



PERFORMANCE SERVICE INTERFACES AND DATA REPRESENTATION - USER MANUAL

WP3

Document Filename: **KWF-WP3-UIBK-v1.0-DRUserManual.doc**

Work package: **WP3**

Partner(s): **UIBK, CYFRONET, FIRST**

Lead Partner: **UIBK**

Document classification: **PUBLIC**

Abstract: This document describes in detail to the user the stable version of the performance service interfaces and data representation (DR) module developed in K-WfGrid.

Delivery Slip

	Name	Partner	Date	Signature
From	Hong-Linh Truong	UIBK	05/01/2006	
Verified by	Piotr Nowakowski	CYFRONET	07/10/2006	
Approved by	Steffen Unger	FIRST	09/10/2006	

Document Log

Version	Date	Summary of changes	Author
0.1	25/11/2005	First version	Hong-Linh Truong
0.2	01/12/2005	Update all chapters	Hong-Linh Truong
0.3	05/01/2006	Revise all chapters	Hong-Linh Truong
0.4	23/08/2006	Revise all chapters	Hong-Linh Truong
0.5	31/08/2006	Revise all chapters	Hong-Linh Truong
1.0	07/10/2006	QA check	Piotr Nowakowski

CONTENTS

1	COPYRIGHT NOTICE.....	4
2	INTRODUCTION.....	5
2.1	ABBREVIATIONS AND ACRONYMS.....	5
2.2	REFERENCES AND SOURCE CODE.....	5
3	SOFTWARE USAGE.....	8
3.1	RUNNING THE SOFTWARE.....	8
3.1.1	<i>Local hardware requirements.....</i>	8
3.1.2	<i>Local software requirements.....</i>	8
3.1.3	<i>Step-by-Step User Setup.....</i>	8
3.2	BASIC OPERATION.....	8
3.2.1	<i>Using XML schemas for data representations and requests.....</i>	8
3.2.2	<i>Binding XML data into a Java Object.....</i>	9
3.2.3	<i>Marshalling Java objects to XML data.....</i>	9
3.2.3.1	Using marshal() method.....	9
3.2.3.2	Using Marshaller object.....	10
3.2.4	<i>Using Ontologies.....</i>	10
4	CONTACT INFORMATION AND CREDITS.....	11
5	THE GPL LICENSE AGREEMENT.....	12
6	REFERENCES.....	13

1 COPYRIGHT NOTICE

Copyright (c) 2004-2006 Distributed and Parallel Systems Group, University of Innsbruck. All rights reserved.

Use of this product is subject to the terms and licenses stated in the GPL license agreement. Please refer to Section 5 for details.

Java™ is a registered trademark of Sun. All rights reserved.

Jena is copyrighted by Hewlett-Packard Development Company, LP.

Castor is copyrighted by Intalio Inc., and others.

Xerces is licensed under the Apache License, The Apache Software Foundation.

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

2 INTRODUCTION

This document describes the stable version of the performance service interfaces and data representations (DR) implemented in WP3 in the framework of the K-WfGrid project (hence DR refers to the performance service interfaces and data representations in K-WfGrid.). DR contains XML-based data representations and languages, and OWL-based ontologies [OWL] defined for

- Describing monitoring and performance data and specifying requests for controlling the performance data query and subscription.
- Describing the structure of workflow applications and specifying requests for controlling the instrumentation of workflows.
- Specifying performance analysis requests.
- Specifying search for performance problems requests
- Describing information about performance data and services, representing ontological form for performance data associated with Grid workflows [WPPERFONTO].

DR provides (standardized) representations for data and requests. It is used by various services which involve in performance analysis and monitoring, e.g., DIPAS and GEMINI, as well services which utilize performance and monitoring data, e.g. KAA and Scheduler. DR helps to simplify the interaction and integration among various K-WfGrid Grid services.

2.1 ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
DIPAS	Distributed Performance Analysis Service
DR	Performance Service Interfaces and Data Representation
GOM	Grid Organizational Memory
GWES	Grid Workflow Execution Service
KAA	Knowledge Assimilation Agent
OWL	Web Ontology Language
PDQS	Performance Data Query and Subscription
SIRWF	Standardized Intermediate Representation for Workflow
WARL	Workflow Analysis Request Language
WIRL	Workflow instrumentation request language

2.2 REFERENCES AND SOURCE CODE

All the source files of DR can be obtained from K-WfGrid CVS. DR consists of two parts

- XML-based data representations and requests: are in \$KWFGRID/dr directory
- OWL-based ontologies for performance data: are in \$KWFGRID/dipas/pmaonto directory

where \$KWFGRID indicates the root directory of the whole K-WfGrid CVS.

Fig. 2-1 depicts the directory structure of kwfgrid/dr which includes all the sources for XML-based data representations and requests

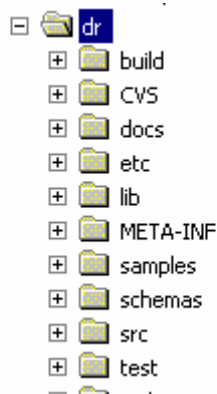


Fig. 2-1: Structure of source directory of XML-based data representations and requests.

