GWD-R (draft-ggf-wsi-rus-19)
WS-I Resource Usage Service (RUS)
RUS-WG

Editors:
X. Chen, Brunel University
A. Khan, Brunel University
J. Ainsworth, ESNW
S. Newhouse, OMII-UK
J. MacLaren, CCT

March, 2007

WS-I Resource Usage Service (RUS) Core Specification ~~Resource Usage Service (RUS) based on WS-I Basic Profile 1.0~~

Status of this Memo

This document defines the Resource Usage Service (RUS) core specification, which is based on~~specification, which uses only~~ Web Services standards contained in the WS-I Basic Profile 1.0 [WSIBP10]. Distribution of this document is unlimited.

Copyright Notice

Abstract

This document describes the Resource Usage Service (RUS) core specification, a normative definition of service port types upon usage information in the format of Usage Record schema developed by OGF Usage Record Working Group. The port types being defined are intended to accommodate requirements on grid resource usage auditing and accounting as well grid economic model. This specification is based on established Web Service specifications contained within the WS-I Basic Profile 1.0 [WSIBP10~~specification as developed by the Market for Computational Services project within the UK e-Science programme. This proposed specification only uses established Web Service specifications contained within the WS-I Basic Profile 1.0 [WSIBP10], and does not require specifications in the OGSI or WSRF families. Therefore it is a 'plain web services' version of the of the original RUS specification [OGSI-RUS], which was dependent on OGSI. Consequently the port types are largely unchanged, whilst the service data elements no longer exist. WSDL and XML schemas are attached as appendices. The RUS uploads (and provides) record of resource usage through an XML document using a format developed by the Usage Record Working Group (UR-WG) in their 'Usage Record XML Format' document [UR-SPEC~~].

Contents

1.1.1

1 Introduction

This document describes the core service interface definitions based on WS-I Basic Profile 1.0 [WS-I Profile] for Resource Usage Service (RUS) that enables grid resource usage auditing and accounting as well as grid economic model. This document originates from the OGSI RUS specification [OGSI-RUS], and the original text has been retained where ever possible.~~service interface and behaviour of a WS-I Basic Profile 1.0 compatible Resource Usage Service (RUS) that enables the recording and retrieval of consumed resource information. This service is needed to provide information about service use to a variety of Grid entities:~~

The RUS service is needed to provide information about resource usage to a variety of Grid entities:

- The service manager who wishes to examine utilisation across their resources.
- A service that wishes to charge for the use of the consumed resources.
- A virtual organisation that wishes to monitor resource activity within their Grid~~.~~

1.1 Background

For resources being shared in the Grid, job usage information is required to be metered and collected in order to provide proofs for cost-benefits analysis, resource planning improvement, security enhancement, quality of service, and economic model. The Resource Usage Service (RUS) is therefore being defined in this document to provide a basic infrastructure to support auditing, accounting and other high-level capabilities requiring usage information and to allow entities within the grid to extract information from the service on potentially aggregate resource use.

1.2 Context

This specification is being defined based on following context:

- Job usage information is formalised in XML format as defined in OGF Usage Record [OGF-UR];
- Job usage information can be either centrally maintained or distributed maintained;
- The service interface definition specified in this document is based on WS-I Basic Profile 1.0 [WSIBP10];

1.3 Usage Scenarios

Within the UK, RUS is currently being developed to support two projects. The first is to support the UK e-Science Grid where the management team of this particular Vir-

tual Organisation (VO) wishes to monitor the consumed resources (e.g. job activity through the Globus).

The second application is as part of the UK e-Science Core programme project – A Market for Computational Services – is to record the consumed resources within grid services to generate charging information.

~~This document is based on the OGSI RUS specification [OGSI-RUS], and the original text has been retained where ever possible.~~

2 Overview

2.1 Architecture

The Resource Usage Service's primary purpose is to normalise operations upon usage records relating to the consumption of resources as described through the OGF Usage Records specification [OGF-UR].~~store records relating to the consumption of resources as described through the Usage Records specification [UR-SPEC]. While the Usage Records specification and the current deployment of the RUS has focussed around recording the resources consumed within computational batch jobs, there is no reason why the Usage Records schema could not be extended to record information relating to the invocation of a Web Service, to record the size of file transfers, etc. However, its primary goal is to collect resource usage information within a virtual organisation, by transferring information from the collection (client) point to the storage (service) point using the Usage Record specification, e.g. as within the UK e-Science Grid. There are therefore two primary functions: the upload of resource usage information into, and the extraction of resource usage data from the service.~~

A RUS implementation is therefore composed of two main components:

- A web service that implements the RUS interface specified in this document and provides for the storage, retrieval and management of records.
- A mechanism which provides for the persistent storage of records and query functionality across the stored data.

The document does not specify particular persistent storages or usage data sharing models. The usage files can be stored in databases, file systems and any other persistent storage. Usage data can also be shared in either central or distribute model.

It is the responsibility of implementations to provide storage management and ensure interoperability to other RUS implementations conforming to the service interface definitions specified within this document.

Another primary concern within RUS is to ensure confidentiality of usage information. The service implementation SHOULD provide access control mechanisms to expose appropriate usage information to suitable users.

2.2 Terminology and Notational Conventions

2.2.1 Terminology

User

~~A RUS implementation is therefore composed of two principal components:~~

- ~~A web service that implements the RUS interface and provides for the storage, retrieval and management of records.~~
- ~~A mechanism which provides for the persistent storage of records and query functionality across the stored data (e.g. an XML database)~~

~~It is the former that is specified in this document as the latter is implementation dependent and never exposed through the service interface.~~

~~Definitions and Notational Conventions~~

Throughout this document we will use the term ‘user’ as a generic term for a client to a RUS which may be an interactive client or a service instance interacting with a RUS instance.

Usage Record

The usage record in the document refers to a single job usage record (*urf:UsageRecord or urf:JobUsageRecord* element).

Usage File

The usage file in this document refers to the XML instance of usage record [OGF-UR] started with (*urf:UsageRecords*) and with one or more usage records embedded.

Usage Repository

The usage repository refers to the container of usage files. The usage repository could be databases, file system or any other type of data storages.

Context Node

The context node is the “context node” of XPATH [XPATH 1.0] expression.

2.2.2 Namespace

This specification uses namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 1: Prefixes and namespaces used in this specification.

Prefix	**Namespace**
wsdl	“http://schemas.xmlsoap.org/wsdl/”
http	"http://www.w3.org/2002/06/wsdl/http"
xsd	"http://www.w3.org/2001/XMLSchema"
xsi	“http://www.w3.org/2001/XMLSchema-instance”
urf	"http://schema.ogf.org/2003/09/urf"
types rus	“http://www.gridforum.org/2007/rus-wg/types” “http://www.gridforum.org/2007/rus-wg/core
~~**Prefix**~~	~~**Namespace**~~
~~wsdl~~	~~“http://schemas.xmlsoap.org/wsdl/”~~
~~http~~	~~"http://www.w3.org/2002/06/wsdl/http"~~
~~xsd~~	~~"http://www.w3.org/2001/XMLSchema"~~
~~xsi~~	~~“http://www.w3.org/2001/XMLSchema-instance”~~
~~urwg~~	~~"http://www.gridforum.org/2003/ur-wg"~~
~~rus~~	~~"http://www.gridforum.org/2005/rus-wg/types”~~

2.2.3 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

2.3 Scope

In the remainder of this document we describe the Web Service interface definitions and configuration requirements for implementation runtimes. Certain usage properties of OGF Usage Records schema [OGF-UR] are further clarified in the context RUS. The Web service description document (approved by Java code generation of Axis 1.4 and Axis2) of RUS are listed in the appendix. ~~to the RUS.~~

3 Configuration

3.1 Users and Authorisation

The security model (see Section 6) defined for the RUS is recommended to be role based. Two main roles are defined in this specification with minimum access control rules that implementations must ensure: ~~requires the identification of 'resource managers' who are permitted to contribute records and view the records that they have contributed and 'administrators' who can view all records. As the identities of these individuals are not exposed through the service interface there is no need to specify how this information is stored.~~

- the role of 'resource managers' who are permitted to contribute records and view the records that they have contributed;
- the role of 'administrators' may be senior to any other role to be defined and have management authorities on all usage records and storage(s) maintained in the RUS service instance as well as service management facilities (e.g.

As the identities of these individuals are not exposed through the service interface there is no need to specify how this information is stored.

3.1.1 Resource Manager

The resource manager has the right to view all activity relating to the resource for which they have managerial responsibility. Responsibility can be defined by resource-specific properties defined in [OGF-UR] including machine name, project name, submit host or other usage extension properties (see 3.13). Wildcard may be used to indicate pattern-matched usage properties, domain name for instance ("*.cfs.ac.uk" for machines in the ".cfs.ac.uk" domain).~~may be defined by machine name, project name, submit host or domain name (e.g., "*.cfs.ac.uk" for machines in the ".cfs.ac.uk" domain).~~

3.1.2 Administrator

The administrator has full read and full or partial write permission on all the records stored within the RUS depending on deployment environment. This is not restricted any further by a usage property~~write permission on all the records stored within the RUS. This is not restricted any further by a resource~~ identifier.

3.1.3 Other Users and extensions

An implementation may define other classes of users that have other access authorities to usage records. For roles identified by usage extension properties, an implementation should also establishes mapping between user identity and extended properties. For example, an extension property can be defined to provide the Virtual Organization properties as *<urf:Resource description="VOName" />*. The role

"VOManager" is therefore entitled to have authority to manage usage records belonging to specific VO property values. ~~degrees of access to the records.~~

1.2 ~~Mandatory Usage Record Elements~~

3.2 Fine-granularity Access Control

The permission model of RUS SHOULD provide fine-granularity access control on the contents of usage records rather than overall usage storage(s). Theoretically, every user should have access, at least read authority, to usage records maintained within RUS but the authorisation should based on partial element of each usage record (e.g. user identity for a grid user who is intended to read usage information of their jobs or resource manager who want to contribute records only with specific resource properties).

