GWD-I
GGF OREP Working Group

Editors:
Ann Chervenak, ISI
Robert Schuler, ISI

Category: Informational

16 August 2005

Web Services NameMapper Specification (WS-NameMapper)

Status of This Memo

This memo provides information regarding the specification of service based interfaces to NameMapper. The specification is presently a draft for discussion. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Copyright © Global Grid Forum (2005). All Rights Reserved.

Abstract

This document proposes an interface specification of a WS-RF service for name mapping. A set of requirements, important terms and key concepts are enumerated to provide background for name mapping and its use in the Grid. Then the core types, message elements, faults, resource properties and the port type are described. The full XML Schema and WSDL documents are included.

Contents

1. Introduction

This document describes a WS-RF interface for a NameMapper service. Mapping among names is useful functionality in a number of Grid contexts. For example, name mapping could be used in replica management to provide mappings between logical names for data items and one or more physical locations of data replicas. Second, name mapping can be used to associate the names of datasets or data collections with the files or other data objects that are members of the collections. Third, name mapping functionality is useful for storage resource managers, which expose a logical name for data items to the external world and map those logical names to physical locations on the storage device.

Requirements for a mapping service include the following:

- Allow clients to register many-to-one or one-to-many associations between names, where these names are arbitrary strings
- Answer queries for registered mappings that contain specified names
- Allow clients to associate descriptive attributes with registered names or mappings
- Answer queries for registered mappings that contain specified attribute values
- Permit users to modify or delete existing names, mappings or attributes
- Support compound or bulk operations that allow efficient registration, query or deletion of multiple names, mappings or attributes
- Support for fine-grained authorization for mapping service operations
- Support for flexible query expression and dialetc

These requirements are described in more detail in the Concepts section of the specification.

This document proposes a set of types, messages, operations and resource property definitions that define the NameMapper Service.

1.1 Specification Scope

The focus of this specification is to introduce a WS-NameMapper service that provides functionality to map from one set of names to another and optionally to attach descriptive attributes to names.

1.2 Specification Organization

This specification defines the key terminology and describes the core concepts in sections 3 and 4, respectively. Section 5 describes the XML types important to understanding the rest of the elements defined in this specification. SOAP messages used in operations of the PortType are described in section 6. Properties, in the sense of WS-ResourceProperties for the WS-Resources in this specification, are described in Section 7. Section 8 describes the custom SOAP Faults. Finally, section 9 defines the core WS-Resource PortType(s) defined by this specification. The appendices A and B present the XML Schema and WSDL documents.

2. Notational Conventions

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

When describing concrete XML schemas and XML instance fragments, this specification uses the notational convention of [WS-Security]. Specifically, each member of an element’s children or attributes property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1 indicates that namespace x is being used, the root

element MyHeader and a child element SomeProperty with an attribute value1). The use of {any} indicates the presence of an element wildcard (<xsd:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xsd:anyAttribute/>).

Italicized element names are used when an element is intended to be specified by the NameMapper specification realizations.

In the body of the specification, when patterns of messages are described, the layout of the XML of each message is presented, as opposed to the XML schema; the XML Schema is provided in Appendix A. The following notation is used to indicate cardinality of XML elements in the XML fragments:

* * zero or more
* \+ one or more
* ? zero or one

Where no notation is added to an element, one instance of the element is expected.

This specification generally adopts the terminology defined in the Open Grid Services Architecture Glossary of Terms [OGSA Glossary]. Domain specific termination is described in Section 3.

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

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema
wsdl	http://schemas.xmlsoap.org/wsdl

3. Terminology

The following section provides definitions for terms pertinent to understanding this specification.

3.1 Mapping

A *mapping* is an association between two names. These associations between names are assumed to be bidirectional. Mappings can be one-to-many or many-to-one among names.

3.2 Names

A *name* is an arbitrary string that is associated with another name in a mapping. In the context of this specification, we refer to the two names associated by a mapping as end1 and end2, where each name represents an end point for the mapping. For example, in the context of data collection management, one end name of a registered mapping might be the name of the data collection and the other end name could be a data item that is a member of the collection. In replica management, one end name for a mapping could be the logical name of a replicated data item while the other end name would be a physical location for a replicated data item. For a storage resource manager, one end name might be the logical name that is exposed by the service to the external world, while the other would be the current physical location of a data item on managed storage.

3.3 Attribute

An *attribute* is a value associated with a name or mapping that describes that name or mapping. An attribute has a specified type.

4. Concepts

The following section explains some of the concepts pertinent to understanding this specification.

4.1 Name Mapping

In this specification, we provide support for name mapping as an important basic functionality used in many contexts in Grid computing. Three examples were discussed earlier for which one-to-many or many-to-one mappings among names are useful:

- In replica management: to associate logical names for data items with one or more locations for physical replicas of the data items
- In collection management: to associate names for data collections or datasets with the names of one or more members that compose the collection
- In storage resource management: to associate a logical name for data items that is exposed to the outside world with names or locations for those data items on the physical storage resource

We distinguish name mapping functionality from namespace management functionality, such as that provided for hierarchical namespaces in the Resource Namespace Service (RNS) Specification being developed by the GFS-WG of GGF. The RNS specification provides interfaces for hierarchical management of a global, hierarchical namespace, including the management of federated and virtual data, services and resources.