Fig. 2-2 presents the structure of source directory of OWL-based ontologies for performance and monitoring data. The root directory pmaonto can be found under \$KWFGRID/dipas.

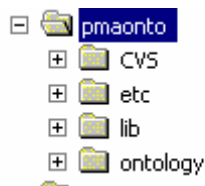


Fig. 2-2: Structure of source directory of OWL-based ontologies.

Table 2-1 shows the list of XML-based data representation and languages which have been defined and developed.

Category	Name	Description	Source File	Java package
Data Representation	host.cpu.used	CPU Usage of a computational node	schemas/cputime.xsd	net.kwfgrid.dr.hostcpuused
	service.available	Availability of services	schemas/ServiceAvailability.xsd	net.kwfgrid.dr.serviceavailable
	host.mem.used	Memory Usage of a computational node	schemas/memusage.xsd	net.kwfgrid.dr.hostmemused
	host.system.loadavg	Load average of a computational node	schemas/loadavg.xsd	net.kwfgrid.dr.hostsystemloadavg
	path.delay.roundtrip	Roundtrip delay of a network path measured at IP level	schemas/deplayroundtrip.xsd	net.kwfgrid.dr.pathdelayroundtrip
	path.bandwidth.capacity.TCP	TCP bandwidth of a network path	schemas/bandwidthtcp.xsd	net.kwfgrid.dr.bandwidthtcp
	host.sysinfo	System information of a computational node	schemas/hostsysinfo.xsd	net.kwfgrid.dr.hostsysinfo

	app.prof	Profiling data of applications	schemas/approfddata.xsd	net.kwfgrid.dr.appprof
	app.event	Application events	schemas/appevent.xsd	net.kwfgrid.dr.appevent
	wfa.event	Workflow events	schemas/genericeventdata.xsd	net.kwfgrid.dr.genericevent
	SIRWF	Standardized intermediate representation for workflow-based application	schemas/sirwf.xsd	net.kwfgrid.dr.sirwf
Requests	PDQS	Performance data query and subscription	schemas/pdqs.xsd	net.kwfgrid.dr.pdqs
	WIRL	Workflow instrumentation request language	schemas/wirl.xsd	net.kwfgrid.dr.wirl
	WARL	Workflow analysis request language	schemas/warl.xsd	net.kwfgrid.dr.warl

Table 2-1: XML-based data representations and requests.

Table 2-2 shows the list of OWL-based ontologies which have been defined and developed.

Name	Description	Source File
WfPerfOnto	Ontology describing performance data associated with workflows	ontology/wfperfonto.owl
WfMetricOnto	Ontology describing performance metrics	ontology/wfmetric.owl
Mondata	Ontology describing information about available performance and monitoring data	ontology/mondato.owl

Table 2-2: OWL-based ontologies for performance and monitoring data.

3 SOFTWARE USAGE

The stable version of DR includes

- a set of XML schemas for describing performance and monitoring data,
- OWL-based ontologies for describing information about monitoring data as well as performance data associated with workflow, and
- a Java-based library used for parsing and unparsing XML performance and monitoring data

3.1 RUNNING THE SOFTWARE

3.1.1 Local hardware requirements

No specific local hardware is required.

3.1.2 Local software requirements

The parsing/unparsing library is built from Java code automatically generated by Castor [CASTOR]. Thus, Castor and Java-based Xerces libraries [XERCES] are required. The ontology vocabularies are built by using Jena [JENA]. Thus an installation of Jena is required.

3.1.3 Step-by-Step User Setup

First, the source can be obtained from K-WfGrid CVS. The following steps are used to compile and install

Step 1: Move into the source directory, either \$KWFGRID/dr or \$KWFGRID/dipas/pmaonto where \$KWFGRID is the directory containing K-WfGrid source files.

Step 2: Generate stub codes and compile the source files

```
$ant jar
```

Step 3: To create java documents

```
$ant javadocs
```

Step 4: To run JUnit test.

```
$ant test
```

Step 5: To install the package. Firstly, edit the `build.properties` to change the location where schemas, ontologies and libraries should be installed. Secondly, type

```
$ant install
```

3.2 BASIC OPERATION

3.2.1 Using XML schemas for data representations and requests

There are two main tasks:

- given an XML document, parse (unmarshal) this data and bind the XML data to a Java object, or
- given a Java object holding monitoring data, analysis request, etc., marshal/serialize the Java object into XML document.

DR/schemas contain several XML schemas for multiple types of performance and monitoring data. Each type of data is described by an XML schema. We use Castor [CASTOR] to generate Java classes for binding XML data to Java objects. The generated Java classes are organized into packages. Basically, each package contains Java classes for processing each type of monitoring data. Generated Java classes include classes for unmarshalling/marshalling XML data/Java objects.

