && OBJECT_MODE=32 ./build_gpt
build_gpt ====> installing GPT into /home/piontek/tmp/ACGT/globus/gt-4.1.2
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/support/Compress-Zlib-1.21
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/support/IO-Zlib-1.01
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/support/makepatch-2.00a
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/support/Archive-Tar-0.22
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/support/PodParser-1.18
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/support/Digest-MD5-2.20
build_gpt ====> building /home/piontek/tmp/ACGT/globus/gt4.1.2-all-source-installer/gpt/packaging_tools
/home/piontek/tmp/ACGT/globus/gt-4.1.2/sbin/gpt-build -srcdir=source-trees/wsrfl/java/common/source gcc32dbg
sh: NOT: command not found
ERROR: Untar failed
make: *** [globus_java_ws_core_common] Error 255
```

Please check if locations of all needed tools in the `/${GLOBUS_PATH}/var/lib/perl/Grid/GPT/LocalEnv.pm` file are set properly.

46.5 Downloading and installing Apache Tomcat

Download and install [Apache Tomcat servlet container \(version 5.5.x\)](#)

IMPORTANT: The guide based on version 5.5.25 but newer ones from branch 5.5 can be also used.

```
wget http://archive.apache.org/dist/tomcat/tomcat-5/v5.5.25/bin/apache-tomcat-5.5.25.tar.gz
tar xzf apache-tomcat-5.5.25.tar.gz
```

Please note, that Tomcat 5.5.x requires java JRE 5.0. If you want to use java 1.4 please install additionally the [JDK 1.4 Compatibility Package](#)

The [Tomcat 5.5.25](#) and [JDK 1.4 Compatibility Package](#) can be also downloaded from alternative site.

46.6 Deploying wsjava into Tomcat

The aim of this chapter is to deploy wscore-4.1.2 into tomcat. Alternatively [the tomcat with already deployed wscore](#) can be downloaded.

Go to the directory where the wsjava package was installed and do following set of actions:

```
$ export GLOBUS_LOCATION=`pwd`
$ ant -f share/globus_wsrfl_common/tomcat/tomcat.xml deploySecureTomcat -Dtomcat.dir=<tomcat.dir>
```

Where `<tomcat.dir>` is an absolute path to the Tomcat installation directory.

Also, `-Dwebapp.name=<name>` property can be specified to set the name of the web application under which the installation will be deployed. By default "wsrf" web application name is used. In our example the `name` was set to `acgt` value.

```
$ export GLOBUS_LOCATION=`pwd`
$ ant -f share/globus_wsrf_common/tomcat/tomcat.xml deploySecureTomcat -Dtomcat.dir=<tomcat.dir> -Dwebapp.name=acgt
```

46.7 Configuring Tomcat

Configuring Security

In addition to the above deployment step it is also needed to configure the Tomcat to use appropriate connectors and valves responsible for GSI-security. Please modify the `<tomcat.dir>/conf/server.xml` configuration file.

- Add a HTTPS Connector in the `<Service name="Catalina">` section and update the parameters appropriately with your local configuration:

```
<Connector
  className="org.globus.tomcat.coyote.net.HTTPSConnector"
  port="8443" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  autoFlush="true"
  disableUploadTimeout="true" scheme="https"
  enableLookups="true" acceptCount="10" debug="0"
  protocolHandlerClassName="org.apache.coyote.http11.Http11Protocol"
  socketFactory="org.globus.tomcat.catalina.net.BaseHTTPSServerSocketFactory"
  proxy="/path/to/proxy/file"
  cert="/path/to/certificate/file"
  key="/path/to/private/key/file"
  cacertdir="/path/to/ca/certificates/directory"
  encryption="true"/>
```

In the above the `proxy`, `cert`, `key` and `cacertdir` attributes are optional. Furthermore, the `proxy` and the combination of `cert` and `key` attributes are mutually exclusive. The `encryption` attribute is also optional (defaults to true if not set). If set, the `cert` and `key` attributes should point to the service certificate requested in the previous step; `cacertdir` should point to the directory where the ACGT CA certificates are located (e.g. `$HOME/.globus/certificates`).

The `mode` attribute can also be set to specify the connection mode. There are two supported connection modes: `ssl` and `gsi`. The `ssl` mode indicates a regular SSL connection mode. The `gsi` mode indicates a SSL connection mode with transport-level delegation support. The `ssl` mode is the default mode if the `mode` attribute is not specified. Please note that the `gsi` mode is intended for advanced users only.

Important The credentials and certificate configuration is used only by the connector and is not used by the rest of the web services stack in Globus Toolkit. To configure credentials inside container please modify the [Security Descriptor](#) file.

Go to the `<tomcat.dir>/webapps/<name>/WEB-INF/etc/globus_wsrf_core` directory and modify content of `global_security_descriptor.xml` file setting paths to credential and key in `<certificate>` section, where `<name>` is value of `-Dwebapp.name` property used during the deployment of wsjava into tomcat. If the property wasn't specified the default value is "wsrf". For this guide the `<name>` value was set to `acgt`. Point to the requested service certificate and key.

```
<?xml version="1.0" encoding="UTF-8"?>
<containerSecurityConfig xmlns="http://www.globus.org/security/descriptor/container"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.globus.org/security/descriptor name_value_type.xsd"
  xmlns:param="http://www.globus.org/security/descriptor" >
  <credential>
    <cert-key-files>
      <key-file value="/PATH/servicekey.pem"/>
      <cert-file value="/PATH/servicecert.pem"/>
    </cert-key-files>
  </credential>
</containerSecurityConfig>
```

- Add a HTTPS Valve in the `<Engine name="Catalina" ... >` section of the file `<tomcat.dir>/conf/server.xml`:

```
<Valve className="org.globus.tomcat.coyote.valves.HTTPSValve55"/>
```

- Remove or comment out the standard HTTP connector to prevent unencrypted connections.

```
<Connector port="8080" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" acceptCount="100"
  connectionTimeout="20000" disableUploadTimeout="true" />
```

- Please change the default shutdown command from "SHUTDOWN" to something else what will be known only to tomcat administrator. Otherwise everyone will be able to shutdown the tomcat sending the command to the shutdown-port.

```
<Server port="8005" shutdown="SHUTDOWN">
```

- You may have to edit `<tomcat.dir>/webapps/<name>/WEB-INF/web.xml` if you are running Tomcat on a non-default port, that is if not using port 8443 (HTTPS). For example, if you run Tomcat on port 443 using HTTPS then the WSRF servlet entry should be modified to have the following `defaultProtocol` and `defaultPort` parameters:

```
<web-app>
```



```

...
<servlet>
  <servlet-name>WSRFServlet</servlet-name>
  <display-name>WSRF Container Servlet</display-name>
  <servlet-class>
    org.globus.wsrif.container.AxisServlet
  </servlet-class>
  <init-param>
    <param-name>defaultProtocol</param-name>
    <param-value>https</param-value>
  </init-param>
  <init-param>
    <param-name>defaultPort</param-name>
    <param-value>443</param-value>
  </init-param>
  <load-on-startup>true</load-on-startup>
</servlet>
...
</web-app>

```

Alternatively, you can use the `setDefault` Ant task to set the default protocol/port in the `web.xml` file:

```

$ cd $GLOBUS_LOCATION
$ ant -f share/globus_wsrif_common/tomcat/tomcat.xml setDefaults \
  -Dtomcat.dir=<tomcat.dir> \
  -DdefaultPort=<port> -DdefaultProtocol=<protocol>

```

Configuring logging

Copy `$GLOBUS_LOCATION/lib/common/log4j-*.jar` and `$GLOBUS_LOCATION/lib/common/commons-logging-*.jar` files to `<tomcat.dir>/webapps/<name>/WEB-INF/lib/` directory. Then configure the Log4j configuration file in `<tomcat.dir>/webapps/<name>/WEB-INF/classes/` directory appropriately. The debugging settings will only affect the web application code.

Please always check the Tomcat log files under the `<tomcat.dir>/logs` directory for any errors or exceptions.

46.7.1 Alternative Transport Layer Security (HTTPG)

For HttpG you need to add another Connector at the `server.xml` configuration file:

```

<Connector
  className="org.globus.tomcat.coyote.net.HTTPSConnector"
  port="8444" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  autoFlush="true"
  disableUploadTimeout="true" scheme="https"
  enableLookups="true" acceptCount="10" debug="0"
  protocolHandlerClassName="org.apache.coyote.http11.Http11Protocol"
  socketFactory="org.globus.tomcat.catalina.net.BaseHTTPSServerSocketFactory"
  cert="/path/to/certificate/file"
  key="/path/to/private/key/file"
  cacertdir="/path/to/ca/certificates/directory"
  encryption="true"
  mode="gsi"/>

```

Please note that it uses a different port (8444) and it has `gsi` specified as mode.

Also in `<tomcat.dir>/webapps/<name>/WEB-INF/etc/globus_wsrif_core/server-config.wsdd` add `<handler type="java.org.globus.axis.handler.CredentialHandler"/>` inside the `<requestFlow>` element like this:

```

<requestFlow>
  <handler type="java.org.globus.wsrif.handlers.AddressingHandler"/>
  <handler type="java.org.globus.wsrif.handlers.URLMapper"/>
  <handler type="java.org.globus.wsrif.impl.security.authentication.transport.TomcatTransportSecurityHandler"/>
  <handler type="AuthenticationServiceHandler"/>
  <handler type="java.org.globus.axis.handler.CredentialHandler"/>
  <handler type="java.org.globus.wsrif.handlers.MessageLoggingHandler"/>
  <handler type="java.org.globus.wsrif.handlers.JAXRPCHandler">
    <parameter name="className" value="org.globus.wsrif.impl.security.authentication.wssec.WSSecurityHandler"/>
  </handler>
  <handler type="java.org.globus.wsrif.impl.security.authentication.SecurityPolicyHandler"/>
  <handler type="java.org.globus.wsrif.impl.security.authorization.AuthorizationHandler"/>
  <handler type="java.org.globus.wsrif.handlers.FaultHandler"/>
</requestFlow>

```

46.8 Starting Tomcat

It is recommended to increase the maximum heap size of the JVM when running the container. By default on Sun JVMs a 64MB maximum heap size is used. The maximum heap size can be set using the `-Xmx` JVM option. Please add the property `-Xmx512M` to `JAVA_OPTS` in the `<tomcat>/bin/catalina.sh` file.

IMPORTANT: By default Sun 1.4.x+ JVMs are configured to use `/dev/random` device as an entropy source. Sometimes the machine can run out of entropy and applications using the `/dev/random` device will block until more entropy is available. One workaround for this issue is to configure the JVM to use `/dev/urandom` (non-blocking) device instead. For Sun JVMs a `java.security.egd` system property can be set to configure a different entropy source. Please add the `-Djava.security.egd=file:/dev/urandom` property to `JAVA_OPTS` in the `<tomcat>/bin/catalina.sh` file.

Start the tomcat using `<tomcat-root>/bin/startup.sh` script and check if it works listing available services. In any web browser try to open following page <https://localhost:8443/<name>/services>, where `<name>` is value of `-Dwebapp.name` property used during the deployment of `wsjava` into tomcat. If the property wasn't specified the default value is `wsrf`. If the tomcat with already deployed `ws-core` was used the name is "acgt".

The list of hosted services should be displayed:

```
And now... Some Services
  NotificationTestService (wsdl)
    generateNotification
    selfSubscribe
  TestAuthzService (wsdl)
    addDeclinedMethod
    SAMLRequest
  CounterService (wsdl)
    add
    createCounter
  TestServiceWrongWSDL (wsdl)
    createResource
    resetNumInstances
    getInstanceInfo
    testLocalInvocation
  ShutdownService (wsdl)
    shutdown
  ...
```

There is some set of `wsrf-core` services, that are unnecessary to start simple `gsi-enabled` web service and can be removed. From the `<tomcat-root>/webapps/<name>/WEB-INF/etc` directory remove all files and directories except the `globus_wsrf_core` directory. If you want you can also remove all services except the `gsi/AuthenticationService` one from the `<tomcat-root>/webapps/<name>/WEB-INF/etc/globus_wsrf_core/server-config.wsdd` file.

The list of services should be:

```
And now... Some Services
  gsi/AuthenticationService (wsdl)
    requestSecurityTokenResponse
    requestSecurityToken
```

IMPORTANT: For unknown reason the hosting environment doesn't work properly if the `globus` deployed into it was compiled when another java then 1.4 was set in environment. If your list of services is limited to three entries:

```
And now... Some Services
  * AdminService (wsdl)
    o AdminService
  * Version (wsdl)
    o getVersion
  * asyncService (wsdl)
    o process
```

please recompile and redeploy `globus` into tomcat having set java 1.4.

47 "Hello World!" Service and Client

This chapter shows how to implement simple gsi-enabled web service and client. The service has only one method *hello* with no arguments and it returns depending on version information about the caller (the user Distinguished Name) or complex information about proxy delegated to the service.

IMPORTANT All classes and jars containing examples presented below were compiled using java 1.5.