By contrast, the NameMapper service is a simple mapping service that allows clients to provide many-to-one and one-to-many associations among names and to associate descriptive attributes with names and mappings. The NameMapper Service does not enforce a particular namespace on its registered names. We assume that if multiple virtual organizations share a NameMapper service, then those VOs will use disjoint namespaces so that registered mappings from different VOs will not conflict.

4.2 Attributes

Our specification includes the concept of attributes or descriptive information that can be associated with names or mappings. The specification includes operations to define an attribute with a specific type and others to associate values for a registered attribute type with a registered name or mapping.

Attributes are useful for several name mapping scenarios. For example, when a NameMapper service is used for data collection management, attributes specified on the collection might describe how the data items are associated. When a mapping service is used for replica management, attributes can provide file sizes, checksums, or other attributes used during data replication or replica selection.

4.3 Compound or Bulk Operations

A key part of this specification is that all operations are defined as compound or “bulk” operations, meaning that they support the manipulation of multiple name mappings in a single NameMapper operation. The requirement to support compound or bulk operations is motivated by the recognition that, while Web services provide the benefit of greater interoperability and widely available tooling, they come with the disadvantage of adding to the overhead of network communication. To mitigate this overhead, coarse-grained operations are essential.

4.4 Authorization of NameMapper Operations

Fine-grained authorization on mapping entries is needed to allow the service to restrict access to particular users or groups for operations on mappings. For example, consider multiple Virtual Organizations that want to share a NameMapper service. The service needs to be able to restrict access to its mappings to authorized members of one VO. It must avoid the possibility of external

(non-VO) users querying or modifying information registered by the VO. The specification must provide the ability to associate permission or policy information with mapping entries and let an external policy engine interpret this information and make decisions to permit or deny operations.

5. Core Types

The following section describes the more primitive, or core, XML types defined in this specification.

5.1 NameType

```
<xsd:complexType name="NameType">
  <xsd:sequence>
    <xsd:element name="attribute" type="tns:AttributeType"
                 minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="fqname" type="xsd:string" use="required"/>
</xsd:complexType>
```

NameType

A NameType element describes a name in the system and contains attribute elements. The @name attribute of the element is used to specify the fully-qualified name string. The NameType element may contain 0 or any number of attribute elements.

5.2 MappingType

```
<xsd:complexType name="MappingType">
  <xsd:attribute name="end1" type="xsd:string" use="required"/>
  <xsd:attribute name="end2" type="xsd:string" use="required"/>
</xsd:complexType>
```

MappingType

A MappingType element describes an association between two names. The names are identified by attributes @end1 and @end2, which are required by the type. The mapping does not specify any navigability. It may be assumed that associations are bidirectional.

5.3 AttributeType

```
<xsd:complexType name="AttributeType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="type" type="tns:AttributeEnumerationType"
                     use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

AttributeType

An AttributeType element describes an “attribute” associated with a NameType name element in the system. This domain specific “attribute” should not be confused with attributes in the XML sense of the term. The AttributeType element uses a name/value pair to express the attribute. The name of the AttributeType element is determined by the @name attribute and the value is specified by a child text element. The value of the AttributeType element conforms to the type as specified by the @type attribute.

5.4 AttributeDefinitionType

```
<xsd:complexType name="AttributeDefinitionType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="required" type="xsd:boolean" default="false"/>
```

```
  <xsd:attribute name="type" type="tns:AttributeEnumerationType"
                 use="required"/>
</xsd:complexType>
```

AttributeDefinitionType

An AttributeDefinitionType element is used to define the supported attributes on names. The name of the definition is specified by the @name attribute. The expected type of value supported by attributes conforming to the definition is specified by the @type attribute. The @required attribute indicates whether the attribute is expected to exist for names in the system.

5.5 AttributeEnumerationType

```
<xsd:simpleType name="AttributeEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="dateTime"/>
    <xsd:enumeration value="decimal"/>
    <xsd:enumeration value="integer"/>
    <xsd:enumeration value="string"/>
  </xsd:restriction>
</xsd:simpleType>
```

AttributeEnumerationType

The attribute enumeration is used to specify the possible types of values for attributes and attribute definitions.

dateTime	A date format of YYYY-MM-DD hh:mm:ss Z
decimal	A decimal number of unspecified range and precision
integer	An integer number of unspecified range
string	A string of characters

5.6 MappingStatusType

```
<xsd:complexType name="MappingStatusType">
  <xsd:complexContent>
    <xsd:extension base="tns:MappingType">
      <xsd:attribute name="status" type="tns:MappingStatusEnumerationType"
                              use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

MappingStatusType

A MappingStatusType element is used when operations involving manipulation of mappings are performed. The MappingStatusType extends the MappingType. In addition to the base type, the MappingStatusType specifies a @status element to indicate the status of the operation on the mapping.

