



## Express Production Real-time e-VLBI Service

EXPReS is funded by the European Commission (DG-INFSO),  
Sixth Framework Programme, Contract #026642

# DJ1.10 eVLBI – Grid interface document

Title: DJ1.10 eVLBI – Grid interface document  
Sub-title:  
Date: 2007/01/15  
Version: 1.0  
Filename: dj1.10\_eVLBI-Grid\_interface\_document\_v1.0

Author: M. Okoń, D. Stokłosa  
Co-Authors: N. Meyer, M. Stroiński, D. Kaliszan, T. Rajtar, M. Lawenda

Summary: This document describes several aspects of Grid - eVLBI interfaces including communication protocols, graphical user interface of the WFM application, the description of the VEX file, and the aspects of correlator integration with the Grid environment, and the introduction of the Grid resource brokers.

Delivery Slip

	Name	Partner	Date	Signature
From	N. Meyer, M. Okoń, D. Stokłosa	PSNC	15.01.2007	
Approved by				

Document Log

Version	Date	Summary of Changes	Authors
0.1		Initial draft	N. Meyer, M. Okoń, D. Stokłosa
1.0		Release	N. Meyer, M. Okoń, D. Stokłosa, M. Stroiński, D. Kaliszan, T. Rajtar, M. Lawenda

Project Information

Project Acronym	EXPRoS
Project Full Title	Express Production Real-Time e-VLBI Service
Proposal/Contract number	DG-INFSO #026642

## Table of Contents

1. Introduction.....	5
2. Conceptual view of the system design.....	6
3. Communication protocols.....	8
3.1. SOAP.....	8
3.2. Web Services.....	9
4. User-Grid interface.....	11
4.1. Workflow Manager (WFM) – central point of the eVLBI.....	11
4.1.1. Users in the eVLBI environment.....	11
4.1.2. WFM – graphical user interface.....	12
4.2. eVLBI experiment showcase.....	14
4.2.1. VEX processing.....	15
4.2.2. Designing VLBI experiment.....	16
4.2.3. Data flows definition.....	18
4.3. VEX file parsing and visualization.....	19
4.3.1. VEX File Definition.....	19
4.4. Software correlator control file.....	21
5. Software correlator in Grid environment.....	23
5.1. Grid resource broker.....	23
5.1.1. Overview.....	23
5.1.2. Job description.....	25
5.2. WFM – GRMS broker interface.....	26
5.2.1. eVLBI resource broker.....	26
6. Summary.....	29
Definitions, abbreviations, acronyms.....	30
References.....	31
Contact Information.....	32

## Table of Figures

Figure 1.	System architecture .....	6
Figure 2.	Soap message .....	8
Figure 3.	Soap response.....	9
Figure 4.	Web Services architecture.....	10
Figure 5.	Data flow between components .....	11
Figure 6.	WFM – main view .....	12
Figure 7.	Information panes - examples .....	13
Figure 8.	Design pane – sample VLBI experiment .....	13
Figure 9.	Message Log .....	14
Figure 10.	WFM – Welcome Screen .....	15
Figure 11.	Setting up the VLBI experiment – radio telescopes net.....	15
Figure 12.	Resource properties .....	16
Figure 13.	Adding new resource.....	16
Figure 14.	Adding new resource – fileserver.....	17
Figure 15.	Sample resource properties dialog .....	17
Figure 16.	Sample VLBI experiment – without data flows.....	18
Figure 17.	Connecting nodes .....	18
Figure 18.	VLBI experiment – complete scenario.....	19
Figure 19.	Sample VEX file .....	20
Figure 20.	Sample CCF file.....	22
Figure 21.	The GRMS .....	24
Figure 22.	Job description .....	26
Figure 23.	eVLBI communication.....	27
Figure 24.	Resource Description Schema.....	28
Figure 25.	Links Description Schema .....	28

# 1. Introduction

Networks of radio telescopes can be used to produce detailed radio images of stars and galaxies. The resolution of the images depends on the overall size of the network (the maximum separation between the telescopes) and the sensitivity depends on the total collecting area of all the telescopes involved and, crucially, the bandwidth of the connection between the telescopes. In this technique, called Very Long Baseline Interferometry (VLBI) the signals between all telescope pairs are combined in a data processor. Typically this processor must be able to cope with data rates up to 1 Gbps per telescope.

This functionality has been implemented by constructing a massively parallel, purpose-built ‘supercomputer’ – usually referred to as a Data Processor or Correlator. The processor capabilities continue to develop, the networking infrastructure is also getting more capable of dynamically handling very large data transfers in long periods in time. All of the factors mentioned so far let us believe that we are able to design and implement eVLBI system. Data processor was replaced by the distributed software correlator spread all over the Grid environment. We have also designed Workflow Manager Application which control and manage observations. The solution presented here has been address in the eVLBI – Grid design document [1].

This document describes the communication interfaces between various system components in eVLBI [10] system. It contains the short review of the system design, the description of the communication protocols and explanation of various interactions between eVLBI modules. It also describes the graphical user interface of the Workflow Management Application – which is the central point of the system. The application is used by the Central VLBI Operator to design and manage the observations.

## 2. Conceptual view of the system design

The detailed description of the system architecture has been presented in the DJ1.6 report: eVLBI – Grid design document [1]. This chapter presents briefly the overall design and creates a proper context for the rest of the document.

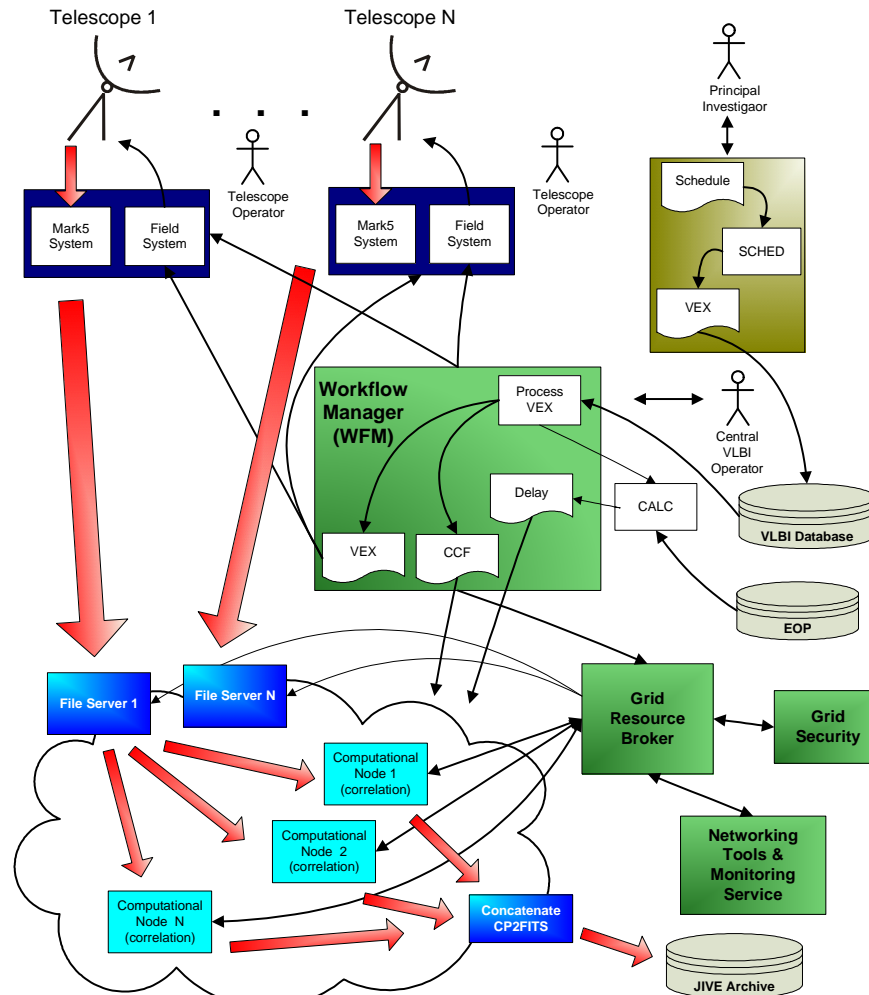


Figure 1. System architecture

The eVLBI process is shown on the Figure 1. The presented system architecture flowchart can be briefly explained in the following steps.

1. The user creates experiment description (VEX file) using SCHED application.
2. The VEX is processed in the Workflow Manager (WFM) by the VLBI operator. Experiment control parameters can be verified and modified if necessary. The central operator will be able to set up more eVLBI parameters, and also he (or she) will be able to associate specific file servers with radio telescopes locations and create a workflow for the post-experiment distributed data correlation. The WFM will also calculate the necessary delay tables before the correlation takes place by calling the external program CALC and Earth Orientation Parameters (EOP).
3. An update VEX file will be created and sent to the telescopes participating in the observations. The WFM also notifies the telescope operators that a new experiment is scheduled, and sends the planned routing information between telescopes and file servers. It also contacts the dedicated Grid Resource Broker to allocate the necessary computational

nodes for each of the correlation tasks, and sends it the newly created software correlator control file (CCF) and Delay file.