```
java version "1.5.0_13"  
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_13-b05)  
Java HotSpot(TM) Client VM (build 1.5.0_13-b05, mixed mode)
```

47.1 Creating WSDL file describing the service interface

This point can be omitted if you want to build the client or service basing on the existing WSDL file.

Interface of every service must be described in *Web Service Definition Language*. The WSDL document can be written manually or generated using appropriate tools. One of such tools that can be used to generate WSLD file is *Java2WSDL* tool provided with Axis.

- Design the service interface as java class making public all methods you want to be available.

IMPORTANT: Do not use java interface to model the service. In such case names of parameters in generated WSDL file will be *arg1, arg2 ... argN* instead of real ones.

```
package acgt.examples.hello;  
  
public class HelloException extends Throwable  
{  
    public int errorCode;  
    public String errorMessage;  
}
```

HelloException.java

```
package acgt.examples.hello;  
public class Hello  
{  
    public String hello() throws HelloException  
    {  
        return null;  
    }  
}
```

Hello.java

- Compile the class modeling the service interface with option switching on additional debug information (-g option of javac tool). Otherwise the *java2WSDL* tool will not be able to obtain real names of parameters.

```
javac -g *.java
```

- Use the **java2WSDL** tool to generate the WSDL file

```
Usage: java org.apache.axis.wsdl.Java2WSDL [options] class-of-portType
```

Detailed information about the *java2WSDL* tool can be found on [Axis Reference Guide](#) page.

The most important options are:

```
-o, --output <argument>  
    output WSDL filename  
-l, --location <argument>  
    service location url  
-P, --portTypeName <argument>  
    portType name (obtained from class-of-portType if not specified)  
-b, --bindingName <argument>  
    binding name (--servicePortName value + "SOAPBinding" if not specified)  
-S, --serviceElementName <argument>  
    service element name (defaults to --servicePortName value + "Service")  
-s, --servicePortName <argument>  
    service port name (obtained from --location if not specified)  
-n, --namespace <argument>  
    target namespace  
-p, --PkgtoNS <argument>=<value>  
    package=namespace, name value pairs  
-N, --namespaceImpl <argument>  
    target namespace for implementation wsdl  
-A, --soapAction <argument>  
    value of the operations soapAction field. Values are DEFAULT,  
    OPERATION or NONE. OPERATION forces soapAction to the name  
    of the operation. DEFAULT causes the soapAction to be set  
    according to the operations meta data (usually ""). NONE forces
```

the soapAction to "". The default is DEFAULT.
 -y, --style <argument>
 The style of binding in the WSDL, either DOCUMENT, RPC, or WRAPPED.
 -u, --use <argument>
 The use of items in the binding, either LITERAL or ENCODED

Following set of options is recommend:

- --soapAction OPERATION
- --style WRAPPED
- --use LITERAL

The --location option is mandatory.

IMPORTANT: Globus was designed to suport and host DOCUMENT/LITERAL style services. Globus Authorization Handler expects that every service's method has one parameter with the name equals to the name of the method. If this requirement is not met (for example for WRAPPED/LITERAL services) authorization process fails with the error presented below. To solve this problem the modified AuthUtil.class has to be copied to <tomcat.dir>/webapps/<name>/WEB-INF/classes/org/globus/wsrf/impl/security/util directory.

```
[JWSSEC-23] Operation name could not be determined
```

The *Which style of WSDL should I use?* article describes differences between different styles of WSDL files.

```
java -cp ../jars/axis.jar:../jars/jaxrpc.jar:../jars/log4j-1.2.13.jar:../jars/commons-logging-1.1.jar:../jars/commons-discovery-0.2.jar:../jars/w
org.apache.axis.wsdl.Java2WSDL \
--style WRAPPED \
--use LITERAL \
--soapAction OPERATION \
--namespace http://www.eu-acgt.org/ \
--namespaceImpl http://www.eu-acgt.org/ \
--location HelloLocation \
--output hello.wsdl \
acgt.examples.hello.Hello
```

All the jars needed for Java2WSDL tool can be taken from GT4.1.2 or [downloaded](#) as a separated tarball.

As a result the Java2WSDL tool should generate the [hello.wsdl](#) file.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap"
  xmlns:impl="http://www.eu-acgt.org/"
  xmlns:intf="http://www.eu-acgt.org/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
  xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.eu-acgt.org/">
  <wsdl:types>
    <schema elementFormDefault="qualified" targetNamespace="http://www.eu-acgt.org/" xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="hello">
        <complexType/>
      </element>
      <element name="helloResponse">
        <complexType>
          <sequence>
            <element name="helloReturn" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>
      <complexType name="ServiceException">
        <sequence>
          <element name="errorCode" type="xsd:int"/>
          <element name="errorMessage" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <element name="fault" type="impl:HelloException"/>
    </schema>
  </wsdl:types>
  <wsdl:message name="helloRequest">
    <wsdl:part name="parameters" element="impl:hello"/>
  </wsdl:message>
  <wsdl:message name="ServiceException">
    <wsdl:part name="fault" element="impl:fault"/>
  </wsdl:message>
  <wsdl:message name="helloResponse">
    <wsdl:part name="parameters" element="impl:helloResponse"/>
  </wsdl:message>
  <wsdl:portType name="Hello">
    <wsdl:operation name="hello">
      <wsdl:input name="helloRequest" message="impl:helloRequest"/>
      <wsdl:output name="helloResponse" message="impl:helloResponse"/>
      <wsdl:fault name="HelloException" message="impl:HelloException"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="HelloSoapBinding" type="impl:Hello">
    <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  </wsdl:binding>
</wsdl:definitions>
```

```

        <wsdl:operation name="hello">
            <wsdlsoap:operation soapAction="hello"/>
            <wsdl:input>
                <wsdlsoap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <wsdlsoap:body use="literal"/>
            </wsdl:output>
            <wsdl:fault name="HelloException">
                <wsdlsoap:fault name="HelloException" use="literal"/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="HelloService">
        <wsdl:port name="Hello" binding="impl:HelloSoapBinding">
            <wsdlsoap:address location="HelloLocation"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>

```

Please modify the value of *soapAction* in binding section to be in url format, otherwise invoking the client you will get following error:

```

ERROR handler.AddressingHandler [main,invoke:120] Exception in AddressingHandler
org.apache.axis.types.URI$MalformedURIException: No scheme found in URI.
    at org.apache.axis.types.URI.initialize(URI.java:653)

```

In the example value of *soapAction* was changed to:

```

<wsdlsoap:operation soapAction="http://www.eu-acgt.org/hello"/>

```

47.2 Implementing gsi-enabled Web Service

47.2.1 Generating stub classes

Having the WSDL file describing the service's interface, use the *WSDL2Java* tool to generate auxiliary axis classes hiding the complexity of SOAP communication.

```

Usage: java org.apache.axis.wsdl.WSDL2Java [options] WSDL-URI

```

Detailed information about the *WSDL2Java* tool can be found on Axis [Reference Guide](#) page.

The most important options are:

```

-s, --server-side
    emit server-side bindings for web service
-S, --skeletonDeploy <argument>
    deploy skeleton (true) or implementation (false) in deploy.wsdd.
    Default is false. Assumes --server-side.
-N, --NStoPkg <argument>=<value>
    mapping of namespace to package
-f, --fileNStoPkg <argument>
    file of NStoPkg mappings (default NStoPkg.properties)
-p, --package <argument>
    override all namespace to package mappings, use this package name instead
-o, --output <argument>
    output directory for emitted files
-u, --allowInvalidURL
    emit file even if WSDL endpoint URL is not a valid URL
-B, --buildFile
    emit Ant Buildfile for web service

```

Use the *--server-side* option if you want the auxiliary code for service side to be created. Otherwise the tool will generate only client side code. It is also recommended to use the *--skeletonDeploy* option set to *true*, what makes the *deploy.wsdd* file, describing the service during the deployment simpler.

```

java -cp ../jars/axis.jar:../jars/log4j-1.2.13.jar:../jars/commons-logging-1.1.jar:../jars/commons-discovery-0.2.jar:../jars/jaxrpc.jar:../jars/s
org.apache.axis.wsdl.WSDL2Java \
--server-side \
--skeletonDeploy true \
--package acgt.examples.hello.stub \
--allowInvalidURL \
--buildFile \
--output wsdl2java \
hello.wsdl

```

All jars needed for *WSDL2Java* tool can be taken from GT4.1.2 or [downloaded](#) as a separated tarball.

Following set of files should be generated:

```

./wsdl2java
|-- acgt
| `-- examples

```

```

|-- hello
|-- stub
|-- HelloException.java
|-- HelloService.java
|-- HelloServiceLocator.java
|-- HelloSoapBindingImpl.java
|-- HelloSoapBindingSkeleton.java
|-- HelloSoapBindingStub.java
|-- Hello_PortType.java
|-- deploy.wsdd
|-- undeploy.wsdd
|-- build.xml

```

Generated files

47.2.2 Implementing the service functionality

Previous step (WSDL2java) should generate set of java classes including the *Hello_PortType* class containing definition of service's portType (interface) and the *HelloSoap_BindingImpl* class one implementing this portType/interface.

```

/**
 * Hello_PortType.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis 1.4 Mar 01, 2007 (10:42:15 CST) WSDL2Java emitter.
 */

package acgt.examples.hello.stub;

public interface Hello_PortType extends java.rmi.Remote {
    public java.lang.String hello() throws java.rmi.RemoteException, acgt.examples.hello.stub.HelloException;
}

/**
 * HelloSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis 1.4 Mar 01, 2007 (10:42:15 CST) WSDL2Java emitter.
 */

package acgt.examples.hello.stub;

public class HelloSoapBindingImpl implements acgt.examples.hello.stub.Hello_PortType{
    public java.lang.String hello() throws java.rmi.RemoteException, acgt.examples.hello.stub.HelloException {
        return null;
    }
}

```

Please, fill in interface methods in *HelloSoapBindingImpl.java* file with real service functionality.

47.2.2.1 Hello World!: Showing who's calling

The service returns information about identity (subject of certificate aka. Distinguished Name) of the user who invoked the *hello* method.