3.3 Mandatory Usage Properties

The Usage Record schema [OGF-UR], which is used by the RUS to describe the record of consumed resources being stored within the service, defines nearly all elements as optional. To ensure consistency and traceability across a set of Usage Records, some elements must be present. For example, it is useful to know who consumed what resources and when. To this end, the RUS can be configured to ensure that a set of elements is present (or mandatory) in a Usage Record. If a record is received without these elements the RUS will reject it as invalid. This document does not specify the configuration format of mandatory elements and leaves it implementation-dependent. The example below shows XML schema example for XML-based configuration for mandatory elements. Implementations may optionally chose other format of configuration (e.g. property file). For those implementations with custom usage properties using [OGF-UR] extensions, corresponding mapping SHOULD also be provided as mandatory properties.

```
<xsd:complexType name="MandatoryElementsType">
<xsd:choice minOccurs="1" maxOccurs="unbounded">
<xsd:element ref="urf:MachineName"/>
<xsd:element ref="urf:ProjectName"/>
<xsd:element ref="urf:SubmitHost"/>
… full list from ur-schema.xsd ...
</xsd:choice>
</xsd:complexType>
```

4 Usage Record Format in RUS

The usage records that move in or out of the RUS MUST be in the exact format as defined by OGF Usage Record schema [OGF-UR]. Clarification of certain usage properties are further described in this section in the context RUS runtime.

4.1 Record History

Each usage record retrieved or inserted into the RUS MUST have record history information represented by two properties defined in the Usage Record XML schema [OGF-UR]:

- The "createTime" property of usage record identity (*urf:RecordIdentity#createTime*) stating when the record is produced;
- The "keyInfo" property of usage record identity (*urf:RecordIdentity#keyInfo*) containing the "X509SubjectName" of the user entity that create the record;

Implementations that require other historic information (e.g. modification history) out of the scope of usage record representation may use Usage Record XML schema's extension framework and declare those properties as mandatory usage properties (see 3.3) or obtain historic information from runtime environment (e.g. logging system).

Considering unlimited number of usage records can be encapsulated within a single usage record file, the size of record instance may oversize the limitation of runtime system or database engine (e.g. less than 5MB per XML file for [Xindice]). It is the implementation's responsibility to enforce the size limitation at runtime. Implementations MAY alternatively restrict only one usage record per file but the usage record SHOULD also started with "urf:UsageRecords" element as a valid usage record instance.

~~The Usage Record schema [UR-SPEC], which is used by the RUS to describe the record of consumed resources being stored within the service, defines nearly all elements as optional. To ensure consistency and traceability across a set of Usage Records, some elements must be present. For example, it is useful to know who consumed what resources and when. To this end, the RUS can be configured to ensure that a set of elements is present (or mandatory) in a Usage Record. If a record is received with out these elements the RUS will reject it as invalid. The XML schema fragment for recording the mandatory Usage Record elements is shown below.~~

```
    <xsd:complexType name="MandatoryElementsType">
        <xsd:choice minOccurs="1" maxOccurs="unbounded">
            <xsd:element ref="urwg:MachineName"/>
            <xsd:element ref="urwg:ProjectName"/>
            <xsd:element ref="urwg:SubmitHost"/>
```

```
                    … full list from ur-schema.xsd ...
          </xsd:choice>
    </xsd:complexType>
```

Resource Usage Record Format

The record of resource consumption that moves in or out of the RUS is based around the Usage Record XML schema [UR-SPEC] and should be consulted in reading this section. The internal storage within the RUS of these records is an implementation issue.

Each usage record retrieved from the RUS is encapsulated within a rus:RUSUsageRecord element. This element contains:

1. A history list with 1 or more entries each encapsulated within a rus:RecordHistory element, which identifies who modified a record and when. The first entry in this list has the following form:
    a. A StoredBy element, containing a XMLDSIG KeyInfo element containing the X509SubjectName of the entity which stored the record.
    b. A TimeStamp element, simply a xsd:dateTime, stating when the record was stored.

    Subsequent entries have the form:
    a. A ModifiedBy element, containing a XMLDSIG KeyInfo element containing the X509SubjectName of the entity which modified the record.
    b. A TimeStamp element, simply a xsd:dateTime, stating when the modified record was stored.
2. A RUSRecordId element, an xsd:unsigned-long uniquely identifying the RUSUsageRecord element *within that RUS*.
3. A single UsageRecord element.

When you query a service for a record, you retrieve the RUSUsageRecord element described below, which encapsulates the original UsageRecord element. This means that the RecordHistory and RUSRecordId can be retrieved along with the associated Usage Record. An example of such an XML document is shown here:

```
<rus:RUSUsageRecord xmlns:rus="http://www.gridforum.org/2005/rus-wg/types">
 <rus:RecordHistory>
  <rus:StoredBy>
   <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:X509Data>
     <ds:X509SubjectName>
      CN=john ainsworth, L=MC, OU=Manchester, O=eScience, C=UK
     </ds:X509SubjectName>
    </ds:X509Data>
   </ds:KeyInfo>
  </rus:StoredBy>
  <rus:TimeStamp>13 January 2005 14:28:12 GMT</rus:TimeStamp>
 </rus:RecordHistory>
 <rus:RUSRecordId>19870</rus:RUSRecordId>
```

```
<UsageRecord xmlns=http://www.gridforum.org/2003/ur-wg
xmlns:urwg="http://www.gridforum.org/2003/ur-wg"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <RecordIdentity urwg:recordId="JSS-UNIQUE-ID"
urwg:createTime="2003-08-13T18:56:56Z" />
    <JobIdentity>
      <GlobalJobId>green147989</GlobalJobId>
      <LocalJobId>147989</LocalJobId>
    </JobIdentity>
  <UserIdentity>
    <LocalUserId>wwmarko</LocalUserId>
    <ds:KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      <X509Data>
        <X509SubjectName>
          CN=jon maclaren, L=MC, OU=Manchester, O=eScience, C=UK
        </X509SubjectName>
      </X509Data>
    </ds:KeyInfo>
  </UserIdentity>
  <JobName>------</JobName>
  <Status>completed</Status>
  <TimeDuration urwg:type="cpuTimeRequested">PT1800S</TimeDuration>
  <TimeDuration urwg:type="wallTimeRequested">PT1800S</TimeDuration>
  <TimeInstant urwg:type="timeSubmitted">2004-11-
29T06:47:30</TimeInstant>
  <Processors>1</Processors>
  <ProjectName>cs5015</ProjectName>
  <Host>green</Host>
  <CpuDuration>PT0.0S</CpuDuration>
  <WallDuration>PT1S</WallDuration>
  <StartTime>2004-11-29T06:48:33</StartTime>
  <EndTime>2004-11-29T06:48:34</EndTime>
  <MachineName>green</MachineName>
  <SubmitHost>wren</SubmitHost>
  <Queue>normal</Queue>
  <Resource urwg:description="quoteReference">contract1234</Resource>
  <Resource urwg:description="contractNumber">escience</Resource>
 </UsageRecord>
</rus:RUSUsageRecord>
```

4.2 Record Uniqueness

For usage records stored in RUS, a record identifier SHOULD ensure the global uniqueness of a single usage record. This is realised by use of the mandatory attribute, "*urf:RecordIdentity#recordId*" defined in Usage Record XML schema [OGF-UR]. The "recordId" property is formatted as string type. Implementations MAY optional transform the data type into numeric data type at runtime in order to obtain efficient record matching at runtime.

The schema for the `rus:RUSUsageRecord` is attached in Section .

5 Service Interface Definition

The RUS has two main functionalities to: upload and retrieve information. The information moved into or out of the RUS is contained within an XML document the syntax of which was described in section 4~~the previous section~~.

5.1 General

5.1.1 Operation Resul~~operationResult Element~~t

Whenever records are stored, retrieved, updated or deleted, there can be failures. It is required that RUS operations could return an operation result document to indicates the execution status of overall affective usage records as well as finer-granularity fault details (if only) of operations on individual usage records. The RUS fault framework is described in section 5.2. ~~In a correctly functioning RUS, three failures are possible: records might not be processed due to permission problems, a non-existent RUSRecordId may be used, a submitted Usage Record maybe invalid.~~

In order to allow the RUS to inform the client of what has happened, the rus:operationResult element is defined and~~we define a rus:operationResult element which~~ contains:

1. A mandatory "Statu~~TotalSuccess~~s" element (xsd:boolean). Only true if there were no records not processed (i.e. returned/inserted/modified/removed) due to problems.
2. An o~~O~~ptional "Processed" element (xsd:unsigned-long). The number of records successfully processed.
3. A sequence of faults that indicates the reason of unprocessed individual usage record~~Optional "PermissionDenied" element (xsd:unsigned-long). The number of records not processed due to permission problem~~s.

