



GRID WORKFLOW EXECUTION SERVICE - USER MANUAL

WP2

Document Filename:	KWF-WP2-D2-FIRST-GWESUserManual.doc
Work package:	WP2
Partner(s):	Fraunhofer FIRST
Lead Partner:	Fraunhofer FIRST
Document classification:	PUBLIC

Abstract: This document describes how to install and use the Grid workflow execution environment of the K-Wf Grid project, which consists of several system components, such as the Grid Workflow Execution Service (GWES), the Grid Workflow User Interface (GWUI), and the Grid Workflow Description Language (GWorkflowDL) Java library.

Delivery Slip

	Name	Partner	Date	Signature
From	Andreas Hoheisel	FIRST	06/09/2006	
Verified by	Piotr Nowakowski	CYFRONET	08/10/2006	
Approved by	Steffen Unger	FIRST	08/10/2006	

Document Log

Version	Date	Summary of changes	Author
0.1	2006-08-28	First version, based on the WP2 appendix to D2_5.2 + updates	Andreas Hoheisel + Tilman Linden
0.2	2006-08-29	Installation section updated	Andreas Hoheisel
0.3	2006-09-06	Updates regarding GWUI an GWES	Andreas Hoheisel + Tilman Linden
0.4	2006-09-26	Included corrections and remarks of the internal review. Update regarding edgeExpressions	Andreas Hoheisel
1.0	2006-10-08	QA check	Piotr Nowakowski

CONTENTS

1. COPYRIGHT NOTICE	5
2. INTRODUCTION.....	7
2.1. K-Wf GRID SETUP.....	7
2.2. INSTANT-GRID SETUP	10
2.3. ABBREVIATIONS AND ACRONYMS	11
2.4. REFERENCES AND SOURCE CODE	11
3. PRODUCT USAGE FOR SYSTEM ADMINISTRATORS.....	12
3.1. RUNNING THE PRODUCT	12
3.1.1. GWES Operating Requirements.....	12
3.1.1.1. Local System Requirements.....	12
3.1.1.2. Grid Infrastructure Requirements.....	12
3.1.2. GWES Step-by-Step Setup	13
3.1.2.1. Compilation of the gwes.war archive.....	13
3.1.2.2. Deployment of the GWES Web Service	13
3.1.2.3. Configuration of the GWES Command Line Program.....	14
3.1.2.4. Deployment of the eXist Database.....	14
3.1.2.5. Deployment of the Linuxtoolbox Example Web Services	15
3.1.2.6. Instant-Grid setup: Deployment of the Resource Matcher Web Service.....	15
3.1.2.7. Instant-Grid setup: Deployment of the Resource Updater Daemon	15
3.1.3. GWUI Operating Requirements	16
3.1.3.1. Local System Requirements.....	17
3.1.3.2. Grid Infrastructure Requirements.....	17
3.1.4. GWUI Step-by-Step Setup.....	17
4. PRODUCT USAGE FOR USERS.....	18
4.1. RUNNING THE GRID WORKFLOW EXECUTION SERVICE (GWES)	18
4.1.1. GWES Concept.....	18
4.1.2. GWES Basic Operation	21
4.1.3. GWES Web Interface.....	22
4.2. RUNNING THE GRID WORKFLOW USER INTERFACE (GWUI).....	25
4.2.1. GWUI Operating Requirements	25
4.2.2. GWUI Step-by-Step User Setup.....	25
4.2.3. GWUI Basic Operation	26
4.3. RUNNING THE GWES COMMAND LINE CLIENT (GWESCLIENT)	28
4.3.1. GWESClient Operating Requirements	28
4.3.2. GWESClient Step-by-Step User Setup.....	28
4.3.3. GWESClient Basic Operation	29
4.4. ADVANCED FEATURES.....	30
4.4.1. Supported GWorkflowDL Operations	30
4.4.2. Supported GWorkflowDL Tokens.....	32
4.4.3. Supported GWorkflowDL Edge Expressions.....	33
4.4.4. Supported GWorkflowDL Conditions.....	37
4.4.5. Supported GWorkflowDL Properties	37
5. INTERFACE REFERENCE GUIDE.....	41
5.1. GRID WORKFLOW USER INTERFACE (GWUI)	41
5.1.1. Workflow Applet.....	41
5.1.2. Task List Applet.....	45
5.2. GWES COMMAND LINE CLIENT (GWESCLIENT)	46
6. Q&A	46

7. KNOWN ISSUES	47
7.1.1. <i>GWES</i>	47
7.1.2. <i>GWUI</i>	48
7.1.3. <i>GWorkflowDL</i>	48
8. CONTACT INFORMATION AND CREDITS	49
9. THE FRAUNHOFER FIRST LICENSE AGREEMENT	50

1. COPYRIGHT NOTICE

Copyright (c) 2005 by **K-Wf Grid, Fraunhofer FIRST**. All rights reserved.

Use of this product is subject to the terms and licenses stated in the Fraunhofer FIRST license agreement. Please refer to Section 9 for details.

This research is partly funded by the European Commission IST-2002-511385 Project “K-WfGrid”.

The components described in this manual make use of external Java libraries, which have their own license agreements:

Library	Version	License	Reference
castor	0.9.7	Apache License; Intalio	http://www.castor.org/
commons-collections	3.0	Apache License	http://ws.apache.org/axis/
commons-digester	1.5	Apache License	http://ws.apache.org/axis/
commons-discovery	0.2	Apache License	http://ws.apache.org/axis/
commons-logging	1.0.4	Apache License	http://ws.apache.org/axis/
dotparser	1.0.1	GNU Lesser General Public License	http://www.netart-datenbank.org/glassbox/
glassbox	0.3.4	GNU Lesser General Public License	http://www.netart-datenbank.org/glassbox/
j2sewssoap	1.06	Free for commercial use	http://www.wingfoot.com/products.html
jaxen	1.1-beta-6	Werken Company License	http://jaxen.org/
jdom	1.0	JDOM License	http://www.jdom.org/
jug	1.1.2	GNU Lesser General Public License	http://jug.safehaus.org/
junit	3.8.1	Common Public License - v 1.0	http://www.junit.org/
addressing	1.0	Apache License	http://ws.apache.org/addressing/
Axis-url_gt	4.0.1	Apache License	http://www.globus.org/
Axis_gt	4.0.1	Apache License	http://www.globus.org/
cog-axis_gt	4.0.1	Globus Toolkit Public License, National Research Council of Canada (Contributor)	http://www.globus.org/
cog-jglobus_gt	4.0.1	Globus Toolkit Public License, National Research Council of Canada (Contributor)	http://www.globus.org/
cog-tomcat_gt	4.0.1	Globus Toolkit Public License, National Research Council of Canada (Contributor)	http://www.globus.org/
cog-url_gt	4.0.1	Globus Toolkit Public License, National Research Council of Canada (Contributor)	http://www.globus.org/
commonj_gt	4.0.1	IBM BEA License	
commons-beanutils_gt	4.0.1	Apache License	http://ws.apache.org/axis/
commons-cli	2.0	Apache License	http://ws.apache.org/axis/
concurrent_gt	4.0.1	Concurrent License	
cryptix-asn1_gt	4.0.1	Cryptix General License	http://www.globus.org/
cryptix32_gt	4.0.1	Cryptix General License	http://www.globus.org/

cryptix_gt	4.0.1	Cryptix General License	http://www.globus.org/
gemini	20060511	GNU General Public License	http://www.gridworkflow.org/kwfgid/jars/
globus_delegation_service_gt	4.0.1	Apache License	http://www.globus.org/
globus_wsrf_mds_aggregator_stubs_gt	4.0.1	Apache License	http://www.globus.org/
gomclient	20051028	K-Wf Grid License	http://www.gridworkflow.org/kwfgid/jars/
gram-client_gt	4.0.1	Apache License	http://www.globus.org/
gram-utils_gt	4.0.1	Apache License	http://www.globus.org/
gworkflowdl	1.0.5d	Fraunhofer FIRST License	http://www.gridworkflow.org/kwfgid/jars/
jaxrpc_gt	4.0.1	Apache License	http://www.globus.org/
jce-jdk13	125	Bouncy Castle License	http://www.bouncycastle.org/
Jgss_gt	4.0.1	Apache License	http://www.globus.org/
jxupdate	0.7.1	Fraunhofer FIRST License	http://www.gridworkflow.org/kwfgid/jars/
kwfgid-dr	20051021	GNU General Public License	http://www.gridworkflow.org/kwfgid/jars/
naming-common_gt	4.0.1	Apache License	http://www.globus.org/
naming-factory_gt	4.0.1	Apache License	http://www.globus.org/
naming-java_gt	4.0.1	Apache License	http://www.globus.org/
naming-resources_gt	4.0.1	Apache License	http://www.globus.org/
opensaml_gt	4.0.1	OpenSAML License, Version 1.1	http://www.globus.org/
puretls_gt	4.0.1	Claymore Systems License	http://www.globus.org/
resolver_gt	4.0.1	Apache License	http://www.globus.org/
saaj_gt	4.0.1	Apache License	http://www.globus.org/
servlet_gt	4.0.1	Apache License	http://www.globus.org/
wSDL4j_gt	4.0.1	Common Public License - v 1.0	http://www.globus.org/
wsrf_core_gt	4.0.1	Apache License	http://www.globus.org/
wsrf_core_stubs_gt	4.0.1	Apache License	http://www.globus.org/
wsrf_provider_jce_gt	4.0.1	Apache License	http://www.globus.org/
wsrf_tools_gt	4.0.1	Apache License	http://www.globus.org/
wss4j_gt	4.0.1	Apache License	http://ws.apache.org/wss4j/
xmlsec_gt	4.0.1	Apache License	http://ws.apache.org/wss4j/
log4j	1.2.8	Apache License	http://logging.apache.org/log4j/docs/
xercesImpl	2.6.2	Apache License	http://xml.apache.org/xerces2-j/
xmlParserAPIs	2.6.2	Apache License	http://xml.apache.org/xerces2-j/
Xpp3	1.1.3.3	Extreme! Lab License	http://www.extreme.indiana.edu/xgws/xsoap/xpp/

This product includes software developed by the

- K-Wf Grid Project (<http://www.kwfgid.eu/>)
- Apache Software Foundation (<http://www.apache.org/>)
- Globus Alliance (<http://www.globus.org/>)
- ExoLab Project (<http://www.exolab.org/>) (castor).
- BEA IBM (<http://dev2dev.bea.com/technologies/commonj/index.jsp>) (commonj)
- JDOM Project (<http://www.jdom.org/>) (jdom)
- University Corporation for Advanced Internet Development <http://www.ucaid.edu> Internet2 Project (opensaml)
- Indiana University Extreme! Lab. For further information please visit <http://www.extreme.indiana.edu/>. (xpp3)
- And others...

2. INTRODUCTION

The Grid workflow orchestration and execution environment consists of several system components which work together in order to assist the user in building and controlling Grid applications in a distributed environment as shown in Figure 1. This user manual describes the usage of the two components *Grid Workflow Execution Service (GWES 1.0.5)* and *Grid Workflow User Interface (GWUI 0.5.2)*. Both components can either be used in combination with further K-Wf Grid components – such as the Automatic Application Builder (AAB), the Workflow Composition Tool (WCT), the Scheduler and the Web Portal (refer to the “*K-Wf Grid Setup*” – Section 2.1) – or as an easy-to-install “stand-alone” deployment with reduced functionality (refer to the “*Instant-Grid Setup*” – Section 2.2).

2.1. K-WF GRID SETUP

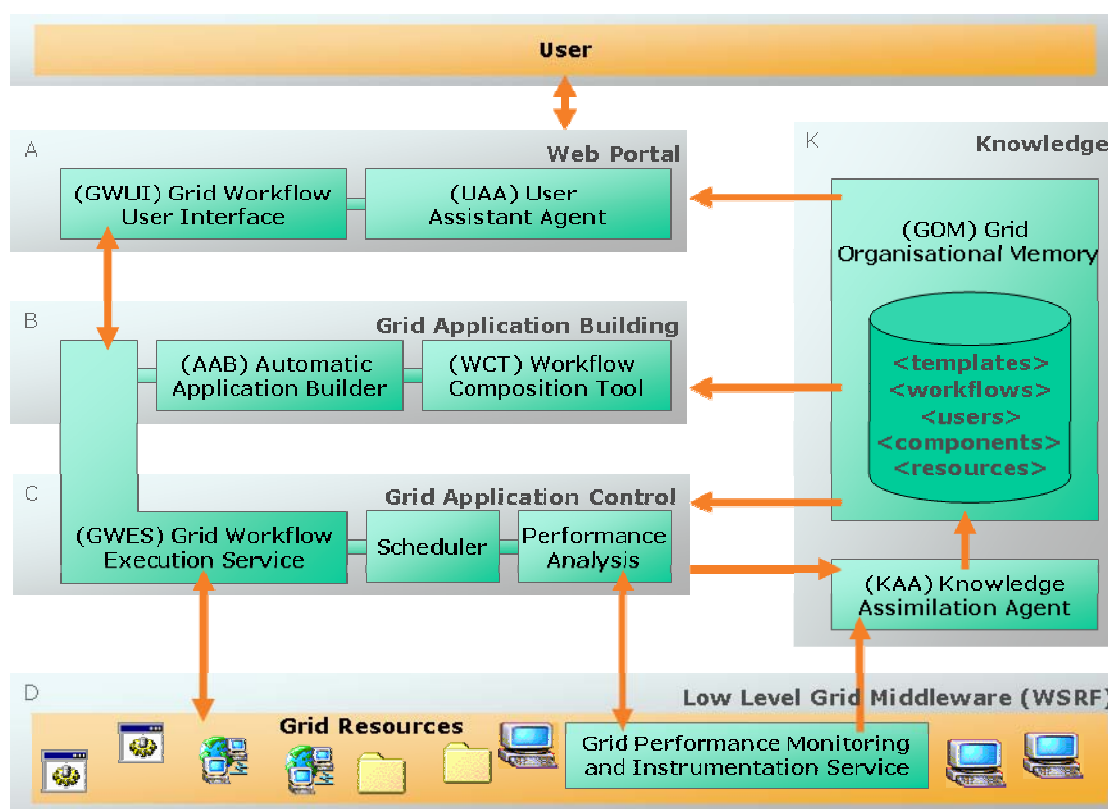


Figure 1. System architecture of the K-Wf Grid system.

The *Grid Workflow Execution Service (GWES)* is the Grid workflow enactment engine of the K-Wf Grid system, which coordinates the composition and execution process of Grid workflows. It implements a highly dynamic workflow concept based on the *Grid Workflow Description Language*

(*GWorkflowDL*). It provides interfaces to the Web Portal for user interaction and to the Low-Level Grid Middleware for the invocation of application operations. The mapping of abstract to concrete workflows is mainly delegated to further system components, such as WCT, AAB, and Scheduler (refer to the corresponding user manuals).

The main purpose of the GWES and its user interface is to:

1. define user requests,
2. initiate Grid workflows based on these user requests,
3. monitor and inspect running and finished workflows,
4. interact with the workflow building and execution process
5. download and upload files from/to the Grid, and
6. provide assistance during the orchestration of workflows.