5.7 MappingStatusEnumerationType

```
<xsd:simpleType name="MappingStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="end1Exists"/>
    <xsd:enumeration value="end1NotFound"/>
    <xsd:enumeration value="end2Exists"/>
    <xsd:enumeration value="end2NotFound"/>
    <xsd:enumeration value="mappingExists"/>
    <xsd:enumeration value="mappingNotFound"/>
    <xsd:enumeration value="end1AttributeRequired"/>
    <xsd:enumeration value="end2AttributeRequired"/>
```

```
  </xsd:restriction>
</xsd:simpleType>
```

MappingStatusEnumerationType

The MappingStatusEnumerationType specifies an enumeration of values pertaining to the results of operations performed involving mappings.

end1Exists	The name identified by @end1 exists.
end1NotFound	The name identified by @end1 does not exist.
end2Exists	The name identified by @end2 exists.
end2NotFound	The name identified by @end2 does not exist.
mappingExists	The mapping exists.
mappingNotFound	The mapping does not exist.
end1AttributeRequired	The name identified by @end1 is missing one or more required attributes.
end2AttributeRequired	The name identified by @end2 is missing one or more required attributes.

5.8 AttributeStatusType

```
<xsd:complexType name="AttributeStatusType">
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="status" type="tns:AttributeStatusEnumerationType"
                 use="required"/>
</xsd:complexType>
```

AttributeStatusType

A AttributeStatusType element is used when operations involving AttributeType elements are performed. The AttributeStatusType element uses the @name attribute to reference the AttributeType element to which the status refers. The @status attribute indicates the result of the operation on the given attribute.

5.9 AttributeStatusEnumerationType

```
<xsd:simpleType name="AttributeStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="attributeExists"/>
    <xsd:enumeration value="attributeNotFound"/>
    <xsd:enumeration value="attributeDefinitionNotFound"/>
    <xsd:enumeration value="attributeRequired"/>
  </xsd:restriction>
</xsd:simpleType>
```

AttributeStatusEnumerationType

The AttributeStatusEnumerationType defines an enumeration of values pertaining to the status of operations performed involving AttributeType elements.

attributeExists	The attribute exists.
attributeNotFound	The attribute does not exist.
attributeDefinitionNotFound	The attribute definition does not exist.
attributeRequired	The attribute is required.

5.10 NameStatusType

```
<xsd:complexType name="NameStatusType">
```

```
  <xsd:sequence>
    <xsd:element name="attributeStatus" type="tns:AttributeStatusType"
                 minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="status" type="tns:NameStatusEnumerationType"
                 use="optional"/>
</xsd:complexType>
```

NameStatusType

The NameStatusType element is used when operations involving manipulation of NameType elements or their associated AttributeType attributes are performed. The NameStatusType element specifies a @name attribute and @status attribute. The @status attribute indicates the status of the operation. The NameStatusType element is composed of 0 to many attributeStatus elements of type AttributeStatusType.

5.11 NameStatusEnumerationType

```
<xsd:simpleType name="NameStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="notFound"/>
  </xsd:restriction>
</xsd:simpleType>
```

NameStatusEnumerationType

The NameStatusEnumerationType defines an enumeration of values pertaining to the status of operations performed involving NameStatusType elements.

notFound	The name element does not exist.

5.12 AttributeDefinitionStatusType

```
<xsd:complexType name="AttributeDefinitionStatusType">
  <xsd:complexContent>
    <xsd:extension base="tns:AttributeDefinitionType">
      <xsd:attribute name="status"
                  type="tns:AttributeDefinitionStatusEnumerationType"
                  use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

AttributeDefinitionStatusType

The AttributeDefinitionStatusType element is used when operations involving manipulation of attribute definitions are performed. The AttributeDefinitionStatusType extends the AttributeDefinitionType by adding a @status attribute to indicate the status of the operation performed.

5.13 AttributeDefinitionStatusEnumerationType

```
<xsd:simpleType name="AttributeDefinitionStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="definitionExists"/>
    <xsd:enumeration value="definitionNotFound"/>
  </xsd:restriction>
</xsd:simpleType>
```

AttributeDefinitionStatusEnumerationType

The AttributeDefinitionStatusEnumerationType defines an enumeration of values pertaining to the status of operations performed involving AttributeDefinitionStatusType elements.

definitionExists	The attribute definition exists.

definitionNotFound	The attribute definition does not exist.

5.14 RenameType

```
<xsd:complexType name="RenameType">
  <xsd:attribute name="old" type="xsd:string" use="required"/>
  <xsd:attribute name="new" type="xsd:string" use="required"/>
</xsd:complexType>
```

RenameType

A RenameType element is used to describe a rename operation. The name referred to by the @old attribute is intended to be replaced by the name specified by the @new attribute.

5.15 RenameStatusType

```
<xsd:complexType name="RenameStatusType">
  <xsd:complexContent>
    <xsd:extension base="tns:RenameType">
      <xsd:attribute name="status" type="tns:RenameStatusEnumerationType"
                     use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

RenameStatusType

The RenameStatusType element is used when operations involving renaming are performed. The RenameStatusType extends RenameType adding a @status attribute to indicate the status of the rename operation.

5.16 RenameStatusEnumerationType

```
<xsd:simpleType name="RenameStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="oldNotFound"/>
    <xsd:enumeration value="newExists"/>
  </xsd:restriction>
</xsd:simpleType>
```

RenameStatusEnumerationType

The RenameStatusEnumerationType defines an enumeration of values pertaining to the status of rename operations.

oldNotFound	The @old name does not exist.
newExists	The @new name exists.