~~Optional "NonExistent" element (xsd:unsigned-long). The number of records not processed due to non-existent RUSRecordIds being used.~~

```
<xsd:element name="operationResult">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Status" type="xsd:Boolean" />
<xsd:element name="Processed" type="xsd:unsignedLong" minOccurs="0"
maxOccurs="1" />
<xsd:element ref="rus:generalFault" minOccurs="0"
manOccurs="unbounded">
</xsd:sequence>
</xsd:complexType>
```

4. ~~Optional "Invalid" element (xsd:unsigned-long). The number of records that were rejected as not conforming to either the Usage Record schema or the mandatory elements list.~~

5. ~~Optional "Duplicate" element (xsd:unsigned-long). The number of records that were rejected as duplicates of records already stored in the RUS.~~

~~Items 2, 3, 4, 5 and 6 are optional because a RUS may not want to say why something has failed, e.g. the RUS might not want the user to be able to distinguish a non-existent record with one which the user has not got permission to see. The XML schema fragment is shown below:~~

```
<xsd:element name="OperationResult">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Status" type="xsd:boolean"
minOccurs="1" maxOccurs="1"/>
            <xsd:element name="Processed" type="xsd:unsignedLong"
minOccurs="0" maxOccurs="1"/>
            <xsd:element name="PermissionDenied"
type="xsd:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="NonExistent" type="xsd:unsignedLong"
minOccurs="0" maxOccurs="1"/>
            <xsd:element name="Invalid"
type="xsd:unsignedLong" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="Duplicate" type="xsd:unsignedLong"
minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
6. </xsd:element>
```

5.1.2 Record Identity~~USRecordIdList Element~~

RUS operations allows finding usage records through flexible search terms (XPath expression) as well as record identifier, the "recordId" attribute of record identity (urf:RecordIdentity) defined in [OGF-UR]. This specification is not intended to define further record identifier for usage records maintained in RUS. Some implementations argues the string-pattern matching of usage record identifier may results in low performance, it is the implementation's responsibility to restrict the record indentifier as numeric format and transform the record identity value as corresponding numeric data type at runtime.

~~For insert, replace, modify and update operations, the RUS returns a list of xsd:long which has to be the same length as the supplied list of elements (to allow a client to determine which records were stored, and which failed). If the record is operated on successfully, its RUSRecordId is returned in this list. If the operation is unsuccessful for the record, then a return code is placed into the RUSRecordIdList to indicate the cause of failure. The order of the RUSRecordId elements in the RUSRecordIdList MUST correspond to the order of the records in the initial input operation.~~

~~These are defined as:~~

RUSRecordId	>0	Any positive non-zero integer indicates a valid RUS Record Id.
Unspecified	0	Used when the cause of the failure is to be kept private
PermissionDenied	-1	User is not authorised to perform the requested operation on this record
NonExistent	-2	The RUSRecordId does not exists
Invalid	-3	The supplied Usage Record is invalid when checked against the schema or has missing mandatory elements
Duplicate	-4	The record already exists in the RUS

The structure of the XML element is defined by the following schema fragment:

```
<xsd:element name="recordIdList complexType name="RUSRecordIdListType">
<xsd:complexTyp <xsd:sequence>
<xsd:sequence <xsd:element name="RUSRecordId" type="xsd:long" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="recordId" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/ </xsd:sequence>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

5.2 Faults

This section describes the RUS fault framework.ree faults may be raised by the RUS:

5.2.1 RusFaultUSProcessingFault – an internal error has occurred

There are several faults may be raised by the RUS:

- RusProcessingFault – an internal error occurs at runtime (e.g. service container, database engine);
- RusInvalidFault – invalid inputs detected by RUS service. Two possible invalid inputs of RUS are invalid usage records (validation-error against OGF-UR schema or against mandatory usage record element list) and invalid XPath expression[1].
- RusUnauthorisedFault – the user does not have permission to invoke the requested operation on the RUS
- RusDuplicatedFault – the fault is thrown when record identifier of usage records to be contributed to RUS conflicts the one of existing usage records in the storage;
- RusRecordNotFoundFault- User operates on usage records that not recognised by RUS.

[1] The invalidity of XPath expression is the operation-logic

These faults are grouped in RUS and collectively known as “RUS Faults”. Every fault defined in RUS MUST extend from the “RUSFault”, an abstract fault element, and substitute to that group so that the operation result element can recognise the faults. As the XML information model below, the “RUSFault” is an abstract element that only contains an optional fault message element.

```
<xsd:element name="RusFault" type="types:RusFaultType"
abstract="true" />
```

```
<xsd:complexType name="RusFaultType" abstract="true">
<xsd:sequence>
<xsd:element name=”faultMessage” type=”xsd:string” minOccurs=”0”
maxOccurs=”1” />
</xsd:sequence>
</xsd:complexType>
```

5.2.2 RusProcessingFault

The “RusProcesssingFault” encapsulates the record identity and runtime error message pairs. The information model of “RusProcessingFault” is described as below:

```
<xsd:element name="processingFaultMessage">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" />
<xsd:element name="runtimeFaultMsg" type="xsd:string" minOccurs="0"
maxOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

```
<xsd:element name="RusProcessingFault"
type="types:RusProcessingFaultType"
substitutionGroup="types:generalFault" />
```

```
<xsd:complexType name="RusProcessingFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:sequence>
<xsd:element ref="types:RusProcessingFaultMessage" minOccurs="0"
maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

5.2.3 RusInvalidFault

The “RusInvalidFault” encapsulates the errors because of invalid input of single usage record operation. There are four predefined invalid input types: invalid XPath

expression, invalid XUpdate expression, invalid usage records input against OGF Usage Record schema [OGF-UR], and invalid input because of missing mandatory elements. The information model of "RusInvalidFault" element is demonstrated as follows:

```
<xsd:simpleType name="invalidType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="invalidUsageRecord"/>
<xsd:enumeration value="missingMandatoryElements" />
<xsd:enumeration value="invalidXpath"/>
<xsd:enumeration value="invalideXupdate" />
</xsd:restriction>
</xsd:simpleType>
```

```
<xsd:element name="RusInvalidFault" type="types:RusInvalidFaultType"
substitutionGroup="types:generalFault" />
<xsd:complexType name="RusInvalidFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="0"
maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="invalidType" type="types:invalidType"
use="required" />
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

5.2.4 RusUnauthorisedFault

The "RusUnauthorisedFault" is thrown by RUS permission model, which acts as a gateway to RUS service endpoint. The "RusUnauthorisedFault" therefore does not support fine-granularity fault notification on individual usage records. As the information model below, the "RusUnauthorisedFault" indicates user identity and the mismatched role the user is claimed to be.

```
<xsd:element name="RusUnauthorisedFault"
type="types:RusUnauthorisedFaultType"
substitutionGroup="types:generalFault" />
```

```
<xsd:complexType name="RusUnauthorisedFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:attribute name="user" type="xsd:string" use="required" />
<xsd:attribute name="role" type="xsd:string" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

5.2.5 RusDuplicateFault

The "RusDuplicateFault" encapsulates a list of record identities of usage records to be inserted which already exist in the usage repository. The information model of "RusDuplicateFault" is illustrated as follows:

```
<xsd:element name="RusDuplicatedFault"
type="types:RusDuplicatedFaultType"
substitutionGroup="types:generalFault" />
```

```
<xsd:complexType name="RusDuplicatedFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"
</xsd:sequence>
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

5.2.6 RusRecordNotFoundFault

The "RusRecordNotFoundFault" encapsulates the record identities of usage records that does not exist in the usage repository. The information model of "RusRecordNotFoundFault" is described as follows:

```
<xsd:element name="recordNotFoundFault"
type="types:recordNotFoundFaultType"
substitutionGroup="types:generalFault" />
```

```
<xsd:complexType name="recordNotFoundFaultType">
<xsd:complexContent>
<xsd:extension base="types:generalFaultType">
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="0"
maxOccurs="1" />
</xsd:sequence>
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

- ~~RUSInputFault – the parameters supplied when an operation was invoked are incorrect. For example, this fault is raised if the supplied XML is invalid when checked against the schema~~
- ~~RUSUserNotAuthorisedFault – the user does not have permission to invoke the requested operation on the RUS~~

~~**Permission Model**~~

~~The Distinguished Name from the client's certificate must be known to the RUS's access control mechanism (see 1.2) otherwise the RUSUserNotAuthorisedFault message is returned. Authorisation for the user to perform the requested operation is determined for each individual record.~~

- **Storage**

5.2.7 RUS::insertUsageRecords

The `insertUsageRecords` port type enables users to populate usage files into usage repository~~laces new usage records into the RUS~~.

Input

- *Mandatory:* A list of usage files (*urf:UsageRecords* XML String);~~*List of Usage Record elements*: A list of *UsageRecord* elements as defined in [UR-SPEC].~~

Output

- *Mandatory: OperationResult* element: As defined in section 5.1.1.

~~*Mandatory: RecordIdList element: As defined in section 5.1.2.*~~

The following XML schema fragment defines the combined input for this port type.

```
<xsd:element name="insertUsageRecordsRequest">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="usagerecords" type="xsd:string" minOccurs="1"
maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>
```

~~The following XML schema fragment defines the combined output for this port type.~~

```
<xsd:element name="recordListOperationResponse" >
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="rus:OperationResult"/>
            <xsd:element ref="rus:RUSRecordIdList"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Faults

- RusP~~USPr~~ocessingFault
- RusUnauthorisedFault~~USUserNotAuthorised~~
- RusDuplicateFault
- RusInvalidFault

The `insertUsageRecords` port type differentiates from the `replaceUsageRecords` port type in that it allows users to insert new usage files into the RUS. The success or failure to insert a *UsageRecord* is recorded in the operation result document. The following operation result example indicates the insert operation status with one usage record failed because of failed schema validation against OGF-UR schema and two failures because of lacking mandatory elements.

```
<operationResult Status="false" Processed="7">
<RusInvalidFault total="1" type="invalidUsageRecords">
<recordId>2345</recordId>
<faultMessage>the record(s) must be OGF-UR compatible</faultMessage>
</RusInvalidFault>
<RusInvalidFault total="2" type="missageMandatoryElements">
<recordId>198437</recordId>
<recordId>274657</recordId>
<faultMessage>the record(s) must contains usage properties as
mandatory element list configuration. Query element list via
rus:listMandatoryUsageRecordElements port type for more details.
</faultMessage>
</RusInvalidFault>
</operationResult>
```

~~The `insertUsageRecords` operation differs from the `replaceUsageRecord` operation in that it will insert a new usage record into the RUS. The success or failure to insert a *UsageRecord* is recorded in the output *RUSRecordIdList* element.~~

1.3 Extraction

Extraction port types enable users to find out usage records relating to certain search term.