Therefore the GWES possesses the following features:

- Analysis and verification of the workflow descriptions (delegated to the *GWorkflowDL* Java library).
- Can act as a workflow engine, cycling through the workflow graph, searching for activated transitions, evaluating arbitrary conditions, and triggering related activities.
- Detection of conflicts within the workflow and delegation of workflow building decisions to the user via the Grid Workflow User Interface (GWUI) in case of conflicts or annotations that request user decisions.
- Invocation of the Workflow Composition Tool (WCT) if abstract workflow elements need to be mapped onto operations of Web Service classes (refer to separate user manual).
- Invocation of the Automatic Application Builder (AAB) if operations of Web Service classes need to be mapped onto lists of concrete Web Service operations (refer to separate user manual).
- Invocation of the Scheduler if a list of concrete Web Service operations needs to be mapped onto a single instance of Web Service operation (refer to separate user manual).
- Reliable invocation of target Web Service operations.
- Transfer of data from one Web Service to another as specified in the Grid workflow description.
- Control the execution of the Web Service operations and throwing workflow-related events to the Event System.
- Execution of remote command line programs using WS-GRAM.
- File transfer by means of RFT

Most of the existing Grid workflow management implementations have three main drawbacks: They don't support dynamic changes of the workflow structure during runtime, they do not allow real interplay between the workflow orchestration and the workflow execution process, and they do not support several levels of abstraction within one workflow description language. The K-WfGrid

workflow orchestration and execution environment aims at overcoming these drawbacks by introducing an innovative workflow management concept.

The approach within K-WfGrid defines a modular architecture for executing workflow applications on the Grid. Most existing approaches to workflow execution wrongly assume a static Grid architecture, consisting of a set of reliable distributed resources. The novelty of this approach comes from taking into consideration the dynamic nature of the Grid, consisting of a set of unreliable and unpredictable resources. The dynamic behaviour of the Grid requires complex scheduling and fault tolerance techniques combined with application and site monitoring. The K-WfGrid system implements a dynamic scheduling approach, where the workflow is scheduled lazily, by using just-in-time mechanisms. An event infrastructure is provided, which can be used to dynamically control and monitor the workflows.

The GWUI (Grid Workflow User Interface) is a client interface for the GWES. The GWUI mainly implements the client part of interfaces 1 and 2 of the Workflow Reference Model of the WfMC (<http://www.wfmc.org/standards/model.htm>):

- User interface for process definition
- Workflow client application

Additionally it has capabilities for:

- Monitoring of workflows
- User interaction with workflows (start, pause, stop, cancel, modify, ...)

For details about the internals of the Grid Workflow Execution Service please refer to the separate Developer Manual (GWESDeveloperManual).

2.2. INSTANT-GRID SETUP

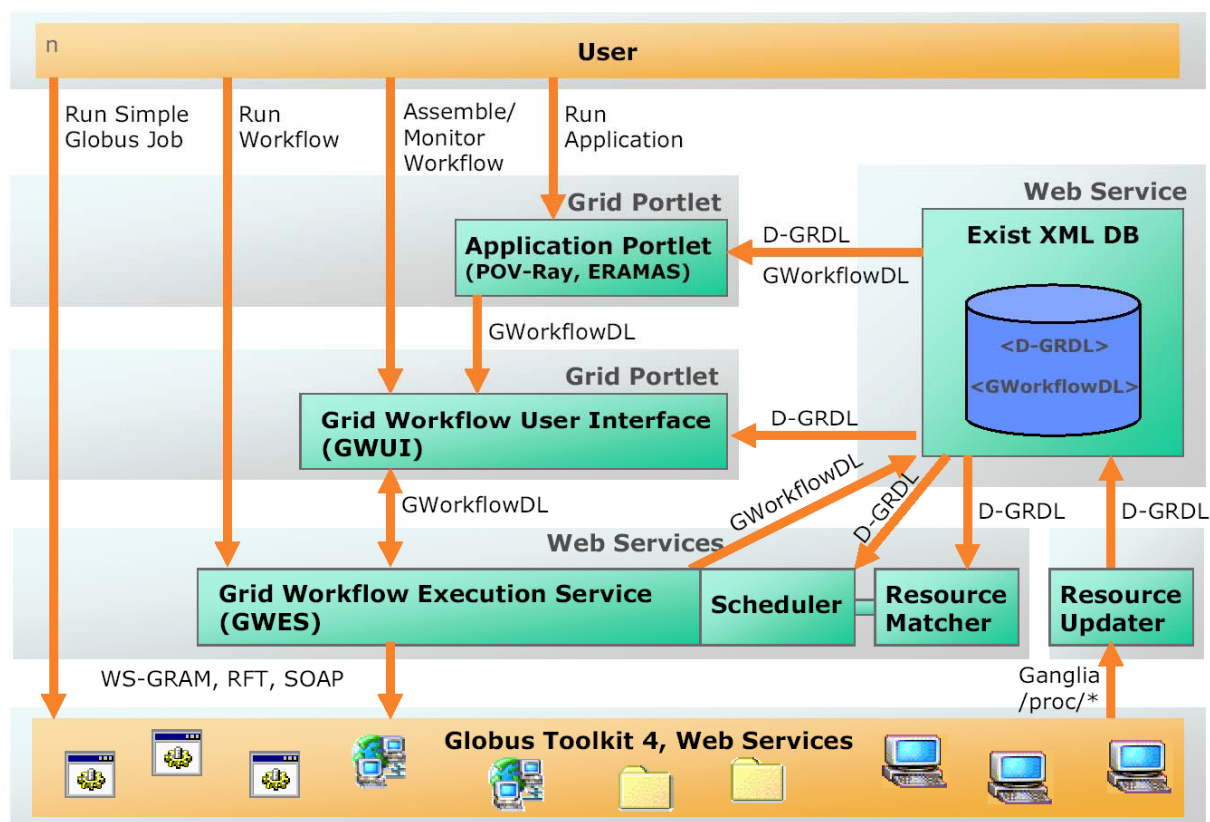


Figure 2. System architecture of the Instant-Grid workflow management system.

Within the K-Wf Grid setup, the GWES depends on further services, in order to provide full automation of the workflow orchestration process. An alternative GWES setup with reduced functionality, using only few components of K-Wf Grid, is realized within the Instant-Grid project (<http://instant-grid.de/>). The Instant-Grid project aims at developing a Knoppix-based live-CD bundled with a pre-configured Grid environment based on the Globus Toolkit. In order to provide Grid workflow management features, it reuses some components developed by Fraunhofer FIRST within the K-Wf Grid project.

Figure 2 shows the architecture of the Instant-Grid setup, where an XML database replaces the K-Wf Grid Organizational Memory. A resource matcher service maps abstract workflows onto service candidates as replacement for the AAB component. While the K-Wf Grid setup takes into account Web and Grid Service invocation, the Instant-Grid setup focuses on the remote execution of programs using WS-GRAM.

The user may use the Instant-Grid setup as a starting point for evaluating the workflow management environment without having to provide the ontologies for the services and applications. This is the reason why we include some documentation related to the Instant-Grid setup in this manual, although parts of the work has not been performed in the scope of the K-Wf Grid project.

2.3. ABBREVIATIONS AND ACRONYMS

AAB	Automatic Application Builder (system component)
CTM	Coordinated Traffic Management (pilot application)
CVS	Concurrent Versions System (software development tool)
D-GRDL	D-Grid Resource Description Language (description language)
ERP	Enterprise Resource Planning (pilot application)
FFSC	Flood Forecasting Simulation Cascade (pilot application)
GEMINI	Generic Monitoring Infrastructure (system component)
GOM	Grid Organizational Memory (system component)
GS	Grid Service (remote procedure technology)
GWES	Grid Workflow Execution Service (system component) – <i>pronounced “ge-wes”</i>
GWorkflowDL	Grid Workflow Description Language (description language)
GWUI	Grid Workflow User Interface (system component) – <i>pronounced “gwui”</i>
RFT	Reliable File Transfer (file transfer technology)
SOAP	Protocol for exchanging XML-based messages (formally know as “Simple Object Access Protocol”) (protocol standard)
WCT	Workflow Composition Tool (system component)
WS	Web Service (remote procedure technology)
WSDL	Web Services Description Language (description language) – <i>pronounced “wiz-dull”</i>
WS-GRAM	Web Service Grid Resource Allocation and Management (remote procedure technology)
WSRF	Web Service Resource Framework (standard)
XML	Extensible Markup Language (standard)

2.4. REFERENCES AND SOURCE CODE

Further online information about the Grid Workflow Execution Service (GWES) and the Grid Workflow User Interface (GWUI) is available at the following links:

- Source and binary software distributions: <http://www.gridworkflow.org/kwfgrid/distributions/>
- GWES software development site: <http://www.gridworkflow.org/gwes/>
- GWES source Xref: <http://www.gridworkflow.org/kwfgrid/gwes/docs/xref/>
- GWES Java Docs: <http://www.gridworkflow.org/kwfgrid/gwes/docs/apidocs/>
- GWES CVS (password protected): <http://cvs.ui.sav.sk/cgi-bin/cvsweb.cgi/kwfgrid/gwes/>
- GWUI software development site: <http://www.gridworkflow.org/kwfgrid/gwui/docs/>
- GWUI source Xref: <http://www.gridworkflow.org/kwfgrid/gwui/docs/xref/>
- GWUI Java Docs: <http://www.gridworkflow.org/kwfgrid/gwui/docs/apidocs/>
- GWUI CVS (password protected): <http://cvs.ui.sav.sk/cgi-bin/cvsweb.cgi/kwfgrid/gwui/>

3. PRODUCT USAGE FOR SYSTEM ADMINISTRATORS

The normal way to install and use the Grid Workflow Execution Service (GWES) is to deploy it as a regular Web Service, e.g., by means of an Apache Tomcat Web Service container. Other system components can then use the GWES by invoking its Web Service method calls. Another possibility is to use the GWES directly by means of its Java API, however this is not within the scope of this manual.

The Grid Workflow User Interface (GWUI) is deployed as a Java Applet, which may be encapsulated by means of a portlet within a GridSphere portal (refer to the separate portal user and developer manual for details). In addition to the main Java Applet, the GWUI has several workflow management servlets, which can be deployed using Apache Tomcat.

3.1. RUNNING THE PRODUCT

The following sections describe how to install and to run the GWES and GWUI from the perspective of a system administrator.

3.1.1. GWES Operating Requirements

The GWES is implemented in Java and deployed as a Web Service; so the main requisite to operate the Grid Workflow Execution Service is a Java Virtual Machine and a Web Service container. The GWES has been tested on Linux, however in principle it could also work using Microsoft Windows™ as operating system. The detailed system and Grid requirements are given below.

3.1.1.1. Local System Requirements

- Java 1.5.x (tested with Sun JDK5.0; Java HotSpot™ version 1.5.0_05-b05)
- An installed Web Service container (tested with Apache Tomcat 4.1 and 5.5.17). The environment variable `{ CATALINA_HOME }` should point to the Apache Tomcat home directory.
- 35MB of free disk space within the webapp folder of the Web Service container in order to deploy the GWES
- 60MB of free disk space within the webapp folder of the Web Service container in order to deploy the eXist XML database (can also be deployed on a separate computer).
- Optional: 75MB of free disk space in order to download and compile the GWES sources
- Optional: Apache Maven 1.0.2 (for compiling the sources and creating the war archive)
- Optional: Online access to the maven jar archive at <http://www.ibiblio.org/maven/> and to the K-WfGrid jar archive at <http://www.gridworkflow.org/kwfgrid/jars/> (for compiling the sources and creating the war archive)
- The GWorkflowDL Java library will be downloaded automatically when compiling the sources using maven, so you do not need to install it separately

3.1.1.2. Grid Infrastructure Requirements

- If you want to invoke Grid Services, WS-GRAM, or RFT activities you need a Globus Toolkit 4 installation on the computer, where the GWES is to be deployed (tested with Globus Toolkit 4.0.1 and 4.0.2). You also need a valid Grid certificate or credential belonging to the user who started Apache Tomcat.

- For testing and demonstration you need some application Web Services and their corresponding WSDL documents accessible via HTTP. In order to create some demo Web Services you may want to use our Linux Toolbox examples, which can be downloaded at <http://www.gridworkflow.org/kwfguid/distributions/linuxtoolbox.war> (refer to the installation steps below and to <http://www.gridworkflow.org/kwfguid/linuxtoolbox/docs/> for documentation)
- *K-Wf Grid setup*: Access to a WCT service if you need to automatically map user requests onto abstract workflows (refer to WCT manuals).
- *K-Wf Grid setup*: Access to an AAB service if you need to automatically map abstract workflows onto Web Service candidates (refer to AAB manuals).
- *K-Wf Grid setup*: Access to a Scheduler service if you need to optimize the selection of the Web Service candidates (refer to Scheduler manuals).
- *K-Wf Grid setup*: Working GEMINI environment if you want to propagate performance monitoring events to the performance analysis services (refer to the corresponding separate manuals for details)

3.1.2. GWES Step-by-Step Setup

The following steps are required for the deployment of the GWES under Debian Linux (sarge) and Apache Tomcat 4.1 (tested with Linux kernel version 2.6.8-2-k7-smp). Other operating systems and Web Service containers may require similar steps. Please replace `localhost:8080` by the host name and port of the server, where you install the services and `<version>` by the latest software version (e.g. `1.0.5` for the current GWES release) .

3.1.2.1. Compilation of the *gwes.war* archive

You may either download the current version of the `gwes.war` file directly from the software distribution download site at <http://www.gridworkflow.org/kwfguid/distributions/gwes.war> or build it by your own from the source distribution, following the instructions below:

1. Install maven 1.0.2 (<http://maven.apache.org/maven-1.x/start/download.html>)
2. Checkout the GWES sources (module `kwfguid/gwes`) from the K-Wf Grid CVS or download the latest source distribution (`gwes-<version>.tar.gz`) from <http://www.gridworkflow.org/kwfguid/distributions/> and unpack the sources:

```
tar -xzf gwes-<version>.tar.gz
ln -s gwes-<version> gwes
```
3. Change to the directory `gwes/src/conf`
4. Edit the configuration files `gwes.properties`, `rft.properties`, and `log4j.properties` due to your needs
5. Change to the directory `gwes`
6. invoke "maven war" in order to create `gwes/target/gwes.war`. It may be necessary to exclude some of the JUnit tests that make use of external Web Services by editing the `maven.project.xml` file in the `gwes` folder.