4. Telescope operator loads the VEX file in the Field System which controls the telescope and in the Mark5 system which records the data. The data recorded by the Mark5 system is sent to the allocated computational nodes via the file servers. The data is correlated using Grid resources according to the defined workflow.

### 3. Communication protocols

One of the most important decisions in every distributed system design is the selection of the communication interface. It should be chosen according to specific environment requirements. In case of eVLBI system, the system is characterized by a large number of different hardware and software platforms which need to communicate in order to produce a successful eVLBI experiment. Nowadays there is a well known and used communication protocol that meets the interoperability requirement. It is the Simple Object Access Protocol (SOAP) [3], and its most known implementation, the Web Services [4].

#### 3.1. SOAP

SOAP Version 1.2 is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. SOAP relies on HTTP [9] as a transport mechanism to send XML based messages, the messages are packed in what is called a SOAP envelop and send to the server to process in a Request/Response fashion. SOAP unlike proprietary protocols like DCOM or RMI does not require strong connection between client and the server and the SOAP messages are sting based messages passed from the Client to Server and vice versa in the form of SOAP envelops.

A sample request - response SOAP messages are shown below. They refer to the hypothetical function **GetStockQuote(Symbol: string)** which returns a float type value of a given stock price.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema" >
  <SOAP-ENV:Body>
    <ns1:GetStockQuote xmlns:ns1="urn:xmethods-quotes">
      <SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        <symbol xsi:type="xsd:string">IBM</symbol>
      </ns1:GetStockQuote>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
  <?xml version="1.0" encoding="UTF-8"?>
```

Figure 2. Soap message

The first tag is the **<SOAP-ENV:Envelope ... >** tag. This tag is an outer shell to the SOAP packet, giving various namespace declarations. A namespace is a way to qualify an XML tag - for instance a namespace cannot have two variables with the same name, but it is allowed if they are in two different namespaces.

The **<SOAP-ENV:Body>** tag is a placeholder that starts off the actual SOAP call.

Coming next is the **<ns1:GetStockQuote ...>** tag. The tag name, **GetStockQuote**, is the function to be called. In SOAP terminology, this is called an *operation*. GetStockQuote is the operation that needs to be executed. *ns1* is a namespace, which points to *urn:xmethods-quotes* in this case.

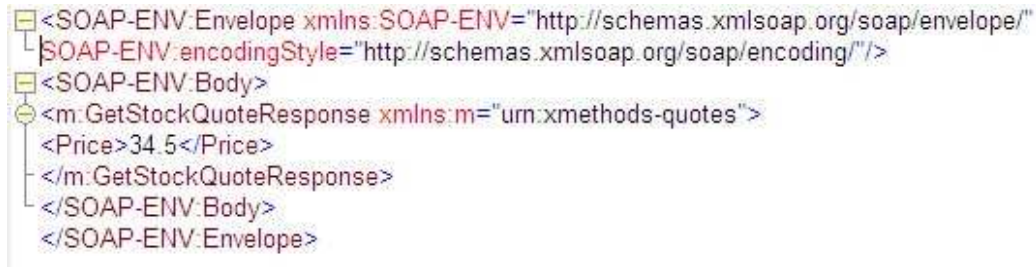
An **encodingStyle** attribute - this attribute specifies how a soap call is serialized. Within the **<GetStockQuote>** tag are the parameters. In this simple case, there is only one parameter, the **<symbol>** tag. It is extended by the definition **xsi:type="xsd:string"**

It is the type definition, as defined in the *xsi* namespace, which is declared in the **<SOAP-ENV:Envelope>** tag, is **xsd:string**. Which is a **string**, as defined in the *xsd* namespace, defined earlier.



Inside of the <symbol> tag there' is the value of the parameter *symbol* to the GetStockQuote function.

And below is a sample response message from the server, with the <Price> parameter with the required information:



```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetStockQuoteResponse xmlns:m="urn:xmethods-quotes">
      <Price>34.5</Price>
    </m:GetStockQuoteResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 3. Soap response

This example shows the flexibility and simplicity of a SOAP protocol itself. The SOAP messages are usually transported using HTTP protocol. An example HTTP request is presented below:

```
POST /StockQuote HTTP/1.1
Host: www.stockquotesever.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"
```

...the soap request packet here...