5.2.8 RUS::extractSpecUsageRecords

Enable the client to find out usage records using record identifier as keywords.

Input

- *Mandatory:* a list of record identifier as string list;

Output

- *Mandatory: OperationResult* element defined in section 5.1.1;
- *Mandatory:* The single usage file (*urf:UsageRecords* XML string) encapsulating matched usage records (urf:UsageRecord or urf:JobUsageRecord element);

The following XML schema fragment defines the output of extraction port types.

```
<xsd:element name="extractionResponse">
<xsd:complexType>
```

```
<xsd:sequence>
<xsd:element name="usagerecords" type="string" />
<xsd:element ref="types:operationResult" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

1.3.1 ~~RUS::extractRUSUsageRecords~~

~~Enable the client to find all usage records relating to a more complicated set of requirements.~~

Faults

- RusProcessingFault;
- RusUnauthorisedFault;
- RusRecordNotFoundFault;

5.2.9 RUS::extractUsageRecords

Enable the client to find out usage records relating to search criteria.

Input

- *Mandatory:* XPath expression as string;~~*searchTerm (1)*: All resource usage records relating to the search criteria that meet the specified access rules should be returned to the client. The search term should be specified as part of an XPath/XQuery string.~~

Output

- *Mandatory:* *OperationResult* element defined in section 5.1.1;~~Zero or more *RUSUsageRecord* elements.~~
- *Mandatory:* The single usage file (*urf:UsageRecords* XML string) encapsulating matched usage records (urf:UsageRecord or urf:JobUsageRecord element);~~*OperationResult.*~~

These two outputs are combined into a single *extractRecordsResponse* element, as in the following XML schema fragment.

```
<xsd:element name="extraction~~RUSUsageRecords~~Response">
~~------~~<xsd:complexType>
~~----------~~<xsd:sequence>
<xsd:element ref="urf:UsageRecords" ~~----------<xsd:element ref="rus:OperationResult"~~/>
<xsd:element ref="types:operationResult" ~~-------<xsd:element ref=”rus:RUSUsageRecord” minOccurs="0" maxOccurs="unbounded"~~/>
~~----------~~</xsd:sequence>
~~------~~</xsd:complexType>
</xsd:element>
```

Faults

- RusUSProcessingFault
- RusUnaUSUserNotAuthorised
- RusInvalidFault with invalid type of "invalidXpath" when the context node is not "urf:UsageRecord or urf:JobUsageRecord";US::extractRUSRecordIds

5.2.10 RUS::extractRecordIds

Enable the client to find all usage records relating to a more complicated set of requirements, returning just the record identifiers and not the full usage records.

Input

- *Mandatory:* XPath expression as string;*searchTerm*: the identifiers of all resource usage records relating to the search criteria that meet the specified access rules should be returned to the client. The search term should be specified as part of an XPath/XQuery string.

Output

- *Mandatory: a list of record identities*;*RUSRecordIdList* element..
- *Mandatory: OperationResult* element defined in section 5.1.1.

The following XML schema fragment defines the output of this port typese two outputs are combined into a single *extractRUSRecordIdsResponse* element, as shown in the following XML schema fragment.

```
<xsd:element name="extractUsageRUSRecordIdsResponse">
    <xsd:complexType>
        <xsd:sequence>
<xsd:element ref="types:recordIdList" />        <xsd:element ref="rus:OperationResult"/>
<xsd:element ref="types:operationResult" />        <xsd:element ref="rus:RUSRecordIdList"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Faults

- RusUSProcessingFault
- RusUnaUSUserNotAuthorised
- RusInvalidFault

5.3 Modification

Modification port types defined in this section allows users to execute update operation on contents of usage records. The modification port types differ from insert and deletion port types that work in the scope of usage files.

5.3.1 RUS::modifySpecUsageRecords

This port type allows users to modify a set of usage records identified by record identity with XUpdate expression. The charge service, for example, could make use of this port type to insert charge information to a usage record as with usage information calculated.

[**Note:** Implementation of this port type should protect record identity and record history properties from being modified.]

[**Note:** Implementation of this port type should ensure data consistence and synchronization when multiple modification operations are executed at same time.]

~~**These operations all modify an existing RUS record in some way. The `rus:RecordHistory` MUST be appended with a ModifiedBy element by these operations.**~~

~~RUS::incrementUsageRecordPart~~

~~This operation will add a numeric increment to an element in a single record. This gives a safe way for multiple writers to update the same record.~~

Input

- *Mandatory:* a list of record identifier as string list;~~*RUSRecordId element*: The record to be updated~~
- *Mandatory:* XUpdate expression as string;~~*ElementPath element (xsd:string = the XPath expression):* This expression should identify a single element within the record which is of some numerical type. The query expression is applied to the UsageRecord element, not the surrounding RUSUsageRecord element, to prevent the alteration of a RUSRecordId.~~

The following XML schema fragment defines the input of this port type.

```
<xsd:element name="modifySpecUsageRecordsRequest">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="recordIdIdList" type="xsd:string" />
<xsd:element ref="types:xupdateExpression" >
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

- ~~***Mandatory: Increment element (xsd:long):* The increment to be applied to the identified element~~
- **Output**
  - *Mandatory: OperationResult* element defined in section 5.1.1;~~: In this case, this refers to the successful location of an accessible record, not the overall success of the operation.~~

~~*Mandatory: Modified element (xsd:boolean):* This will be true if only if the record was updated correctly, which implies that the record was found, permissions were OK, *and* the XPath identified a single numeric element~~

Faults

- Rus~~US~~ProcessingFault
- RusUnauthorisedFault~~USUserNotAuthorised~~
- RusInvalidFault ~~US::modifyUsageRecordPart~~
- RusRecordNotFoundFault

5.3.2 RUS::modifyUsageRecords

This operation will modify a set of Usage Records according to a XUpdate expression. The resource manager, for example, may use this port type to modify resource-specific property (e.g. *urf:MachineName*) values.

[**Note:** Implementation of this port type should protect record identity and record history properties from being modified.]

[**Note:** Implementation of this port type should also ensure synchronisation of property values of usage records with values in the configuration. For example, the change of machine name should also affect the change of corresponding machine name value defined in configuration file of the resource manager.]

~~This operation will modify a Usage Record according to an XUpdate expression. Note that this expression will be applied to a single UsageRecord element, NOT the surrounding RUSUsageRecord element, thus preventing the modification of the RUSRecordId, time of storage, etc.~~

[**Note:** Implementation of this port type should ensure data consistence and synchronization when multiple modification operations are executed at same time.]

~~It is possible to create races between multiple writers depending upon the XUpdate expressions used.~~

Input

- *Mandatory: XUpdate* expression string;~~*RUSRecordId element*: The element to be updated~~

Mandatory: XUpdate element: This element will dictate how the usage record will be updated.

Output

- *Mandatory: OperationResult element*: *In this case, this refers to the successful location of an accessible record, not the overall success of the operation*.

Optional: XUpdate results: If the RUS record was found, this part MUST be present.

Faults

- RusUSProcessingFault
- RusUnauthorisedFaultUSUserNotAuthorised
- RusInvalidFault

5.3.3 RUS::replaceUsageRecords

The `replaceUsageRecords` replaces records held in the RUS.

We define a *ReplacementRecord* element, which contains a *RUSRecordId* element, and a *UsageRecord* element, as defined in [UR-SPEC].

Input

- *Mandatory: List* of usage records embedded into a single usage file as string:*ReplacementRecord elements*: A list of elements, each being a paired RUSRecordId and usage record.

The following XML schema fragment defines this input:

```
<xsd:complexType name="ReplacementRecordType">
    <xsd:sequence>
        <xsd:element name="RUSRecordId" type="xsd:unsignedLong"
minOccurs="1" maxOccurs="1"/>
        <xsd:element ref=”urwg:Usage" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:element name="ReplacementRecords">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="List" type="ReplacementRecordType"
minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Output

- *Mandatory: RUSRecordIdList element: This is a list of elements which MUST be the same length and order as the inputted list. Each element in this list will be a RUSRecordId (an xsd:long).*
- *Mandatory: OperationResult element*

~~**The following XML schema fragment defines the combined output for this port type.**~~

```
~~<xsd:element name="RecordListOperationResult" >~~
~~    <xsd:complexType>~~
~~        <xsd:sequence>~~
~~            <xsd:element ref="rus:OperationResult"/>~~
~~            <xsd:element ref="rus:RUSRecordIdList"/>~~
~~        </xsd:sequence>~~
~~    </xsd:complexType>~~
~~</xsd:element>~~
```

Faults

- RusUSProcessingFault
- RusRecordNotFoundFaultUSUserNotAuthorised
- RusUnauthorisedFault

5.4 Deletion

5.4.1 RUS::deleteSpecificUsageRecords

Enable the client to delete remove usage record(s) from a usage file;~~all usage records that match the specified criteria.~~

Input

- *Mandatory:* a list of record identifier as string list;~~*searchTerm (1)*: All resource usage records relating to the search criteria that meet the specified access rules should be returned to the client. Defined in the same way as *searchTerm* in extractRecords, Section 1.1.2.~~

Output

- ~~*Mandatory: RUSRecordIdList element: As defined in section 5.1.2. This will provide the RUSRecordIds of the records that have been deleted and a reason as to why the record has not been deleted.*~~
- *Mandatory: OperationResult element*

Faults

- RusUSProcessingFault
- RusUnauthorisedFaultUSUserNotAuthorised
- RusRecordNotFoundFault

5.4.2 RUS::deleteUsage~~Specific~~Records

Enable the client to delete all usage records with the specified RUSRecordIds

Input

- *Mandatory:* XPath expression as string;~~*RUSRecordIdList element*: This is a list of RUSRecordId elements which contain the RUSRecordIds of the records to be deleted.~~

Output

- ~~*Mandatory: RUSRecordIdList element. As defined in section 5.1.2. This will provide the RUSRecordIds of the records that have been deleted and a reason as to why the record has not been deleted.*~~
- *Mandatory: OperationResult element*

Faults

- RUSProcessingFault
- RusUnauthorisedFault~~USUserNotAuthorised~~
- RusInvalideFault

5.5 Configuration

5.5.1 RUS::listMandatoryUsageRecordElements

Enable the client to retrieve a list of the usage record elements required by this RUS implementation.