6. Messages and Related Element Types

The following section describes the SOAP messages along with element types related to the messages defined in this specification.

6.1 Message Convention Example

```
<wsdl:message name="RegisterInputMessage">
  <wsdl:part name="parameters" element="tns:Register"/>
</wsdl:message>

<wsdl:message name="RegisterOutputMessage">
  <wsdl:part name="parameters" element="tns:RegisterResponse"/>
</wsdl:message>

…
```

RegisterInputMessage

The RegisterInputMessage specifies a wsdl:message composed of one wsdl:part, a single Register element. The more interesting aspects of the message can be seen in the specification for the Register element and its composite elements and attributes.

RegisterOutputMessage

The RegisterOutputMessage specifies a wsdl:message composed of one wsdl:part, a single RegisterResponse element. The more interesting aspects of the message can be seen in the specification for the RegisterResponse element and its composite elements and attributes.

The RegisterInputMessage and RegisterOutputMessage are provided above as an example to illustrate the structural and naming pattern for input and output messages. In the PortType section, it can be seen that the Register operation of the NameMapperPortType specifies the above messages for its corresponding input and output parameters.

While the remaining messages are not described in this section, they follow a similar pattern. Thus the Remove operation (see NameMapperPortType) specifies RemoveInputMessage and RemoveOutputMessage for its input and output messages, which in turn rely on the Remove element type and RemoveReponse element type respectively. Since the interesting details of the input and output parameters are contained within the Remove and RemoveResponse elements, they are described in further detail in this section.

6.2 Register

```
<xsd:element name="Register">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapping"
                   type="tns:MappingType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RegisterResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="mappingStatus"
                   type="tns:MappingStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Register

The Register element specifies bulk parameters for mapping registration. The input allows for 1 to many mapping elements of MappingType.

RegisterResponse

The RegisterResponse element specifies bulk results for mapping registration. The output provides 0 to many mappingStatus elements of MappingStatusType.

6.3 Remove

```
<xsd:element name="Remove">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapping"
                   type="tns:MappingType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="RemoveResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="mappingStatus"
                   type="tns:MappingStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Remove

The Remove element specifies bulk parameters for mapping removal. The input allows for 1 to many mapping elements of MappingType.

RemoveResponse

The RemoveResponse specifies bulk results for mapping removal. The output provides 0 to many mappingStatus elements of MappingStatusType.

6.4 Rename

```
<xsd:element name="Rename">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="rename"
                   type="tns:RenameType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="RenameResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="renameStatus"
                   type="tns:RenameStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Rename

The Rename element specifies bulk parameters for renaming. The input allows for 1 to many rename elements of RenameType.

RenameResponse

The RenameResponse element specifies bulk results for renaming. The output provides for 0 to many renameStatus elements of RenameStatusType.

6.5 DefineAttributes

```
<xsd:element name="DefineAttributes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="definition"
                   type="tns:AttributeDefinitionType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="DefineAttributesResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="definitionStatus"
```

```
                  type="tns:AttributeDefinitionStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

DefineAttributes

The DefineAttributes element specifies bulk parameters for defining attribute definitions. The input allows for 1 to many definition elements.

DefineAttributesResponse

The DefineAttributeResponse element specifies bulk results for defining attribute definitions. The output provides for 0 to many definitionStatus elements of AttributeDefinitionStatusType.

6.6 UndefineAttributes

```
<xsd:element name="UndefineAttributes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="definition"
                   type="tns:AttributeDefinitionType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="UndefineAttributesResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="definitionStatus"
                   type="tns:AttributeDefinitionStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

UndefineAttributes

The UndefineAttributes element specifies bulk parameters for undefining attribute definitions. The input allows for 1 to many definition elements.

UndefineAttributesResponse

The UndefineAttributesResponse element specifies bulk results for undefining attribute definitions. The output provides for 0 to many definitionStatus elements of AttributeDefinitionStatusType.

6.7 SetAttributes

```
<xsd:element name="SetAttributes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="name"
                   type="tns:NameType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="SetAttributesResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="nameStatus"
                   type="tns:NameStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

SetAttributes

The SetAttributes element specifies bulk parameters for setting attributes on name elements. The input allows for 1 to many name elements which correspondingly contain attribute elements.

SetAttributesResponse

The SetAttributesResponse element specifies bulk results for setting attributes on name elements. The output provides for 0 to many nameStatus elements which correspondingly contain attributeStatus elements.