And the sample response would be as follows:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
```

...Soap Response packet here...

The HTTP protocol is the most common way of sending SOAP messages, because network administrators do not have to worry about opening separate ports for SOAP calls. A web server would do the job, and port 80 is usually opened to the world to handle incoming web requests. The other advantage is that web servers are usually extensible using CGI, ISAPI or other native modules. This extensibility allows us to write a module that will handle a SOAP request, while not affecting any other web content.

It is also very common to use HTTPS instead of HTTP, which adds a very high level of security to the message parsing interface.

### 3.2. Web Services

The Web Services [4] platform is a simple, interoperable, messaging framework, incorporating the SOAP and HTTP(S) protocols. It is a set of technologies that exposes business functionality over the Web as a set of automated interfaces. These automated interfaces allow businesses to discover and bind to interfaces at run-time, supposedly minimizing the amount of static preparation that is needed by other integration technologies.

Firstly, a standard way of capturing service descriptions is necessary. The Web Services Description Language (WSDL) [8] has been developed for this purpose. WSDL describes a service as a set of 'ports' which group related interactions that are possible between the application (service requestor)

and the Web Service (service provider). The interactions that are possible through a port are described as 'operations' which may have an input message and optionally a resulting output message. Each operation describes a potential interaction with the Web Service. This may be a request from the application to the web service. It could also be an interaction that can be initiated by the web service for which the application needs to take action. Interactions in either direction can be one-way or can require a response to be sent.

There are two different kinds of user for WSDL documents. During development of an application that will use a web service, the developer needs to know the *interface* to the service that the application will bind to. When the application is running it needs details of a specific *implementation* of that service so that it can bind to it. WSDL can be used to specify both interfaces and their implementations.

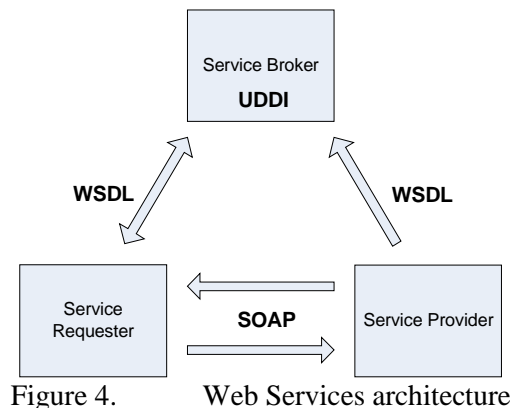


Figure 4. Web Services architecture

WSDL describes a service in terms of possible interactions with it. A WSDL document provides the potential information content of interactions with a web service but doesn't explain how to communicate that information between an application and a web service. For this purpose, the SOAP protocol is used. SOAP is typically transmitted over HTTP or HTTPS providing a platform for communication with/between web services.

The purpose of Universal Discovery, Description and Integration (UDDI) [7] is to enable mechanisms to “discover” the existing and available Web Services. In other words, UDDI is a specification for distributed registries of Web Services.

A UDDI web services registry is itself a web service which can be accessed via SOAP from an application that wishes to discover web services. UDDI specifies interfaces for applications to publish web services (as WSDL documents) and to discover web services (via their WSDL documents).

A UDDI entry actually contains more than just a WSDL interface and implementation, it can also include further metadata such as quality of service parameters, payment mechanisms, security and keywords for resource discovery.

These standards complete the infrastructure to publish (WSDL, UDDI), find (WSDL, UDDI) and bind (WSDL, SOAP) Web Services in an interoperable manner.

## 4. User-Grid interface

This section describes the interaction between end user and eVLBI system according to the system architecture and design. We have decided to create an application – Workflow Manager (WFM), which will be the central point of the eVLBI. The WFM and its interface are described in this chapter. As well as the VEX File format [1], which is used to describe VLBI experiment, software correlator control file is presented in details.

### 4.1. Workflow Manager (WFM) – central point of the eVLBI

As it was mentioned in the introduction Workflow Manager Application has been designed to take a role of the central and control place for the system users. The advantage of such an approach is that there is only one entry point to the system. One entry point means also one unique and user friendly interface. The interaction between user and system components is presented in the figure below (see Figure 5)

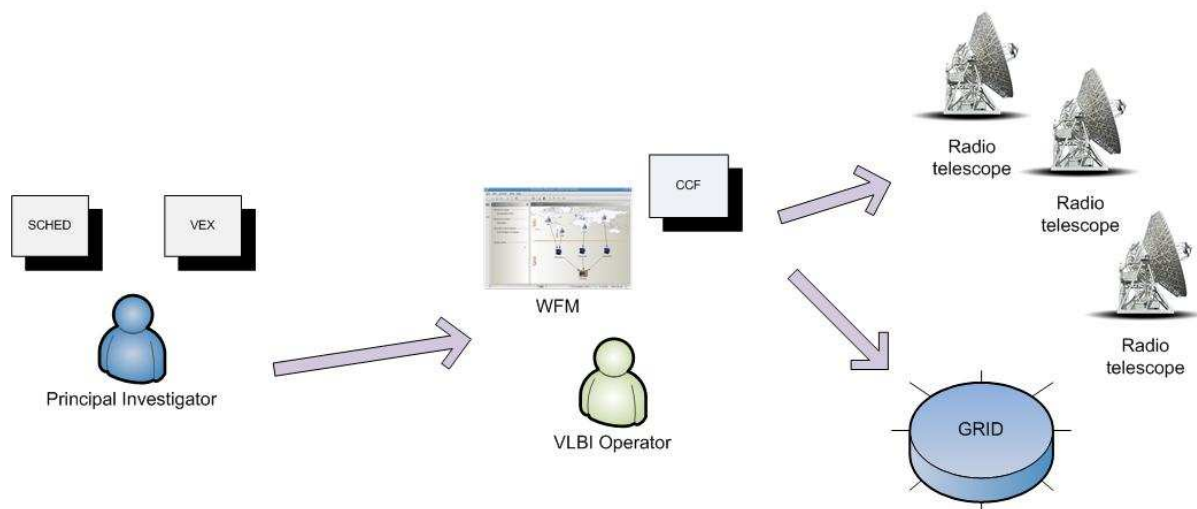


Figure 5. Data flow between components

The data flow is initiated by the user – Principal Investigator, who creates the observation schedule file (VEX). After that Central VLBI Operator loads the VEX file into the WFM system. The VEX file is described in more detail later on in this chapter. The VEX file is validated by the WFM and based on the parameters found in the VEX file the eVLBI experiment is initiated. The list of radio telescopes is taken from the experiment description file and devices are located on the application design pane. The last step required before experiment submission into the Grid environment is mapping between some parameters from VEX file and Correlator Control File (CCF). This issue is also discussed in more details soon in this chapter.