3.1.2.2. Deployment of the GWES Web Service

1. Upload and install the `gwes.war` into your Tomcat container, e.g., by using your web browser and the Tomcat manager, often available at <http://localhost:8080/manager/html>

2. If Tomcat has the Java security enabled, copy the file `${CATALINA_HOME}/webapps/gwes/05gwes.policy` to the directory `/etc/tomcat4/policy.d/` (Debian sarge) or append it to the existing `${CATALINA_HOME}/conf/catalina.policy` file (standard Tomcat installation)
3. Edit the configuration files `gwes.properties`, `rft.properties`, and `log4j.properties` (directory `${CATALINA_HOME}/webapps/gwes/WEB-INF/classes`) due to your needs
4. Restart Tomcat
5. Go with your web browser to <http://localhost:8080/gwes> to get an overview and to <http://localhost:8080/gwes/servlet/GWESConfigurationServlet> to test the GWES installation. If the `GWESConfigurationServlet` returns “The GWES is operating”, then the GWES has been successfully deployed. Logging messages are available at `${CATALINA_HOME}/logs/catalina.out`
6. If you want to use the GWES to invoke WS-GRAM, RFT or other Grid Services, the user who started Tomcat needs a valid Grid credential. This can be obtained by invoking the Globus Toolkit command line program “`grid-proxy-init`”. All Grid jobs are then invoked in the name of the user who started Tomcat. Globus Toolkit 4 logging is normally available at `/usr/local/globus-<version>/var/container.log`.

3.1.2.3. Configuration of the GWES Command Line Program

The `gwes.war` also contains a GWES command line client, which internally uses Axis libraries in order to interact with the GWES Web Service.

1. Make the shell scripts executable:
`chmod +x ${CATALINA_HOME}/webapps/gwes/bin/*.sh`
2. Export the `GWES_HOME` environment variable and add the bin directory to your path:
`export GWES_HOME=${CATALINA_HOME}/webapps/gwes`
`export PATH=$PATH:${CATALINA_HOME}/webapps/gwes/bin`
3. Test the GWES command line program:
`GWESClient.sh -gwes http://localhost:8080/gwes \`
`-is ${CATALINA_HOME}/webapps/gwes/examples/\`
`instant-grid/gworkflowdl_test.xml -monitor`

3.1.2.4. Deployment of the eXist Database

The GWES uses the eXist database for storing workflow checkpoints and further workflow data. Within the Instant-Grid setup the resource updater stores additional resource descriptions to the eXist database. The resource descriptions are later used by the resource matcher and the scheduler.

1. Download the latest `exist-<version>.war` from <http://exist.sourceforge.net/index.html#download> (tested with `eXist-1.0b2-build-1107.war` and `exist-1.1rc2-newcore.war`)
2. rename it to `exist.war`:
`mv exist-<version>.war exist.war`
3. Upload and install the `exist.war` into your Tomcat container, e.g., by using your web browser and the Tomcat manager
4. Configure the eXist database regarding the documentation at <http://exist.sourceforge.net/>. The default login to the eXist database is “admin” without password. This is also the default for the `gwes.properties` file.

5. The eXist database is deployed at <http://localhost:8080/exist/>.

3.1.2.5. Deployment of the Linuxtoolbox Example Web Services

The Linuxtoolbox provides some example web services that wrap Linux command line programs, such as `sort` and `tail`. In addition, the Linuxtoolbox provides a graph layout service called GraphVizWS which is required by the GWUI, in order to display the workflows.

1. Install the Linux programs `/usr/bin/sort`, `/usr/bin/tail` and `/usr/bin/dot`. Within Debian this is achieved by:
`apt-get install coreutils graphviz`
2. Download `linuxtoolbox.war` from <http://www.gridworkflow.org/kwfgrid/distributions/linuxtoolbox.war> or build it with “maven war” from the latest source distribution.
3. Upload and install the `linuxtoolbox.war` into your Tomcat container, e.g., by using your web browser and the Tomcat manager
4. If Tomcat has the Java security enabled, copy the file `${CATALINA_HOME}/webapps/linuxtoolbox/05linuxtoolbox.policy` to the directory `/etc/tomcat4/policy.d/` (Debian) or append it to the existing `${CATALINA_HOME}/conf/catalina.policy` file (standard Tomcat installation)
5. Test the installation by opening <http://localhost:8080/linuxtoolbox/happyaxis.jsp>. The list of deployed web services is available at <http://localhost:8080/linuxtoolbox/servlet/AxisServlet>.

3.1.2.6. Instant-Grid setup: Deployment of the Resource Matcher Web Service

The Resource Matcher replaces the AAB in the reduced Instant-Grid setup. In the full K-Wf Grid setup it is not required to install the Resource Matcher service.

1. Download `resmatch.war` from <http://www.gridworkflow.org/kwfgrid/distributions/resmatch.war>.
2. Upload and install the `resmatch.war` into your Tomcat container
3. Modify `${CATALINA_HOME}/webapps/resmatch/ResourceMatcher.properties` due to your needs.

The resource matcher web service accesses the eXist database in order to retrieve matching hardware and software resource descriptions. The hardware resource descriptions are included and updated automatically by means of the resource updater daemon (refer to next section), whereas the software resource descriptions currently have to be stored manually to the eXist database. Figure 3 and Figure 4 show examples of resource descriptions which use the D-GRDL format. Example applications and the corresponding resource descriptions are also available in the following directories:

```
${CATALINA_HOME}/webapps/gwes/examples/programexecution/cat  
${CATALINA_HOME}/webapps/gwes/examples/programexecution/povray
```

3.1.2.7. Instant-Grid setup: Deployment of the Resource Updater Daemon

The resource updater is a daemon, which is installed on all computers, in order to update the resource descriptions (e.g. CPU load, CPU count, etc.) within the eXist database in regular intervals. When the daemon starts, it creates a new resource description, which will then be updated using the XUpdate standard.

1. Download the latest `resourceupdater-<version>.zip` from <http://www.gridworkflow.org/kwfgrid/distributions/>.

2. Unzip the resource updater on all computers with Globus Toolkit participating in the Grid:
`unzip -d /usr/local/ resourceupdater-<version>.zip`
3. Configure the resource updater:
`cd /usr/local/resourceupdater/bin`
`vi ResourceUpdater.properties`
4. Install the init.d script (the resource updater will be started as user "globus"):
`cp /usr/local/resourceupdater/bin/resourceupdater /etc/init.d/`
`update-rc.d resourceupdater defaults`
5. Start the daemon:
`/etc/init.d/resourceupdater start`
Logging is available in `/var/log/resourceupdater.log`

```
<?xml version="1.0" encoding="UTF-8"?>
<resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
    "http://www.gridworkflow.org/kwfggrid/src/xsd/instantgrid-d-grdl.xsd"
  uri="hardware:server">
  <ofClass uri="urn:dgrdl:hardware"/>
  <name>server</name>
  <description>Hardware server</description>
  <provides>
    <resourceRef uri="software:cat-fhrg"/>
    <resourceRef uri="software:povray-3-5"/>
    <resourceRef uri="software:povray-post-convert-0-1"/>
    <resourceRef uri="software:povray-post-merge-0-1"/>
    <resourceRef uri="software:povray-post-statistics-0-1"/>
    <resourceRef uri="software:povray-pre-0-1"/>
  </provides>
  <simpleProperty id="WSRF.ManagedJobFactoryService" type="uri" unit="">
    https://server:8443/wsrf/services/ManagedJobFactoryService
  </simpleProperty>
  <simpleProperty id="cpucount" type="int" unit="pcs">2</simpleProperty>
  <simpleProperty id="cpuload" type="float" unit="percent">0.3</simpleProperty>
</resource>
```

Figure 3. Example of a D-GRDL hardware resource description for the Instant-Grid setup.

```
<?xml version="1.0" encoding="UTF-8"?>
<resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
    "http://www.gridworkflow.org/kwfggrid/src/xsd/instantgrid-d-grdl.xsd"
  uri="software:povray-3-5">
  <ofClass uri="urn:dgrdl:software:povray"/>
  <name>povray</name>
  <description>Persistence of Vision(tm) Ray Tracer Version 3.5.x</description>
  <simpleProperty id="executable" type="string" unit="">
    /usr/local/workflows/povray/povray.sh
  </simpleProperty>
</resource>
```

Figure 4. Example of a D-GRDL software resource description for the Instant-Grid setup.

3.1.3. GWUI Operating Requirements

The GWUI is implemented in Java and deployed as a Java Applet inside a GridSphere portlet; so the main requisite to operate the Grid Workflow User Interface is Web Browser with a Java Plugin on the client side and a GridSphere portal or a regular Web Server to host the applet on the server side. The detailed system requirements are given below.

3.1.3.1. Local System Requirements

- Web Browser with Java Plugin \geq 1.4.x on client computer
- Online access to the Grid portal and to the GWES Web Service from the client computer.

3.1.3.2. Grid Infrastructure Requirements

- GridSphere-based portal installation with online access to the GWES Web Service (refer to portal manual for details)
- User Assistant Agent installed (refer to UAA manual)

3.1.4. GWUI Step-by-Step Setup

1. Checkout the GWUI sources (module kwfgrid/gwui) from the K-Wf Grid CVS or download the latest source distribution from <http://www.gridworkflow.org/kwfgrid/distributions/> if you want to compile the applet by your own. A compiled and signed version of the GWUI library is also available as part of the GWES web service in the directory:
\${CATALINA_HOME}/webapps/gwes/gwui
2. Either you directly use the servlets provided by the GWES (e.g. <http://localhost:8080/gwes/servlet/GWUIServlet>) in order to open the applet or copy the directory “\${CATALINA_HOME}/webapps/gwes/gwui” to the GridSphere workflow portlet location on the web server (refer to portal manual for details) or to another location.
3. The GWUIServlet returns an HTML page with the embedded applet. You also can provide your own HTML, e.g.:

```
<html>
<head>
  <title>GWUI 0.5.2</title>
</head>
<body>
<applet code="net/kwfgrid/gwui/applet/IGApplet.class"
  codebase="./gwui"
  archive="signedjar/gwui-0.5.2.jar,lib/dotparser-1.0.1.jar,lib/glassbox-
0.3.4.jar,lib/gworkflowdl-1.0.6b.jar,lib/jaxen-1.1-beta-6.jar,lib/jdom-
1.0.jar,lib/jug-1.1.2.jar,lib/log4j-1.2.8.jar,lib/j2sewsoap-1.06.jar,lib/jxupdate-
0.7.1.jar,lib/xpp3-1.1.3.3.jar,lib/xercesImpl-2.6.2.jar,lib/xmlParserAPIs-2.6.2.jar"
  width="1000" height="800">
  <param name="user.id" value="gwui-0.5.2">
  <param name="workflow.id" value="">
  <param name="service.gwes.uri" value="http://localhost:8080/gwes/services/GWES">
  <param name="service.graphviz.uri"
    value="http://localhost:8080/linuxtoolbox/services/GraphVizWS">
  <param name="gworkflowdl.xsd.path" value="http://localhost:8080/gwes/xsd">
</applet>
</body>
</html>
```

4. PRODUCT USAGE FOR USERS

While the previous chapter described the installation and administration of the Grid Workflow Execution Service (GWES) and its components, this chapter now focuses on how users can interact with the GWES in order to manage their workflows in a Grid environment. Section 4.1 describes the main concept of the GWES and explains its basic features. End users normally access the GWES by means of the Grid Workflow User Interface (GWUI) which is deployed as a Java Applet in a Grid portal and which we describe in Section 4.2 whereas Section 4.3 focuses on the GWES command line client, which is an alternative way to interact with the GWES without a graphical user interface. Detailed information about the interfaces is also available in the interface reference guide in Chapter 5.

4.1. RUNNING THE GRID WORKFLOW EXECUTION SERVICE (GWES)

The Grid Workflow Execution Service is normally deployed as a Web Service, so all interaction is done remotely using a remote procedure call mechanism or via servlets that can be accessed using a web browser. This section describes the main concept of the GWES and its basic operation.

4.1.1. GWES Concept

The basis for the workflow management is the Grid Workflow Description Language (GWorkflowDL), which is a Petri Net-based standard for describing workflows using XML. For the features and the specification of the Grid Workflow Description Language please refer to the GWorkflowDL software development site at <http://www.gridworkflow.org/gworkflowdl/>.

The GWorkflowDL and the GWES support several levels of workflow abstraction that are displayed in Figure 5. An initial input workflow provided by the user or by the user assistant may possess different abstraction levels, ranging from an abstract user requests to the concrete (executable) workflow which can be invoked directly on the available Grid resources. The supported abstraction levels are:

- **User Request (Red):** The user request represents an abstract operation which has still not been mapped onto potential workflows.
- **Abstract Workflow (Yellow):** An abstract (non-executable) workflow consists of operations of Web Service or program execution classes. The WCT automatically composes abstract workflows from the user requests, if the corresponding ontologies are represented within the Grid Organizational Memory (GOM). The user can also directly provide abstract workflows without using the WCT (refer to *Instant-Grid setup*).
- **Workflow of Service Candidates (Blue):** Consists of lists of Web Service or program execution candidates, which match the Web Service (or program execution) classes. The mapping is done by the AAB (or the resource matcher).
- **Workflow of Service Instances (Green):** Consists of concrete (executable) instances of Web Service operations or program executions. The selection of the optimal instance out of the list of candidates is delegated to the Scheduler.
- **Control Flow (Black):** Black transitions within the workflow denote a control flow (separate from the data flow) which is not connected to any real operation.

Only green or black workflow nodes can directly be executed by the GWES. If the workflow contains red, yellow or blue nodes, the WCT, AAB, Resource Matcher, or Scheduler is invoked, in order to

refine the workflow. If these components are not able to provide a solution, then the GWES will suspend the workflow and the user has to refine the workflow.

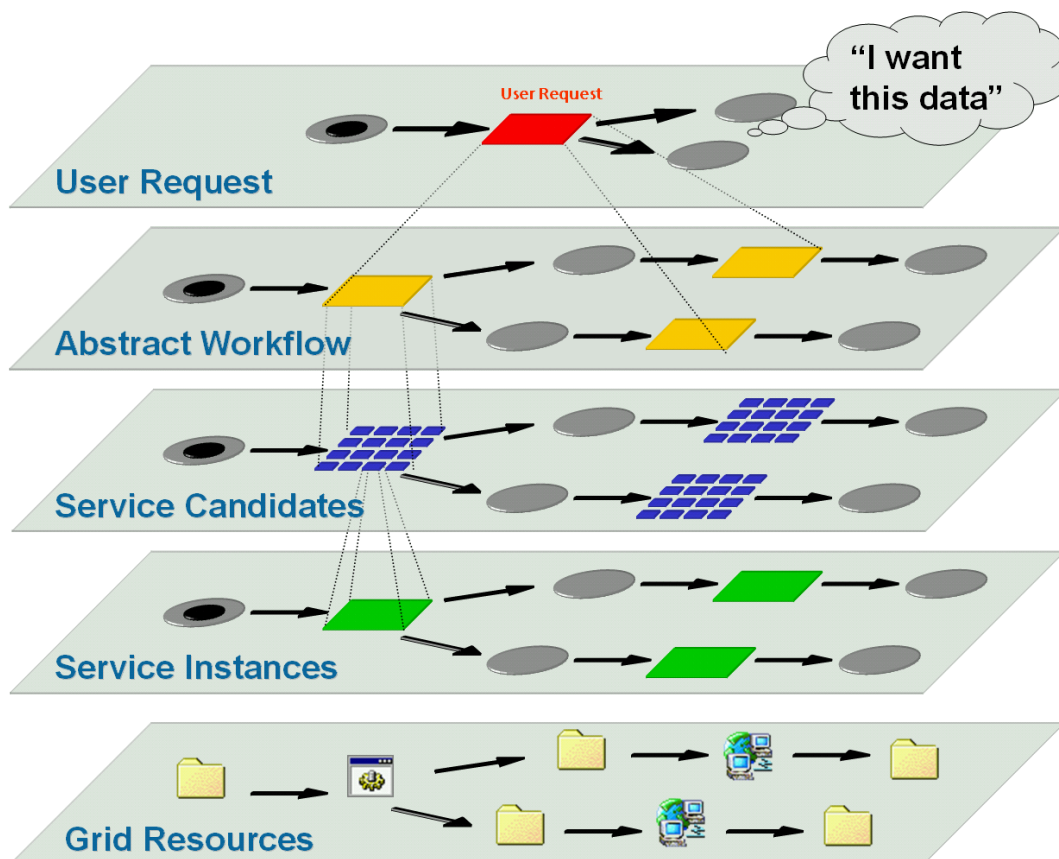


Figure 5. The abstraction levels that are supported by the GWorkflowDL and the GWES.