IMPORTANT: The *MessageContext.getCurrentContext()* method can be invoked only from the main thread created by Axis to serve the request. Invocation from another thread created by the user will return *NULL* as a result.

```

/**
 * HelloSoapBindingImpl.java
 */

package acgt.example.hello.stub;

import java.security.Principal;
import javax.security.auth.Subject;
import org.apache.axis.MessageContext;
import org.globus.wsrfl.impl.security.authentication.Constants;

public class HelloSoapBindingImpl implements acgt.examples.service.stub.Hello_PortType
{
    public java.lang.String hello() throws java.rmi.RemoteException, acgt.examples.hello.stub.HelloException
    {
        String userDN = null;
        Subject subject = (Subject)MessageContext.getCurrentContext().getProperty(Constants.PEER_SUBJECT);

        if( subject != null)
        {
            Principal principal = null;
            if( subject.getPrincipals().isEmpty() == false)
            {
                principal = (Principal)subject.getPrincipals().iterator().next();
            }
        }
    }
}

```

```

        if( principal != null)
        {
            userDN = principal.getName();
        }
    }
}
return userDN;
}
}
}

```

HelloSoapBindingImpl.java

Following set of jars is needed to compile the code:

- axis.jar
- jaxrpc.jar
- wsrf_core.jar

All these jars can be taken from Globus Toolkit 4.1.2 or [downloaded](#) as a separated tarball.

Add the listed above jars to the classpath definition in build.xml file created by wsdl2java tool.

Compile all classes generated by WSDL2Java tool including the modified *HelloSoapBindingImpl.java* one and generate jar archive.

To generate the *hello.jar* yourself [download the tarball](#) (containing ant build file, all needed jars and code) file and simply invoke *ant* tool:

```
ant
```

The [Apache Ant](#) is required!

47.2.2.2 Hello World!: Showing information about delegated proxy

The service returns full information about proxy certificate that was delagated to the service (Distinguished Name, proxy type, proxy lifetime, etc.).

IMPORTANT: The *MessageContext.getCurrentContext()* method can be invoked only from the main thrad created by Axis to serve the request. Invocation from another thread created by the user will return *NULL* as a result.

```

/**
 * HelloSoapBindingImpl.java
 */

package acgt.examples.hello.stub;

import java.security.Principal;
import javax.security.auth.Subject;
import javax.security.auth.Subject;
import org.apache.axis.MessageContext;
import org.ietf.jgss.GSSCredential;
import org.globus.wsrfl.impl.security.authentication.Constants;
import org.globus.gsi.GlobusCredential;
import org.globus.gsi.gssapi.GlobusGSSCredentialImpl;

public class HelloSoapBindingImpl implements acgt.examples.hello.stub.Hello_PortType
{
    public java.lang.String hello() throws java.rmi.RemoteException, acgt.examples.hello.stub.HelloException
    {
        GSSCredential credential = null;
        Subject subject = (Subject)MessageContext.getCurrentContext().getProperty(Constants.PEER_SUBJECT);
        if( subject != null)
        {
            try
            {
                if( subject.getPrivateCredentials().isEmpty() == false)
                {
                    credential = (GSSCredential)subject.getPrivateCredentials().iterator().next();
                    if (credential != null && credential instanceof GlobusGSSCredentialImpl)
                    {
                        GlobusCredential globusCred = ((GlobusGSSCredentialImpl)credential).getGlobusCredential();

                        return globusCred.toString();
                    }
                    else
                        return "No Globus proxy";
                }
            }
            catch( Exception e)
            {
                return "Failed to get user's credential: " + e;
            }
        }
        return "ERROR";
    }
}

```

HelloSoapBindingImpl.java

Following set of jars is needed to compile the code:

- axis.jar
- jaxrpc.jar
- wsrf_core.jar
- cog-jglobus.jar

All these jars can be taken from Globus Toolkit 4.1.2 or [downloaded](#) as a separated tarball.

Add the listed above jars to the classpath definition in build.xml file created by wsdl2java tool.

Compile all classes generated by WSDL2Java tool including the modified *HelloSoapBindingImpl.java* one and generate jar archive.

To generate the *hello.jar* yourself [download the tarball](#) (containing ant build file, all needed jars and code) file and simply invoke *ant* tool:

```
ant
```

The [Apache Ant](#) is required!

47.2.3 Deploying Service

1. copy jar with code implementing the service's functionality to the `<tomcat>/webapps/<name>/WEB-INF/lib` directory
2. create a new directory in `<tomcat>/webapps/<name>/WEB-INF/etc` one.
3. copy **deploy.wsdd** file generated by *WSDL2Java* tool to the created directory under the **server-config.wsdd** name.
 - ◆ Hello World!: Showing who's calling - [server-config.wsdd](#)
 - ◆ Hello World!: Showing information about delegated proxy - [server-config.wsdd](#)

IMPORTANT: Before you copy any jar, which is needed for your service, to the `<tomcat>/webapps/<name>/WEB-INF/lib` directory please ensure that it is not already present either in `<tomcat>/common/lib` or in `<tomcat>/server/lib`.

Please do not copy *cog-jglobus.jar* to `<tomcat>/webapps/<name>/WEB-INF/lib` it is already in `<tomcat>/common/lib`.

After tomcat restart (`<tomcat>/bin/shutdown.sh` and `<tomcat>/bin/startup.sh`) the deployed service should be listed on list of deployed service. Using any Web browser check the list of services enter the following location <http://localhost:8080/<name>/services>, where name is *name* chosen during the deploying globus to the tomcat. If the name wasn't specified the default one is "wsrf".

And now... Some Services

```
* Hello (wsdl)
  o hello
* gsi/AuthenticationService (wsdl)
  o requestSecurityTokenResponse
  o requestSecurityToken
```

47.3 Implementing gsi-enabled Web Service Client

The aim of this part of guide is to show step by step how to write and configure simple gsi-enabled command line java client, able to contact service described in previous chapter without and with delegation of user proxy certificate.

47.3.1 Generating stub classes

Having the WSDL file describing the service's interface, use the *WSDL2java* tool to generate auxiliary axis classes hiding the complexity of SOAP communication. You can skip this point and use stub classes generated for the service or generate them once again limiting the process only to classes needed on client side.

```
Usage: java org.apache.axis.wsdl.WSDL2Java [options] WSDL-URI
```

Detailed information about the WSDL2java tool can be found on [Axis Reference Guide](#) page.

The most important options are:

```
-N, --NStoPkg <argument>=<value>
    mapping of namespace to package
-f, --fileNStoPkg <argument>
    file of NStoPkg mappings (default NStoPkg.properties)
-p, --package <argument>
    override all namespace to package mappings, use this package name instead
-o, --output <argument>
    output directory for emitted files
-u, --allowInvalidURL
    emit file even if WSDL endpoint URL is not a valid URL
```



```
-B, --buildFile
    emit Ant Buildfile for web service
```

Do not use the `--server-side` option if you want only the client side code to be generated.

```
java -cp ../jars/axis.jar:../jars/log4j-1.2.13.jar:../jars/commons-logging-1.1.jar:../jars/commons-discovery-0.2.jar:../jars/jaxrpc.jar:../jars/
org.apache.axis.wsdl.WSDL2Java \
--package acgt.examples.hello.stub \
--allowInvalidURL \
--buildFile \
--output wsdl2java \
hello.wsdl
```

All jars needed for *WSDL2Java* tool can be taken from GT4.1.2 or [downloaded](#) as a separated tarball.

Following set of files should be generated:

```
./wsdl2java
|-- acgt
|  |-- examples
|     |-- hello
|        |-- stub
|           |-- HelloException.java
|           |-- HelloService.java
|           |-- HelloServiceLocator.java
|           |-- HelloSoapBindingStub.java
|           |-- Hello_PortType.java
|-- build.xml
```

Generated files

47.3.2 Client code

The client code is relatively simple, but its more important parts were marked by numbers in comments and explained below.

```
package acgt.examples.hello;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
import acgt.examples.hello.stub.Hello_PortType;
import acgt.examples.hello.stub.HelloServiceLocator;
import acgt.examples.hello.stub.HelloException;
import org.apache.axis.client.Stub;
import org.globus.axis.gsi.GSIConstants;
import org.globus.axis.util.Util;
import org.globus.wsrfl.impl.security.authentication.Constants;
import org.globus.wsrfl.impl.security.authorization.IdentityAuthorization;
import org.gridforum.jgss.ExtendedGSSManager;
import org.ietf.jgss.GSSCredential;
import org.ietf.jgss.GSSException;

public class HelloClient
{
    public static void main( String[] args)
    {
        String url = args[0];
        URL hello_url = null;
        try
        {
            hello_url = new URL( url);
        }
        catch( MalformedURLException e1)
        {
            System.err.println( "Malformed url: " + url);
            System.exit( 1);
        }
        String hello_dn = args[1];
        ExtendedGSSManager manager = (ExtendedGSSManager)ExtendedGSSManager.getInstance();
        GSSCredential credential = null;
        try
        {
            /* 1 */
            credential = manager.createCredential( GSSCredential.INITIATE_AND_ACCEPT);
        }
        catch( GSSException e)
        {
            System.err.println( "Failed to load user's proxy");
            System.exit( 1);
        }
        HelloServiceLocator locator = new HelloServiceLocator();
        Hello_PortType hello = null;
        try
        {

```

```

/* 2 */
        hello = locator.getHello( hello_url);
    }
    catch( ServiceException e)
    {
        System.err.println( "Failed to get port type");
        System.exit( 1);
    }
/* 3 */
((Stub)hello)._setProperty( GSIConstants.GSI_CREDENTIALS, credential);
/* 4 */
((Stub)hello)._setProperty( GSIConstants.GSI_MODE, GSIConstants.GSI_MODE_FULL_DELEG);
/* 5 */
((Stub)hello)._setProperty( Constants.GSI_TRANSPORT, Constants.ENCRYPTION);
/* 6 */
((Stub)hello)._setProperty( Constants.GSI_SEC_CONV, Constants.SIGNATURE);
/* 7 */
((Stub)hello)._setProperty( Constants.AUTHORIZATION, new IdentityAuthorization( hello_dn));
/* 8 */
Util.registerTransport();
try
{
/* 9 */
    String user = hello.hello();
    System.out.println( user);
}
catch( HelloException e)
{
    System.err.println( "Operation failed:");
    System.err.println( "errorCode: " + e.getErrorCode());
    System.err.println( "errorMessage: " + e.getErrorMessage());
}
catch( RemoteException e)
{
    System.err.println( e);
}
}
}

```

HelloClient.java

Marked parts have following meanings:

1. Loads user credential from default location. Please see "Client configuration" section for details,
2. Creates and returns object representing Hello interface,
3. Sets user certificate to be used to authenticate the user,
4. Sets delegation mode to "full". If you do not want to delegate the proxy replace *GSIConstants.GSI_MODE_FULL_DELEG* constant with *GSIConstants.GSI_MODE_NO_DELEG* one
5. Forces client to encrypt communication, that guaranties as well privacy as integrity,
6. Tells the client to use only digital signatures to protect messages integrity for "Secure Conversation", encryption is not needed on this level because the whole "channel" is encrypted,
7. Sets service Distinguished Name that will be used during the mutual authentication, must be used if delegation of certificate was chosen. Delegation of proxy certificate to unknown service is not allowed,
8. Registers axis transports for https and httpg protocols,
9. Invokes remote method.