6.8 UnsetAttributes

```
<xsd:element name="UnsetAttributes">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="name"
                   type="tns:NameType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="UnsetAttributesResponse">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="nameStatus"
                   type="tns:NameStatusType"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

UnsetAttributes

The SetAttributes element specifies bulk parameters for unsetting attributes on name elements. The input allows for 1 to many name elements which correspondingly contain attribute elements.

UnsetAttributesResponse

The UnsetAttributesResponse element specifies bulk results for unsetting attributes on name elements. The output provides for 0 to many nameStatus elements which correspondingly contain attributeStatus elements.

7. Faults

The following section describes the SOAP fault messages defined in this specification.

7.1 NameMapperFault

```
<xsd:complexType name="NameMapperFaultType">
  <xsd:complexContent>
    <xsd:extension base="wsbf:BaseFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="NameMapperFault" type="tns:NameMapperFaultType"/>
```

NameMapperFault

The NameMapperFault specifies a fault that indicates that a general exception occurred when performing an operation.

7.2 Internal Error Fault

```
<xsd:complexType name="InternalErrorFaultType">
  <xsd:complexContent>
    <xsd:extension base="tns:NameMapperFaultType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="InternalErrorFault" type="tns:InternalErrorFaultType"/>
```

InternalErrorFault

The InternalErrorFault specifies a fault that indicates that an internal system exception occurred when performing an operation.

8. Properties

The following section describes the WS-ResourceProperties elements defined in this specification.

8.1 NameMapperResourceProperties

```
<xsd:element name="NameMapperResourceProperties">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="definition"
                   type="tns:AttributeDefinitionType"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="name"
                   type="tns:NameType"/>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="mapping"
                   type="tns:MappingType"/>
      <xsd:element maxOccurs="unbounded" minOccurs="1" ref="rpns0:Topic"/>
      <xsd:element maxOccurs="unbounded" minOccurs="1"
                   ref="rpns0:TopicExpressionDialects"/>
      <xsd:element maxOccurs="1" minOccurs="1" ref="rpns0:FixedTopicSet"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

/wsname:NameMapperProperties

The NameMapperProperties specifies the resource properties document for the interface. It is composed of definitions, names, mappings, along with standard topic-related resource properties. The standard properties are not further described in this document.

/wsname:NameMapperProperties/wsname:definition

The resource properties contain 0 or many definition elements of AttributeDefinitionType.

/wsname:NameMapperProperties/wsname:name

The resource properties contain 0 or many name elements of NameType.

/wsname:NameMapperProperties/wsname:mapping

The resource properties contain 0 or many mapping elements of MappingType.

9. PortTypes

The following section describes the WS-Resource and other SOAP PortTypes defined in this specification.

9.1 NameMapperPortType

```
<wsdl:portType name="NameMapperPortType"
    wsrp:ResourceProperties="tns:NameMapperResourceProperties">
```

```
    <wsdl:operation name="Register">
      <wsdl:input message="tns:RegisterInputMessage"/>
      <wsdl:output message="tns:RegisterOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="Remove">
      <wsdl:input message="tns:RemoveInputMessage"/>
      <wsdl:output message="tns:RemoveOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="Rename">
      <wsdl:input message="tns:RenameInputMessage"/>
      <wsdl:output message="tns:RenameOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="SetAttributes">
      <wsdl:input message="tns:SetAttributesInputMessage"/>
      <wsdl:output message="tns:SetAttributesOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="UnsetAttributes">
      <wsdl:input message="tns:UnsetAttributesInputMessage"/>
      <wsdl:output message="tns:UnsetAttributesOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="DefineAttributes">
      <wsdl:input message="tns:DefineAttributesInputMessage"/>
      <wsdl:output message="tns:DefineAttributesOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    <wsdl:operation name="UndefineAttributes">
      <wsdl:input message="tns:UndefineAttributesInputMessage"/>
      <wsdl:output message="tns:UndefineAttributesOutputMessage"/>
      <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
    </wsdl:operation>

    …standard operations ommitted…

</wsdl:portType>
```

NameMapperPortType

The NameMapperPortType defines the PortType for the WS-Resource NameMapper. Supported operations include Register, Remove, Rename, SetAttributes, UnsetAttributes, DefineAttributes, UndefineAttributes, along with standard operations (not shown above) for getting resource properties [WS-ResourceProperties] and notifications [WS-BaseNotification].

The normal form of input is a single input parameter and single output parameter each composed of multiple elements, which represent a bulk operation. Transactional boundaries are narrowly defined around a single parameter within a bulk operation. Thus when registering numerous mappings, the transaction protects a single mapping registration within the bulk operation. It is possible for a subset of an operation to succeed while the remainder fails.

The results of operations return status information to the caller. In most cases, rather than expressing most user errors in the fault message, the errors are reported within the output

message as defined by the various status types defined in this document. Below when the description of an operation mentions error reporting it is referring to the status contained in the output message rather than in the fault message.

Register

The Register operation supports registration of mappings. When a mapping is registered, a mapping element is created as part of the resource properties document. In addition, if there exists no name element corresponding to a mapping endpoint, the name element is also created and added to the resource properties document. An error is reported if the mapping already exists.

Remove

The Remove operation supports removal of mappings. When a mapping is removed, the corresponding mapping element is removed from the resource properties document. In addition, if after the removal of the mapping one or more of the names formerly associated with the mapping is no longer referenced by any other mapping, then the name element is also removed from the resource properties document. An error is reported if the mapping does not exist.

Rename

The Rename operation supports renaming of name elements. When renaming, the resource properties document is searched for a name element corresponding to the “old” name (see RenameType) and the name element’s name attribute (see NameType) is changed to the specified “new” name. And error is reported if the name element corresponding to the “old” name does not exist or if a name element corresponding to the “new” name already exists.