Figure 6 shows the graphical representation of an example workflow which contains an explicit loop. Figure 7 lists the corresponding GWorkflowDL document. This workflow uses an external “Calculator” service in order to increment the loop counter by means of the method “plus”. The token on the place which is connected with the edge expression “b” represents the increment of the loop. The conditions of the two black transitions define the exit criterion of the loop.

More details about the technical concept behind the GWES are available in the developer manual (GWESDeveloperManual).

4.1.2. GWES Basic Operation

This chapter describes the most important GWES operations. Normally the user does not directly interface the GWES; nevertheless it is important to understand the sequence of operations which are necessary to run a workflow. Workflows (as well as workflow activities) possess the states UNDEFINED, INITIATED, RUNNING, ACTIVE, SUSPENDED, COMPLETED and TERMINATED. The state of a workflow can be modified by invoking the methods which are deployed by the GWES:

Initiate: The initiate method initiates a new workflow by means of its workflow description (GWorkflowDL document) and the user identifier, which is required for authorization, accounting and logging issues. During initialization, the GWES analyses the workflow description and stores a first snapshot of the workflow to the database. The initiate method returns a unique workflow identifier as reference to this workflow instance.

Start: With the workflow identifier, the user can start the processing of the workflow by invoking the start method. The state of the workflow switches to RUNNING. If there are one or more instances of activities related to the workflow, then the state switches to ACTIVE. If there does not exist any further activity to invoke and the workflow has not been aborted yet, then the state switches to COMPLETED. If the workflow fails or is aborted by the user, then the state switches to TERMINATED.

Suspend: The suspend method pauses a workflow with a certain workflow identifier. If the current state is ACTIVE, then the GWES first waits until the state RUNNING is reached, before suspending the workflow. This may need some time. After this, the workflow switches to the SUSPENDED state.

Resume: A suspended workflow can be resumed using the resume method. The state then switches back to RUNNING or ACTIVE.

Abort: The user can abort workflows that are not COMPLETED nor TERMINATED by means of the abort method. The state then switches to TERMINATED.

Restart: Restart a workflow with a certain identifier from the beginning. The restarted workflow will have a new identifier. This method looks for the earliest snapshot of the workflow in the database. Then the GWES initiates and starts the workflow.

Store: Store a snapshot of the workflow in the XML workflow database. The workflow can be restored using the restore method.

Restore: Restores a workflow from the XML database. If the input is the clean workflow identifier (without version number), then the GWES will restore the latest available workflow snapshot. If the workflow identifier string contains the whole XML database collection identifier, then the GWES will restore the given workflow (e.g. /db/gworkflowdl/[workflowID]/000000221.xml). The method call returns the new workflow identifier of the restored workflow. A restored workflow has the status INITIATED and needs to be started using the start method. Workflows are stored, e.g., by using the store method.

Get Data: Get some data that is hold inside a specific workflow and that is referenced by a data place identifier. Another alternative to get the data hold by the workflow is to retrieve to whole workflow description.

Get Workflow Description: Get the current Grid Workflow Description document of the workflow specified by its identifier.

Get Status: Get the current status code of the workflow specified by its identifier.

Get Workflow Identifiers: Get the identifiers of all the workflows that are handled by this Grid Workflow Execution Service

Get Workflow Status Array: Get the workflow status and additional information of all currently available workflows

Figure 8 shows a typical workflow lifecycle: First, the user initiates the workflow with a given workflow description and his user identifier. As a result, the user gets a workflow identifier back from the GWES. The user takes this identifier in order to start the workflow. After completion of the workflow, the user gets specific data elements from the workflow by means of the “get data” method or he retrieves the whole workflow document.

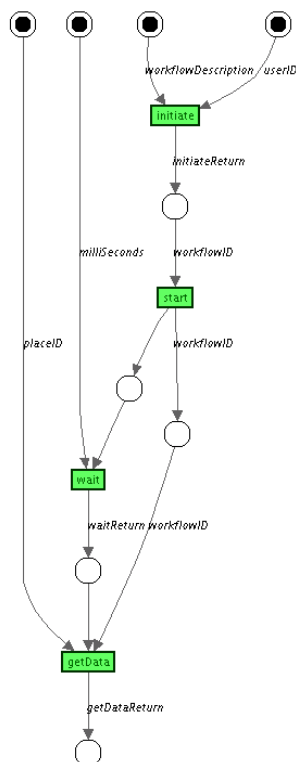


Figure 8. Workflow that represents a typical process of running a workflow.

4.1.3. GWES Web Interface

After installation the user can directly access the GWES using the built-in web interface (see Figure 9), which is available at <http://localhost:8080/gwes/> (replace localhost:8080 by the host name and port of the server where the GWES is installed). From within this web interface the user can:

- configure and test the GWES,
- list current workflows (see Figure 10),
- launch the GWUI,
- view a list of deployed web services (see Figure 11),
- browse some workflow examples (see Figure 12),
- browse the workflow history database (see Figure 13), and
- browse the resource description database (Instant-Grid setup)

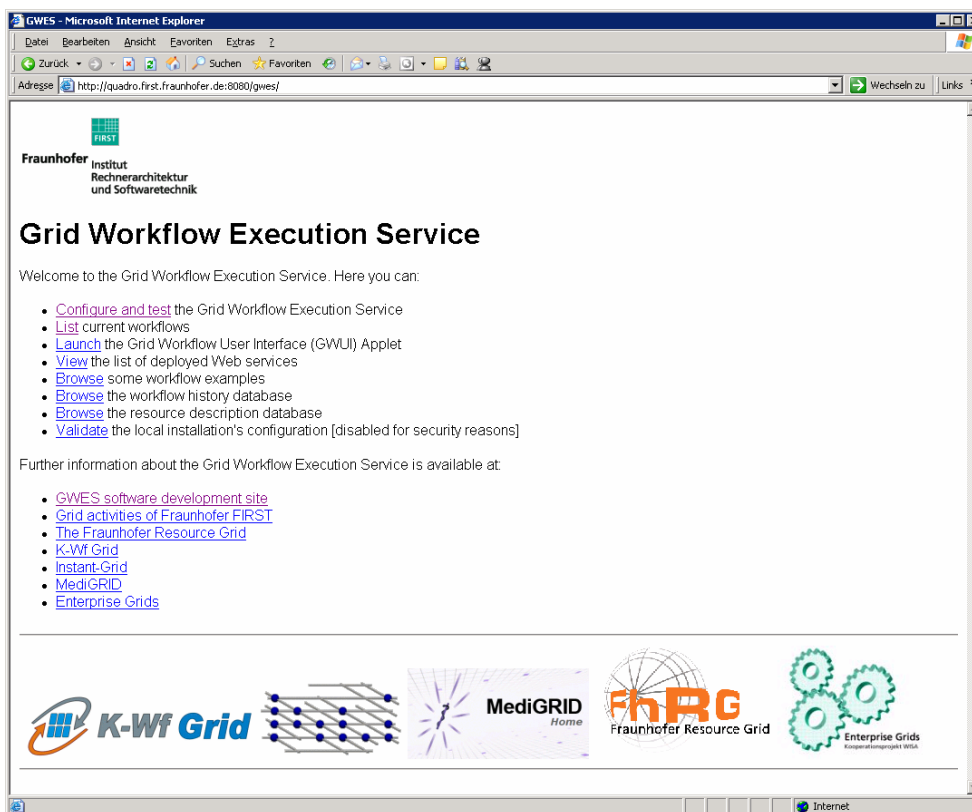


Figure 9. Screenshot of the main view of the GWES web interface.

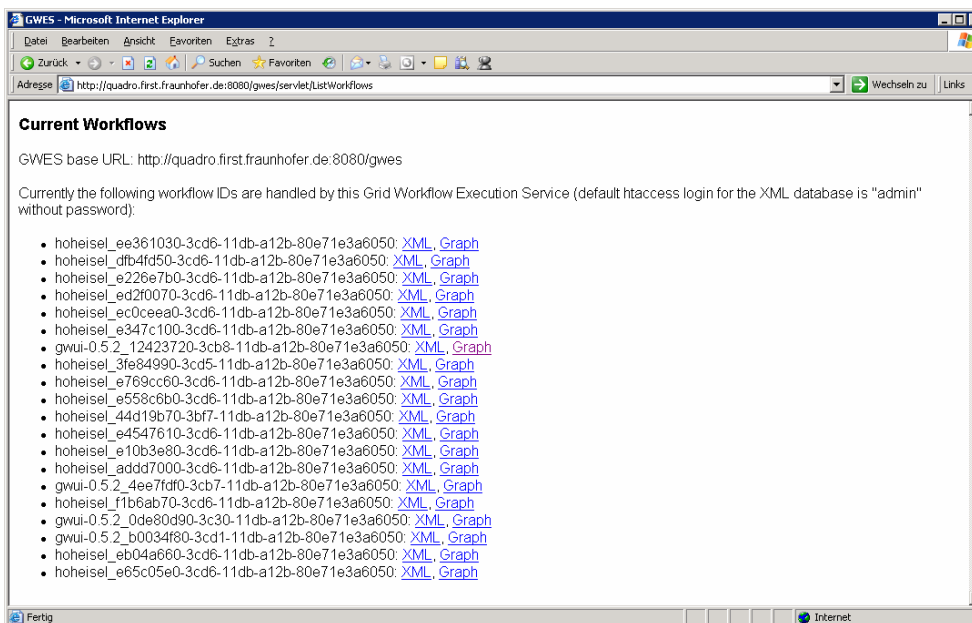


Figure 10. Screenshot of the web interface showing the list of current workflows. “XML” links to the XML workflow database; “Graph” starts the GWUI with the corresponding workflow.

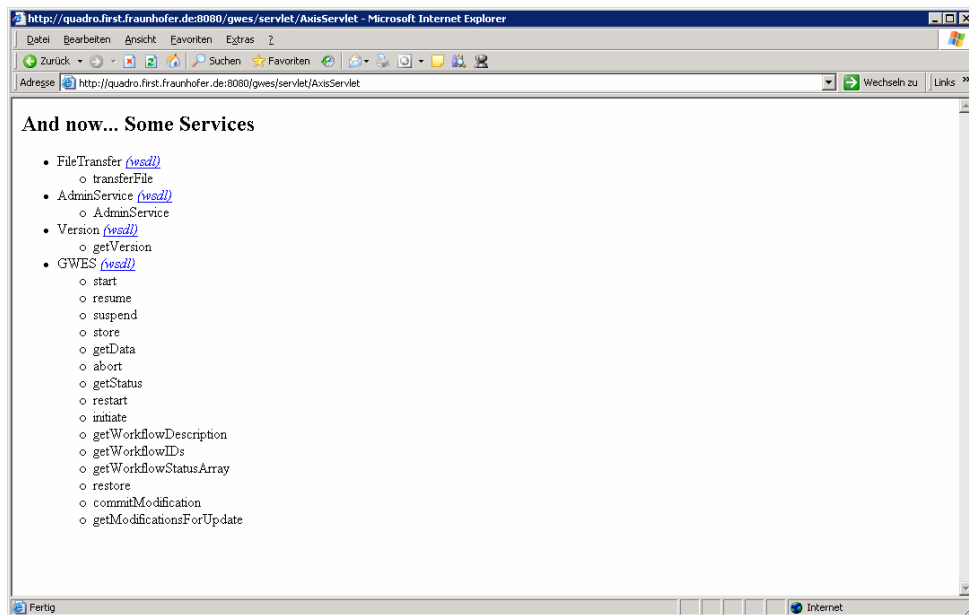


Figure 11. Screenshot of the web interface showing the list of deployed web services with links to the corresponding WSDLs.

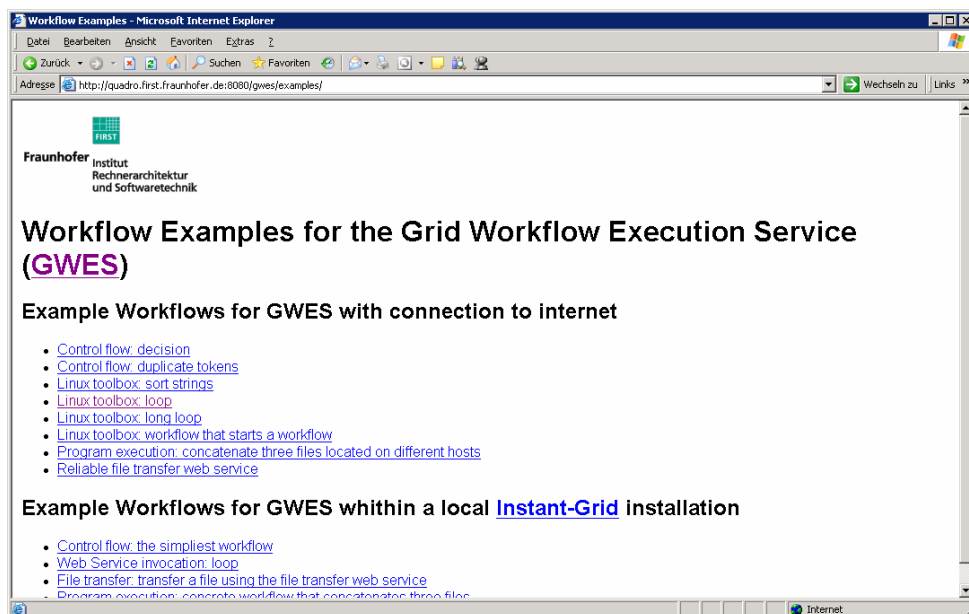


Figure 12. Screenshot of the web interface showing a list of workflow examples.