In the following, we explain how to use the parsing/unparsing library. Further information can be found in example programs in the source directory.

3.2.2 Binding XML data into a Java Object

Firstly, put XML data into a Java Reader object (e.g., `StringReader`, `BufferedReader`). For example,

```
StringReader reader = new StringReader(pdqsRequest);
```

with `pdqsRequest` is a string containing XML data.

Secondly, unmarshal the data in the Reader into a corresponding Java object. For example,

```
PDQS pdqs = (PDQS)PDQS.unmarshal(reader);
```

`PDQS` is the Java class that implements the mapping between Java objects and XML data described by the `PDQS` schema.

Here the important note is to select the right Java class for the right data representation. Refer to section 2.2 for existing Java packages.

3.2.3 Marshalling Java objects to XML data

There are two ways for marshalling Java objects into XML data. First, we can use method `marshal()` of the corresponding Java object to marshal the data hold by a Java object into XML data. Or, second, we can use `Marshaller` class provided by Castor to marshal the Java object data into XML data.

3.2.3.1 Using marshal() method

First, define a Java Writer where the XML data will be written. For examples,

```
StringWriter writer= new StringWriter();
```

Second, call the method `marshal()` of the Java object to store Java data into XML data. For example,

```
pdqs.marshal(writer);
```

writer now contains XML data and we can get XML string from writer, for instance, by using toString() method.

3.2.3.2 Using Marshaller object

First, import the class Marshaller:

```
import org.exolab.castor.xml.Marshaller ;
```

Second, define a Java Writer object. For example,

```
StringWriter writer= new StringWriter();
```

Third, define a marshaller

```
Marshaller marshaller = new Marshaller(writer);
```

Fourth, optionally, configure the marshaller. For example,

```
marshaller.setSuppressNamespaces(true);
```

Finally, marshal the data

```
marshaller.marshal(monitoringData);
```

with monitoringData is the Java object holding XML data.

3.2.4 Using Ontologies

Currently, using an ontology means that we describe information complying with the ontology. Then the information can be stored or utilized, depending the information creator and user. For example, mondata ontology is integrated into GOM and GEMINI can publish information about monitoring data by using mondata ontology.

4 CONTACT INFORMATION AND CREDITS

DR module is developed at:

Distributed and Parallel Systems Group,
Institute of Computer Science, University of Innsbruck
Technikerstrasse 21A, A-6020, Innsbruck, Austria

Developers:

- Hong-Linh Truong

With contributions from

- Peter Brunner
- Robert Samborski

Part of DR code is based on previous developments in SCALEA-G framework.

Visit <http://www.dps.uibk.ac.at/projects/pma> for further information about our work on performance monitoring and analysis. Further comments, suggestions, and bug reporting, contact Hong-Linh Truong at truong@dps.uibk.ac.at.

5 THE GPL LICENSE AGREEMENT

Copyright (c) 2004-2006 Distributed and Parallel Systems Group, University of Innsbruck. All rights reserved.

This software includes voluntary contributions made to K-WfGrid. For more information on K-WfGrid, please see <http://www.kwfgrid.eu>.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

6 REFERENCES

- [CASTOR] <http://www.castor.org/>
- [DRDEV] K-WfGrid Consortium, "K-WfGrid Performance Service Interfaces and Data Representation - Development Manual", August, 2006
- [JENA] Jena Semantic Web Toolkit –home page: <http://jena.sourceforge.net/>
- [JUNIT] <http://www.junit.org/>
- [OWL] "OWL Web Ontology Language Reference", <http://www.w3.org/TR/owl-ref/>
- [SCALEA-G] Hong-Linh Truong and Thomas Fahringer. "SCALEA-G: a Unified Monitoring and Performance Analysis System for the Grid", Scientific Programming, 12(4):225-237, IOS Press, 2004.
- [SIR] Clovis Seragiotto, Hong-Linh Truong, Thomas Fahringer, Bernd Mohr, Michael Gerndt, and Tianchao Li, "Standardized Intermediate Representation for Fortran, Java, C and C++ Programs", Technical Report AURORATR2004-18, Institute for Software Science, University of Vienna, October 2004. Also APART Working Group TR <http://www.kfajuelich.de/apart/>.
- [WPPERFONTO] Hong-Linh Truong, Thomas Fahringer, Francesco Nerieri, Schahram Dustdar "Performance Metrics and Ontology for Describing Performance Data of Grid Workflows, IEEE International Symposium on Cluster Computing and Grid 2005 (CCGrid2005), 1st International Workshop on Grid Performability, IEEE Computer Society Press, Cardiff, UK, 9 - 12 May 2005.
- [XPATH] XML Path Language, <http://www.w3.org/TR/xpath.html>
- [XERCES] <http://xerces.apache.org/>
- [XMLAPACHE] <http://xml.apache.org/>
- [XQUERY] XQuery: an XML Query Language, <http://www.w3.org/TR/xquery/>