SetAttributes

The SetAttributes operation supports setting of attributes on name elements. When setting attributes, the resource properties document is searched to locate name elements matching the input parameter names. When the name element is located, the attributes are inspected. If a matching attribute element does not exist, it is added to the name element. If a matching attribute element does exist, then its value is overwritten by the input attribute element’s value. An error is reported if the name element does not exist or if there exists no attribute definition element corresponding to the input attribute element name and type.

UnsetAttributes

The UnsetAttributes operation supports unsetting of attributes on name elements. When unsetting attributes, the resource properties document is searched to locate name elements matching the input parameter names. When the name element is located, the attributes are inspected. Then the matching attribute element is entirely removed from the name element. An error is reported if the name element does not exist, if there exists no matching attribute in the name element, or if the attribute is defined as required.

DefineAttributes

The DefineAttributes operation supports defining of attribute definitions. When defining attribute definitions, attribute definition elements are added to the resource properties document. An error is reported if an attribute definition with the specified name from the input message already exists.

UndefineAttributes

The UndefineAttributes operation supports undefining of attribute definitions. When undefining attribute definitions, attribute definition elements are removed from the resource properties document. An error is reported if an attribute definition with the specified name from the input message does not exist.

10. Security Considerations

The realizations of a grid NameMapper service will use standard grid security mechanisms as specified by the OGSA AUTHZ Working Group combined with standard ways of relating grid credentials and authorities to resource access rights. The assumption is that these standards will also indicate how to make information related to authentication, authorization, security, etc., available.

11. Conclusions

This document has described a proposal for an interface for name mapping. The interface proposed is intended to be compatible with the architecture to be proposed by the GGF Open Grid Services Architecture working group.

Editor Information

Ann Chervenak,
University of Southern California,
Information Sciences Institute,
4676 Admiralty Way
Suite 1001,
Marina del Rey, CA 90292,
USA

Robert Schuler,
University of Southern California,
Information Sciences Institute,
4676 Admiralty Way
Suite 1001,
Marina del Rey, CA 90292,
USA

Contributors

TODO

Acknowledgements

TODO

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.

Full Copyright Notice

Copyright (C) Global Grid Forum (2005). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

References

[OGSA]
I. Foster (Ed), H. Kishimoto (Ed), A. Savva (Ed), D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, R. Subramaniam, J. Treadwell, J. Von Reich. The Open Grid Services Architecture, Version 1.0. Global Grid Forum. GFD-I.030. 29 January 2005. http://forge.gridforum.org/projects/ggf-editor/document/GFD.30/en/1.

[OGSA Glossary]
J. Treadwell, Open Grid Services Architecture Glossary of Terms, GFD-I.044, January 25th 2005. http://forge.gridforum.org/projects/ggf-editor/document/GFD.44/en/1.

[RFC2119]
S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, Internet Engineering Task Force, RFC 2119, http://www.ietf.org/rfc/rfc2119.txt, March 1997.

[SOAP]
D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer. *Simple Object Access Protocol (SOAP) 1.1*. W3C Note 08 May 2000. Available from: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[WSDL]
E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. Web Services Description Language 1.1. World Wide Web Consortium. W3C Note 15 March 2001. http://www.w3.org/TR/wsdl.

[WS-BaseNotification]
S. Graham (Ed), B. Murray (Ed). Web Services Base Notification 1.2 (WS-BaseNotification), Working Draft 03, 21 June 2004. http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf

[WS-Resource]
S. Graham (Ed), A. Karmarkar (Ed), J. Mischkinsky (Ed), I. Robinson (Ed), I. Sedukhin (Ed). Web Services Resource 1.2 (WS-Resource), Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-cd-01.pdf

[WS-ResourceProperties]

S. Graham (Ed), J. Treadwell (Ed). Web Services Resource Properties 1.2 (WS-ResourceProperties), Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-cd-01.pdf.

[XPath]
J. Clark and S. DeRose. XML Path Language (XPath), Version 1. W3C Recommendation 16 November 1999. See: http://www.w3.org/TR/xpath.

Appendix A – NameMapper Types XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
    targetNamespace="http://www.globus.org/namespaces/2005/07/name/mapper"
    xmlns:tns="http://www.globus.org/namespaces/2005/07/name/mapper"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing"
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns:wsbf="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:import schemaLocation="../../ws/addressing/WS-Addressing.xsd"
        namespace="http://schemas.xmlsoap.org/ws/2004/03/addressing"/>
    <xsd:import schemaLocation="../../wsrf/faults/WS-BaseFaults.xsd"
        namespace="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd"/>


    <!-- AttributeEnumerationType -->
    <xsd:simpleType name="AttributeEnumerationType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="dateTime"/>
        <xsd:enumeration value="decimal"/>
        <xsd:enumeration value="integer"/>
        <xsd:enumeration value="string"/>
      </xsd:restriction>
    </xsd:simpleType>

    <!-- AttributeDefinitionType -->
    <xsd:complexType name="AttributeDefinitionType">
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="required" type="xsd:boolean" default="false"/>
      <xsd:attribute name="type" type="tns:AttributeEnumerationType"
                     use="required"/>
    </xsd:complexType>