Name	Permissions	Owner	Group	Created	Last Modified
NN_0eac3910-2072-11db-a74c-a070ee05d750	rwur-ur-u	admin	dba	Jul 31 2006 10:53:38	
NN_1e23b9e0-2072-11db-a74c-a070ee05d750	rwur-ur-u	admin	dba	Jul 31 2006 10:54:04	
NN_23eb8600-2072-11db-a74c-a070ee05d750	rwur-ur-u	admin	dba	Jul 31 2006 10:54:13	
NN_3b4e4f30-2072-11db-a74c-a070ee05d750	rwur-ur-u	admin	dba	Jul 31 2006 10:54:53	
NN_49903630-2072-11db-a74c-a070ee05d750	rwur-ur-u	admin	dba	Jul 31 2006 10:55:17	
NN_f50d6290-2071-11db-a74c-a070ee05d750	rwur-ur-u	admin	dba	Jul 31 2006 10:52:55	
guest_352aff0-18d6-11db-ba57-ebd2f1fabd8a	rwur-ur-u	admin	dba	Jul 21 2006 18:30:23	
guest_3d9f1640-18cd-11db-815b-8f0fb69a7a5b	rwur-ur-u	admin	dba	Jul 21 2006 17:26:12	
guest_40dea550-18d7-11db-b4db-cf2342c3f7ef	rwur-ur-u	admin	dba	Jul 21 2006 18:37:52	
guest_6e8369f0-18cd-11db-815b-8f0fb69a7a5b	rwur-ur-u	admin	dba	Jul 21 2006 17:27:34	
guest_8ff78f0-18cd-11db-815b-8f0fb69a7a5b	rwur-ur-u	admin	dba	Jul 21 2006 17:28:30	
guest_97813460-18c6-11db-8537-ebca4b84c8a7	rwur-ur-u	admin	dba	Jul 21 2006 16:38:38	
guest_ae068620-18d7-11db-b4db-cf2342c3f7ef	rwur-ur-u	admin	dba	Jul 21 2006 18:40:55	
guest_c13a4ca0-18cc-11db-815b-8f0fb69a7a5b	rwur-ur-u	admin	dba	Aug 21 2006 17:22:45	
gwui-0.5.1_16e8c1e0-3830-11db-b876-d99e1b640a7c	rwur-ur-u	admin	dba	Aug 30 2006 16:01:53	
gwui-0.5.1_18c4dee0-2b8d-11db-b16b-b780cee16614	rwur-ur-u	admin	dba	Aug 14 2006 14:04:54	

Figure 13. Screenshot of the web interface showing the workflow history database.

4.2. RUNNING THE GRID WORKFLOW USER INTERFACE (GWUI)

The following sections describe how to use GWUI from the perspective of the end user.

4.2.1. GWUI Operating Requirements

In order to access the Grid Workflow User Interface the user needs a web browser that possesses a Java plugin (version ≥ 1.4) and access to the Grid portal where the system administrator has installed the GWUI applet. In addition you need online access from the web browser to the GWES web service (usually working on port 8080). Furthermore the user may need some example workflows, which are available in the directory `gwes/examples/` in the source distribution or in the `gwes.war` archive (download at <http://www.gridworkflow.org/kwfgrid/distributions/>). The user may also directly access the GWES web interface in order to browse workflow examples or to launch the GWUI (refer to 4.1.3).

4.2.2. GWUI Step-by-Step User Setup

K-Wf Grid setup

For the following instructions we assume that the user has access to the K-Wf Grid portal at <https://portal.ui.sav.sk/kwfportal/>. Details how to use the portal are available in the portal manual.

1. Open the URL <https://portal.ui.sav.sk/kwfportal/> in your favourite web browser
2. Log into the Grid Sphere portal by typing in your user name and password
3. Select the tab “KWf-Grid” – “Workflows”
4. If the web browser asks to accept the certificate of the signed Java applet, please accept it.
5. Now the web browser should download and display the workflow applet.
6. Use either the User Assistant (right panel) or the workflow loader (refer to chapter 5.1.1) to initiate new workflows.

7. Use the workflow status controls (refer to chapter 0) in order to start, suspend, resume or abort the workflow
8. Click on the nodes of the graph in order to modify the workflow

Instant-Grid Setup

If you do not have access to the K-Wf Grid portal or to another portal with deployed GWUI, then you can directly access the GWUI by means of the GWES web interface:

1. Open the URL <http://localhost:8080/gwes/servlet/GWUIServlet> in your favourite web browser (replace `localhost:8080` by the host name and port of the server where the GWES is installed)
2. If the web browser asks to accept the certificate of the signed Java applet, please accept it.
3. Now the web browser should download and display the workflow applet.
4. Use the workflow loader to initiate new workflows.
5. Use the workflow status controls in order to start, suspend, resume or abort the workflow.
6. Click on the nodes of the graph in order to modify the workflow

4.2.3. GWUI Basic Operation

Using the workflow applet described in chapter 5.1.1 the user is able to initiate, start, suspend, resume, abort, modify, and monitor workflows, that have been specified in XML according to the Grid Workflow Description Language (GWorkflowDL).

Figure 14 shows an example workflow from the CTM pilot application, containing red and yellow nodes within the workflow applet (left panel). The user creates this workflow by entering a free text query to the User Assistant (right panel), which initiates the workflow. The user then starts the workflow and the GWES invokes the WCT and the AAB in order to refine the workflow to the current abstraction level.

Figure 15 shows an example workflow from the Instant-Grid setup with a POV-Ray workflow, which renders an image of a 3D scene using three parallel instances of the ray tracing tool POV-Ray. The pre processing component splits the input file into several files, which are processed by each of the concurrent POV-Ray instances. The two post processing transitions merge the results into aggregated statistics and image files. The convert transition then converts the image into a specific format which is defined by one of the input tokens. The transfer transition then stores the resulting image file to a user defined location. The green transitions have already been mapped onto concrete resources by means of the scheduler, whereas the blue transitions are related to lists of matching candidates.

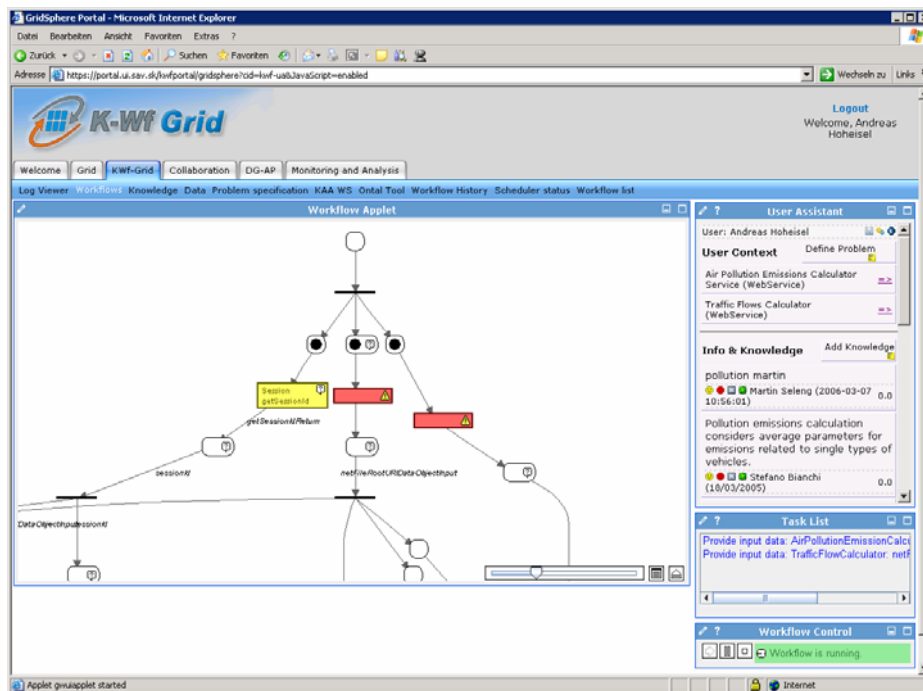


Figure 14. Workflow Applet with an example workflow from the CTM pilot application.

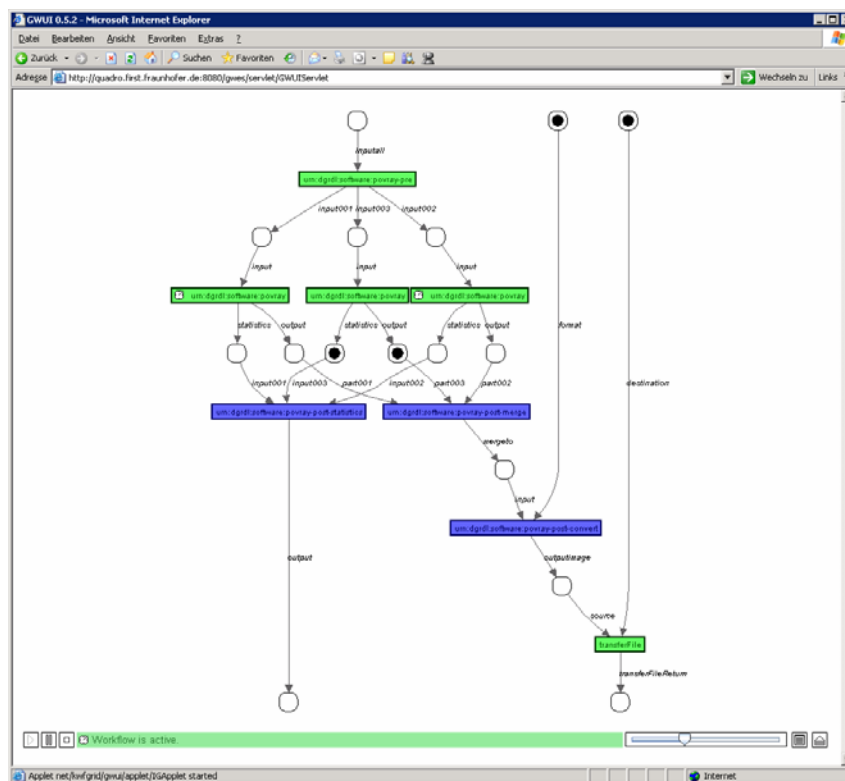


Figure 15. Workflow Applet in the *Instant-Grid* setup with a POV-Ray workflow.

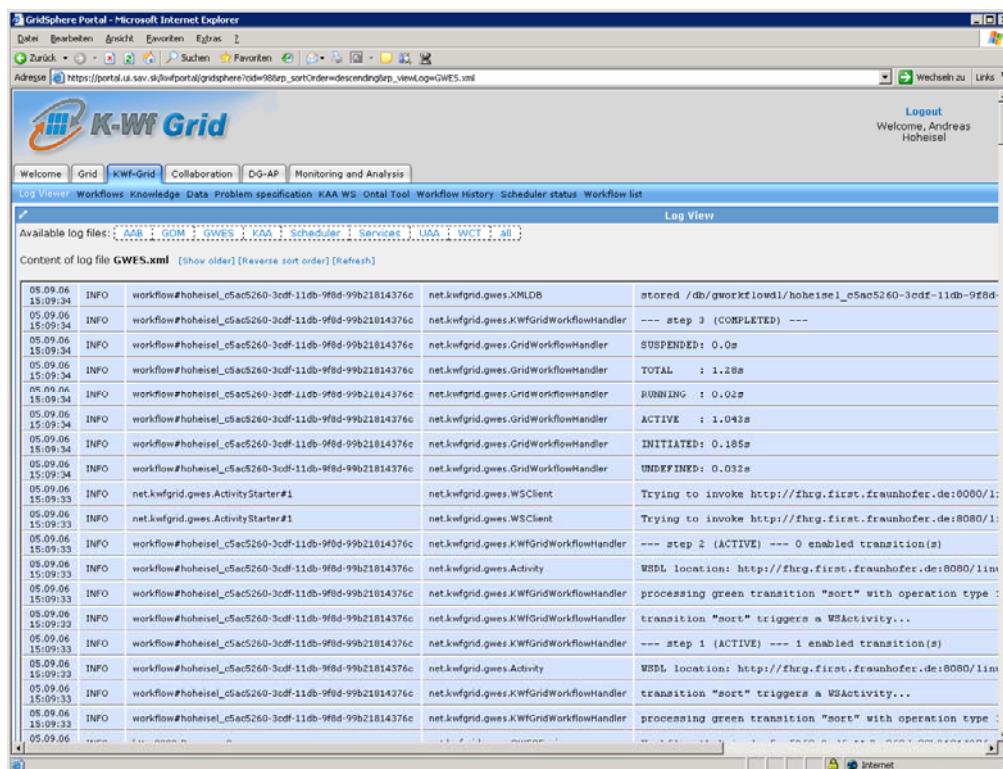


Figure 16. The Log Viewer that displays the GWES debug, info, and error messages.

Within the Kwf-Grid tab, the user can also access the GWES log viewer that displays the logging messages of the remote GWES web service (see Figure 16). The system administrator can adjust the amount of logging messages displayed here by editing the log4j.properties file within the GWES Web Service container. As default all info, warn, error, and fatal messages are displayed, and all debug messages are suppressed.

4.3. RUNNING THE GWES COMMAND LINE CLIENT (GWESCLIENT)

An alternative way to use the GWES is by means of the GWES command line client, which makes it possible to interact with the GWES without any graphical user interface. It is also possible to call the GWES command line client from within scripts or other programs, in order to easily integrate the workflow management features to the own environment.

4.3.1. GWESClient Operating Requirements

In order to invoke the GWESClient you need the Java libraries provided by the GWES and access to the script GWESClient.sh, which is located at gwes/bin/GWESClient.sh. Furthermore you need Java 1.5 and online access to the GWES web service (usually running on port 8080). In order to invoke the script GWESClient.sh you need a bash installed at /bin/bash. For more information regarding the installation procedure refer to section 3.1.2.3.

4.3.2. GWESClient Step-by-Step User Setup

The user may also use the GWES command line client as an alternative, which is located at gwes/bin/GWESClient.sh. In order to execute the client, Java 1.5 is required.

4.3.3. GWESClient Basic Operation

The GWESClient supports the full set of GWES operations, which have been described in Section 4.1.2. Here we give a short example of how to use the GWESClient in order to run workflows from the command line. For the complete interface reference please refer to Section 5.2:

```
GWESClient.sh -gwes http://localhost:8080/gwes \  
-initiateStart ${GWES_HOME}/examples/linuxtoolbox/gworkflowdl_long-wait-loop.xml \  
-monitor
```

This initiates and starts the workflow specified in the file `gworkflowdl_long-wait-loop.xml` using the GWES installed at `http://localhost:8080/gwes`.

The `-monitor` switch indicates that the current workflow state should be displayed continuously on the client side, e.g.:

```
workflowID = hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3  
ACTIVE  
RUNNING  
ACTIVE.....  
.....
```

If the workflow state does not change, the client outputs one “.” per second to the standard output, to show that the process is still alive.