#### 4.1.1. Users in the eVLBI environment

According to the system architecture we have distinguished three different types of users: *Principal Investigator (PI)*, *Telescope Operator (TO)* and *Central VLBI Operator (CO)*. PI is the external user of the eVLBI system. He is interested in getting access to the radio telescope infrastructure. PI is also responsible for creating observation schedule file, containing details like telescopes used in the experiment, sources of observation or exact observation time specification. On the other hand, TO is responsible for setting up the radio telescope for the experiment. One TO is devoted to one radio telescope, so in order to prepare all devices for the data acquisition TO operators need to cooperate with each other. Moreover each TO needs to know all the settings in advance. There has to be also one person responsible for managing the observation. His job is to make sure all radio telescopes are properly set up by TO before the experiment start up. Moreover, he is also designing data flows

between system components: File Servers and Correlator. This super user is called VLBI Central Operator in our design.

Because of the complexity of this issue, tools required for creation of the VLBI observation schedule, as well as the time limitation of the project we have decided to support only Central VLBI Operator in the WFM.

### 4.1.2. WFM – graphical user interface

The main interface between user and system has been constructed using Java technology. The graphical user interface has been designed with Swing library and great help of JGoodies libraries [6]. We have deployed the prototype of the WFM with Java Web Start technology. Using Java Web Start technology, standalone Java software applications can be deployed with a single click over the network. Java Web Start ensures the most current version of the application will be deployed, as well as the correct version of the Java Runtime Environment will be used. The main advantage of such an approach is easy, unique and intuitive interface. Moreover, the application can be run at every computer connected to the Internet and equipped with web browser.

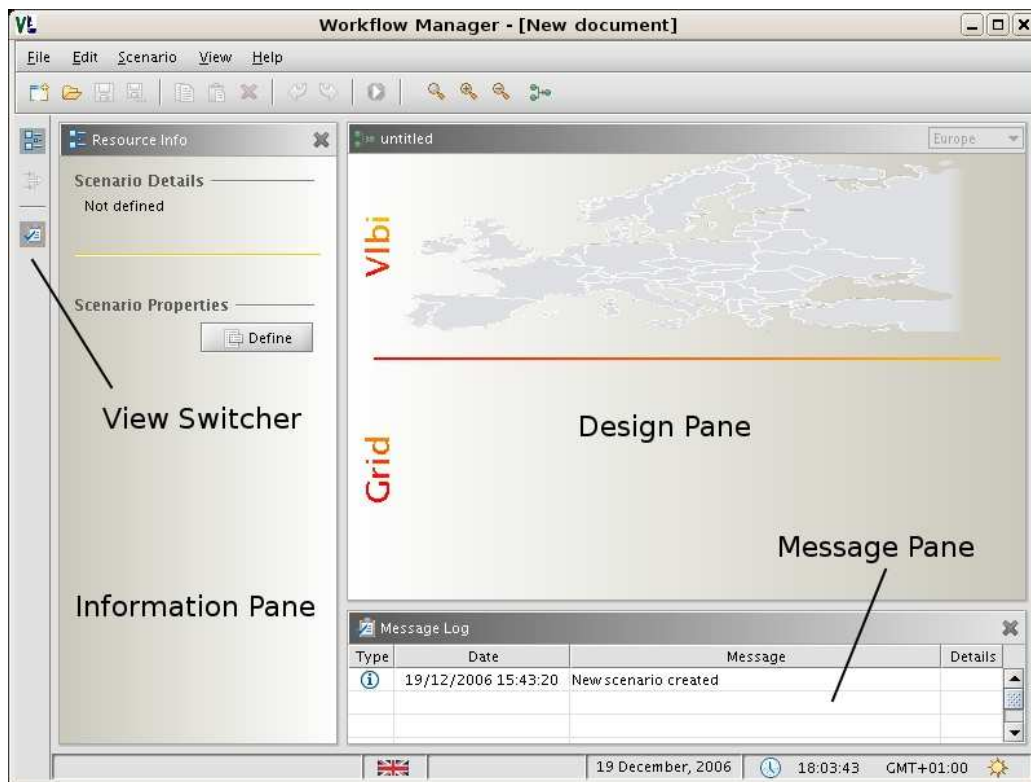


Figure 6. WFM – main view

The Workflow Manager Application is divided into several working panes: *Information Pane*, *Design Pane* and *Log Message Pane*. Each view has its own place in the application window (see Figure 6). User can manage the visibility of the different views, adjusting the application to so it meets his needs.

#### 4.1.2.1. Information Pane

Information Pane is the only “dynamic” pane which means that it presents different information based on the application state. This allows user to get the most important (more general) data quickly and without tiring and time consuming interaction.

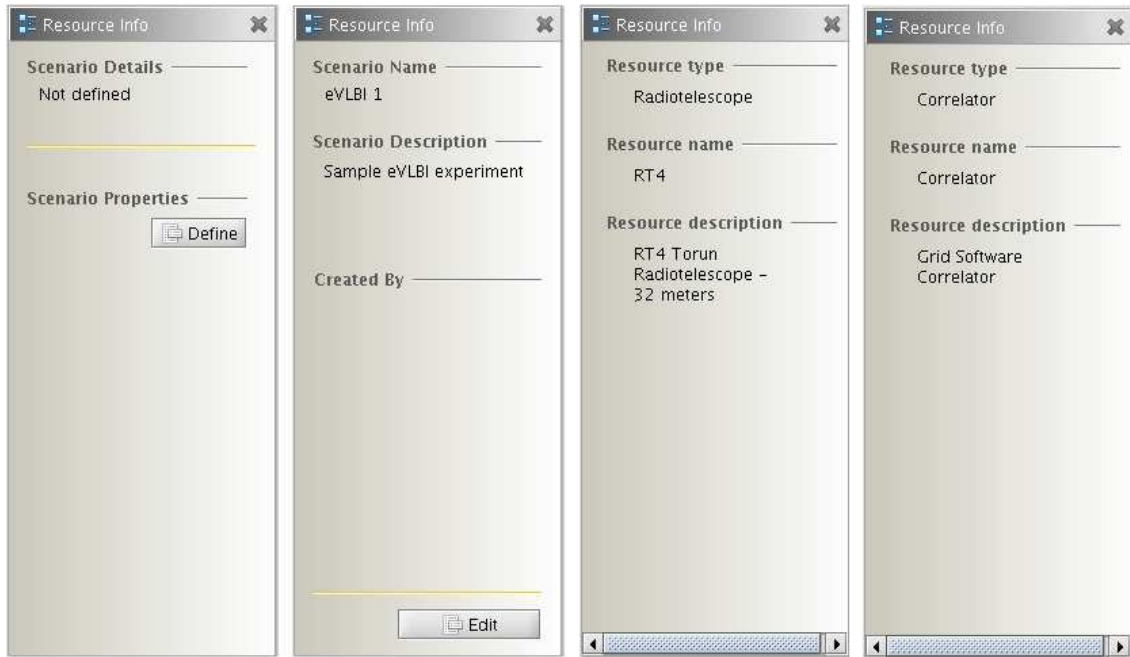


Figure 7. Information panes - examples

The Figure 7 presents several combinations of information panes. They contain data like short summary about current experiment scenario and resources: radio telescope and correlator.

#### 4.1.2.2. Design Pane

Design pane it is used mainly by the VLBI operator for the VLBI scenario design and management. The pane itself is divided into two parts: *VLBI* and *Grid* by the horizontal separator line. The *VLBI pane* is reserved only for the radio telescopes. The system analyzes the VEX file and visualizes all the radio telescopes, which take part in the VLBI experiment the given map. Be default it is the smallest map capable of displaying all the nodes. The map view can be changed into different

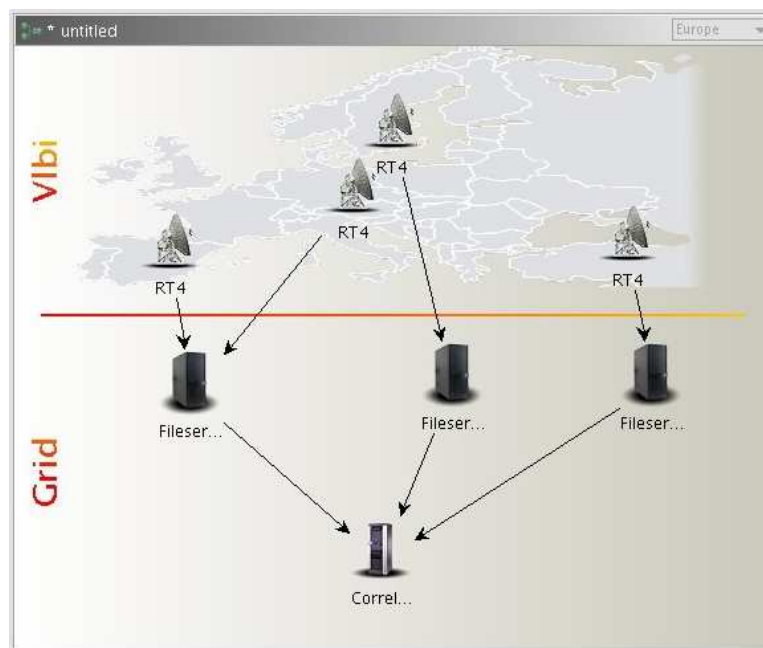


Figure 8. Design pane – sample VLBI experiment

World region. The VLBI part cannot be changed. This means that it is not allowed to add new radio telescopes or remove them. All nodes are not moveable – each telescope has its own location. One eye glimpse and Central VLBI operator knows exactly which radio telescopes are taking part in the experiment. In the future work we plan to add monitoring system to the existing model. Such a tool will be very powerful and useful for the experiment operator. Detailed description of the monitoring system is not a scope of this document.

*Grid pane* is intended to set up (design) data flows between the radio telescopes and Grid. Based on his experience the Central VLBI Operator decides how many file servers should be used, what is the best way to connect radio telescopes with those servers. Moreover, he also defines parameters of the correlator node, manage the transformation between the VEX and CCF.

#### 4.1.2.3. Message Log Pane

The message pane is used as an information board between user and system. There are several types of messages: information, error, notification. The display can be customized, it can be turned on or off at any time.

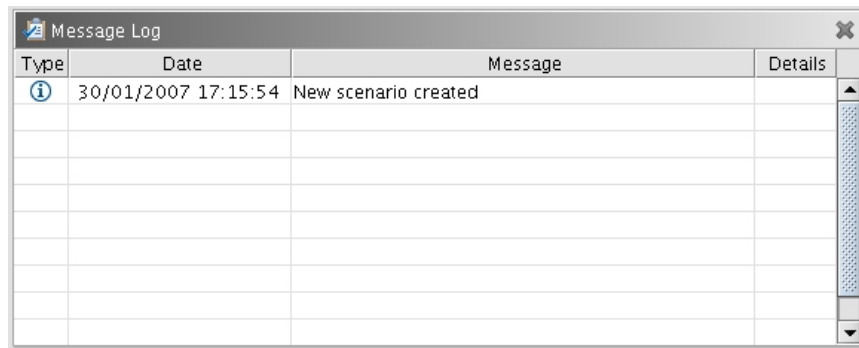


Figure 9. Message Log

## 4.2. eVLBI experiment showcase

This section describes the interaction between WFM and Central VLBI Operator. We will present how to design VLBI experiment easily and quickly. The given description refers to the prototype version. The process of defining the eVLBI workflow has been divided into the several steps, which are summarized below.