47.3.3 Client configuration

To work properly, client has to be able to load user's proxy certificate and validate service credential during the handshake procedure. To do this it needs to know location of the file containing proxy certificate and directory containing public keys of Certificate Authorities it should trust.

- Client looks for proxy according to following rules:

It first checks the X509_USER_PROXY system property. If the property is not set, it checks next the 'proxy' property in the current configuration. If that property is not set, then it defaults to a value based on the following rules:
 If a UID system property is set, and running on a Unix machine it returns /tmp/x509up_u\${UID}. If any other machine then Unix, it returns \${tempdir}/x509up_u\${UID}, where tempdir is a platform-specific temporary directory as indicated by the java.io.tmpdir system property.
 If a UID system property is not set, the username will be used instead of the UID. That is, it returns \${tempdir}/x509up_u_\${username}

- Client looks for the CA directory according to following rules:

It first checks the X509_CERT_DIR system property. If the property is not set, it checks next the 'cacert' property in the current configuration. If that property is not set, it tries to find the certificates using the following rules:
 First the \${user.home}/.globus/certificates directory is checked.
 If the directory does not exist, and on a Unix machine, the /etc/grid-security/certificates directory is checked next.
 If that directory does not exist and GLOBUS_LOCATION system property is set then the \${GLOBUS_LOCATION}/share/certificates directory is checked.

CoG library configuration can be modified using the COG properties file **~/.globus/cog.properties**

```
#Java CoG Kit Configuration File
```

```
proxy=/tmp/x509up_u501
cacert=/etc/grid-security/certificates/
```

47.3.4 Running the client

Following set of jars is needed to run the client:

- Globus jars ([download tarball](#)):
 - ◆ addressing-1.0.jar
 - ◆ axis.jar
 - ◆ cog-axis.jar
 - ◆ cog-jglobus.jar
 - ◆ commons-codec-1.3.jar
 - ◆ commons-discovery-0.2.jar
 - ◆ commons-httpclient-3.0.jar
 - ◆ commons-logging-1.1.jar
 - ◆ cryptix-asn1.jar
 - ◆ cryptix.jar
 - ◆ cryptix32.jar
 - ◆ globus_java_authz_framework.jar
 - ◆ jaxrpc.jar
 - ◆ jce-jdk13-131.jar
 - ◆ log4j-1.2.13.jar
 - ◆ opensaml-1.1.jar
 - ◆ puretls.jar
 - ◆ saaj.jar
 - ◆ wsdl4j-1.5.1.jar
 - ◆ wsrf_core.jar
 - ◆ wsrf_core_stubs.jar
 - ◆ wsrf_provider_jce.jar
 - ◆ wss4j.jar
 - ◆ xalan-2.6.jar
 - ◆ xmlsec-1.2.1.jar
- Hello Example jars:
 - ◆ [hello.jar](#)
 - ◆ [client.jar](#)

Additionally the [client-config.wsdd](#) file from Globus 4.1.2 is has to be placed in directory listed on CLASSPATH.

IMPORTANT: In the client-config.wsdd file, the original declaration of transport for "https" protocol was replaced with another one to avoid undeterministic and inexplicable "NoHttpResponseException" on client side.

IMPORTANT: By default Sun 1.4.x+ JVMs are configured to use /dev/random device as an entropy source. Sometimes the machine can run out of entropy and applications and using the /dev/random device will block until more entropy is available. One workaround for this issue is to configure the JVM to use /dev/urandom (non-blocking) device instead. For Sun JVMs a java.security.egd system property can be set to configure a different entropy source.

```
-Djava.security.egd=file:///dev/urandom
```

Client usage:

```
java -Djava.security.egd=file:///dev/urandom acgt.examples.hello.HelloClient <SERVICE_URL> <SERVICE_DN>
```

for example:

```
#!/bin/bash
CLASSPATH=.
for i in ./jars/*.jar; do CLASSPATH=$CLASSPATH:$i; done
java -cp $CLASSPATH -Djava.security.egd=file:///dev/urandom \
    acgt.examples.hello.HelloClient \
    https://localhost:8443/HELLO/services/Hello \
    "/C=PL/O=GRID/O=PSNC/CN=Hello/druid-bis.man.poznan.pl"
```

Following results should be displayed:

- for **Hello World!: Showing who's calling**

```
/C=PL/O=GRID/O=PSNC/CN=Tomasz Piontek
```

- for **Hello World!: Showing information about delegated proxy**

```
subject      : C=PL,O=GRID,O=PSNC,CN=Tomasz Piontek,CN=1168110104,CN=2070173049
```

issuer : C=PL,O=GRID,O=PSNC,CN=Tomasz Piontek,CN=1168110104
strength : 512 bits
timeleft : 31931 sec
proxy type : Proxy draft compliant impersonation proxy

48 Globus Authorization Framework

If the Globus Authorization Framework is going to be used to authorize access to the service please note that:

- The Globus Authorization Framework has significantly changed in Globus 4.1.x
 - ◆ please see [Authorization Framework Release Notes](#) for list of changes
 - ◆ please see [Authorization Framework Documentation](#) for more details.
 - ◆ The architecture of Generic Java Authorization Engine is described in detail in this [document](#). It also describes interfaces and writing custom PDPs/PIPs.
- Globus Authorization Handler expects that every service's method has one parameter with the name equals to the name of the method.
 - ◆ if this requirement is not met authorization process fails with the error presented below. To solve this problem the modified `AuthUtil.class` has to be copied to `<tomcat.dir>/webapps/<name>/WEB-INF/classes/org/globus/wsrf/impl/security/util` directory.

```
[JWSSEC-23] Operation name could not be determined
```

48.1 GAS Plug-in

Configuring hosting environment to use Grid Authorization Service as a Policy Decision Point.

Copy the `gas-plugin.jar` into the `<tomcat>/webapps/<name>/WEB-INF/lib` directory

Please modify the `server-config.wsdd` file

- modify the value of "provider" attribute in "<service ...>" section from "java:RPC" to "Handler"

```
<service name="Hello" provider="Handler" style="wrapped" use="literal">
```

- please add "handlerClass" parameter

```
<parameter name="handlerClass" value="org.globus.axis.providers.RPCProvider"/>
```

- please add parameter pointing location of security descriptor

```
<parameter name="securityDescriptor" value="etc/hello/security-descriptor.xml"/>
```

Copy the `security descriptor` file into location specified in the previous step

```
<serviceSecurityConfig xmlns="http://www.globus.org/security/descriptor/service"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.globus.org/security/descriptor name_value_type.xsd"
  xmlns:param="http://www.globus.org/security/descriptor">
<auth-method>
  <GSISecureTransport/>
</auth-method>

  <authzChain>
    <pdps>
      <interceptor name="acgt:pl.psn.c.gas.plugins.gt41.GASAuthorization">
        <parameter>
          <param:nameValueParam>
            <param:parameter name="GASLocation" value="https://acgt-node.custodix.com:12355"/>
            <param:parameter name="GASDn" value="/C=EU/O=ACGT/OU=Custodix/CN=acgt-node.custodix.com"/>
          </param:nameValueParam>
        </parameter>
      </interceptor>
    </pdps>
  </authzChain>
</serviceSecurityConfig>
```

Please restart the tomcat.

Until the GAS is not configured to return PERMIT as an authorization decision the client's calls will fail with following error:

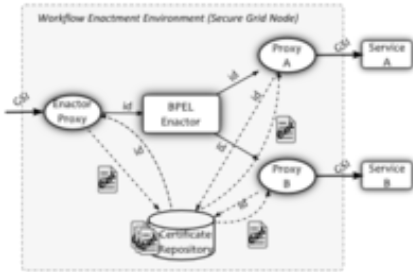
```
org.globus.security.authorization.AuthorizationDeniedException: [JWSSEC-161] "[/C=PL/O=GRID/O=PSNC/CN=Tomasz Piontek]"
is not authorized to invoke "hello" operation on this service
```

For further information regarding GAS please contact the technical support: `acgt-rt@ics.forth.gr`

49 PM:Developing a new Proxy service

49.1 Introduction

In designing and building the ACGT workflow environment the integration of the Grid security to the workflow enactor proved to be the most challenging task. In particular, the workflow engine should support the delegation of user rights so that all the services that participate in a workflow are contacted by the enactor in the name of the end user with no need for the user to be present during the workflow execution. What is more, there are numerous existing third party services which it would be nice to be integrated in scientific workflows and which do not support these security standards and it is neither possible nor desirable that these services be re-implemented. Thus, what is needed is a design approach which permits the mixture of heterogeneous systems. Taking into consideration that this mechanism must conform also to legal requirements, it becomes obvious why it is challenging to comply with the above and at the same time provide a functional, efficient and non-restrictive platform.



Secure enactor implemented through proxy services

There's an old adage saying that *Every problem in computer science can be solved by another layer of indirection*. The incompatibilities between the BPEL processes and the GSI secured ACGT services can be overcome by supplying the necessary layer of indirection: the Proxy Services. These proxies or wrapper services provide BPEL friendly facades of the original, real ACGT services, effectively working as calls transformation bridges between the two worlds.

The core idea behind wrapper services is that we transform the interface of the underlying service and we pass extra pieces of information, which empowers us to detour the usual flow of credential delegation and bypass the enactor. This information is an ID, unique for each enactment, with which the proxy service can retrieve the delegated credentials and pass them on to the service which is proxied. The BPEL workflow therefore is constructed in such a way as to pass this extra parameter in every outgoing request. The value of this extra parameter enters the BPEL engine through the Enactor Proxy: this is an inverse proxy service that presents a GSI compliant interface of the BPEL workflow. The Enactor Proxy saves the security context of the workflow enactment in the database and submits the corresponding database ID to the BPEL engine with the other parameters. The BPEL engine accepts these parameters and forwards the ID to the Proxy Services, which, based on this ID, subsequently retrieve the security context and make the GSI compliant request to the real service.

49.2 Generic Guidelines for the Implementation

The implementation of Proxy services depends a lot on the kind of the original service that will be "proxied". From the above discussion we see two major uses of proxies:

- Implement the security delegation mechanism in an enactor transparent way. Since this type of proxies provide the same functionality as the original services but they also augment ("decorate") it with the proper security context we call them **Decorators**.
- Provide either a high level view of the original service functionality (e.g. as is the case with **OGSA-DAI** services where we don't want the user to be aware of low level details like "perform documents") or an enactor friendly interface (i.e. a **WS-I** compliant web service interface). Let's call this specific type of proxies as **Adapters**

In either case the proxy service presents a modified WSDL interface to the enactor but the details differ. The main difference is that in the case of credential delegation every operation supported by the Proxy service must require an additional "enactmentId" parameter of type **(XML Schema) String**. **This parameter should have this exact name and type and should be the first in the list of parameters of every operation of the Proxy service interface.**

An additional point to make: A Proxy service could be very well both a Decorator and an Adapter. In fact this is the case for the **OGSA-DAI Data Access (Proxy) Services** because they require the user credentials to perform the delegation (and therefore the "enactmentId" parameter) and they are also **Adapters** since their WSDL is a lot different than the **OGSA-DAI** one.