Input

- *None*

Output

- *Optional: MandatoryElements element*
- *Mandatory: OperationResult element*

Faults

- RUSProcessingFault
- RUSUserNotAuthorised

6 Security Considerations

As the RUS is a system on which financial transactions may ultimately depend, security is of the utmost importance. The following measures are required to ensure that service consumers trust service providers to record the usage correctly and that fraudulent use is prevented or traceable. This is the usual trust model employed when consumption is metered.

6.1 Authentication

Underpinning much of the following discussion is the ability to authenticate users. A RUS implementation must accept messages transported in a way that allows user authentication (e.g. mutual TLS or signed messages using WS-Security). If confidentiality is required then the messages between the RUS client and the RUS can be encrypted (e.g. TLS or MLS within WS-Security).

6.2 Authorisation

The authentication data is used to implement a role-based access control model to govern access to the RUS on a per record basis thereby ensuring the privacy of the stored data. A minimal model, defining administrators and resource managers, is defined in section 3.1. For example, a resource manager must be able to only retrieve usage information relating to their resources, and no one else's. Likewise, write access must be restricted to those entities that own the metered resources.

6.3 Audit

The data stored in the RUS must, from the perspective of the RUS, be non-repudiatable. It is not the responsibility of the RUS to determine if the data it is storing is correct, but it must be able to demonstrate the source of this data and how it may have been altered by subsequent operations if it is challenged. This can be achieved by ensuring message integrity between RUS client and RUS, and by maintaining an audit trail of each operation on the RUS that changes the usage data. The audit trail should include the message requesting the operation, the identity of the entity performing the operation, a digital signature of the message and the date and time the operation was requested. The message and its signature are not part of the urf:UsageRecords~~rus:RUSUsageRecord~~ but should be recorded by a RUS so that they are available for audits and resolving disputes.

7 Editor Information

Xiaoyu Chen
BITLab
Brunel University
Uxbridge Middlesex
London UB8 3PH
United Kingdom
Email: xiaoyu.chen@brunel.ac.uk

Akram Khan
School of Engineering and Design
Brunel University
Uxbridge Middlesex
London UB8 3PH
United Kingdom
Email: akram.khan@brunel.ac.uk

John Ainsworth
e-Science NorthWest
Kilburn Building
University of Manchester
Manchester M13 9PL
United Kingdom
Email: john.ainsworth@manchester.ac.uk

Steven Newhouse
Open Middleware Infrastructure Institute UK
Suite 6005, Faraday Building (B21), Highfield Campus,
University of Southampton, Highfield, Southampton, SO17 1BJ
United Kingdom
Email: s.newhouse@omii.ac.uk

Jon MacLaren
Centre for Computation and Technology (CCT)
352 Johnston Hall
Louisiana State University
Baton Rouge, LA 70803
United States of America
Email: maclaren@cct.lsu.edu

8 Contributors

The editors would like to acknowledge the contribution of Anthony Mayer (LeSC) and Asif Saleem (LeSC) for discussions relating to this document.

9 Acknowledgements

The creation of this document has been supported by the Computational Markets project funded under the UK e-Science Core programme by the Department of Trade and Industry and the Engineering and Physical Sciences Research Council (http://www.lesc.ic.ac.uk/markets/). Additional support to attend Global Grid Forum meetings has been provided by the UK e-Science GridNet programme.

10 Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

11 Full Copyright Notice

12 References

12.1 Normative References

[RFC 2119]

Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, RFC 2119, March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt

[WSDL 1.2]

Web Services Description Language (WSDL) Version 1.2, Published W3C Working Draft, World Wide Web Consortium. Available at http://www.w3.org/TR/wsdl12/

[WSDL 1.2 DRAFT]

Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 3 March 2003, World Wide Web Consortium. Available at http://www.w3.org/TR/2003/WD-wsdl12-20030303

[WSIBP10]

Web Services Interoperability Basic Profile Verion 1.0, Web Services Interoperability Organization. Available at http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[OGF-UR~~UR-SPEC~~]

Usage Record – XML Format, Draft GGF Recommendation. Available from: http://www.psc.edu/~lfm/Grid/UR-WG/

12.2 Informative References

[OGSI-RUS]

S. Newhouse and J. MacLaren, "Resource Usage Service RUS" Global Grid Forum Resource Usage Service Working Group draft-ggf-rus-service-4. Available online at https://forge.gridforum.org/projects/rus-wg/document/draft-ggf-rus-service-4-public/en/1

[JAX-RPC]

Java™ API for XML-Based RPC (JAX-RPC). *http://java.sun.com/xml/jaxrpc/docs.html*

[Web Services Book]

Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, s. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, R. Neyama. Sams, 2001.

[WSIF]

Welcome to WSIF: Web Services Invocation Framework, http://www.apache.org/wsif/

2 XSD and WSDL Specifications

This section contains the full XSD and WSDL definitions for everything described in this document. The definitions in this section MUST be considered definitive, if there are any discrepancies between the definitions in this section and those portions described in other sections above.

2.1 RUS Base Types schema

```
<xsd:schema
        targetNamespace="http://www.gridforum.org/2005/rus-wg/types"
        elementFormDefault="qualified"
        xmlns="http://www.gridforum.org/2005/rus-wg/types"
```

13 Appendix

13.1 rus-core-types.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema  targetNamespace="http://www.gridforum.org/2007/rus-
wg/core/types"
elementFormDefault="qualified"
attributeFormDefault="qualified"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:types="http://www.gridforum.org/2007/rus-wg/core/types"
xmlns="http://www.w3.org/2001/XMLSchema">
<xsd:element name="xpathExpression" type="xsd:string" />

<xsd:element name="xupdateExpression" type="xsd:string" />

<xsd:element name="recordIdList">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="1"
maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
        xmlns:urwg="http://www.gridforum.org/2003/ur-wg"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">

        <xsd:import namespace="http://www.gridforum.org/2003/ur-wg"
schemaLocation="./urwg-schema-11.xsd" />

        <xsd:complexType name="MandatoryElementsType">
            <xsd:choice minOccurs="1" maxOccurs="unbounded">
                <xsd:element ref="urwg:MachineName"/>
                <xsd:element ref="urwg:ProjectName"/>
                <xsd:element ref="urwg:SubmitHost"/>
                <xsd:element ref="urwg:Processors"/>
                <xsd:element ref="urwg:Memory"/>
                <xsd:element ref="urwg:Disk"/>
```

```
                <xsd:element ref="urwg:Network"/>
                <xsd:element ref="urwg:TimeDuration"/>
                <xsd:element ref="urwg:CpuDuration"/>
                <xsd:element ref="urwg:StartTime"/>
                <xsd:element ref="urwg:EndTime"/>
                <xsd:element ref="urwg:NodeCount"/>
                <xsd:element ref="urwg:Queue"/>
                <xsd:element ref="urwg:ServiceLevel"/>
                <xsd:element ref="urwg:WallDuration"/>
                <xsd:element ref="urwg:UserIdentity"/>
                <xsd:element ref="urwg:JobIdentity"/>
            </xsd:choice>
        </xsd:complexType>

        <xsd:element name="RUSUsageRecords"
type="RUSUsageRecordsType" />

        <xsd:element name="OperationResult">
            <xsd:complexType>
            <xsd:sequence>
                    <xsd:element name="Status" type="xsd:boolean"
minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="Processed"
type="xsd:unsignedLong" minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="PermissionDenied"
type="xsd:unsignedLong" minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="NonExistent"
type="xsd:unsignedLong" minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Invalid" type="xsd:unsignedLong"
minOccurs="0" maxOccurs="1"/>
                    <xsd:element name="Duplicate"
type="xsd:unsignedLong" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

        <xsd:complexType name="WhoAndWhenType">
            <xsd:sequence>
                    <xsd:element ref="ds:KeyInfo"/>
                    <xsd:element name="TimeStamp" type="xsd:dateTime"
maxOccurs="1" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="RecordHistoryType">
            <xsd:sequence>
                    <xsd:element name="StoredBy"
type="WhoAndWhenType"/>
                    <xsd:element name="ModifiedBy"
type="WhoAndWhenType" maxOccurs="unbounded" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>

<xsd:element name="operationResult"><xsd:complexType
name="RUSUsageRecordType">
<xsd:complexTyp<xsd:sequence>
```