- 1) VEX file validation and processing
- 2) Designing VLBI experiment
  - a) Definition of File Servers
  - b) Definition of correlation node
  - c) Resource properties definition
- 3) Definition of the data flows between resources
- 4) eVLBI scenario submission

The detailed description of each phase can be found at the following sections.

## 4.2.1. VEX processing

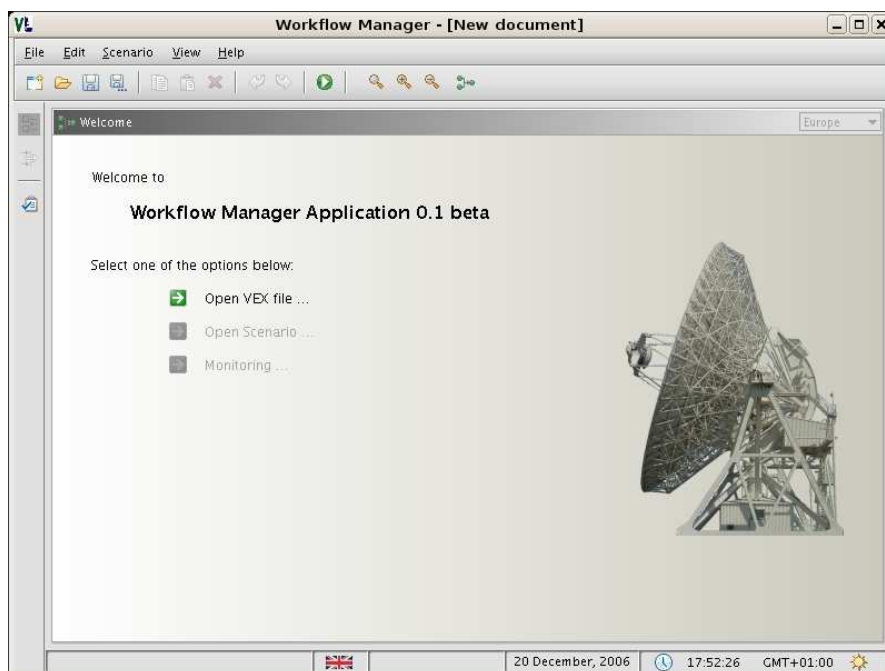


Figure 10. WFM – Welcome Screen

The welcome screen is presented on the Figure 10. Choose “*Open VEX file ...*” option from the welcome screen in order to open a VEX file. WFM will validate and analyze the VEX file and draw radio telescopes net on the *Design Pane* (the VLBI part). All the telescopes from the VLBI experiment will be drawn on the map.

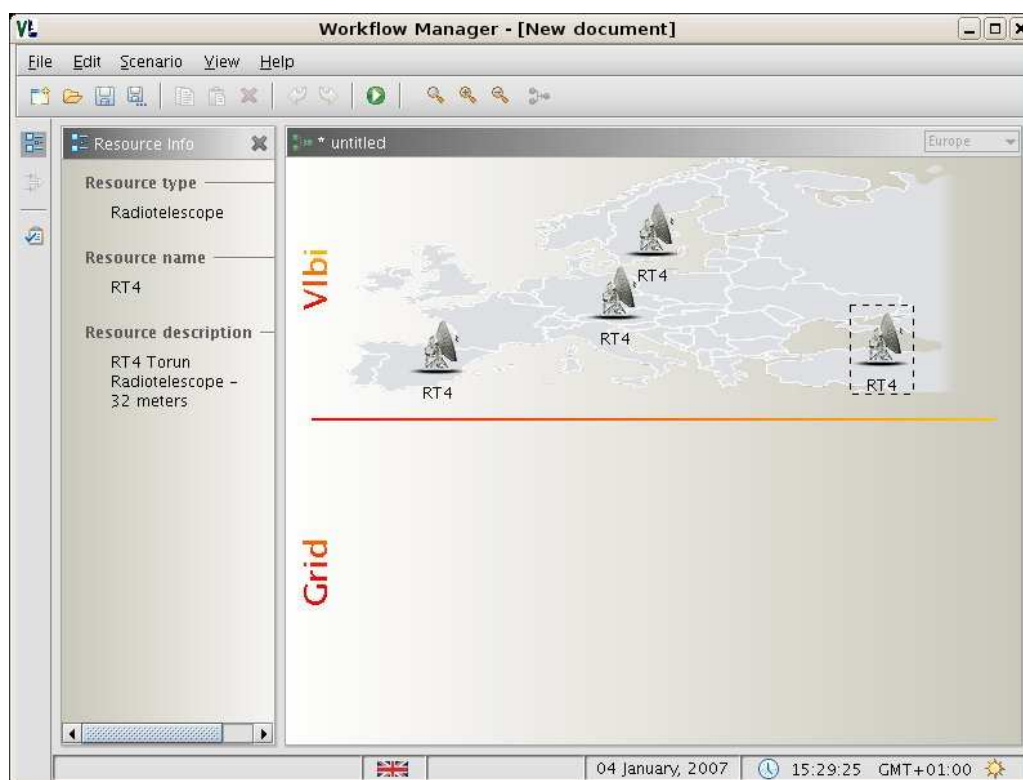


Figure 11. Setting up the VLBI experiment – radio telescopes net

In the prototype version the radio telescopes are drawn at random locations. In the real VLBI experiment, the devices will be drawn at their given locations. This gives an overview of all radio telescopes which take part in the VLBI experiment. At this stage the VLBI Operator can view or change the radio telescope parameters. To preview/edit parameters double click on the device icon or right click on the device icon and press “*Properties*” (see Figure 12)

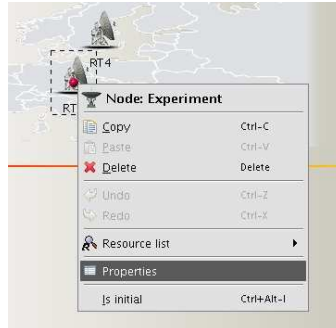


Figure 12. Resource properties

### 4.2.2. Designing VLBI experiment

At this stage we have defined the VLBI part of the experiment. Now we have to design the GRID part: choose file servers, describe correlator parameters and finally define data flows.

#### File Servers

File servers can be added into the workflow by inserting new resources into the *Grid part* and

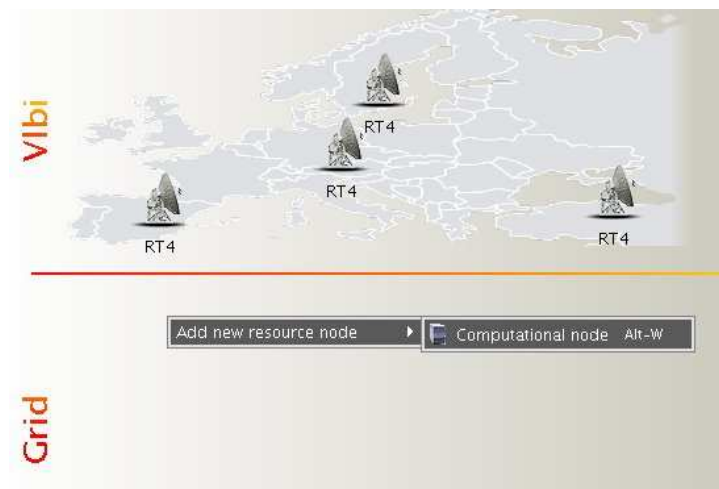


Figure 13. Adding new resource

defining their parameters. New resource can be added by right mouse click on the design area (see Figure 13.)



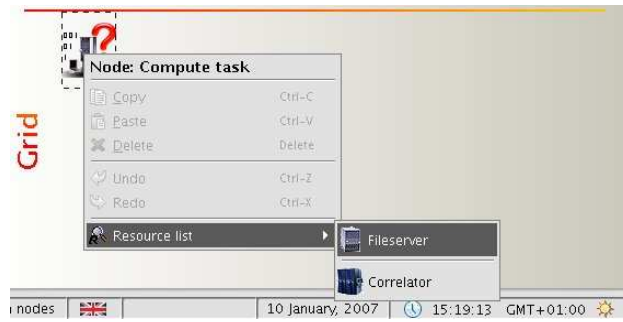


Figure 14. Adding new resource – fileserver

After a new resource icon appears on the Design Pane press the right mouse button and choose node type: file server or correlator. The choice will be indicated by icon change. When a File Server or Correlator is placed on the Design Pane it is possible to change their properties.

#### Correlator

Adding a correlator node into the design is done similar to adding a File Server. The only difference is that from the *Resource List* you need to choose *Correlator* option.

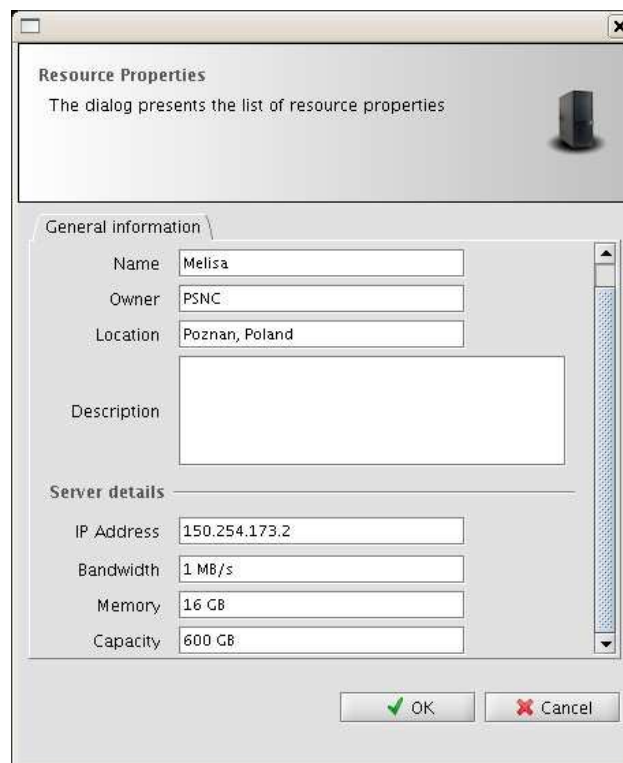


Figure 15. Sample resource properties dialog

The following figure shows the VLBI experiment with three file servers defined and one correlation node.



Figure 16. Sample VLBI experiment – without data flows

### 4.2.3. Data flows definition

The last phase of the VLBI experiment definition is to show the system data flows between the components. This can be done by connecting two nodes by an arrow. Whenever you cross the mouse

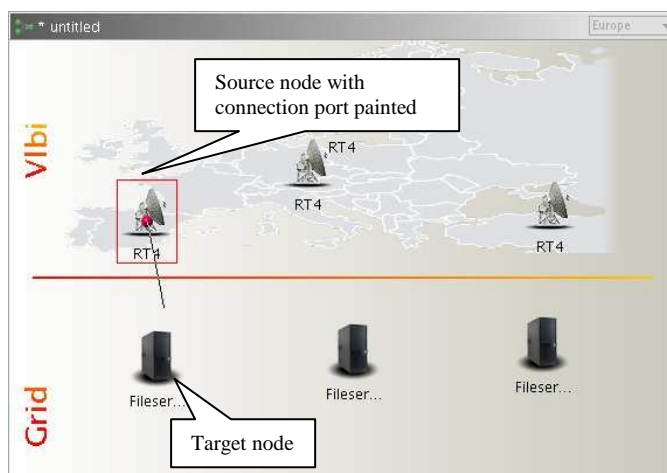


Figure 17. Connecting nodes

over a resource on the design pane, there is a dot painted in the middle of the resource icon. The resource is also encircled with border, which symbolizes its special state – ready for a connection with other node (see Figure 17).

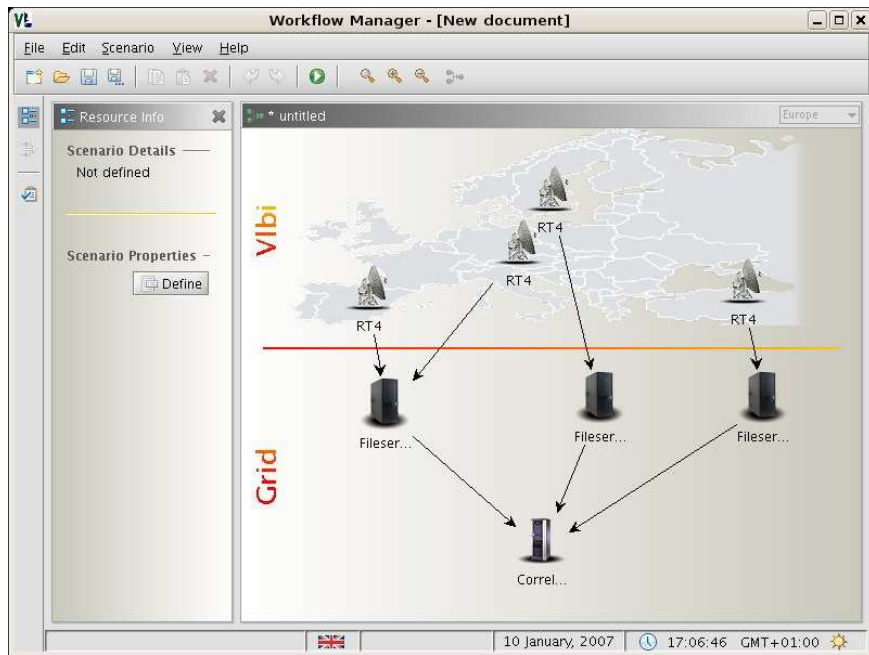


Figure 18. VLBI experiment – complete scenario

The Figure 18 presents a final eVLBI scenario, defined by the user (in our case the VLBI experiment supervisor). We have four radio telescopes taking part in the experiment. The data from the radio telescopes will be transferred to the three specified File Servers and then correlated in the GRID environment using software correlator. Such a scenario can be submitted into the system.

### 4.3. VEX file parsing and visualization

#### 4.3.1. VEX File Definition

The 'VEX-file' format (VEX = 'VLBI Experiment'), has been invented to prescribe a complete description of a VLBI experiment, including scheduling, data-taking and correlation. This includes all setup and configuration information, as well as the schedule of observations. VEX is designed to be independent of any particular VLBI data-acquisition system or correlator, and is expandable to accommodate new equipment, recording and correlation modes. Every attempt has been made to consider the requirements and concerns of both the astronomy and geodetic VLBI communities in the construction of the VEX format. Files in the VEX format are targeted at three particular types of files in the experiment process:

- The schedule file (generated by your scheduling program of choice): This VEX-format file will be completely self-contained file which details the experiment setup and execution for all sites. (The example VEX file in this document is primarily of this type.)
- The station experiment summary file, detailing the actual as-observed-experiment: Each participating station will create a VEX-format file with 'as-observed' (i.e. log) information. As of this writing, much work needs to be done to specify the details of this file.
- The Mark IV and EVN correlators (at least) are planning to use a VEX file format as the primary correlator-control file. Sample VEX file is presented on the figure below. Because of the high capacity of the file, only part of it will be presented.