    <!-- AttributeType -->
    <xsd:complexType name="AttributeType">
      <xsd:simpleContent>
        <xsd:extension base="xsd:string">
          <xsd:attribute name="name" type="xsd:string" use="required"/>
```

```
        <xsd:attribute name="type" type="tns:AttributeEnumerationType"
                       use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <!-- NameType -->
  <xsd:complexType name="NameType">
      <xsd:sequence>
          <xsd:element name="attribute" type="tns:AttributeType"
                       minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="fqname" type="xsd:string" use="required"/>
  </xsd:complexType>

  <!-- MappingType -->
  <xsd:complexType name="MappingType">
    <xsd:attribute name="end1" type="xsd:string" use="required"/>
    <xsd:attribute name="end2" type="xsd:string" use="required"/>
  </xsd:complexType>

  <!-- CatalogType -->
  <xsd:complexType name="CatalogType">
    <xsd:sequence>
      <xsd:element name="definition" type="tns:AttributeDefinitionType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="mapping" type="tns:MappingType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="name" type="tns:NameType"
                   minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>


  <!-- Operation types
  -->

  <!-- MappingStatusEnumerationType -->
  <xsd:simpleType name="MappingStatusEnumerationType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="end1Exists"/>
      <xsd:enumeration value="end1NotFound"/>
```

```
      <xsd:enumeration value="end2Exists"/>
      <xsd:enumeration value="end2NotFound"/>
      <xsd:enumeration value="mappingExists"/>
      <xsd:enumeration value="mappingNotFound"/>
      <xsd:enumeration value="end1ReplicaAttributeRequired"/>
      <xsd:enumeration value="end2ReplicaAttributeRequired"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- MappingStatusType -->
  <xsd:complexType name="MappingStatusType">
    <xsd:complexContent>
      <xsd:extension base="tns:MappingType">
        <xsd:attribute name="status" type="tns:MappingStatusEnumerationType"
                       use="required"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- AttributeStatusEnumerationType -->
  <xsd:simpleType name="AttributeStatusEnumerationType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="attributeExists"/>
      <xsd:enumeration value="attributeNotFound"/>
      <xsd:enumeration value="attributeDefinitionNotFound"/>
      <xsd:enumeration value="attributeRequired"/>
    </xsd:restriction>
  </xsd:simpleType>

  <!-- AttributeStatusType -->
  <xsd:complexType name="AttributeStatusType">
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="status" type="tns:AttributeStatusEnumerationType"
                   use="required"/>
  </xsd:complexType>

  <!-- NameStatusEnumerationType -->
  <xsd:simpleType name="NameStatusEnumerationType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="notFound"/>
    </xsd:restriction>
  </xsd:simpleType>
```

```
<!-- NameStatusType -->
<xsd:complexType name="NameStatusType">
  <xsd:sequence>
    <xsd:element name="attributeStatus" type="tns:AttributeStatusType"
                 minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="status" type="tns:NameStatusEnumerationType"
                 use="optional"/>
</xsd:complexType>

<!-- RenameType -->
<xsd:complexType name="RenameType">
  <xsd:attribute name="old" type="xsd:string" use="required"/>
  <xsd:attribute name="new" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- RenameStatusEnumerationType -->
<xsd:simpleType name="RenameStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="oldNotFound"/>
    <xsd:enumeration value="newExists"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- RenameStatusType -->
<xsd:complexType name="RenameStatusType">
  <xsd:complexContent>
    <xsd:extension base="tns:RenameType">
      <xsd:attribute name="status" type="tns:RenameStatusEnumerationType"
                     use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<!-- AttributeDefinitionStatusEnumerationType -->
<xsd:simpleType name="AttributeDefinitionStatusEnumerationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="definitionExists"/>
    <xsd:enumeration value="definitionNotFound"/>
  </xsd:restriction>
```

```
    </xsd:simpleType>

    <!-- AttributeDefinitionStatusType -->
    <xsd:complexType name="AttributeDefinitionStatusType">
      <xsd:complexContent>
        <xsd:extension base="tns:AttributeDefinitionType">
          <xsd:attribute name="status"
                  type="tns:AttributeDefinitionStatusEnumerationType"
                  use="required"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>


    <!-- Fault Types
    -->

    <!-- NameMapperFaultType -->
    <xsd:complexType name="NameMapperFaultType">
        <xsd:complexContent>
            <xsd:extension base="wsbf:BaseFaultType"/>
        </xsd:complexContent>
    </xsd:complexType>

   <!-- InternalErrorFaultType -->
    <xsd:complexType name="InternalErrorFaultType">
        <xsd:complexContent>
            <xsd:extension base="tns:NameMapperFaultType"/>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- NameMapperFault -->
    <xsd:element name="NameMapperFault" type="tns:NameMapperFaultType"/>
    <xsd:element name="InternalErrorFault" type="tns:InternalErrorFaultType"/>