```
<xsd:sequence>                <xsd:element name="RecordHistory"
type="RecordHistoryType"/>
<xsd:element name="Status" type="xsd:string" />
<xsd:element name="RUSRecordId" type="xsd:long"/>
<xsd:element name="Processed" type="xsd:unsignedLong" minOccurs="0"
maxOccurs="1"                <xsd:element ref="urwg:Usage"/>
<xsd:element ref="types:RusFault" minOccurs="0" maxOccurs="unbounded"
/>          </xsd:sequence>
</xsd:sequenc     </xsd:complexType>
</xsd:complexType>
</xsd:element>

<!-- RUS Faults -->
<xsd:element name="RusFault" type="types:RusFaultType"
abstract="true" />
<xsd:complexType name="RusFaultType" abstract="true">
<xsd:sequence>
<xsd:element name="faultMessage" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>

<xsd:element name="RusUnauthorisedFault"
tye="types:RusUnauthorisedFaultType"
substitutionGroup="types:RusFault" />

<xsd:complexType name="RusUnauthorisedFault     <xsd:complexType
name="RUSUsageRecordsType">
<xsd:complexContent          <xsd:sequence>
<xsd:extension base="types:RusFaultType"          <xsd:element
name="RUSUsageRecord" type="RUSUsageRecordType" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:attribute name="user" type="xsd:string" use="required" />
</xsd:sequence>
<xsd:attribute name="role" type="xsd:string" use="required" />
</xsd:complexType>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="RusRecordNotFoundFault"
type="types:RusRecordNotFoundFaultType"
substitutionGroup="types:RusFault" />
<xsd:complexType name="RusRecordNotFoundFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="0"
maxOccurs="1" />
</xsd:sequence>
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="RusProcessingFaultMessage">
<xsd:complexType>
```

```
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" />
<xsd:element name="runtimeFaultMsg" type="xsd:string" minOccurs="0"
maxOccurs="1" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="RusProcessingFault"
type="types:RusProcessingFaultType"
substitutionGroup="types:RusFault" />

<xsd:complexType name="RusProcessingFault<xsd:complexType
name="ReplacementRecord">Type">
<xsd:complexContent<xsd:sequence>
<xsd:extension base="types:RusFaultType"
<xsd:element name="RUSRecordId" type="xsd:long" minOccurs="1"
maxOccurs="1"/>
<xsd:sequence<xsd:element ref="urwg:Usage"
minOccurs="1" maxOccurs="1"/>
<xsd:element ref="types:RusProcessingFaultMessage" minOccurs="0"
maxOccurs="unbounded" /</xsd:sequence>
</xsd:sequenc</xsd:complexType>e>
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="invalidType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="invalidUsageRecords"/>
<xsd:enumeration value="missingMandatoryElements" />
<xsd:enumeration value="invalidXpath"/>
<xsd:enumeration value="invalideXupdate" />
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="RusInvalidFault" type="types:RusInvalidFaultType"
substitutionGroup="types:RusFault" />
<xsd:complexType name="RusInvalidFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="0"
maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="invalidType" type="types:invalidType"
use="required" />
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:element name="RusDuplicatedFault"
type="types:RusDuplicatedFaultType"
substitutionGroup="types:RusFault" />
```

```
        <xsd:element name="ReplacementRecords">
                <xsd:complexType>
                <xsd:sequence>
                <xsd:element name="List" type="ReplacementRecordType"
minOccurs="1" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

        <xsd:element name="RUSRecordIdList">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="RUSRecordId" type="xsd:long"
minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>


            <xsd:complexType name="RUSRecordIdListType">
                <xsd:sequence>
                    <xsd:element name="RUSRecordId" type="xsd:long"
minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
</xsd:schema>
```

Service Interface WSDL specifications

2.1.1 resource-usage-porttype.wsdl

```
<wsdl:definitions name="resourceusage"
    targetNamespace="http://www.gridforum.org/2005/rus-wg/service"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:intf="http://www.gridforum.org/2005/rus-wg/service"
    xmlns:types="http://www.gridforum.org/2005/rus-
wg/service/types"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:urwg="http://www.gridforum.org/2003/ur-wg"
    xmlns:rus="http://www.gridforum.org/2005/rus-wg/types"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <types>
        <xsd:schema
            targetNamespace="http://www.gridforum.org/2005/rus-
wg/service/types"
            elementFormDefault="qualified"
            attributeFormDefault="qualified"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:rus="http://www.gridforum.org/2005/rus-wg/types"
          xmlns:urwg="http://www.gridforum.org/2003/ur-wg"
            xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
          xmlns="http://www.w3.org/2001/XMLSchema">
```

```
            <xsd:import namespace="http://www.gridforum.org/2003/urwg" schemaLocation="./urwg-schema-11.xsd" />
            <xsd:import namespace="http://www.gridforum.org/2005/rus-wg/types" schemaLocation="./RUS.xsd" />

            <xsd:element name="RusInputFault" type="xsd:string"/>
            <xsd:element name="RusProcessingFault" type="xsd:string"/>
            <xsd:element name="RusUserNotAuthorisedFault" type="xsd:string"/>

            <xsd:element name="recordListOperationResponse" >
                <xsd:complexType>
                    <xsd:sequence>
                    <xsd:element ref="rus:OperationResult"/>
                    <xsd:element ref="rus:RUSRecordIdList"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

<xsd:complexType name="RusDuplicatedFaultType">
<xsd:complexContent>
<xsd:extension base="types:RusFaultType">
<xsd:sequence>
<xsd:element name="recordId" type="xsd:string" minOccurs="0" maxOccurs="unbounded" />
</xsd:sequence>
<xsd:attribute name="total" type="xsd:unsignedLong" use="required" />
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- insert data types -->
<xsd:element name="insertUsageRecordsRequest">
        <xsd:complexType>
            <xsd:sequence>
<xsd:element name="usagerecords" type="xsd:string" minOccurs="1" maxOccurs="unbounded" <xsd:element name="records" type="urwg:UsageRecordType" minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

<!-- extraction data types -->
<xsd:element name="extractionResponse"  <xsd:simpleType name="xPathQuery" type="xsd:string"/>
<xsd:complexType>
<xsd:sequence  <xsd:element name="extractRUSUsageRecordsResponse">
<xsd:element name="usagerecords" type="string" /
<xsd:complexType>
<xsd:element ref="types:operationResult" /
<xsd:sequence>
</xsd:sequence  <xsd:element ref="rus:OperationResult"/>
```

```
</xsd:complexType>            <xsd:element name="Records"
type="rus:RUSUsageRecordType" minOccurs="0" maxOccurs="unbounded"/>
</xsd:element>                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

            <xsd:element name="extractRUSRecordIdsResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="rus:OperationResult"/>
                    <xsd:element ref="rus:RUSRecordIdList"
minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

        <xsd:element name="replaceUsageRecordsRequest" >
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="rus:ReplacementRecords"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

            <xsd:element name="deleteRecordsRequest"
type="types:xPathQuery"/>

            <xsd:element name="extractRUSUsageRecordsRequest"
type="types:xPathQuery"/>
            <xsd:element name="extractRUSRecordIdsRequest"
type="types:xPathQuery"/>
            <xsd:element name="deleteSpecificRecordsRequest"
type="rus:RUSRecordIdListType"/>
            <xsd:element name="extractRUSUsageRecordsRequest"
type="rus:RUSRecordIdListType"/>

            <xsd:element name="incrementUsageRecordPartRequest">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="RUSRecordId"
type="xsd:long"/>
                                    <xsd:element name="path"
type="types:xPathQuery"/>
                    <xsd:element name="increment"
type="xsd:long"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

            <xsd:element name="incrementUsageRecordPartResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="rus:OperationResult"/>
                                    <xsd:element name="modified"
type="xsd:boolean" minOccurs="0" />
                </xsd:sequence>
```

```
            </xsd:complexType>
        </xsd:element>

            <xsd:element name="modifyUsageRecordPartRequest">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="RUSRecordId"
type="xsd:long"/>
                    <xsd:element name="XUpdate"
type="types:xPathQuery"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

            <xsd:element
name="listMandatoryUsageRecordElementsResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="rus:OperationResult"/>
                    <xsd:element name="MandatoryElements"
type="rus:MandatoryElementsType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

            <xsd:element name="modifyUsageRecordPartResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element ref="rus:OperationResult"/>
                    <xsd:element name="XUpdateResults"
type="xsd:boolean" minOccurs="0"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

            <xsd:element
name="listMandatoryUsageRecordElementsRequest">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="request"
type="xsd:boolean" minOccurs="0"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>

    </xsd:schema>
  </types>

  <wsdl:message name="RusUserNotAuthorisedFaultMessage">
    <part name="FaultDiagnostic"
element="types:RusUserNotAuthorisedFault"/>
  </wsdl:message>

  <wsdl:message name="RusInputFaultMessage">
    <part name="FaultDiagnostic" element="types:RusInputFault"/>
  </wsdl:message>
```

```
    <wsdl:message name="RusProcessingFaultMessage">
        <part name="FaultDiagnostic"
element="types:RusProcessingFault"/>
    </wsdl:message>

    <wsdl:message name="listMandatoryUsageRecordElementsInput"/>
    <wsdl:message name="listMandatoryUsageRecordElementsOutput">
        <wsdl:part name="output"
element="types:listMandatoryUsageRecordElementsResponse"/>
    </wsdl:message>

    <wsdl:message name="insertUsageRecordsOutput">
        <wsdl:part name="output"
element="types:recordListOperationResponse"/>
    </wsdl:message>