```

VEX_rev = 1.5;
* SCHED vers: March 2006
* VEX/SCHED: 1.5.86
* Other versions: Sched: 6.0 Plot: 1.06 JPL-ephem: 1.01
*-----
$GLOBAL;
  ref $EXPER = N06C2;
*
*           +-----+
*           PI revision number: | 2.0000 |
*           +-----+
*-----
$EXPER;
*
def N06C2;
  exper_name = N06C2;
  exper_description = "Network Monitoring Expt";
  PI_name = "Zsolt Paragi";
  PI_email = paragi@jive.nl;
* address: JIVE
*       Postbus 2
*       7990 AA Dwingeloo
*       The Netherlands
* phone: +31-521-596536
* during obs:+31-521-596536
* fax:
*
* year, doy: 2006, 168
* date   : Sat 17 Jun 2006
* MJD    : 53903
  target_correlator = JIVE;
*
* integr_time : 2.000 s
* number_channels: 16
*
* number_antenna : 10
* cross_polarize : Yes
* weight_func   : UNIFORM
* distrib_medium : DAT
* source_pos_cat : STANDARD
* distribute_to :
*               Zsolt Paragi
*
endif;

```

Figure 19. Sample VEX file

As it was stated in the previous chapters VEX file among other things contains observation description, the list of radio telescopes and their parameters. Furthermore, WFM needs this file in order to get all the input data required for the experiment diagram creation. This file is also needed for the Correlator Control File (CCF) creation. The CCF is discussed in more detail in the next chapter.

#### 4.4. Software correlator control file

Correlator Control File is used by the Grid Software Correlator. This file contains all the settings and parameters required for the proper work of the software correlator i.e. number of nodes which will be used during computations, observation start time, observation stop time and many others details concerning experiment. The CCF file is created with great help of Workflow Manager Application. Furthermore, the description is based on the parameters found in the VEX file. The WFM pulls out all the parameters, which can be used in the CCF file and gives user an opportunity to change their values with easy and intuitive interface. The other settings, which cannot be found at the VEX file has to be submitted by user with the support of WFM interface. A part of the CCF file is presented on the figure below (see Figure 20).

```
MESSAGELVL 1
# Description: set the message level.
# Optional
# MAN
# Default value : 0
# Values:0 : only error and abort messages will appear
#   1 : 0 + higher level progress and warning messages
#   2 : 1 + lower-level warning and progress messages + information
messages
INTERACTIVE 0
# Description: Confirm to continue or not.
# Optional
# MAN
# Default value : 0
# Values:0 : run without user interaction, run automatically
#   1 : run with user interaction, only useful when MESSAGELVL > 0
RUNOPTION 1
# Description: determines which part of the application is executed
# Optional
# MAN
# Default value 1
# Values:0 : determine the file offset to get to the START. Usually this option
#   is run in combination with MESSAGELVL>0 and INTERACTIVE=1
#   1 : execute all main processing steps: offset, unpack, filter,
#   delay, correlate
```

The second part of the Correlator Control File is presented on the next page.

```

# GENERAL EXPERIMENT PARAMETER SETTINGS AND
# INFORMATION
# -----
EXPERIMENT N06C2
# Description: formal name of the experiment
# Optional
# VEX
# Default value : DefExp
# Values: any ascii character string without spaces
START    2006 168 07 32 34
# Description: Requested start time
# Compulsory
# VEX
# Values: yyyy ddd hh mm ss, where ddd=0..365, hh=0..23, mm=0..59,
ss=0..59
#     START has to be earlier then STOP
STOP     2006 168 07 32 36
# Description: Requested stop time
# Compulsory
# VEX
# Values: yyyy ddd hh mm ss, where ddd=0..365, hh=0..23, mm=0..59,
ss=0..59
#     STOP has to later then START

```

Figure 20. Sample CCF file

## 5. Software correlator in Grid environment

The scope of the FABRIC project covers the creation of VLBI software correlator, which has to be embedded in the distributed environment. The project assumes that the Grid environment will be used, which has some significant benefits, such as lower running costs of the infrastructure, and compensating for increased resource demands from the application.

In order to successfully combine the software correlator with the Grid environment, a certain system elements have to be introduced. One of them is a VLBI resource broker – the “heart” of the system, which receives the experiment description from the Workflow Manager, and directs the execution of the distributed correlation process. To manage such task, the VLBI broker interacts with some kind of lower level Grid resource broker

The Grid resource broker is responsible for the whole process of remote job submission to various batch queuing systems, clusters or resources.

### 5.1. Grid resource broker

#### 5.1.1. Overview

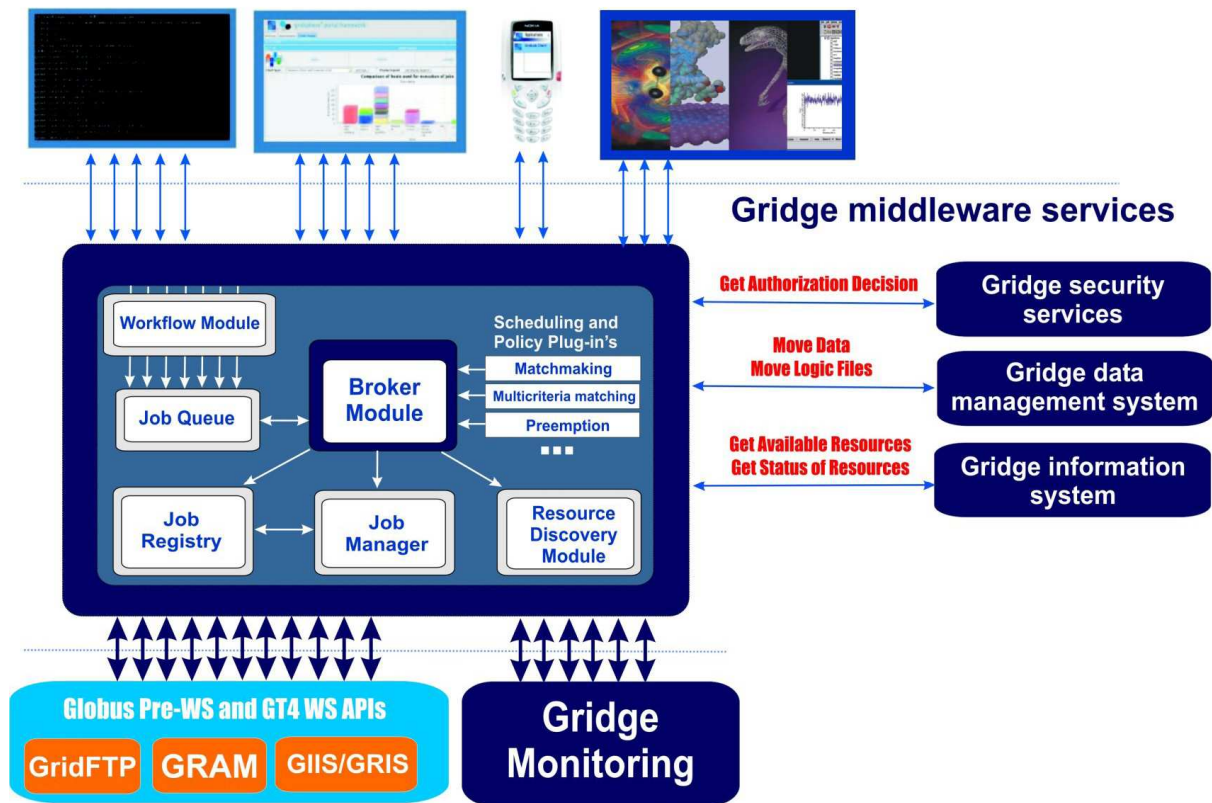
As it was stated in FABRIC deliverable DJ1.6 “eVLBI –Grid design document” the ideal candidate for the Grid resource broker should provide the following functionality:

- *Basic resource broker functionality* – this contains a basic set of functions like job submitting, job monitoring, resource discovery, etc.
- Integration with the Globus toolkit
- *Web Service interfaces* – well-defined and documented API, which allows to control and interact with the broker from the low level
- *Open Source* - possibility to add new functionality to the resource broker is required, which implies that open source is a great advantage.

In the same deliverable the detailed study of state-of-the-art resource brokers was provided, with the complete analysis of 8 different Grid-enabled brokers. After this analysis, the Grid Resource Management System (GRMS) was selected as a first choice for the resource broker. It incorporates all of the required functionality and the latest standards in managing Grid job scheduling, and it is also developed in PSNC allowing to actively participate in its modifications and extensions, making it even better suited for the Grid-VLBI purpose.

Furthermore, the first testbed environment will be crated at PSNC. It will be used to test the system components and to do a reference benchmark of the software correlation. The next stem would be to move the correlator into the “real” Grid environment, distributed among project partners, using their available resources.

## Grid portals and application tools



## Various Domains, Physical Resources and Queuing Systems

Figure 21. The GRMS

The GRMS, based on dynamic resource selection, mapping and advanced scheduling methodology, combined with feedback control architecture, deals with dynamic Grid environment and resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support. Therefore, the main goal of the GRMS is to manage the whole process of remote job submissions to various batch queuing systems, clusters or resources directly. It has been designed as an independent set of core components for resource management processes, which can take advantage of various low-level Core Services and existing. Finally, the GRMS can be considered as a robust system, which provides an abstraction of the complex grid infrastructure as well as a toolbox, which helps to form and adapt to distributing computing environments.

In order to perform job management remotely the GRMS has to stage-in and stage-out files (input files, output files, stdin, stdout, stderr) required by jobs and users before and after executions in a very efficient way. To deal with these issues GRMS is able to use Core Services taken from Globus GridFTP/GASS/FTP/RFT and also use data management middleware service – the Grid Data Management System. The idea of collaboration with this middleware service is to use more abstract operations on data and logical files rather than deals with physical file locations and low level data operations. The whole process of data management, mapping between logical and physical files, virtual collections and directories. The current release of GRMS uses basic functions invoking it's Web Service interface to convert between logical collection files and their physical locations. To start using DMS middleware service together with GRMS, DMS must be appropriately deployed in a computing environment, and proper configuration variables should be set and point to deployed service.



## 5.1.2. Job description

An XML based GRMS Job Description (GJD) language was specified to allow users to define the computational jobs and resource requests. For each task there is a section in a job description document describing resource requirements and user preferences used for dynamic resource discovery. Another section defines the application: executable, input and output files required, arguments, environment, *etc.*

GRMS job description can be divided into several parts describing the way the job should be processed as a whole. Job description starts with general properties characterizing the job in GRMS system. User has to specify a string distinguishing the job from other ones. The name of a job will be used by system as a part of final GRMS job identifier. Optionally it is possible to specify the project, a job belongs to or to specify if the processing of the job needs commitment to be started. The job consists of set of dependent task and job as well as each single task can have notes containing informal and human readable descriptions. Every task forming a job has a set of general properties. It has to have a unique identifier, that distinguishes it from other tasks.

In the example described below the job consists of two applications – Master and Slave. Slave is launched as soon as Master is running on one of Grid resources. Master controls the execution of the experiment in a few ways:

- Monitoring progress of the Slave application
- Migrating the Slave application due to decrease of application performance on current resource
- Spawning additional jobs based on some internal indicators

To serve the application calls, not only application – service communication takes place, but there are also a lot of interactions between services.