The final thing to keep in mind is that Proxy services are contacted by the BPEL enactor so their interface needs to comply with its requirements. In designing the web service interface they are apparently **many different WSDL styles**. Nevertheless it seems that BPEL and the Apache ODE Workflow Enactor specifically do not support Web Services implemented in accordance to the **RPC/Encoded** style. The **WS-I** consortium also recommends against the use of **SOAP Encoding** rules. The services implemented following the **Document/Literal** style in WSDL are the ones that are fully compliant with BPEL and ODE. Furthermore the BPEL transformation tool employed by the **ACGT Workflow Editor** works with the WSDL version 1.1 and its **SOAP 1.1** binding. So in conclusion the requirements for the Proxy service interface description are the following:

- Use the **WSDL 1.1** web service description language
- Use (and specify in the WSDL document) **SOAP 1.1**
- Use (and specify in the WSDL document) the **Document/Literal wrapped** style

49.2.1 Getting access to the proxy certificate

In cases where the original service that is "proxied" is a GSI enabled Grid service, the delegated proxy certificate can be retrieved using the *enactmentId* parameter. In order to make this easy and transparent to the proxy service developers a **proxyAuth.jar** jar library is provided. The ProxyAuth class of this jar offers a *getCertificate()* method that given the enactmentId returns the **GSSCredential** proxy certificate.

Having the credential the proxy service code should use it and proceed to the actual call to the "original" service as described [here](#) as if it were the real client.

49.3 Examples

In this section we provide various examples of building Proxy services...

49.3.1 Data Access Services

The original ACGT Data Access Service (DAS) presents an **OGSA-DAI** compliant interface. The target (original) OGSA-DAI service requires valid ACGT user credentials when contacted and therefore the corresponding proxy services would be a "Decorator" as described above and all its operations will have an "enactmentId" string parameter. Furthermore we aim to provide a simple interface to the user based on the core functionality of the service, which is "data access" in relational (or Semantic Web) data. Therefore we design the Proxy interface to offer the following operations:

- `submitSQL (enactmentId: String, DataSet: String, SQL_query: String, DirName: String, FileName: String): String`

This is an operation to send an SQL query. *DataSet* is the (OGSA-DAI) resource id of the database, *DirName* and *FileName* designate the names of the folder and the file in DMS where the results will be stored. The return value is a *job id* that references this specific request (query) so that the client (i.e. the enactor) can use in order to find the status thereof.

- `submitSPARQL (enactmentId: String, DataSet: String, SPARQL_query: String, DirName: String, FileName: String): String`

The same as above but for SPARQL queries for Semantic Web databases.

- `hasFinished(enactmentId: String, jobId: String): boolean`

This operation returns the status (finished or not) of the query identified by the *jobid*

- `getResults(enactmentId: String, jobId: String): int`

This returns the DMS id of the file that persists the results of the query (if finished).

The resulted WSDL is here: <https://iapetus.ics.forth.gr/proxies/DASProxyService.wsdl>

49.3.1.1 Implementation

For the implementation of the Proxy service we use the **wSDL2java** tool from [Apache Axis](#)] to build the server skeleton file:

```
wSDL2java -s -u -W DASProxyService.wsdl
```

Note: Some notable command line options for wSDL2java are:

```
-s, --server-side
    emit server-side bindings for web service
-W, --noWrapped
    turn off support for "wrapped" document/literal
-u, --allowInvalidURL
    emit file even if WSDL endpoint URL is not a valid URL
-S, --skeletonDeploy <argument>
    deploy skeleton (true) or implementation (false) in deploy.wsdd.
    Default is false. Assumes --server-side.
-N, --NStoPkg <argument>=<value>
    mapping of namespace to package
-f, --fileNStoPkg <argument>
    file of NStoPkg mappings (default NStoPkg.properties)
-p, --package <argument>
    override all namespace to package mappings, use this package name instead
-o, --output <argument>
    output directory for emitted files
-B, --buildFile
    emit Ant Buildfile for web service
```

49.3.2 GridR Scripts

49.3.3 Biomoby Services

50 PM:ACGT Testbed Security Guide

50.1 Security

After Globus Toolkit installation, but before starting Globus services several aspects of GSI security need to be configured:

- configure Globus to trust a particular set of CAs (Certificate Authorities), i.e. place certificates of trusted CAs into designated directory. CA is trusted only if its CA certificate exists with the appropriate name in an appropriate directory. Moreover, for pre-ws services, signing policy file must exist in the same location as CA certificate. In other words, one needs two files to trust given CA: `cert_hash.0` - the trusted CA certificate and `cert_hash.signing_policy` - the signing policy. Java-based components ignore it and accept all valid certificates issued by trusted CAs. Globus services and tools look for that directory in following locations:
 - ◆ the value of `$X509_CERT_DIR` environment variable if it is set and the directory exists,
 - ◆ otherwise, in `$HOME/.globus/certificates` if it exists,
 - ◆ otherwise, in `/etc/grid-security/certificates` if it exists,
 - ◆ otherwise, in `$GLOBUS_LOCATION/share/certificates` if it exists.

Note: We suggest to use `/etc/grid-security/certificates` as system wide trusted CAs directory, but remember that `$X509_CERT_DIR` and `$HOME/.globus/certificates` have higher priority. `cert_hash.0`, i.e. certificate of the CA, is provided by CA, usually with appropriate hash name. Hash name consists of 8 hex-digits and suffix `.0` (e. g. `8a661490.0`). Valid hash can be obtained by the following command (available in `$GLOBUS_LOCATION/bin`): `openssl x509 -hash -noout -in ca_certificate`

- ◆ `cert_hash.signing_policy` usually is also provided by CA, but it can be constructed manually. The signing policy file has the following format:

```
access_id_CA X509 'CA Distinguished Name'
pos_rights globus CA:sign
cond_subjects globus '"Name Pattern1" "Name Pattern2" ...'
```

- to get 'CA Distinguished Name' execute:

```
openssl x509 -subject -noout -in cert_hash.0
```

- *name pattern* is a string used to match the distinguished names of certificates granted by the given CA. Usually, it is a CA name with common name replaced by wild card "*", e.g.:

```
"/C=PL/O=GRID/CN=Polish Grid CA" -> '"/C=PL/O=GRID/*'
```

it accepts `/C=PL/O=GRID/OU=PSNC/CN=Bogdan Ludwiczak` it rejects `/O=GRID/OU=PSNC/CN=Bogdan Ludwiczak` * pattern accepts all certificates.

- Configure appropriate default values for use by the `grid-cert-request` command which is used to generate certificates requests. The following files have to be properly configured to enable Globus tools to generate valid certificate requests:
 - ◆ `/etc/grid-security/globus-user-ssl.conf` - defines the distinguished name to use for a user's certificate request.
 - ◆ `/etc/grid-security/globus-host-ssl.conf` - defines the distinguished name for a host and service certificate request.
 - ◆ `/etc/grid-security/grid-security.conf` - is a main configuration file that contains the name and email address for the given CA.

These files are usually provided by the CA, particularly ACGT CA does provide these files. Typically, CA configuration files are placed in `/etc/grid-security/certificates/` directory with additional extension `.CA_hash_name` and only appropriate symbolic links are created in `/etc/grid-security/`. Globus Toolkit provides `grid-default-ca` command which can be used to automatically create appropriate links.

50.1.1 Accepting ACGT CA and installation of related CA files

ACGT CA certificate and configuration files are provided in a form of a tar ball available at: <http://moss1.man.poznan.pl/acgt-ca-2008.tar>. The hash name of ACGT CA is `e622f687`. To accept ACGT CA and install its configuration files simply:

- download the tar ball
- and as *root* user, `untar` it into `/etc/grid-security` directory.

This should place all files into `/etc/grid-security/certificates/` and create required links in `/etc/grid-security/` directory. Figure below shows the whole procedure of installing ACGT CA files:

```
bogdanl@cress ~ $ su -
cress ~ # cd /etc/grid-security/
cress grid-security # wget http://moss1.man.poznan.pl/acgt-ca-2008.tar
cress grid-security # tar xvf acgt-ca-2008.tar
./certificates/e622f687.0
./certificates/e622f687.signing_policy
./certificates/globus-host-ssl.conf.e622f687
./certificates/globus-user-ssl.conf.e622f687
./certificates/grid-security.conf.e622f687
./certificates/5b58e31d.0
./certificates/5b58e31d.signing_policy
cress grid-security # grid-default-ca -ca e622f687
```

50.1.2 Requesting host and user X.509 certificates

All Globus service (including GridFTP) require a host (or service) certificate to operate. Also every user needs a user certificate to use Globus services. You can use Globus command `grid-cert-request` to generate host/users certificate request which should be sent to your CA to be signed. Make sure that CA configuration files and certificate are in place before generating requests. This command will create 3 files:

- an empty (file length is 0) `/etc/grid-security/hostcert.pem` or `~/.globus/usercert.pem`,
- `/etc/grid-security/hostkey.pem` or `~/.globus/userkey.pem` file containing host/user private key which must be kept secret - make sure that the unix access mode is set to 0400 or 0600 at most,
- `/etc/grid-security/hostcert_request.pem` or `~/.globus/usercert_request.pem` file containing actual request to be sent to and signed by CA.

Note: Before you can use `grid-cert-request` command you have to source Globus configuration script appropriate for your shell, i.e. `$GLOBUS_LOCATION/etc/globus-user-env.sh` or `$GLOBUS_LOCATION/etc/globus-user-env.sh`

Send newly generated certificates request to the appropriate CA and wait for the certificate which should be sent in return by your CA. Save the new certificate in `hostcert.pem` or `usercert.pem`. Now, request file can be deleted. Host certificate and private key should be owned by `root` user.

50.1.3 Specify identity mapping information

Globus services map distinguished names (retrieved from certificates) to local identities (unix account) by means of `grid-mapfile`. Mappings have the following form: "Distinguished Name" local_name (every file of the file defines one mapping) To let user in, create an account and add mapping to `grid-mapfile`. Globus looks for the file in the following locations:

- the value of `GRIDMAP` environment variable if it is set,
- otherwise, if service is run as root then grid map file is `/etc/grid-security/grid-mapfile`,
- otherwise, the grid map file is `$HOME/.gridmap`
- otherwise, in `/etc/grid-security/grid-mapfile`.

50.1.4 Firewall configuration

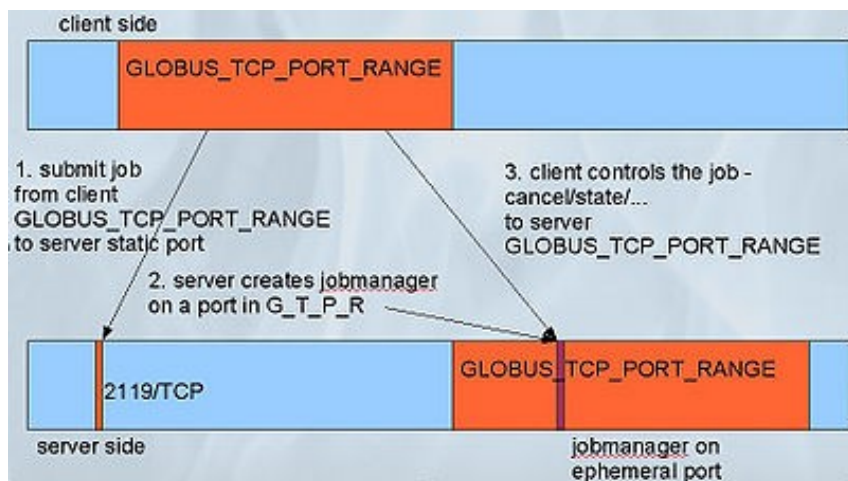
Detailed information on GT firewall issues can be found at <http://www.globus.org/toolkit/security/firewalls/>. This paragraph only lists the minimum required firewall configuration and gives a short overview of the basic issues. To enable remote access to Globus Toolkit services the following TCP port should be opened for the incoming connections:

- static ports:
 - ◆ 2119 for pre-ws GRAM
 - ◆ 2811 for GridFTP
 - ◆ 8080,8443 for Globus service container
 - ◆ arbitrary chosen port for gsi-enabled ssh, we suggest to leave regular ssh on standard 22/TCP port
- ephemeral TCP ports
- Various Globus services require an arbitrary chosen TCP port range. It is controllable by environmental variable `GLOBUS_TCP_PORT_RANGE` for pre-ws components or by jav system property "org.globus.tcp.port.range" for java-based ws components. It can be set as follows:

```
$ java -Dorg.globus.tcp.port.range=5000,6000
$ export GLOBUS_TCP_PORT_RANGE=5000,6000
```

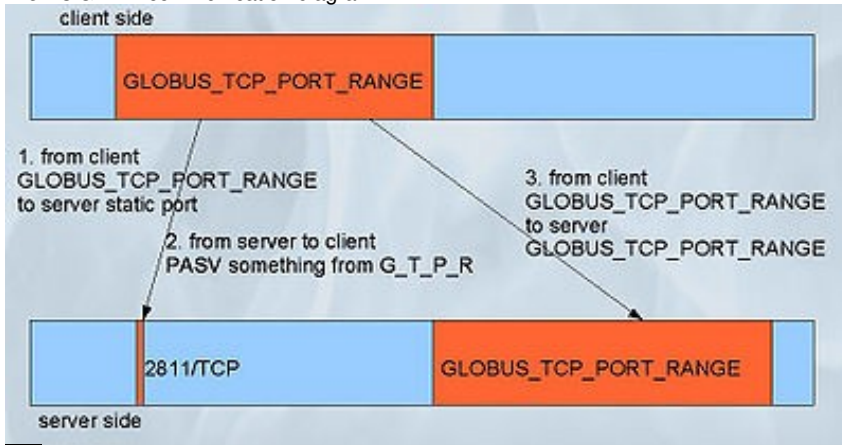
- make sure it is defined in daemons environment (i.e. put it in (x)inetd configuration entries and in `GLOBUS_OPTIONS` env variable for Globus service container)
- mind that number of ports limits the number of controllable jobs which can be submitted via pre-ws GRAM

Figures below gives an overview of typical Globus communication flow for pre-ws GRAM and Grid-FTP appropriately. They show the meaning of the aforementioned Globus TCP port range.





Pre-ws GRAM communication diagram



GridFTP communication diagram.

50.1.5 GridFTP

After successful installation of GT, the GridFTP daemon binary is located in: \$GLOBUS_LOCATION/sbin/globus-gridftp-server file. The daemon could be run in standalone mode, but we suggest using (x)inetd mode. First add the following entry to /etc/services file:

```
gsiftp          2811/tcp          # GridFTP
```

(Port 2811 is the IANA registered GridFTP port) Next add appropriate entry to (x)inetd configuration file(s). Remember to set LD_LIBRARY_PATH, GLOBUS_LOCATION and GLOBUS_TCP_PORT_RANGE in the (x)inetd entry environment, like in the xinetd example below:

```
service gsiftp
{
    instances            = 100
    socket_type          = stream
    wait                 = no
    user                 = root
    env                  = LD_LIBRARY_PATH=/opt/globus/lib
    env                  += GLOBUS_LOCATION=/opt/globus
    env                  += GLOBUS_TCP_PORT_RANGE=9000,9999
    server                = /opt/globus/sbin/globus-gridftp-server
    server_args          = -i -c /opt/globus/etc/gridftp.conf
    log_on_success       += DURATION
    log_on_failure       +=
    nice                  = 10
    disable              = no
}
```

Note that the above example specifies grid-ftp configuration file (-c ../gridftp.conf in server_args line). It is not required - GridFTP daemon should work without this file. A useful example of this file is:

```
log_level ALL (or 'EERROR', 'WARN', 'INFO', 'DUMP')
log_module stdio
log_single /var/log/gridftp
```

it enables logging which can be helpful by resolving potential problems. Finally, restart (x)inetd to activate the GridFTP daemon.

50.1.6 WS-GRAM

Since Globus container acts on non-privileged account, it need a mean to execute jobs on the submitters behalf. The current GT version uses *sudo* command for that purpose. WS GRAM requires that the *sudo* command is installed and functioning on the service host. Appropriate authorization rules will need to be added to the sudoers file to allow the WS GRAM service account to execute (without a password) the scheduler adapter in the accounts of authorized GRAM users (job submitters). Below are the required sudoers rules:

```
globus ALL=(ALL) NOPASSWD: \
/opt/globus/libexec/globus-gridmap-and-execute \
-g /etc/grid-security/grid-mapfile \
/opt/globus/libexec/globus-job-manager-script.pl *

globus ALL=(ALL) NOPASSWD:
/opt/globus/libexec/globus-gridmap-and-execute \
-g /etc/grid-security/grid-mapfile \
/opt/globus/libexec/globus-gram-local-proxy-tool *
```

50.1.7 RFT

RFT requires JDBC compliant database system to operate. The preferred database system is PostgreSQL. To enable RFT:

- install PostgreSQL in a way suitable for the resource's operating system.
- Configure the postmaster daemon so that it accepts TCP connections. This can be done by adding the "-o -i" switch to the postmaster script.
- create *globus* PostgreSQL user:

```
$ su postgres
$ createuser globus
```

- create *rftDatabase* database owned by *globus*

```
$ su postgres
$ createdb -O globus rftDatabase
```

- set the security on the *rftDatabase* - i.e. add the following line to *pg_hba.conf*:

```
host rftDatabase globus "host-ip" 255.255.255.255 md5
```

- and restart/reload PostgreSQL
- initialize database structure:

```
psql -U globus -d rftDatabase \
-f $GLOBUS_LOCATION/share/globus_wsrft/rft_schema.sql
```

- Check database parameters in RFT configuration file *\$GLOBUS_LOCATION/etc/globus_wsrft/jndi-config.xml*
- Find the *<resource name="dbConfiguration">* element and change the *connectionString* parameter to point to the machine on which you installed PostgreSQL and to the name of the RFT database,
- Change the *userName* and *password* to the name of the user who owns the RFT database
- default values for other parameters should work well in most cases.

50.1.8 MDS4

As described in "ACGT D4.1 Prototype and report of the ACGT GRID layer" WS MDS4 consist of:

- Index Service - collects monitoring and discovery information from Grid resources, and publishes it in a single location. It aggregates the WS-Resources properties
- Trigger Service - collects data from resources on the grid and, if administrator defined rules match, can perform various actions.
- Aggregator Framework - the software framework on which Index and Trigger are built.
- WebMDS - an additional, helper component. It is a web browser interface for viewing monitoring information (not required)

This document covers only minimum Index Service configuration which is required for proper operation of ACGT common grid infrastructure layer. More information on configuring MDS4 is available in [1]. By default every GT4 container provides container-wide MDS4 index service. Local container's services (by default WS-GRAM and RFT) automatically registers to container index when correctly configured MDS4 Index Services can register to each other comprising multi level hierarchy. Particularly, local indexes can be registered to VO wide index. For the ACGT current Grid infrastructure the VO-Index service is provided and maintained by PSNC. It is available at:

<https://moss1.man.poznan.pl:8443/wsrft/services/DefaultIndexService>

PSNC also provides WebMDS interface for the ACGT VO Index. It is available at:

<http://moss1.man.poznan.pl/webmds/>

50.1.8.1 ACGT required MDS4 Configuration

- turn on host name publishing (instead of default IP publishing) - add two parameters to *<globalConfiguration>* section in file:

```
$(GLOBUS_LOCATION)/etc/globus_wsrft_core/server-config.wsdd
```

```
<parameter name="publishHostName"
value="true"/>
<parameter name="domainName"
value="man.poznan.pl"/>
```

this will change information available in index: from: <https://150.254.173.173:8443/wsrft/services/ManagedJobFactoryService> to: <https://moss1.man.poznan.pl:8443/wsrft/services/ManagedJobFactoryService>

- Building MDS4 hierarchy

MDS4 supports both, upstream and downstream registration. Its configured in *\$(GLOBUS_LOCATION)/globus_wsrft_mds_index/hierarchy.xml* by any combination of any number of the following entries:

```
<upstream>
```

```
    https://vo-host:8443/wsrf/services/DefaultIndexService
</upstream>
<downstream>
    https://member-host:8443/wsrf/services/DefaultIndexService
</downstream>
```

to register in ACGT VO-level index add the following entry:

```
<upstream>
    https://moss1.man.poznan.pl:8443/wsrf/services/DefaultIndexService
</upstream>
```

51 PM:ACGT Bug Reporting

52 ACGT Bug Reporting - General Guidelines

52.1 Introduction

The ACGT Portal comprises a series of tools and services, each one offering specific features.

Descriptions of each available service, as well as manuals, are located at the ACGT Handbook ([Main_Page](#)).

The following instructions are targeted at end-users of the ACGT Portal. They intend to give some general guidelines on how to properly report a found bug within the portal.

52.2 Which types of bugs to report

- **Unavailable Tool/Service:** a required tool or service does not load in the ACGT Portal.
- **Not-working feature:** a feature of a service does not work as intended or they results cannot be visualized/stored properly.

Example: *After uploading a query with the Query Tool, it is not shown as available in the Workflow Editor.*

- **Incomplete documentation:** the ACGT Handbook does not contain enough information for performing some action in the ACGT Portal.

Example: *The Handbook does not give enough detail on how to save a query created in the Query Tool.*

- **Outdated documentation:** documentation in the handbook does not reflect the actual functioning of the service.

Example: *The handbook depicts the button required for saving a query created in the Query Tool, however this button is not available in the actual tool.*

52.3 How to report a bug

In case you find an issue as described above, you can notify it by email. Some information should be provided:

- **Contact information (email):** the developers might need to contact you to obtain some more details on the issue, or perform some tests.
- **Detailed bug description:** Please give as much detail as possible when describing the bug.

Example: *?Running the workflow wf01 from my account shows an error window after 4-5 minutes? **is better than** ?The Workflow Editor does not work!!!?.*

- **Provide reproducible steps when possible:** if possible, describe the steps you took since you logged into the Portal until you reached the bug.

Example:

1. I log in to the Portal.
2. I select the Query Tool in the main page.
3. The Query Tool loads and shows the available data sources.
4. I select the data source named ?mcmp?.
5. An error window is shown.

- **Provide the timestamp if possible:** if the bug is not documentation related, please include the exact date and time at which you found it. It will help us finding the source of the error.

Example: *The Query Tool did not load. The time was May 15th 2010, at around 11:30am GMT+1.*

53 EX:ACGT End to End Scenario

Preliminary note: The present scenario covers most uses of the ACGT platform. However, it does NOT form a fully integrated scenario, as it has some pre-established "short cuts" to minimize the need to have interactions with administrators. This is specially the case with ObTiMA for which two trials have already been setup: one demonstrating the preparation of the CRFs for a trial (DOEDEMO), one demonstrating the use of ObTiMA in the patient data collection phase (MCMP). Such a setup is required as it is only possible to collect patient data when the design of the trials (the CRFs) is frozen on the administrative side.

54 Overview of the scenario

- **Step 1& Step 2: ObTiMA - Trial Management** - Create a trial & Run a trial

Note: The DOEDEMO trial exists only to demonstrate the setup of a trial. It is not used in later steps.

- **Steps 3-7: ACGT Data Mining** - The MCMP simulated trial is based on a dataset which has been published in the scientific literature. The data of this trial will be used as source to illustrate *the use of the ACGT data mining environment*.

◇ **Step 3:** The creation of a mediator **query to access** the MCMP **data** is demonstrated.