    <wsdl:message name="deleteRecordsOutput">
        <wsdl:part name="output"
element="types:recordListOperationResponse"/>
    </wsdl:message>

    <wsdl:message name="deleteSpecificRecordsOutput">
        <wsdl:part name="output"
element="types:recordListOperationResponse"/>
    </wsdl:message>

    <wsdl:message name="extractRUSRecordIdsOutput">
        <wsdl:part name="output"
element="types:extractRUSRecordIdsResponse"/>
    </wsdl:message>

    <wsdl:message name="extractRecordsOutput">
        <wsdl:part name="output"
element="types:extractRecordsResponse"/>
    </wsdl:message>

    <wsdl:message name="extractSpecificRUSRecordsOutput">
        <wsdl:part name="output"
element="types:extractRecordsResponse"/>
    </wsdl:message>

    <wsdl:message name="incrementUsageRecordPartOutput">
        <wsdl:part name="parameters"
element="types:incrementUsageRecordPartResponse"/>
    </wsdl:message>

    <wsdl:message name="modifyUsageRecordPartOutput">
        <wsdl:part name="parameters"
element="types:modifyUsageRecordPartResponse"/>
    </wsdl:message>

    <wsdl:message name="replaceUsageRecordsOutput">
        <wsdl:part name="parameters"
element="types:recordListOperationResponse"/>
    </wsdl:message>
```

```
    <wsdl:message name="insertUsageRecordsInput">
        <wsdl:part name="parameters"
element="types:insertUsageRecordsRequest"/>
    </wsdl:message>

    <wsdl:message name="deleteRecordsInput">
        <wsdl:part name="parameters"
element="types:deleteRecordsRequest"/>
    </wsdl:message>

    <wsdl:message name="deleteSpecificRecordsInput">
        <wsdl:part name="parameters"
element="types:deleteSpecificRecordsRequest"/>
    </wsdl:message>

    <wsdl:message name="extractRUSRecordIdsInput">
        <wsdl:part name="parameters"
element="types:extractRUSRecordIdsRequest"/>
    </wsdl:message>

    <wsdl:message name="extractRecordsInput">
        <wsdl:part name="parameters"
element="types:extractRecordsRequest"/>
    </wsdl:message>

    <wsdl:message name="extractSpecificRUSRecordsInput">
        <wsdl:part name="parameters"
element="types:extractSpecificRUSRecordsRequest"/>
    </wsdl:message>

    <wsdl:message name="incrementUsageRecordPartInput">
        <wsdl:part name="parameters"
element="types:incrementUsageRecordPartRequest"/>
    </wsdl:message>

    <wsdl:message name="modifyUsageRecordPartInput">
        <wsdl:part name="parameters"
element="types:modifyUsageRecordPartRequest"/>
    </wsdl:message>

    <wsdl:message name="replaceUsageRecordsInput">
        <wsdl:part name="parameters"
element="types:replaceUsageRecordsRequest"/>
    </wsdl:message>

    <wsdl:portType name="ResourceUsagePortType" >
        <wsdl:operation name="listMandatoryUsageRecordElements" >
            <wsdl:input
message="intf:listMandatoryUsageRecordElementsInput" />
            <wsdl:output
message="intf:listMandatoryUsageRecordElementsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
```

```
        </wsdl:operation>
        <wsdl:operation name="insertUsageRecords" >
            <wsdl:input message="intf:insertUsageRecordsInput" />
            <wsdl:output message="intf:insertUsageRecordsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="extractRecords" >
            <wsdl:input message="intf:extractRecordsInput" />
            <wsdl:output message="intf:extractRecordsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="extractRUSRecordIds" >
            <wsdl:input message="intf:extractRUSRecordIdsInput" />
            <wsdl:output message="intf:extractRUSRecordIdsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="extractSpecificRUSRecords" >
            <wsdl:input message="intf:extractSpecificRUSRecordsInput"
/>
            <wsdl:output
message="intf:extractSpecificRUSRecordsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="deleteSpecificRecords" >
            <wsdl:input message="intf:deleteSpecificRecordsInput" />
            <wsdl:output message="intf:deleteSpecificRecordsOutput" /
>
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="deleteRecords" >
            <wsdl:input message="intf:deleteRecordsInput" />
            <wsdl:output message="intf:deleteRecordsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
```

```
        </wsdl:operation>

        <wsdl:operation name="incrementUsageRecordPart" >
            <wsdl:input
message="intf:incrementUsageRecordPartInput" />
            <wsdl:output
message="intf:incrementUsageRecordPartOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="modifyUsageRecordPart" >
            <wsdl:input message="intf:modifyUsageRecordPartInput" />
            <wsdl:output message="intf:modifyUsageRecordPartOutput" /
>
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
            <fault name="RusInputFault"
message="intf:RusInputFaultMessage"/>
        </wsdl:operation>

        <wsdl:operation name="replaceUsageRecords" >
            <wsdl:input message="intf:replaceUsageRecordsInput" />
            <wsdl:output message="intf:replaceUsageRecordsOutput" />
            <fault name="RusUserNotAuthorisedFault"
message="intf:RusUserNotAuthorisedFaultMessage"/>
            <fault name="RusProcessingFault"
message="intf:RusProcessingFaultMessage"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