```
<grmsJob appid="MYJOB">
  <task asked="MASTER">
    <resource>
      <hostname>host1.man.poznan.pl</hostname>
    </resource>
    <executable type="single">
      <execfile name="master">
        <url>file:///bin/master</url>
      </execfile>
      <arguments>
        <value>--xf</value>
        <value>--verbose</value>
        <file name="parameters" type="in">
          <logicalId>master_param1</logicalId>
        </file>
      </arguments>
      <environment>
        <variable name="SLAVE_ID">SLAVE</variable>
      </environment>
    </executable>
  </task>
```

```

<task asked="SLAVE" commitWait="true">
  <resource>
    <applications>
      <application>Slave</application>
    </applications>
  </resource>
  <executable type="single checkpointable="true">
    <execfile name="slave">
      <url>file:///bin/slave</url>
    </execfile>
    <arguments>
      <value>2</value>
      <value>25</value>
      <file name="input1" type="in">
        <logicalId>input_from_slave1</logicalId>
      </file>
      <file name="output1" type="out">
        <logicalId>output_from_slave1</logicalId>
      </file>
    </arguments>
  </executable>
  <workflow>
    <parent triggerState="RUNNING">MASTER
    </parent>
  </workflow>
</task>
</grmsJob>

```

Figure 22. Job description

There are two tasks in the job description: MASTER and SLAVE. In a resource requirement section of Master, host name is specified directly, but for Slave, the machine to execute an application will be chosen from the list of resources that have the specified application installed locally (dynamic resource discovery). Executable description contains information about location of file, arguments of the execution, input files required and generated output. Workflow section in the SLAVE task, denotes that Slave has one parent (MASTER task) and will be executed as soon, as the Master passes to the RUNNING state. Of course it is possible to define more than two tasks and with very complex precedence constraints – everything is up to the application developer. Tasks can communicate with each other or can be entirely independent, for instance one can define two independent pairs of Master and Slave in one job description. There is only one condition: Maser has to know the identifier of Slave task, but that requirement is very simple to meet (*e.g.* using environment variable).

## 5.2. WFM – GRMS broker interface

In order to build a fully functional eVLBI system, a proper communication between GRMS and WFM is required. It must be done by introducing a new module, responsible for translation of eVLBI scenario from WFM application into the set of correlation tasks for the GRMS. This module must have some implemented logic that will allow selecting the best routing from the telescopes to the computational nodes, based on information on network monitoring modules. This module will also have the possibility of making the network bandwidth reservation for real-time eVLBI data transfers.

### 5.2.1. eVLBI resource broker

This new module will be placed between WFM application and computational grid (with GRMS as its gateway). Its main role will be to transform the data coming from WFM into XML job definition for

GRMS module. eVLBI resource broker will also communicate with network monitoring modules in order to find the optimal network routing for experiment data transfer. The networking modules will also make a bandwidth reservation if it will be necessary.

The following diagram shows the general dataflow between WFM, eVLBI broker and GRMS modules.

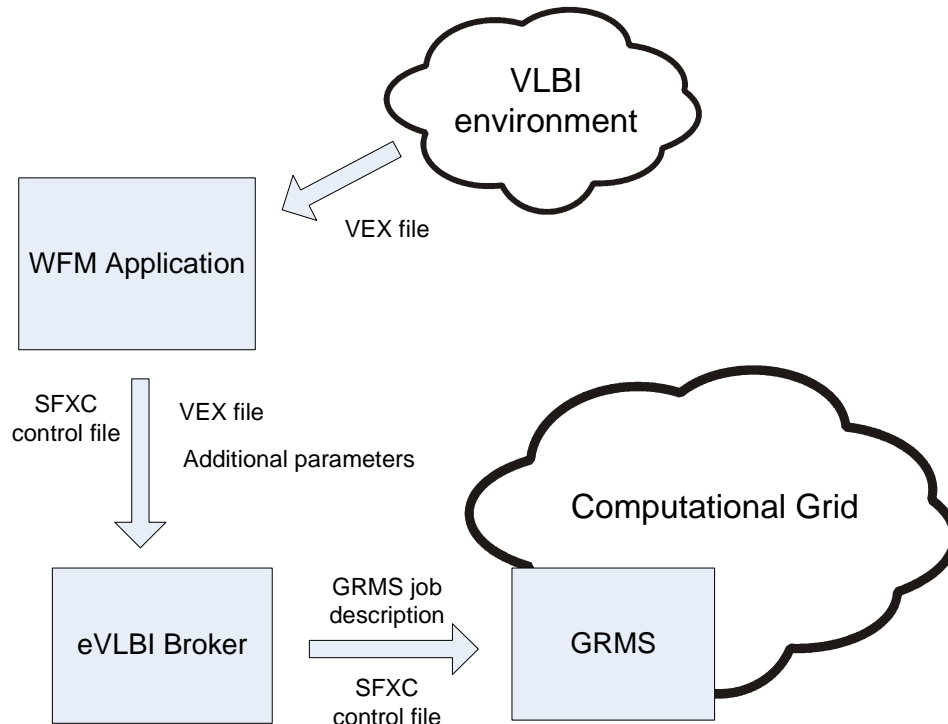


Figure 23. eVLBI communication

The whole process can be described in the following steps:

The WFM application obtains the VLBI VEX experiment description file. The file is downloaded from the VLBI server or from the local storage media (cd-rom, pen drive etc.)

The application visualizes the data and allows the user to enter additional correlation parameters and set-up the global correlation process using the graphical user interface.

The application converts the gathered data into software correlator control file, and passes it, together with VEX file and additional parameters, to the eVLBI broker module.

The eVLBI broker reads the parameters and VEX file, consults with networking modules and creates the correlation job description for the GRMS module. The job is submitted together with SFXC control file and path to the radio telescope data streams (or files). The correlation process is monitored and the user is informed of its status changes.

The additional parameters are stored in the specially designed XML file using XSD Schema technology. XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide means for defining the structure, content and semantics of XML documents. The purpose of an XML Schema is to define the legal building blocks of an XML document. We have defined several XSD schemas which stores crucial information about different aspect of the eVLBI experiment. First of all, the Resource Description Schema (RDS) [5], which main purpose is to describe resources in the eVLBI system. The schema defines a list of the resources which are all attached to the resources node. Every resource element contains the following sections:

- nodeType – contains all the information about type of the given resource i.e. type identifier, type name, etc.
- tabs - the resource properties are grouped in tabbed panes which are displayed in the JTabbedPane; each tab node contains a set of elements; each element represents a

resource property; the elements can also be put together in a group; the relationship between elements in the group can also be defined.

- monitoring - this optional section contains information about the current status of each resource in the experiment.

The RDS schema is presented on the figure blow (see Figure 24).

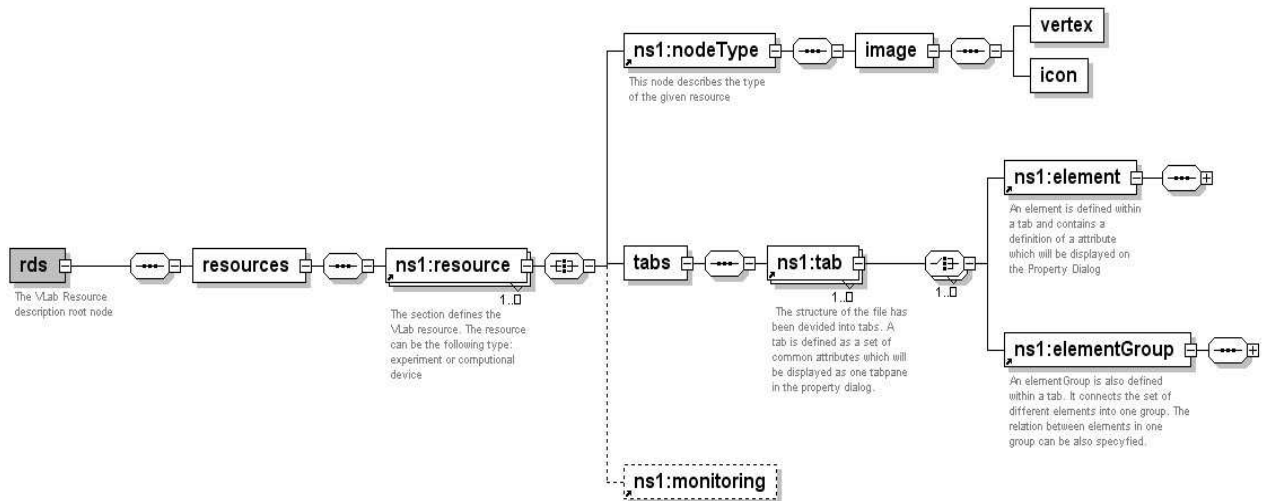


Figure 24. Resource Description Schema

Second of all, we have designed also Links Description Schema (LDS), which describes the available connections between resources. The schema structure is presented on the following figure (see Figure 25).

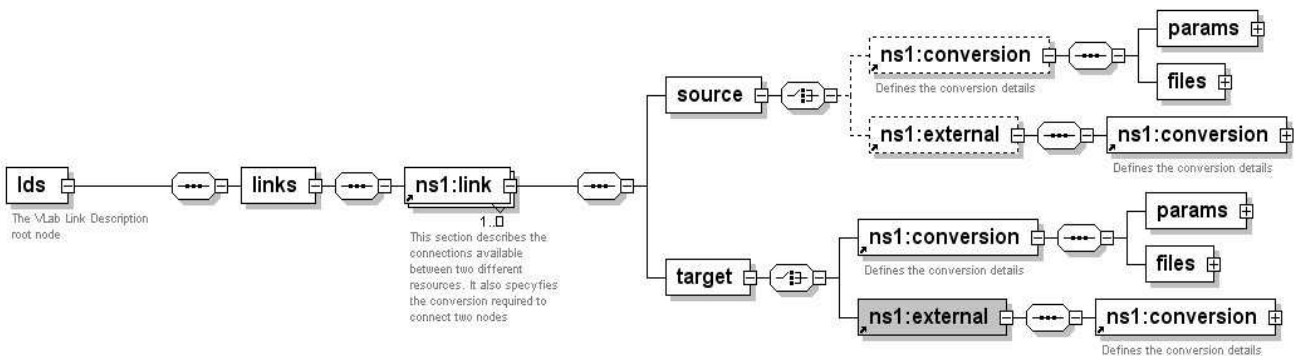


Figure 25. Links Description Schema

The link element contains connection data between two nodes. If there is an entry in the LDS definition file, given source node can be connected with given target node.

## 6. Summary

In this document, a several aspects of Grid - eVLBI interfaces were presented, including communication protocols, graphical user interface of the WFM application, the description of the VEX file, and the aspects of correlator integration with the Grid environment, and the introduction of the Grid resource brokers.

The described interfaces cover all communication layers between eVLBI and Grid, starting from data acquisition and ending with distributed correlation in the Grid environment. The interfaces between functional modules were described as well, providing the information on how it is plan to successfully integrate the software correlator with the Grid, and ensure its optimal performance by selecting the optimal Grid resources.

The first system version, the prototype, will deliver the basic functionality allowing users to correlate data coming from pre-recorded files instead of “live” telescopes (so called virtual radio telescopes). The data will be stored on one of the file servers and correlation will be done using the distributed environment provided by PSNC. The next step will involve using the computational resources provided by project participants, resulting in geographical distribution of the Grid environment.

## Definitions, abbreviations, acronyms

CCF	–	Correlator Control File.
CO	–	Central VLBI operator
DMS	–	Data Management System
eVLBI	–	Electronic Very Long Baseline Interferometry
FABRIC	–	Future Arrays of Broadband Radio-telescopes on Internet Computing
GJD	–	GRMS Job Description
GRMS	–	Gridge Resource Management System
HTTP	–	Hypertext Transfer Protocol
Java	–	Java Technology <a href="http://java.sun.com/">http://java.sun.com/</a>
LDS	–	Links Description Schema
PI	–	Principal investigator
RDS	–	Resource Description Schema
RDS	–	Resource Description Schema
SOAP	–	Simple Object Access Protocol
TO	–	Telescope operator
UDDI	–	Universal Description, Discovery and Integration
WFM	–	Workflow Manager Application
WSDL	–	Web Services Description Language
XML	–	Extensible Markup Language
XSD	–	XML Schema Definition

## References

- [1] – eVLBI – Grid design document,  
– [http://www.jive.nl/dokuwiki/lib/exe/fetch.php/fabric:grid\\_vlbi\\_design\\_v1.0\\_final.pdf?id=fabric%3Awp2\\_distributed\\_correlation&cache=cache](http://www.jive.nl/dokuwiki/lib/exe/fetch.php/fabric:grid_vlbi_design_v1.0_final.pdf?id=fabric%3Awp2_distributed_correlation&cache=cache)
- [2] – VEX File Format, <http://lupus.gsfc.nasa.gov/vex/vex.html>
- [3] – SOAP W3C Recommendation  
– <http://www.w3.org/TR/soap12>
- [4] – Web Services, <http://www.w3.org/2002/ws/>
- [5] – Lawenda M., Meyer N., Rajtar T., Okon M., Stoklosa D., Kaliszan D., Kupczyk M., Stroinski M, Workflow with Dynamic Measurement Scenarios in the Virtual Laboratory,  
– [http://vlab.psnc.pl/pub/Workflow\\_With\\_Dynamic\\_Measurement\\_Scenarios\\_In\\_The\\_Virtual\\_Laboratory.pdf](http://vlab.psnc.pl/pub/Workflow_With_Dynamic_Measurement_Scenarios_In_The_Virtual_Laboratory.pdf)
- [6] – JGoodies focuses on Java look, UI design and usability <http://jgoodies.com/>
- [7] – UDDI, [www.uddi.org](http://www.uddi.org)
- [8] – WSDL, [www.w3.org/TR/wsdl](http://www.w3.org/TR/wsdl)
- [9] – HTTP, <http://www.w3.org/Protocols/>
- [10] – eVLBI, <http://www.evlbi.org/evlbi/evlbi.html>

## Contact Information

All authors affiliation:  
Poznań Supercomputing and Networking Center  
ul. Noskowskiego 10  
61-704 Poznań, Poland

URL: <http://www.man.poznan.pl>  
Tel. (+48 61) 858-20-00  
Fax (+48 61) 852-59-54

Marcin Okoń  
Dominik Stokłosa  
Damian Kaliszan  
Tomasz Rajtar  
Norbert Meyer  
Maciej Stroiński  
Marcin Lawenda

[marcin.okon@man.poznan.pl](mailto:marcin.okon@man.poznan.pl)  
[d.stoklosa@man.poznan.pl](mailto:d.stoklosa@man.poznan.pl)  
[damian,kaliszan@man.poznan.pl](mailto:damian,kaliszan@man.poznan.pl)  
[tomasz.rajtar@man.poznan.pl](mailto:tomasz.rajtar@man.poznan.pl)  
[meyer@man.poznan.pl](mailto:meyer@man.poznan.pl)  
[stroins@man.poznan.pl](mailto:stroins@man.poznan.pl)  
[lawenda@man.poznan.pl](mailto:lawenda@man.poznan.pl)