You may interrupt the monitoring by pressing `<CTRL>-<C>`. This will only stop the client – the workflow itself will continue running. Now you may suspend the workflow using the following command (replacing `hoheisel_6220...` by the correct workflow identifier):

```
GWESClient.sh -gwes http://localhost:8080/gwes \  
-suspend hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3
```

Use the following command to display the current state of the workflow:

```
GWESClient.sh -gwes http://localhost:8080/gwes \  
-getStatus hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3
```

which will return, e.g.:

```
status = SUSPENDED
```

Now you may resume the workflow again and switch on monitoring by means of the following two commands:

```
GWESClient.sh -gwes http://localhost:8080/gwes \  
-resume hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3  
GWESClient.sh -gwes http://localhost:8080/gwes \  
-monitor hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3
```

The client now again displays the current workflow state until the workflow completes or terminates:

```
ACTIVE.....  
.....  
COMPLETED
```

Now the workflow is completed and you can retrieve the resulting workflow description:

```
GWESClient.sh -gwes http://localhost:8080/gwes \  
-getWorkflowDescription hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3
```

You can also download specific output data stored on one of the places within the workflow:

```
GWESClient.sh -gwes http://localhost:8080/gwes \  
-getData hoheisel_6220ba80-3d7d-11db-a61f-e455c60107e3 end
```

This returns the contents of the XML token, which is stored on the place with the identifier “end”:

```
<data xmlns:xsd="http://www.w3.org/2001/XMLSchema "  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance "  
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">  
  <plusReturn xsi:type="xsd:int">10</plusReturn>  
</data>
```

4.4. ADVANCED FEATURES

The main users of the Grid Workflow Execution service are:

- End users, who are composing and invoking predefined workflows using the GWUI or the GWES command line client, and
- Application developers, who develop and test new workflows for distributed Grid applications.

The following chapters target the second group of users, the application developers that need more insight to the advanced features of the workflow execution service.

The principal format of the workflow description language is defined by the GWorkflowDL XML Schema, available at <http://www.gridworkflow.org/gworkflowdl/>. In some parts, however, the specification of the GWorkflowDL gives some freedom to the implementation of the enactment machine, in order to make the language specification reusable in other context. This mainly affects the workflow operations, tokens, conditions, and properties. Here we describe, how these elements are specifically supported by the GWES.

4.4.1. Supported GWorkflowDL Operations

Currently the GWES supports operations that invoke web services methods as well as remote program executions. The format of the two alternative operation child elements is defined in separate XML schemas available at <http://www.gridworkflow.org/gworkflowdl/>. Here are some examples:

Web Service operations (*K-Wf Grid setup*):

```
<!-- red transition with unknown operation -->  
<operation/>
```

```
<!-- yellow transition with class of service, annotated with K-Wf Grid ontologies -->  
<operation>  
  <ws:WClassOperation xmlns:ws="http://www.gridworkflow.org/gworkflowdl/wsclassoperation">  
    <ws:owl>http://gom.kwfguid.net/gom/ontology/ServiceRegistry/CTM/NFP#Profile</ws:owl>  
  </ws:WClassOperation>  
</operation>
```

```
<!-- blue transition with service candidates -->  
<operation>  
  <ws:WClassOperation xmlns:ws="http://www.gridworkflow.org/gworkflowdl/wsclassoperation">  
    <ws:WOperation wsdl="http://server1/NFP.wsdl" operationName="generate" />  
    <ws:WOperation wsdl="http://server2/NFP.wsdl" operationName="generate" />  
  </ws:WClassOperation>  
</operation>
```

```
<!--green transition with selected service -->
<operation>
  <ws:WSClassOperation xmlns:ws="http://www.gridworkflow.org/gworkflowdl/wsclassoperation">
    <ws:WSOperation wsdl="http://server1/NFP.wsdl" operationName="generate" />
    <ws:WSOperation wsdl="http://server2/NFP.wsdl" operationName="generate" selected="true"/>
  </ws:WSClassOperation>
</operation>
```

```
<!-- green transition with selected service, annotated with K-Wf Grid ontologies -->
<operation>
  <ws:WSClassOperation xmlns:ws="http://www.gridworkflow.org/gworkflowdl/wsclassoperation">
    <ws:owl>http://gom.kwfgrid.net/gom/ontology/ServiceRegistry/CTM/NFP#Profile</ws:owl>
    <ws:WSOperation wsdl="http://server1/NFP.wsdl" operationName="generate">
      <ws:owl>http://gom.kwfgrid.net/gom/ontology/ServiceRegistry/CTM/NFP#Service</ws:owl>
    </ws:WSOperation>
    <ws:WSOperation wsdl="http://server2/NFP.wsdl" operationName="generate" selected="true">
      <ws:owl>http://gom.kwfgrid.net/gom/ontology/ServiceRegistry/CTM/NFP#Service</ws:owl>
    </ws:WSOperation>
  </ws:WSClassOperation>
</operation>
```

Remote program execution (*Instant-Grid setup*):

```
<!-- yellow transition with software class -->
<operation>
  <pe:programClassExecution
    xmlns:pe="http://www.gridworkflow.org/gworkflowdl/programclassexecution"
    softwareClass="urn:dgrdl:software:cat"/>
</operation>
```

```
<!-- blue transition with software/hardware candidates -->
<operation>
  <pe:programClassExecution
    xmlns:pe="http://www.gridworkflow.org/gworkflowdl/programclassexecution"
    softwareClass="urn:dgrdl:software:cat">
    <pe:programExecution software="software:cat-fhrg" hardware="hardware:server1"/>
    <pe:programExecution software="software:cat-fhrg" hardware="hardware:server2"/>
  </pe:programClassExecution>
</operation>
```

```
<!-- green transition with software/hardware selected -->
<operation>
  <pe:programClassExecution
    xmlns:pe="http://www.gridworkflow.org/gworkflowdl/programclassexecution"
    softwareClass="urn:dgrdl:software:cat">
    <pe:programExecution software="software:cat-fhrg" hardware="hardware:server1"
      quality="0.9" selected="true"/>
    <pe:programExecution software="software:cat-fhrg" hardware="hardware:server2"
      quality="0.1"/>
  </pe:programClassExecution>
</operation>
```

```
<!-- green transition with software/hardware selected and mapped onto real resources -->
<operation>
  <pe:programClassExecution
    xmlns:pe="http://www.gridworkflow.org/gworkflowdl/programclassexecution"
    softwareClass="urn:dgrdl:software:cat">
    <pe:programExecution software="/usr/bin/cat.sh"
      hardware="https://server1:8443/wsrp/services/ManagedJobFactoryService"
      quality="0.9" selected="true"/>
    <pe:programExecution software="software:cat-fhrg" hardware="hardware:server2"
      quality="0.1"/>
  </pe:programClassExecution>
</operation>
```

4.4.2. Supported GWorkflowDL Tokens

The data is stored as tokens within the GWorkflowDL description of the workflow. The content of the token either contains the real data itself or (when dealing with big amount of data) a reference to the real data (e.g. an URL). The syntax of tokens depends on the implementation of the workflow engine. The GWES currently supports token formats that are related to the invocation of Web Services via SOAP and to the invocation of remote programs using file I/O. Another class of tokens contains control tokens that are used to model the control flow of the workflow. In the following we give some examples of tokens that can be processed by the GWES.

The following token represents a simple integer value used as an input parameter of a Web Service operation:

```
<token>
  <data xmlns="">
    <duration xsi:type="xsd:int">1000</duration>
  </data>
</token>
```

This token represents a string that is used in a Web Service operation that processes the regex format:

```
<token>
  <data xmlns="">
    <param xsi:type="xsd:string">bl.$</param>
  </data>
</token>
```

Remember that the names of the child elements of the <data> element are not processed by GWES nor by SOAP, so they can be arbitrary names, such as duration or param in the examples above.

This is a control token that is placed on control places. It contains a boolean value that expresses the exit state of the last transition:

```
<token>
  <control xmlns="" xsi:type="xsd:boolean">true</control>
</token>
```

This token contains a SOAP fault as an return value of a failed remote program execution:

```
<token>
  <data xmlns="" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
    <soapenv:Fault xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
      <soapenv:Code>
        <soapenv:Value>env:Server</soapenv:Value>
      </soapenv:Code>
      <soapenv:Reason>
        <soapenv:Text xml:lang="en">
          Activity 'hoheisel_4825fbd0-3d92-11db-a61f-e455c60107e3_0000000001': Exception
          during WS-GRAM invocation
        </soapenv:Text>
      </soapenv:Reason>
      <soapenv:Detail>
        https://server1:8443/wsrf/services/ManagedJobFactoryService /usr/bin/cat.sh
        GSSException: Expired credentials detected</soapenv:Detail>
      </soapenv:Fault>
    </data>
  </token>
```

This token specifies a file which is used by a remote program execution and can be transferred automatically by means of the Reliable File Transfer (RFT):

```
<token>
  <data>
    <file xsi:type="xsd:string">gsiftp://server1//tmp/catdir/d27.dat</file>
  </data>
</token>
```

This token specifies a command line parameter for a remote program execution:

```
<token>
  <data>
    <param xsi:type="xsd:string">jpg</param>
  </data>
</token>
```

4.4.3. Supported GWorkflowDL Edge Expressions

Edge expressions are labels attached to the arcs that connect places with transitions or vice versa. Depending on the type of operation which is represented by the transition, the edge expressions have slightly different meaning for the workflow processing within the GWES.

Control Transitions:

Control transitions are not related to any operation, they are simply used for synchronization issues or for simple token processing. Every *incoming edge expression* (attached to an arc that connects an input place with the control transition) is treated as a label for the incoming data token. The name of the first child element after the <data> element of the input token is replaced by the corresponding edge expression. After renaming the child elements, the GWES joins all input tokens within one <data> element.

Outgoing edge expressions (attached to an arc that connects the control transition with an output place) are evaluated as *XPath 1.0* expressions. The context of the evaluation is the joint <data> element, which contains the merged child elements of all data input tokens. If the evaluation of the expression results in an empty set of elements, then an empty <data/> element will be put in the output token.

Figure 17 displays a simple workflow example that makes use of edge expressions, in order to rearrange the contents of two input tokens. Figure 18 shows the corresponding GWorkflowDL document. The result after the workflow execution is displayed in Figure 19.

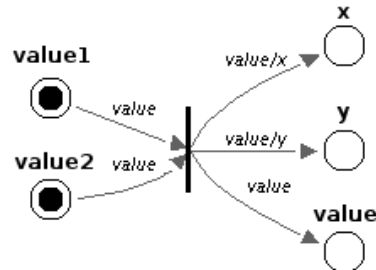


Figure 17. Example of using edge expressions to join two input tokens and to distribute the result on three output tokens.

```
<workflow xmlns="http://www.gridworkflow.org/gworkflowdl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation="http://www.gridworkflow.org/gworkflowdl
  http://www.gridworkflow.org/kwfguid/src/xsd/gworkflowdl_1_0.xsd" ID="No_ID">
  <place ID="value1">
    <token><data xmlns="">
      <value1>
        <x>15</x>
        <y>23</y>
      </value1>
    </data></token>
  </place>
  <place ID="value2">
    <token><data xmlns="">
      <value2>
        <x>16</x>
        <y>24</y>
      </value2>
    </data></token>
  </place>
  <place ID="x"/>
  <place ID="y"/>
  <place ID="value"/>
  <transition ID="joinSplitTokens">
    <inputPlace placeID="value1" edgeExpression="value"/>
    <inputPlace placeID="value2" edgeExpression="value"/>
    <outputPlace placeID="x" edgeExpression="value/x"/>
    <outputPlace placeID="y" edgeExpression="value/y"/>
    <outputPlace placeID="value" edgeExpression="value"/>
  </transition>
</workflow>
```

Figure 18. XML representation of the workflow example which uses edge expressions to join two input tokens and to distribute the result on three output tokens.

```
<!-- Place "x" -->
<data>
  <x>15</x>
  <x>16</x>
</data>
```

```
<!-- Place "y" -->
<data>
  <y>23</y>
  <y>24</y>
</data>
```

```
<!-- Place "value" -->
<data>
  <value>
    <x>15</x>
    <y>23</y>
  </value>
  <value>
    <x>16</x>
    <y>24</y>
  </value>
</data>
```

Figure 19. Result of the workflow displayed in Figure 18.

For more details about XPath 1.0, please refer to the specification at <http://www.w3.org/TR/xpath>. Here are some examples of outgoing edge expressions and their corresponding result, regarding the example above:

XPath Expression	Description	Result
value	Return all elements <value>	<value> <x>15</x> <y>23</y> </value> <value> <x>16</x> <y>24</y> </value>
value/x	Return all children <x> of <value>	<x>15</x> <x>16</x>
value/y	Return all children <y> of <value>	<y>23</y> <y>24</y>
value[1]/x	Return all children <x> of the first <value>	<x>15</x>
value[x=15]	Return all elements <value> that have a child <x> with contents "15"	<value> <x>15</x> <y>23</y> </value>
value[x=15]/y	Return all children <y> of all elements <value> that have a child <x> with contents "15"	<y>23</y>
value/*	Return all children of all elements <value>	<x>15</x> <y>23</y> <x>16</x> <y>24</y>

Web Service Transitions:

The edge expressions for web service transitions are treated similar as for control transitions. The *incoming edge expressions* are labels for the input tokens, which need to be identical to the name of the message part of the corresponding SOAP operation, as defined in the WSDL of the web service.

The *outgoing edge expressions* are evaluated as *XPath 1.0* expressions. The context of the evaluation is the result of the web service invocation. The name of the first child of the <data> element is identical to the corresponding message part name as defined in the WSDL of the web service. If the

evaluation of the expression results in an empty set of elements, then an empty <data/> element will be put in the output token.

Figure 20 shows an example WSDL of an Calculator web service which offers the operation “plus”. The SOAP input message parts are defined as “a” and “b”, the output message part is called “plusReturn”. Valid incoming edge expressions for the invocation of this operation are “a” and “b”. Valid outgoing edge expressions are arbitrary XPath 1.0 expressions that are evaluated against the return XML data <data><plusReturn>...</plusReturn></data>.

A valid outgoing edge expression would be, e.g., “plusReturn[@type='xsd:int']”.

SOAP faults that are thrown during the invocation of the web service plus operation are encoded as <data><soapenv:Fault>...</soapenv:Fault></data>. If you want to put the contents of the SOAP fault on a place, use the edge expression “soapenv:Fault”. If you are only interested in the fault string, you may use the edge expression “soapenv:Fault/faultstring”.

```
<wsdl:definitions ...>
  <wsdl:message name="plusResponse">
    <wsdl:part name="plusReturn" type="xsd:int"/>
  </wsdl:message>
  <wsdl:message name="plusRequest">
    <wsdl:part name="a" type="xsd:int"/>
    <wsdl:part name="b" type="xsd:int"/>
  </wsdl:message>
  ...
  <wsdl:portType name="Calculator">
    <wsdl:operation name="plus" parameterOrder="a b">
      <wsdl:input message="impl:plusRequest" name="plusRequest"/>
      <wsdl:output message="impl:plusResponse" name="plusResponse"/>
    </wsdl:operation>
    ...
  </wsdl:portType>
  ...
</wsdl:definitions>
```

Figure 20. Example WSDL of a Calculator web service with input message parts “a” and “b” and with output message part “plusReturn”.