resource-usage-service.wsdl

```
<definitions name="resourceusage"
    targetNamespace="http://www.gridforum.org/2005/rus-wg/service"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://www.gridforum.org/2005/rus-wg/service"
    xmlns:porttype="http://www.gridforum.org/2005/rus-wg/service"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" >

    <import location="./resource-usage-porttype.wsdl"
        namespace="http://www.gridforum.org/2005/rus-wg/service"/>

    <binding name="ResourceUsagePortTypeSOAPBinding"
type="porttype:ResourceUsagePortType">
        <documentation>SOAP Binding for the
ResourceUsagePortType</documentation>
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="listMandatoryUsageRecordElements">
```

```
            <soap:operation
                soapAction="ResourceUsagePortType#listMandatoryUsageR
ecordElements"/>
          <input>
            <soap:body use="literal"/>
          </input>
            <output>
                <soap:body use="literal"/>
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
<operation name="insertUsageRecords">
            <soap:operation
                soapAction="ResourceUsagePortType#insertUsageRecords"
/>
            <input>
                <soap:body use="literal"  />
            </input>
            <output>
                <soap:body use="literal"  />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>

        <operation name="extractRecords">
            <soap:operation
                soapAction="ResourceUsagePortType#extractRecords"/>
            <input>
                <soap:body use="literal"  />
            </input>
            <output>
                <soap:body use="literal"  />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
```

```
        <operation name="extractRUSRecordIds">
            <soap:operation
                soapAction="ResourceUsagePortType#extractRUSRecordIds
"/>
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
        <operation name="extractSpecificRUSRecords">
            <soap:operation
                soapAction="ResourceUsagePortType#extractSpecificRUSR
ecords"/>
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
        <operation name="deleteRecords">
            <soap:operation
                soapAction="ResourceUsagePortType#deleteRecords"/>
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
```

```
        <operation name="deleteSpecificRecords">
            <soap:operation
                soapAction="ResourceUsagePortType#deleteSpecificRecor
ds"/>
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
        <operation name="incrementUsageRecordPart">
            <soap:operation
                soapAction="ResourceUsagePortType#incrementUsageRecor
dPart"/>
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
        <operation name="modifyUsageRecordPart">
            <soap:operation
                soapAction="ResourceUsagePortType#modifyUsageRecordPa
rt"/>
            <input>
                <soap:body use="literal" />
            </input>
            <output>
                <soap:body use="literal" />
            </output>
            <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
```

```
            <fault name="RusInputFault">
                <soap:fault name="RusInputFault" use="literal"/>
            </fault>
        </operation>
        <operation name="replaceUsageRecords">
            <soap:operation
                soapAction="ResourceUsagePortType#replaceUsageRecords
"/>
            <input>
                <soap:body use="literal"  />
            </input>
            <output>
                <soap:body use="literal"  />
            </output>
             <fault name="RusUserNotAuthorisedFault">
                <soap:fault name="RusUserNotAuthorisedFault"
use="literal"/>
            </fault>
            <fault name="RusProcessingFault">
                <soap:fault name="RusProcessingFault"
use="literal"/>
            </fault>
        </operation>
    </binding>
    <service name="ResourceUsageService">
        <port name="ResourceUsagePortTypeSOAPPort"
            binding="tns:ResourceUsagePortTypeSOAPBinding">
            <soap:address location="http://localhost"/>
        </port>
    </service>
</definitions>

<xsd:element name="extractUsageRecordIdsResponse">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="types:recordIdList" />
<xsd:element ref="types:operationResult" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- update data types -->
<xsd:element name="modifySpecUsageRecordsRequest">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="recordIdIdList" type="xsd:string" />
<xsd:element ref="types:xupdateExpression" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:element name="listMandatoryUsageRecordElementsRequest"
type="xsd:string" />

<xsd:element name="listMandatoryUsageRecordElementsResponse">
<xsd:complexType>
```

```
<xsd:sequence>
<xsd:element name="mandatoryElements" type="xsd:string" minOccurs="0"
maxOccurs="1" />
<xsd:element ref="types:operationResult" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

13.2 rus-core-porttype.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.gridforum.org/2007/rus-wg/core"
xmlns:rus="http://www.gridforum.org/2007/rus-wg/core"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:types="http://www.gridforum.org/2007/rus-wg/core/types"
>

<wsdl:import namespace="http://www.gridforum.org/2007/rus-
wg/core/types"
location="rus-core-types.xsd" />

<wsdl:message name="unauthorisedFaultMessage">
<wsdl:part element="types:RusUnauthorisedFault"
name="unauthorisedFaultMessage" />
</wsdl:message>

<wsdl:message name="recordNotFoundFaultMessage">
<wsdl:part element="types:RusRecordNotFoundFault"
name="recordNotFoundFaultMessage" />
</wsdl:message>

<wsdl:message name="processingFaultMessage">
<wsdl:part element="types:RusProcessingFault"
name="processingFaultMessage" />
</wsdl:message>

<wsdl:message name="invalidFaultMessage">
<wsdl:part element="types:RusInvalidFault" name="invalidFaultMessage"
/>
</wsdl:message>

<wsdl:message name="duplicatedFaultMessage">
<wsdl:part element="types:RusDuplicatedFault"
name="duplicatedFaultMessage" />
</wsdl:message>

<wsdl:message name="insertUsageRecordsInput">
<wsdl:part element="types:insertUsageRecordsRequest"
name="insertUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="insertUsageRecordsOutput">
<wsdl:part element="types:operationResult"
name="insertUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="extractSpecUsageRecordsInput">
<wsdl:part element="types:recordIdList"
name="extractSpecUsageRecordsInput" />
</wsdl:message>
```

```
<wsdl:message name="extractSpecUsageRecordsOutput">
<wsdl:part element="types:extractionResponse"
name="extractSpecUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="extractUsageRecordsInput">
<wsdl:part element="types:xpathExpression"
name="extractUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="extractUsageRecordsOutput">
<wsdl:part element="types:extractionResponse"
name="extractUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="extractUsageRecordIdsInput">
<wsdl:part element="types:xpathExpression"
name="extractUsageRecordIdsInput" />
</wsdl:message>

<wsdl:message name="extractUsageRecordIdsOutput">
<wsdl:part element="types:extractUsageRecordIdsResponse"
name="extractUsageRecordIdsOutput" />
</wsdl:message>

<wsdl:message name="modifySpecUsageRecordsInput">
<wsdl:part element="types:modifySpecUsageRecordsRequest"
name="modifySpecUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="modifySpecUsageRecordsOutput">
<wsdl:part element="types:operationResult"
name="modifySpecUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="modifyUsageRecordsInput">
<wsdl:part element="types:xupdateExpression"
name="modifyUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="modifyUsageRecordsOutput">
<wsdl:part element="types:operationResult"
name="modifyUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="replaceUsageRecordsInput">
<wsdl:part type="xsd:string" name="replaceUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="replaceUsageRecordsOutput">
<wsdl:part element="types:operationResult"
name="replaceUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="deleteSpecUsageRecordsInput">
```

```
<wsdl:part element="types:recordIdList"
name="deleteSpecUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="deleteSpecUsageRecordsOutput">
<wsdl:part element="types:operationResult"
name="deleteSpecUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="deleteUsageRecordsInput">
<wsdl:part element="types:xpathExpression"
name="deleteUsageRecordsInput" />
</wsdl:message>

<wsdl:message name="deleteUsageRecordsOutput">
<wsdl:part element="types:operationResult"
name="deleteUsageRecordsOutput" />
</wsdl:message>

<wsdl:message name="listMandatoryUsageRecordElementsInput">
<wsdl:part element="types:listMandatoryUsageRecordElementsRequest"
name="listMandatoryUsageRecordElementsInput" />
</wsdl:message>

<wsdl:message name="listMandatoryUsageRecordElementsOutput">
<wsdl:part element="types:listMandatoryUsageRecordElementsResponse"
name="listMandatoryUsageRecordElementsOutput" />
</wsdl:message>

<wsdl:portType name="ResourceUsagePortType" >

<wsdl:operation name="insertUsageRecords" >
<wsdl:input message="rus:insertUsageRecordsInput" />
<wsdl:output message="rus:insertUsageRecordsOutput" />
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusDuplicatedFault" message="rus:duplicatedFaultMessage"
/>
<fault name="RusInvalidFault" message="rus:invalidFaultMessage" />
</wsdl:operation>

<wsdl:operation name="extractSpecUsageRecords">
<wsdl:input message="rus:extractSpecUsageRecordsInput" />
<wsdl:output message="rus:extractSpecUsageRecordsOutput" />
<fault name="RusRecordNotFoundFault"
message="rus:recordNotFoundFaultMessage" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
</wsdl:operation>

<wsdl:operation name="extractUsageRecords">
<wsdl:input message="rus:extractUsageRecordsInput" />
```

```
<wsdl:output message="rus:extractUsageRecordsOutput" />
<fault name="RusInvalidFault"
message="rus:recordNotFoundFaultMessage" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
</wsdl:operation>

<wsdl:operation name="extractUsageRecordIds">
<wsdl:input message="rus:extractUsageRecordIdsInput" />
<wsdl:output message="rus:extractUsageRecordIdsOutput" />
<fault name="RusInvalidFault"
message="rus:recordNotFoundFaultMessage" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
</wsdl:operation>

<wsdl:operation name="modifySpecUsageRecords">
<wsdl:input message="rus:modifySpecUsageRecordsInput" />
<wsdl:output message="rus:modifySpecUsageRecordsOutput" />
<fault name="RusRecordNotFoundFault"
message="rus:recordNotFoundFaultMessage" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
<fault name="RusInvalidFault" message="rus:invalidFaultMessage" />
</wsdl:operation>

<wsdl:operation name="modifyUsageRecords">
<wsdl:input message="rus:modifyUsageRecordsInput" />
<wsdl:output message="rus:modifyUsageRecordsOutput" />
<fault name="RusInvalidFault"
message="rus:recordNotFoundFaultMessage" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
</wsdl:operation>

<wsdl:operation name="replaceUsageRecords">
<wsdl:input message="rus:replaceUsageRecordsInput" />
<wsdl:output message="rus:replaceUsageRecordsOutput" />
<fault name="RusRecordNotFoundFault"
message="rus:recordNotFoundFaultMessage"/>
<fault name="RusInvalidFault" message="rus:invalidFaultMessage" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
</wsdl:operation>

<wsdl:operation name="deleteSpecUsageRecords">
```

```
<wsdl:input message="rus:deleteSpecUsageRecordsInput" />
<wsdl:output message="rus:deleteSpecUsageRecordsOutput" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
<fault name="RusRecordNotFoundFault"
message="rus:recordNotFoundFaultMessage" />
</wsdl:operation>

<wsdl:operation name="deleteUsageRecords">
<wsdl:input message="rus:deleteUsageRecordsInput" />
<wsdl:output message="rus:deleteUsageRecordsOutput" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
<fault name="RusInvalidFault"
message="rus:recordNotFoundFaultMessage" />
</wsdl:operation>

<wsdl:operation name="listMandatoryUsageRecordElements">
<wsdl:input message="rus:listMandatoryUsageRecordElementsInput" />
<wsdl:output message="rus:listMandatoryUsageRecordElementsOutput" />
<fault name="RusProcessingFault"
message="rus:processingFaultMessage"/>
<fault name="RusUnauthorisedFault"
message="rus:unauthorisedFaultMessage"/>
</wsdl:operation>

</wsdl:portType>
```

13.3 rus-core-service.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
        targetNamespace="http://www.gridforum.org/2007/rus-wg/core"
        xmlns:rus="http://www.gridforum.org/2007/rus-wg/core"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:types="http://www.gridforum.org/2007/rus-wg/core/types"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
>
<wsdl:import namespace="http://www.gridforum.org/2007/rus-wg/core"
              location="rus-core-porttype.wsdl" />
<wsdl:binding name="ResourceUsagePortTypeSOAPBinding"
type="rus:ResourceUsagePortType">
<wsdl:documentation>SOAP Binding for the
ResourceUsagePortType</wsdl:documentation>
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="insertUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#insertUsageRecords"/>
<wsdl:input>
```

```
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusDuplicatedFault">
<soap:fault name="RusDuplicatedFault" use="literal" />
</fault>
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="extractSpecUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#extractSpecUsageRecords"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusRecordNotFoundFault">
<soap:fault name="RusRecordNotFoundFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="extractUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#extractUsageRecords"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
</wsdl:operation>
```

```
<wsdl:operation name="extractUsageRecordIds">
<soap:operation
soapAction="ResourceUsagePortType#extractUsageRecordIds"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="modifySpecUsageRecords">
<soap:operation
soapAction="ResourceUsagePortyType#modifySpecUsageRecords" />
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusRecordNotFoundFault">
<soap:fault name="RusRecordNotFoundFault" use="literal" />
</fault>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="modifyUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#modifyUsageRecords"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
```

```
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="replaceUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#replaceUsageRecords"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
<fault name="RusRecordNotFoundFault">
<soap:fault name="RusRecordNotFoundFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="deleteSpecUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#deleteSpecUsageRecords"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusRecordNotFoundFault">
<soap:fault name="RusRecordNotFoundFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="deleteUsageRecords">
<soap:operation
soapAction="ResourceUsagePortType#deleteUsageRecords"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
```

```
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
<fault name="RusInvalidFault">
<soap:fault name="RusInvalidFault" use="literal" />
</fault>
</wsdl:operation>
<wsdl:operation name="listMandatoryUsageRecordElements">
<soap:operation
soapAction="ResourceUsagePortType#listMandatoryUsageRecordElements"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
<fault name="RusUnauthorisedFault">
<soap:fault name="RusUnauthorisedFault" use="literal" />
</fault>
<fault name="RusProcessingFault">
<soap:fault name="RusProcessingFault" use="literal" />
</fault>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="resource-usage-service">
<wsdl:port binding="rus:ResourceUsagePortTypeSOAPBinding"
name="ResourceUsagePortTypeSOAPPort">
<soap:address
location="http://localhost:8080/axis2/services/resource-usage-
service"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```