Note: prerecorded queries will be used in the actual workflow

◇ **Step 4:** The registration of an **external** (user-defined) **database** is also illustrated (dynamic registration of a database in the system); this shows the ability of the system to be fed with arbitrary data.

◇ **Step 5** The **creation** and registration of a **GridR** analytical **service** supporting the analysis of the association between a patient treatment and tumor relapse.








◇ **Step 6** Then the data sources and the analytical service are combined in a **workflow**.

◇ **Step 7 Results** are **visualized** in the ACGT Data Management System (DMS).

55 Prerequisites

- An user account in the ObTiMA system, with trial manager rights on the DOEDEMO and the MCMP clinical trial.
- An ACGT user account, with **Group membership** (Query Tool, DMS, Data Access, GridR, weeditor, metadataregistration2, howto)

56 Scenario

1. Create a trial  **Not done**
2. Run a trial  **Not done**
3. Create a query  **Done**
4. Register an external database  **Done**
5. Register a GridR service  **Done**
6. Create and run a workflow  **Done**
7. Visualize the results  **Done**

56.1 ObTiMA: Creation of a trial (CRF preparation)

- Go to the ObTiMA page: <https://acgt-obtima.ibmt.fhg.de/> and log in
- A new trial can only be created by the ObTiMA administrator, this can be requested by sending an e-mail to the address listed under "Contact / Administration".
- Clinical trials which are in preparation phase are listed in "Trials / Create/Develop trials"
- Select the DOEDEMO trial.
- General / administrative information of the trial can be defined in the "General" node of the tree (in particular the "Characteristics of the trial".
- CRFs can be created/edited through the "Treatment / List of CRFs" tree item.
- **CRF creation**
- Click on "Add new CRF" (this immediately creates the new CRF)
- The general information about the CRF is set in the "Metadata" box in the lower right side.
Set a title for the CRF.
- Edit questions on the CRF:
 - ◆ **Add non-ontology based question:** (Question with red background)
 - ◆ Type in the question text.
 - ◆ Select the type of answer.
 - ◆ If list-based answer: Provide the list of possible values, separated by semi-colons (;) .
 - ◆ Then "Save edited item" or "Save as new item" as needed.
 - ◆ **Add ontology based question:** (Question with green background)
 - ◆ Click on "New from ontology"
 - ◆ Select the ontology term to associate to the question.
(E.g. Treatment / Pharmacotherapy / Chemotherapy / ABVDChemo)
 - ◆ Click on "Exist item" checkbox on the right and edit the question as required, e.g. "Patient received ABVD chemotherapy?"
 - ◆ Click on "Add item to preview"
 - ◆ Click on "Create items"
 - ◆ **Organize the CRF**
 - ◆ The order of questions on a CRF can be changed by clicking on the question on the list and moving it.
 - ◆ **Save the CRF:** click on the corresponding button.
- Give roles to users in a trial (as trial chairman)
 - ◆ "Administration / Register User in trial".
 - ◆ Select the trial to which to register a user.
 - ◆ Select the user.
 - ◆ Select the second tab for roles and rights.
 - ◆ Activate the checkbox for the appropriate role. ("Local physician" is required to see patients.)
 - ◆ "Save Roles"

56.2 ObTiMA: Running trial / Edition of patient information

- Go to the ObTiMA page: <https://acgt-obtima.ibmt.fhg.de/> and log in
- View CRFs associated to a running trial
 - ◆ Click on the menu "Trials / List my trials"
 - ◆ Select the MCMP trial
 - ◆ Visualize the forms for the trial
- List patients
 - ◆ Menu "Patients / List my patients"
(Actually all patients visible by a clinician are visible in this view, independently of the trial.)
 - ◆ Select one patient (e.g. Cressida Bunya-Bunya), then the data associated to the patient are displayed.
 - ◇ General information about the patient are on the first tab
 - ◇ Select the CRF tab; all CRFs associated to the patient are shown
 - ◇ Select one CRF (e.g. "Tumor categorization"); the data associated to the CRF are shown.
 - ◇ Change the ER status for the patient and save the form.
 - ◇ Navigation to other CRFs is done through the "Back to ..." buttons on the right.
 - ◆ Adding a patient is made through the main menu "Patient / Enter new patient".
Please do not save new patients with MCMP, some scripts may fail later on in the exercise".

56.3 Data mining: Creation of a mediator query

See this section for additional instructions on [Retrieving data](#)
See this section for additional description of the user interface: [Query_Tool](#)

The next step will create a mediator query which can be used to retrieve data from a clinical trial. A later execution of this query will return a file which contains a snapshot of the clinical trial database at the time of execution (for the selected variables **only**).

- Login in the [ACGT_Portal](#)
- Go to the [Query_Tool](#) tab.
- Click on the *Show Help* button to see the 5 areas of the tool (optional more detailed description [here](#)):

- **Step 1:** Select the trial from which you want to get the data from:
 - Locate the *Create Query* box (on the right [area 1](#))
 - Select the data source (the clinical trial) e.g. *MCMP Simple C*

- **Step 2:** Select what information (input) you want the query to retrieve from the trial data.
 - Locate the *Entries* list (on the right [area 2](#))
 - Select (one at a time) each entry you see as a possible question to be "asked" from the "Entries" list (e.g. Tamoxifen status) and add them using the < **Add Entry** button.



Note: Alternatively, you can add all the entries at once using the < **Add All Entries** button. You will have available all the entries in step 3 and sort out what you want and what you don't want there!

- **Step 3:** Choose what data you want to be returned by the query from the list of available variables. These variables can be selected from the ones associated with the entries you have selected.



Caution: The entries in step 2 and the variables in step 3 are **not the same!**

Entries are **relationships** closer to ideas expressed in a natural language, while variables denote actual **data** seen as columns in a database.

E.g. Some entries such as *Gender of Patient - Male* have more than one corresponding variable: *Gender and Male*

- Locate the *Query description* box (on the left, [area 3](#)). This list of data fields/variables are **available** to be selected as columns of the result table of the query.



Note: The variables corresponding to the entries you have added in step 2 are *available* but not actually *selected* to be returned by the query. Keep in mind that selected variables will be **actual columns** of the result table while the remaining available variables which are not selected at this step will remain unused.

- Select the variables that you want the query to return as data fields will form part of the result of our query



Caution: The order in which you add them will be the exact order of the columns of the data returned by the query

- Select each of them and move them to the **Selected variables** area, selecting each of the desired data fields and clicking the >> button.
- You can always change your mind and move selected variables back to the available variables list by clicking the << button.

- **Step 4:** Click on *Show* to see the corresponding SPARQL query
- **Step 5:** (Optional) Click on *Submit* to test the query (i.e. see what will be returned), this opens the *Results* tab.
- **Step 6:** Once the data returned looks OK, click on *Upload query to metadata repository* to save the query and make it available as a service.



Note: Remember the name you give to the query as will have to use it later in the [Workflow Editor](#).

56.4 Data mining: Registration of an external database

See this section for additional instructions on [Importing data](#)
See this section for additional description of the user interface: [Data Access](#)

You can connect your private database to the rest of the ACGT environment in the [Data Access area](#) of the [ACGT_Portal](#).

- Login in the [ACGT_Portal](#)
- Go to the [Data Access](#) tab.

- Click on *Add new data resource*

- Specify:

◊ the **location** of the database - the JDBC connection string

Example: `jdbc:mysql://iapetus.ics.forth.gr/acgt_japan`

◊ the **username** required to access the database

◊ the **password** required to access the database

◊ the **dialect** of the database (*MySQL, PostgreSQL, Oracle*)

Example of connection settings to a public database can be found [here](#).

- Click on *Next*
- Provide a name which will be used in the resource's actual name:

Example: If you name it XXXX, the resource will actually be accessible later with the name: "Dynamic-XXXX-ogsadai-nnnnnnnnn" where nnnnnnnnn is a random unique identifier assigned by the system.

- A default RDF mapping is shown so that you can modify it if you want, before finishing the import process.
- Click on *Register resource* to finalize the import process

56.5 Data mining: Registration of a GridR analytical service

For a more detailed description of how to register a service see [HB:Browsing/registration_of_services](#).

- Go to the **Metadata registration** tab in the ACGT Portal.
- Select the functional category under which the service should be registered in the left tree.



Note: Services which are not generic (i.e. linked to a specific dataset/clinical trial) should fall under the *Bioinformatics/Scenario* category, e.g. *HomburgNovember2009*

- Click on the **AddRScript** button.
- Define all required inputs and outputs for the script.

◊ Each input is the result of a R command `read.csv()` which result is stored as an element of the special R list "gridr.input".

◊ Outputs are passed as strings in the `gridr.output` list, strings which can be used as file names for storage.

◊ E.g. a script which has 4 inputs and 1 output would typically have an R interface such as:

```
map      = gridr.input[[1]]
patients = gridr.input[[2]]
tumors   = gridr.input[[3]]
survival = gridr.input[[4]]
pdf(file = gridr.output[[1]], paper="a4", width=28./2.54, height=18./2.54)
```

- Look at *Scenarios / HokkaidoSeptember2009 / Hokkaido_MCMP_light* for an actual implementation of an R script
- Alternatively you can get from the [Appendix Section of this page](#) the same R script.
- Fill in the name of the service (**mandatory**)
- Fill the other associated metadata (**recommended**) such as **text/csv** or **application/pdf**.

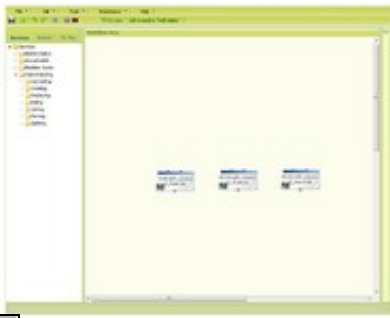
56.6 Data mining: Creation and Registration of a workflow

Background: We are creating a scientific workflow which is making a statistical analysis on the clinical trial data. It is based on several previously defined queries which are already registered in the ACGT and previously registered R script (such as the one in the previous step).

More information regarding the environment - the editor in which we will create the workflow can be found here: [HB:Workflow_Editor](#)

More information on designing and running workflows can be found here: [HB:Setting_up_workflows](#)

- Select the **Workflow editor** tab.
- Select from the menu **File / New**



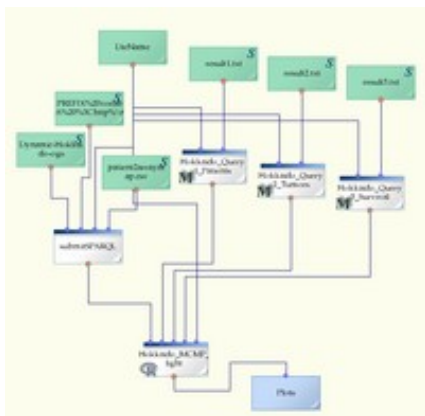
After step 1 - the workflow area should look like this

- **Step 1** - Insert queries that will get us the **input clinical data** in the workflow
 - ◊ We will insert 3 previously registered mediator queries in the workflow, one by one.
 - ◊ Click on the **"M"** icon in the **toolbar** under the **menu bar** to see all the mediator queries available (registered).
 - ◊ Select the query *Hokkaido_Query1_Patients* and click **Add** to insert it in the workflow.
 - ◊ A small rectangle with an **"M"** in the bottom left corner will appear in the **center frame - workflow area**.
 - ◊ Repeat the operation for the queries *"Hokkaido_Query2_Tumors"* and *"Hokkaido_Query3_Survival"*.