Program Transitions:

Program transitions have been introduced to automatically invoke command line programs on remote computing nodes (refer to Section 4.4.1). Legacy command line programs are usually encapsulated by wrapper scripts, which provide a uniform structure of command line parameters for the GWES. The structure of command line parameters is as follows:

```
<executable> -<param1> <directory1>/<filename1> \  
              -<param2> <directory2>/<filename2> \  
              -<param3> <directory3>/<filename3> \  
              ...
```

<param1>...<paramN> are the names of the input and output parameters of the command line program which are identical to the *incoming* and *outgoing edge expressions* defined in the corresponding transition. “*stdin*”, “*stdout*”, and “*stderr*” are additional reserved edge expressions, which connect the token on a place with the corresponding standard I/O of the process.

Here is an example of a program transition, which is connected to two input and one output place:

```
<transition ID="cat">
  <inputPlace placeID="d25" edgeExpression="input1"/>
  <inputPlace placeID="d26" edgeExpression="input2"/>
  <outputPlace placeID="d25-26" edgeExpression="stdout"/>
  <operation>
    <pe:programClassExecution ... />
  </operation>
</transition>
```

The GWES will then invoke the command line program using a syntax similar to:

```
cat.sh -input1 ${token_on_d25} -input2 ${token_on_d26} > ${token_on_d25-26}
```

`${token_on_*}` is replaced by the local directory and file name as specified within the corresponding token.

4.4.4. Supported GWorkflowDL Conditions

Transitions can be annotated with conditions (also know as "transition guards"). An enabled transition only fires if all its conditions are fulfilled. Conditions are used in order to decide between two or more branches in an OR split region of a Petri net. The Grid Workflow Execution Service supports conditions that follow the XPath 1.0 syntax, which is evaluated over the whole workflow description. Variables (e.g. `$input/`) that are equal to edge expressions are expanded by the corresponding input place (e.g. `place[@ID="begin"]/`).

Here are some examples of supported GWorkflowDL conditions:

```
<!-- true if text length of the string within the token on the place connected with the
edgeExpression "value" is greater than "0" -->
<condition>
  string-length($value/gwdl:token/data/*)>0
</condition>

<!-- true if token content is of type string -->
<condition>
  $params/gwdl:token/data/*/@xsi:type="xsd:string"
</condition>

<!-- this condition always returns true -->
<condition>true</condition>

<!-- this condition is true, if the token contains a number less than 10. -->
<condition>$plusReturn/gwdl:token/data<10</condition>
```

4.4.5. Supported GWorkflowDL Properties

Properties within the workflow description are used to annotate specific workflow elements. As specified in the GWorkflowDL XML schema the elements `<workflow>`, `<place>`, and `<transition>` can contain `<property>` elements with the following format:

```
<property name="NAME">VALUE</property>
```

Please remind that the sequence of child elements within the GWorkflowDL is specified, and that the `<property>` element has to be the second (optional) child element after `<description>`.

Currently the GWES is using the following properties:

Workflow properties:

Name	Values	Description
status	UNDEFINED, INITIATED, RUNNING, ACTIVE, SUSPENDED, COMPLETED, TERMINATED	Current status of the workflow
faultManagementPolicy	AbortOnActivityTerminated, ContinueOnActivityTerminated, SuspendOnActivityTerminated,	What to do, if one activity of the workflow terminates. Default value is "AbortOnActivityTerminated". If the fault management is explicitly modeled within the workflow, then you should select "ContinueOnActivityTerminated".
domain	CTM, FFSC, ERP, CTM;FFSC, ...	<i>K-Wf Grid setup:</i> Application domain of the Ontologies (used by AAB and WCT). Set this property to speed up the refinement process.
resource.repository.collection	e.g.: /db/instantgrid, /db/dgrdl/, ...	<i>Instant-Grid setup:</i> Name of the XML database collection, which contains the resource descriptions to be used during the refinement process
wct.refinement.version	Number	<i>K-Wf Grid setup:</i> version of the last WCT refinement
warn.1, warn.2, ...	Warning message	Workflow warning messages
error.1, error.2, ...	Error message	Workflow error messages
birthdayMs	e.g.: 1153469237249	Timestamp of the first occurrence of the workflow in milliseconds since 1970.
durationUndefinedMs	e.g.: 141	Time in milliseconds status being undefined and not yet initialized
durationInitiatedMs	e.g.: 1492	Time in milliseconds status being initiated and not running
durationRunningMs	e.g.: 5	Time in milliseconds status being running (not including active)
durationActiveMs	e.g.: 444	Time in milliseconds status being active
durationSuspendedMs	e.g.: 0	Time in milliseconds status being suspended
durationTotalMs	e.g.: 2082	Total duration of workflow in milliseconds.
endTimeMs	e.g.: 1153469239331	Timestamp when the workflow completes or terminates in milliseconds since 1970

isUnbounded	true, false	<i>Workflow Analysis:</i> “true” if the workflow contains an unbounded place
decision.X (X=1, 2, 3, ...)	TAKE_CONFLICT, TAKE_CHOICE, PUT_CONFLICT, PUT_CHOICE	<i>Workflow Analysis:</i> This workflow contains a decision (e.g., a conflict where two transitions compete for the same token). Conditions which resolve the decision are not regarded.
complexity	number	<i>Workflow Analysis:</i> measure for the Karp-Miller-Tree complexity of this workflow

Place properties:

Name	Values	Description
isQuasiLive	true, false	<i>Workflow Analysis:</i> “false” if the place can never get a token
isFinallyMarked	true, false	<i>Workflow Analysis:</i> “true” if the place will be marked when the workflow completes.
decision.X (X=1, 2, 3, ...)	TAKE_CONFLICT, TAKE_CHOICE, PUT_CONFLICT, PUT_CHOICE	<i>Workflow Analysis:</i> This place is part of a conflict (e.g., two transitions compete for a token). Conditions which resolve the decision are not regarded.

Transition properties:

Name	Values	Description
status	UNDEFINED, INITIATED, RUNNING, ACTIVE, SUSPENDED, COMPLETED, TERMINATED	Status of the last activity triggered by this transition
breakpoint	empty, “REACHED”, “RELEASED”	If a transition contains a breakpoint property, the GWES suspends the execution of the workflow when reaching the transition. The GWES then sets the property value to “REACHED”. When resuming the workflow, the GWES sets the value to “RELEASED” until the breakpoint is reached again.
aab.refinement.failed	Number or reason	<i>K-Wf Grid setup:</i> Property is set if the AAB was not able to refine the transition
wct.refinement.failed	Number or reason	<i>K-Wf Grid setup:</i> Property is set if the WCT was not able to refine the transition

resourcematcher.refinement.failed	Number or reason	<i>Instant-Grid setup:</i> Property is set if the Resource Matcher was not able to refine the transition
decision.X (X=1, 2, 3, ...)	TAKE_CONFLICT, TAKE_CHOICE, PUT_CONFLICT, PUT_CHOICE	<i>Workflow Analysis:</i> This transition is part of a conflict (e.g., two transitions compete for the same token). Conditions which resolve the decision are not regarded.
isQuasiLive	true, false	<i>Workflow Analysis:</i> “false” if the transition will never fire

5. INTERFACE REFERENCE GUIDE

This chapter contains a detailed description of the GWUI interfaces, such as the Grid Workflow User Interface (Section 5.1) and the GWES command line client (Section 5.2)

5.1. GRID WORKFLOW USER INTERFACE (GWUI)

5.1.1. Workflow Applet

The workflow applet is the central component of the GWUI, which dynamically displays the running workflow as a graph. It offers an interactive view of the workflow graph as well as several controls to inspect and control a workflow.

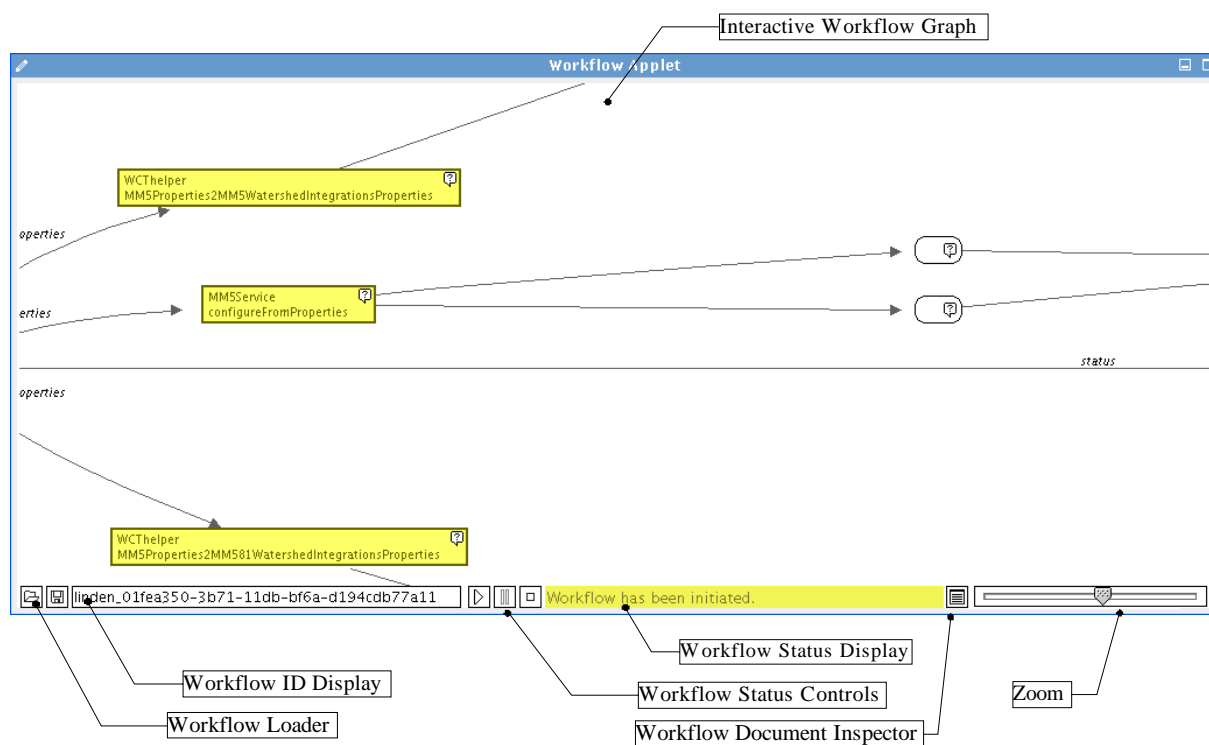


Figure 21. The display of the workflow applet showing part the graph of an initiated workflow.

Interactive Workflow Graph

The workflow graph offers several tools to inspect places and transitions of the workflow. The places and transitions of the graph may be dragged around by the user in order to adjust the view of the graph.

Transitions of the workflow are painted in 5 different states according to the abstraction level of the transition (Figure 22). In any state the user may click on a transition to open the transition inspector (Figure 23). In red state a transition may have a warning symbol which indicated that the transition could not be concretized by the WCT. In this case the user must use the place inspector (Figure 25) to

remove the input tokens of the transition and add the output tokens manually. In yellow state the node contains a little question mark. Clicking the question mark will call the User Assistant (*K-Wf Grid setup*) in a different browser frame, showing additional information about the Web Service class operation of the according transition.

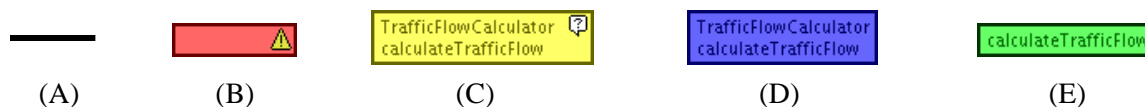


Figure 22. (A) Control Transition, (B) abstract Transition with Warning Symbol, (C) Transition with Web service Class Operation, (D) Transition with several selected Web service Operations, (E) Transition with one scheduled Web service Operation.

Places of the workflow are painted as control, data or effect places (Figure 22). Clicking on a place will open the place inspector (Figure 25) for that place. Data and effect places have a little question mark which can be clicked by the user in order to let the User Assistant display hints about the data / effect class of that place.



Figure 23. (A) Control Place, (B) Effect / Data Place

Transition Inspector

Clicking on a transition in the workflow graph will open a dialog configured for the according transition (Figure 24). The transition inspector shows some information about the abstraction level of the transition, explains warnings if there exist any for the transition and offers an editor for the conditions and properties of the transition. Existing conditions can be inspected by selecting them from the according combo box. A selected condition can be removed by clicking on the remove button. New conditions can be added by entering an XPath expression manually or selecting a preconfigured one from a popup menu (right-click the text area) and pressing the add button. By editing conditions the user can solve conflicts in the workflow which can not be solved automatically. A new property can be added by first double clicking the table cell saying “Enter new Property...” in the properties table. Then the key of the new property needs to be typed and confirmed by pressing the “Enter” key. Afterwards the value of that property can be added into the table cell next to the key cell. After the value has been confirmed by pressing “Enter” the new property will be added to the transition. The value of an existing property can be edited by double clicking the according table cell and entering a new value for that property. To remove a property, double click the table cell showing the value of the property, completely delete the value and press “Enter”.

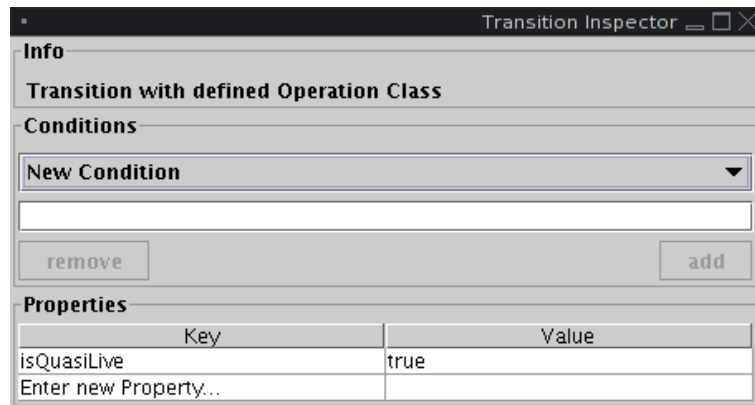
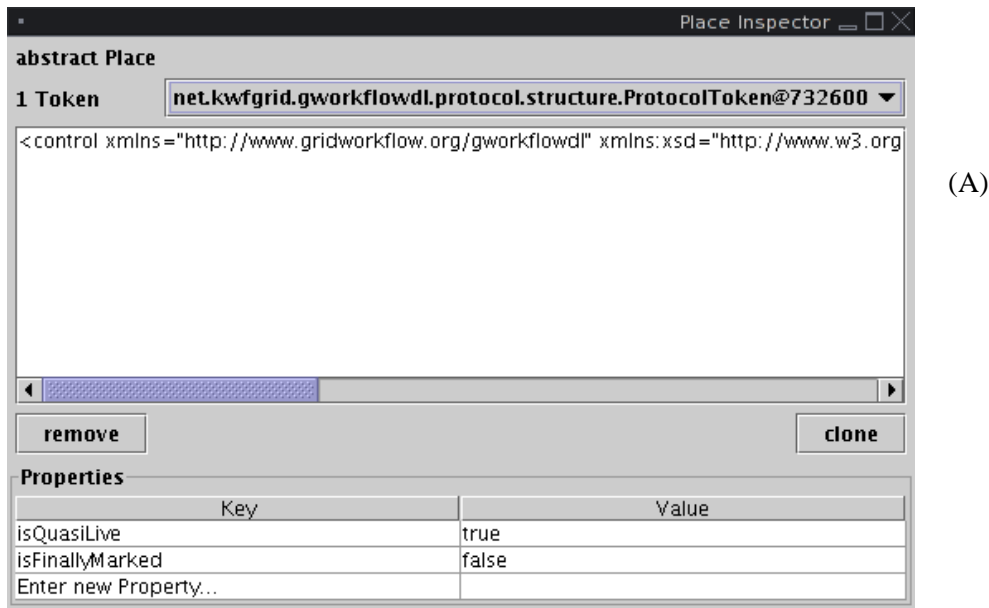


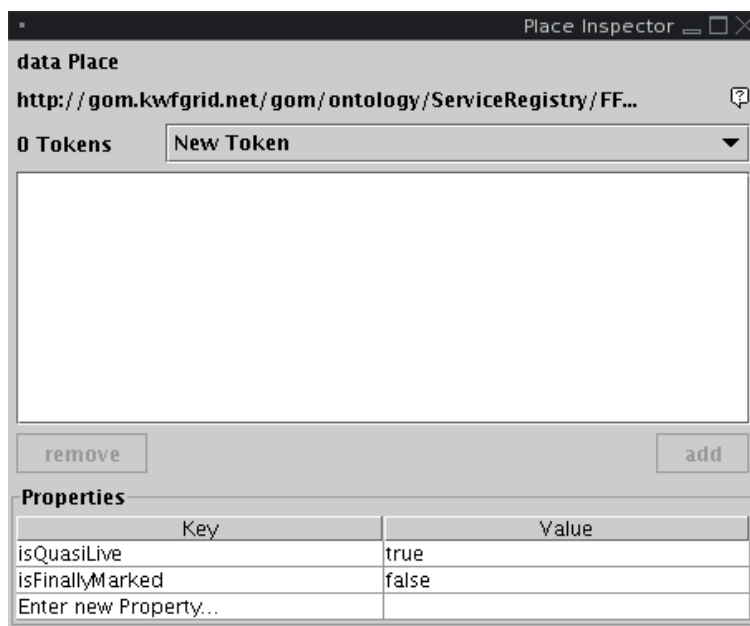
Figure 24. Transition Inspector for a Transition with defined Operation Class (Yellow)

Place Inspector

Clicking on a place in the workflow graph will open a dialog configured for the according place (Figure 25). The place inspector shows some information about the place's type and – if it's a data or effect place – the URI of the OWL-S description of the data/effect class. Further information about the data/effect class can be retrieved from User Assistant by clicking on the question mark next to the URI. The place inspector also offers an editor for the tokens located on the place and for the properties of the place. Existing tokens can be inspected by selecting them from the combo box. A selected token can be removed by clicking the remove button. Existing tokens can be cloned by pressing the clone button. New tokens can be added by entering their XML description manually, loading it from a file or selecting a preconfigured token from the popup menu which can be activated by right-clicking the text area. After the XML description has been entered into the text area, the add button must be pressed to add the token to the workflow. The properties table works just like described for the Transition Inspector.



(A)



(B)

Figure 25. (A) Place Inspector for a control Place with one Token, (B) Place Inspector for a data Place without any tokens.

Workflow Status Controls *

Using the workflow status controls the user can start/resume, pause or abort the workflow. It depends on the current status of the workflow if the controls are enabled or disabled.

Workflow Status Display*

The workflow status display informs the user about the runtime status of the workflow. This may be one of undefined, initiated, running, suspended, terminated, and completed.

The status display is also used to communicate exceptions to the user. Exceptions have to be confirmed by clicking the close button displayed to the right of the message.

Zoom

Dragging the slider left or right will zoom in / out the graph.

Workflow Document Inspector

Clicking on the workflow document inspector, a dialog will open which displays the current state of the workflow document in XML syntax. The content of the according text area can be updated by clicking the update button. The workflow document inspector also features a table of the workflow's properties which can be operated like the property tables shown in the place and transition inspectors.

Workflow ID Display

The workflow ID display shows the ID of the currently displayed workflow. If an ID is entered into the text field and confirmed with "Enter", the workflow with the typed ID will be shown in the Grid Workflow User Interface.

Workflow Loader

The workflow loader can be used to upload a workflow description from the user's local hard disc to the workflow enactment engine and to save the current workflow as XML file to the user's local hard disc.

5.1.2. Task List Applet

The task list applet (Figure 16) displays a list of tasks the user has to accomplish in order to complete a workflow and offer the according dialogs. These tasks are concerning the input of missing data or the decision about conflicts during workflow execution.

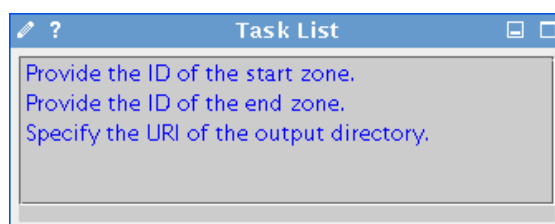


Figure 26. Task List Applet displaying some data input tasks the user has to accomplish.

Clicking on one of the displayed tasks will open a dialog suitable to solve that task. These dialogs are application and data specific. Their functioning is therefore out of scope of this reference guide.

*In the K-Wf Grid Portal the Workflow Status Controls and the Workflow Status Display are displayed in a separate portlet.

5.2. GWES COMMAND LINE CLIENT (GWESCLIENT)

```
Usage: GWESClient.sh -gwes | -g <gwesURL>
      -initiate | -i <GWorkflowDL>
      -initiateStart | -is <GWorkflowDL> [-monitor]
      -start | -s <workflowID> [-monitor]
      -suspend <workflowID>
      -resume <workflowID>
      -store <workflowID>
      -restore <workflowID>
      -restart <workflowID> [-monitor]
      -abort | -a <workflowID>
      -getWorkflowDescription | -gwd <workflowID>
      -getStatus | -gws <workflowID>
      -getWorkflowIDs | -gwi
      -getWorkflowStatusArray | -gwa
      -getData <workflowID> <placeID>
      -monitor | -m <workflowID>
```

The parameter `-gwes <gwesURL>` is mandatory and specifies the home URL where the GWES is located, e.g., `http://server:8080/gwes`.

In addition you must specify exactly one of the operations `-initiate`, `-initiateStart`, `-start`, `-suspend`, etc. The operations are described in more detail in Section 4.1.2 and 4.3.3.

`<GWorkflowDL>` is the path and filename of the GWorkflowDL file containing the workflow to enact.

`<workflowID>` is the identifier of the workflow returned by the GWES after the workflow initialization.

You may either use the full parameter name (e.g. `-getWorkflowDescription`) or the shortcut if available (e.g. `-gwd`).

`-monitor` or `-m` may be used to monitor the running workflow until it terminates or completes.

6. Q&A

Q: Where can I download the sources and binaries of the Grid Workflow Execution Service?

A: At <http://www.gridworkflow.org/kwfgrid/distributions/>

Q: Is the Grid Workflow Execution Service open source?

A: It is open source for non-commercial usage. For commercial usage, or if you want to (re-)use parts of the software in your own product, you must contact steffen.unger@first.fraunhofer.de to get a special license agreement. So the Grid Workflow Execution Service is not open source with respect to the Open Source Initiative's (OSI) definition of the term "open source", whereas the source is open in the sense, that everybody can download it. You find the license agreement at <http://www.gridworkflow.org/kwfgrid/gwes/docs/license.html>

Q: How should I pronounce "GWES"?

A: Within the K-Wf Grid project we pronounce it "ge-wes".

Q: How should I pronounce “K-Wf Grid”?

A: Well, we are almost famous in the Grid community to have the most ugly acronym in history:-) But you could pronounce it “quif-grid”.

Q: What is "GWorkflowDL"?

A: GWorkflowDL is an acronym for "Grid Workflow Description Language". The GWorkflowDL is an XML-based description language for Grid workflows developed within the K-Wf Grid project.

Q: What are these side effects? Do we have to assign a side effect for each operation?

A: No, you do not have to assign a side effect for each operation. We need this side effects just in order to be able to automatically compose a workflow that contains web service operations without real response values. For example: operation "void A.store()" writes its output data to a SQL database and returns without any return value. Then we need some side effect, e.g. "DataXY stored in SQL database" in order to use this operation in a workflow. The operation "String A.getData()" however does not need any side effect.

Q: Why do the tokens on the input places contain real data?

A: We need tokens with real data in order to enact concrete workflows. In this case we have different (instances of) workflows for different input instances. However it is also possible to put an abstract token without any contents onto a place. Then some refinement tool or the user must provide the real values before running the workflow.

Q: What is the owl attribute meant for?

A: The owl attribute can be used in order to annotate the corresponding element with some meta data. In the K-Wf Grid project we use this attribute in order to refer to a certain ontologic property. This property is then used in an RDQL query to the Grid Organizational Memory (kind of registry) for retrieving additional information about the element.

Further questions and answers will be included during the progress of the project.

7. KNOWN ISSUES

If you have access to the K-Wf Grid issue tracking system (currently only for K-Wf Grid developers), you will find more details by following the link connected to each issue ID. For issue/bug reporting please contact Andreas Hoheisel (andreas.hoheisel@first.fraunhofer.de) or report it directly to the issue tracking system at <http://cvs.ui.sav.sk/mantis/>.

7.1.1. GWES

ID	Category	Severity	Summary
0000085	gwes	major	GWES should fire the transition AFTER completing the activity
0000094	gwes	minor	GWES: Exclude JUnit tests in project.xml which depend on external resources

0000092	gwes	minor	GWES: GRAMActivity NullPointerException when aborting workflow
0000089	gwes	minor	GWES source distribution contains confidential data
0000088	gwes	minor	GWES: The RFTService should throw an exception if the output token contains a soap fault
0000054	gwes	minor	GWES Activity Statistics are sometimes wrong
0000032	gwes	minor	Exception Management for Activities
0000014	gwes	minor	GWES sends activity_terminated without activity_initiated
0000019	gwes	minor	error when getting workflowdescription when workflow is finished
0000008	gwes	minor	GWES: Problem when output place has capacity=1 and there are several input tokens
0000061	gwes	text	Document GWorkflowDL <property> elements processed by GWES
0000013	gwes	trivial	GWES: Use wsdl4j for parsing the wsdl file
0000080	gwes	feature	GWES should store all workflow tokens in XML database
0000066	gwes	feature	Test and optimize the scalability of the GWES
0000063	gwes	feature	Testscript for Remote GWES Test
0000031	gwes	feature	GUI for configuration of GWES
0000033	gwes	feature	Include Fault Tolerance Mechanisms

7.1.2. GWUI

0000075	gwui	crash	Portal GWUI applet sometimes hangs (firefox too)
0000037	gwui	major	upload new workflow after termination of old workflow hangs
0000093	gwui	minor	GWUI-0.5.2 sometimes does not display the workflow after calling graph layouter
0000071	gwui	minor	GWUI should center the workflow after loading it.
0000064	gwui	minor	Scrollbar of Workflow Inspector is deactivated if workflow is running, terminated or completed
0000027	gwui	minor	Invisible arrow heads
0000028	gwui	feature	visualize the state of the operations related to transistions
0000067	gwui	feature	GWUI: Provide buttons/user interface for all main GWES Web Service operations
0000043	gwui	feature	Button for storing workflows
0000060	gwui	feature	GWUI: Tokens with SOAPFaults

7.1.3. GWorkflowDL

ID	Category	Severity	Summary
0000025	gworkflowdl	minor	NullpointerException when parsing wrong GWorkflowDL
0000069	gworkflowdl	feature	Implement service for creating huge parallel and/or sequential workflows

8. CONTACT INFORMATION AND CREDITS

The following persons were involved in the development of the GWES and GWUI:

- Andreas Hoheisel, Fraunhofer FIRST (main developer and workpackage leader)
- Hans-Werner Pohl, Fraunhofer FIRST (developer – GworkflowDL)
- Tilman Linden, Fraunhofer FIRST (developer – GWUI)
- Helge Rose, Fraunhofer FIRST (developer – Prorater (Scheduler), Resource Matcher, Resource Updater)
- Armin Wolf, Fraunhofer FIRST (developer – D-GRDL, Resource Matcher)

For further information or bug reporting please contact Andreas Hoheisel (andreas.hoheisel@first.fraunhofer.de).

9. THE FRAUNHOFER FIRST LICENSE AGREEMENT

Copyright (c) 2005 Fraunhofer Gesellschaft/Fraunhofer FIRST. All rights reserved.

This software includes voluntary contributions made by Fraunhofer FIRST under the K-WfGrid-project. For more information on K-WfGrid, please see <http://www.kwfgrid.eu>.

Installation, use, reproduction, display, and modification of this software, with or without modification, in source and binary forms, are permitted, however only for Licensee's own scientific or educational purposes. Any use beyond – especially but not limited for commercial purposes – and/or distribution to third parties requires K-WfGrid's prior written consent. Any exercise of rights under this license by you is subject to the following conditions:

1. Any use of this software, with or without modification, must contain the above copyright notice and the above license statement as well as this list of conditions, in the software, the user documentation and any other materials provided with the software.
2. The user documentation, if any, must include the following notice: "This product includes software developed by K-WfGrid (www.kwfgrid.eu)." Alternatively, if that is where third-party acknowledgments normally appear, this acknowledgment must be reproduced in the software itself.
3. The names "K-WfGrid" and "Knowledge Workflow Grid" may not be used to endorse or promote software, or products derived therefrom, except with prior written permission by steffen.unger@first.fraunhofer.de.
4. You are under no obligation to provide anyone with any bug fixes, patches, upgrades or other modifications, enhancements or derivatives of the features, functionality or performance of this software that you may develop. However, if you publish or distribute your modifications, enhancements or derivative works without contemporaneously requiring users to enter into a separate written license agreement, then you are deemed to have granted participants in K-WfGrid a worldwide, non-exclusive, royalty-free, perpetual license to install, use, reproduce, display, modify, redistribute and sub-license your modifications, enhancements or derivative works, whether in binary or source code form, under the license conditions stated in this list of conditions.

5. DISCLAIMER

THIS SOFTWARE IS PROVIDED BY Fraunhofer FIRST AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. Fraunhofer FIRST AND CONTRIBUTORS MAKE NO REPRESENTATION THAT THE SOFTWARE, MODIFICATIONS, ENHANCEMENTS OR DERIVATIVE WORKS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.

6. LIMITATION OF LIABILITY

Fraunhofer FIRST AND CONTRIBUTORS SHALL HAVE NO LIABILITY TO LICENSEE OR OTHER PERSONS FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Fraunhofer FIRST takes no maintenance obligations.