- **Step 2** - Connect our external database to the workflow. For this we need to add a SPARQL service that will take create a mapping from patients to micro-array data. Such a connection/mapping is done through a SPARQL query. The ACGT environment already has such a generic service that can be used to add arbitrary SPARQL queries to a workflow - called **Submit SPARQL**.
 - ◊ Locate the Services Panel - **east frame of the editor**
 - ◊ Navigate in the tree and select **Services / Bioinformatics / Proxies / DynamicDAS / SubmitSPARQL**
 - ◊ Click on the **"SubmitSPARQL"** item
 - ◊ Drag-and-drop it on the workflow editor

- **Step 3** - Insert a GridR analytical component that will be used to analyze data (using R programming language) and plot the results.
 - ◊ Click on the **"R"** icon in the **toolbar** under the **menu bar** to see all the available R-based services(registered).
 - ◊ Select **Hokkaido_MCMP_light** script (alternatively you can use your GridR script registered in the previous step)
 - ◊ Click **Add** to insert it in the workflow.
 - ◊ A small rectangle with an **"R"** in the bottom left corner will appear in the **center frame - workflow area**.

- **Step 4** - Connect the different components of the workflow so that it looks as in the figure below



The final workflow should look like this




Note: Blue dots are inputs and red dots are outputs from each box. Query help you to get data and R scripts to analyze this data - so the R script uses as input the outputs of the queries!

- ◊ First look at the R script and then work your way backwards to collect the data from the queries:



Caution: The output of the four queries (3 Mediator queries and the SPARQL query) must be connected as input to the GridR component. The inputs must be connected in the order expected by the script. Input slot 1 (leftmost), corresponds to the 1st element of *gridr.input*, etc.


- ◊ Hovering with the mouse over its 5 inputs (blue circles on the top) will show you each input parameter as it was defined in the R code.
- ◊ The three middle ones are associated to the queries 1,2,3 that get the data regarding patients, tumors and survival.
- ◊ Connect the red dot of the query to the corresponding blue dot of the R box
- ◊ Connect the red dot (output) of the SPARQL query to the first blue dot of the R box
- ◊ Drag the last (5th from left to right) blue dot of the R box (it's called "DirName") to an empty space. A green box will be created. This will be the output directory that the R script will use to store its results.
- ◊ Double-click on the green box and give it a name such as "MyResultsAreHere" so that you can easily remember where the results will be.
- ◊ Connect this result box to the left blue dot of each of the M Queries (3 connections needed) so that they will all use the same directory to store the results to.

 **Note:** Clicking on a connection selects it (the rest of the environment appears to fade in the background coloured in grey). You can simply press the delete key on the keyboard to get rid of unwanted connections or boxes anytime!

- ◊ Drag the remaining blue dot on each query to an empty space to create 3 separate files for each of the queries result.
- ◊ Double-click them and rename them to *result1.txt* and so on..
- ◊ Let's finish with the **SubmitSPARQL** box
- ◊ Connect the first blue dot in this box to an empty space to create the dynamic resource box.
- ◊ Double-click it and give it the name you wrote down after registering your dynamic database (ex:Dynamic-Hokkaido-ogsadai-124777a375b)
- ◊ Connect the second blue dot of the SubmitSPARQL box to an empty space to create a SPARQL Query.
- ◊ Double-click this box and copy-paste the following SPARQL Query:

```
PREFIX vocab: <http://gridnode.ehv.campus.philips.com/d2rq/automatic#>
SELECT ?patient_id ?marrayA ?marrayB
WHERE {
  ?tumors
  vocab:tumors_patient_id ?patient_id ;
  vocab:tumors_marrayA      ?marrayA ;
  vocab:tumors_marrayB     ?marrayB ;
}
```

- ◊ Connect the fourth and last blue dot of the SubmitSPARQL box to an empty space to create the CSV file where it will store its results. Rename it to *patient2arraymapping.csv*
- ◊ Finally drag the red dot of the R script to an empty space to create the green box with the resulting plots.

 **Note:** It is recommended to use static strings for file names, and a execution-time-specified string (i.e. do not edit, leave it as the green box was created) for the directory name.

- **Step 5 - Save** the workflow under a new name (e.g. using your name as prefix) using **File / Save as**
- **Step 6 - Deploy** the workflow (this step converts our drawing to an actual BPEL document that can be run by the workflow environment) using **Tools / Make Executable**. Please wait for the "BPEL process deployed" message.
- **Step 7 - Run** the workflow: **Tools / Run**.
- You can see the status in the south panel called **Status Panel**
- You will see all the queries start to blink at once (This is a grid environment, they can be run on different machines at the same time).
- Each of them will stop blinking after finishing their execution.
- After all of them finish, the R script will begin to analyze the data.
- After the R script has completed its analysis you will see *status FINISHED* in the Status panel

56.7 Data mining: ACGT DMS and Visualization of results

See this section for additional instructions on how data is stored in ACGT [Storing data](#)

See this section for additional description and information regarding the DMS user interface: [DMS](#)

After the execution of the workflow the results files will be stored in the **ACGT Data Management System** called (**DMS**). To find the results:

- Login in the [ACGT_Portal](#)
- Go to the *My Files* tab.
- Check that you're in the *File-Directory Mode* which shows 3 panels.
- Click on the the directory in the *Left Panel* to open it.
- Find the output file and click on it
- A visualization tool will show the content of the file in the *Right Panel*

56.8 Appendix: GridR script for data mining

```
map = gridr.input[[1]]
for (i in 1:length(colnames(map))) { # Hack to remove hardcoded blank in ACGT DAS
  map[,i] = sub("^ ", "", map[,i])
}
d1 = gridr.input[[2]]
d2 = gridr.input[[3]]
```

```

d3 = gridr.input[[4]]
pdf(file = gridr.output[[1]], paper="a4", width=28./2.54, height=18./2.54)

pat.id1 = as.character(d1[, "clinicaltrialpatientnumberstring_2"])
birth.date = as.Date(d1[, "patientbirthdate_3"])
recr.date = as.Date(d1[, "recruitmentdate_4"])
treatment = as.character(d1[, "tamoxifen_5"])
pat.age = floor(as.numeric((as.Date(d1[, 4]) - as.Date(d1[, 3])) / 365.25) + 0.5)
names(recr.date) = pat.id1
names(birth.date) = pat.id1
names(pat.age) = pat.id1
names(treatment) = pat.id1
pat.id.all = pat.id1

pat.id2 = as.character(d2[, "clinicaltrialpatientnumberstring_1"])
er.status = as.numeric(as.character(d2[, "malignneoplasmerstatus_3"]))
names(er.status) = pat.id2

pat.id3 = as.character(d3[, "clinicaltrialpatientnumberstring_1"])
e.rfs = as.character(d3[, "diagnosticprocesstumorrelapse_2"])
e.rfs[e.rfs=="No"] = 0
e.rfs[e.rfs=="Yes"] = 1
e.rfs = as.numeric(e.rfs)
rel.date = as.Date(d3[, "tumorrelapsediagnosticprocessdate_3"])
t.rfs = as.numeric(rel.date - recr.date[pat.id3])
names(t.rfs) = pat.id3
names(e.rfs) = pat.id3

library(survival)

sel = names(e.rfs)
sel = sel[-which(is.na(er.status[sel]))]
sf = survfit(Surv(t.rfs[sel], e.rfs[sel]) ~ er.status[sel])
plot(sf, col=c(1,2), main="Patient survival (ER stratification)")

sel = names(e.rfs)
sf = survfit(Surv(t.rfs[sel], e.rfs[sel]) ~ treatment[sel])
plot(sf, col=c(1,2), main="Patient survival (Treatment stratification)")

# ----- New material -----

#
# Verify that the status of Patient OXFU_12 has changed
#

pat.0=names(which(er.status==0))
pat.1=names(which(er.status==1))
if (length(which(pat.0=="OXFU_12"))) {
  pat.0=pat.0[-which(pat.0=="OXFU_12")]
}
if (length(which(pat.1=="OXFU_12"))) {
  pat.1=pat.1[-which(pat.1=="OXFU_12")]
}

demo.mat=matrix(c(
  er.status[pat.1[1]], er.status[pat.0[1]], er.status[pat.1[4]],
  er.status[pat.1[2]], er.status["OXFU_12"], er.status[pat.1[5]],
  er.status[pat.1[3]], er.status[pat.0[3]], er.status[pat.1[6]]), nrow=3)

image(demo.mat, col=c(0,2), xaxt="n", yaxt="n")
z = ((0:2)/2)
text(c(z[1], z[1], z[1], z[2], z[2], z[2], z[3], z[3], z[3]),
     z,
     c(pat.1[1], pat.0[1], pat.1[4],
       pat.1[2], "OXFU_12", pat.1[5],
       pat.1[3], pat.0[3], pat.1[6]))

dev.off()

```


57 EX: Integrated demonstrators

57.1 Hokkaido wf07

Goal: Query a dynamically deployed database.

57.1.1 Dynamic integration of database

1. Select "Data Access" portlet
2. Click on "Add new data resource"
3. Database access information:
 1. Connection string: jdbc:mysql://iapetus.ics.forth.gr/acgt_japan
 2. Username/Password: janedoe / m.....cp.g
 3. Database type: MySQL
4. Give resource name: "Hokkaido" (The prefix "Dynamic-" and the suffix "-ogsadai-XXXXXXXXXXXX" are added automatically.)

The full name of the resource (e.g. Dynamic-Hokkaido-ogsadai-1240b95a240) will be used to query the database in the workflow environment.

57.1.2 Create simple workflow to query the database

1. Select the "Workflow Editor" tab
2. Create a new workflow
3. Add a SPARQL query component to the workflow
Services tab -> Bioinformatics -> Proxies -> DynamicDAS -> submitSPARQL
4. Add inputs to the component by click-drag-and-drop the input slots to the working area.
Double-click the newly created blocks to open a dialog which lets assigning the input value.
 1. DataSet: use the name of the dynamic resource defined above
 2. SPARQL_query: SPARQL query to be executed (see below)
 3. DirName: output directory (can be left unspecified to be queried at execution time)
 4. FileName: name of the file that will contain the query output (for storage on the DMS)
5. The output of the component can be left unconnected.

The SPARQL query to be used in "SPARQL_query" to retrieve fields "patient_id", "marrayA", and "marrayB" from table "tumors" is:

```
PREFIX vocab: <http://gridnode.ehv.campus.philips.com/d2rq/automatic#>
SELECT ?patient_id ?marrayA ?marrayB
WHERE {
  ?tumors
  vocab:tumors_patient_id ?patient_id ;
  vocab:tumors_marrayA ?marrayA ;
  vocab:tumors_marrayB ?marrayB ;
}
```

57.2 Hokkaido wf06

Subset of wf05 and wf04.

Goal: Use mediator queries to retrieve information from an ObTiMA database, and execute a GridR script.

57.3 Hokkaido wf05

Subset of wf04.

Goal: Retrieve information from ObTiMA database using the mediator and from a dynamically registered database to feed a GridR analysis workflow.

57.4 Hokkaido wf04

Goal: Retrieve clinical information from ObTiMA database using the mediator and microarray meta-information (chip names) from a dynamically registered database. The latter is used to query BASE to retrieve actual microarray data. All data are combined in a GridR analysis workflow.

58 EX:Multi-centric multi-platform data analysis

Some or all of the information on this page is inconsistent, irrelevant or confusing.

Please help clean it up if you are able.

In this scenario, we will demonstrate how to use ACGT to do a multi-centric multi-platform data analysis. In particular, you will see how to use ACGT's workflow editor and the ACGT data analysis tools.