</xsd:schema>
```

Appendix B – NameMapper WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="NameMapper" targetNamespace="http://www.globus.org/namespaces/2005/07/name/mapper"
xmlns:wsrp="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd"
xmlns:wsbf="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd"
xmlns:tns="http://www.globus.org/namespaces/2005/07/name/mapper" xmlns:wsrpw="http://docs.oasis-
open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.wsdl" xmlns:wsrlw="http://docs.oasis-
open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.wsdl"
xmlns:wsdlpp="http://www.globus.org/namespaces/2004/10/WSDLPreprocessor"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsntw="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.wsdl"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:import namespace="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.wsdl"
location="../../wsrf/lifetime/WS-ResourceLifetime.wsdl"/>
    <wsdl:import namespace="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.wsdl"
location="../../wsrf/properties/WS-ResourceProperties.wsdl"/>
    <wsdl:import namespace="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.wsdl"
location="../../wsrf/notification/WS-BaseN.wsdl"/>
    <wsdl:types>
        <xsd:schema elementFormDefault="qualified"
targetNamespace="http://www.globus.org/namespaces/2005/07/name/mapper" xmlns:rpns0="http://docs.oasis-
open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd"
xmlns:tns="http://www.globus.org/namespaces/2005/07/name/mapper" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-01.xsd"
schemaLocation="../../wsrf/notification/WS-BaseN.xsd"/>


            <xsd:import namespace="http://schemas.xmlsoap.org/ws/2004/03/addressing"
schemaLocation="../../ws/addressing/WS-Addressing.xsd"/>

            <xsd:include schemaLocation="mapper_types.xsd"/>

            <xsd:import namespace="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-01.xsd"
schemaLocation="../../wsrf/faults/WS-BaseFaults.xsd"/>

            <!-- NameMapper ResourceProperties -->
            <xsd:element name="NameMapperResourceProperties">
                <xsd:complexType>
                    <xsd:sequence>
```

```
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="definition"
type="tns:AttributeDefinitionType"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="name" type="tns:NameType"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="mapping" type="tns:MappingType"/>

                        <xsd:element maxOccurs="unbounded" minOccurs="1" ref="rpns0:Topic"/>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" ref="rpns0:TopicExpressionDialects"/>
                        <xsd:element maxOccurs="1" minOccurs="1" ref="rpns0:FixedTopicSet"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <!-- Register -->
            <xsd:element name="Register">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapping" type="tns:MappingType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="RegisterResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="mappingStatus"
type="tns:MappingStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <!-- Remove -->
            <xsd:element name="Remove">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapping" type="tns:MappingType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="RemoveResponse">
                <xsd:complexType>
                    <xsd:sequence>
```

```
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="mappingStatus"
type="tns:MappingStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <!-- Rename -->
            <xsd:element name="Rename">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="rename" type="tns:RenameType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="RenameResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="renameStatus"
type="tns:RenameStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <!-- DefineAttributes -->
            <xsd:element name="DefineAttributes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="definition"
type="tns:AttributeDefinitionType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="DefineAttributesResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="definitionStatus"
type="tns:AttributeDefinitionStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
```

```
            <!-- UndefineAttributes -->
            <xsd:element name="UndefineAttributes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="definition"
type="tns:AttributeDefinitionType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="UndefineAttributesResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="definitionStatus"
type="tns:AttributeDefinitionStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <!-- SetAttributes -->
            <xsd:element name="SetAttributes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="name" type="tns:NameType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="SetAttributesResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="nameStatus" type="tns:NameStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <!-- UnsetAttributes -->
            <xsd:element name="UnsetAttributes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="1" name="name" type="tns:NameType"/>
```

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

            <xsd:element name="UnsetAttributesResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element maxOccurs="unbounded" minOccurs="0" name="nameStatus" type="tns:NameStatusType"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="RenameOutputMessage">
        <wsdl:part name="parameters" element="tns:RenameResponse"/>
    </wsdl:message>
    <wsdl:message name="DefineAttributesInputMessage">
        <wsdl:part name="parameters" element="tns:DefineAttributes"/>
    </wsdl:message>
    <wsdl:message name="UndefineAttributesOutputMessage">
        <wsdl:part name="parameters" element="tns:UndefineAttributesResponse"/>
    </wsdl:message>
    <wsdl:message name="SetAttributesInputMessage">
        <wsdl:part name="parameters" element="tns:SetAttributes"/>
    </wsdl:message>
    <wsdl:message name="NameMapperFaultMessage">
        <wsdl:part name="fault" element="tns:NameMapperFault"/>
    </wsdl:message>
    <wsdl:message name="UnsetAttributesInputMessage">
        <wsdl:part name="parameters" element="tns:UnsetAttributes"/>
    </wsdl:message>
    <wsdl:message name="RegisterOutputMessage">
        <wsdl:part name="parameters" element="tns:RegisterResponse"/>
    </wsdl:message>
    <wsdl:message name="UndefineAttributesInputMessage">
        <wsdl:part name="parameters" element="tns:UndefineAttributes"/>
    </wsdl:message>
    <wsdl:message name="RenameInputMessage">
        <wsdl:part name="parameters" element="tns:Rename"/>
    </wsdl:message>
    <wsdl:message name="RegisterInputMessage">
```

```
        <wsdl:part name="parameters" element="tns:Register"/>
    </wsdl:message>
    <wsdl:message name="SetAttributesOutputMessage">
        <wsdl:part name="parameters" element="tns:SetAttributesResponse"/>
    </wsdl:message>
    <wsdl:message name="UnsetAttributesOutputMessage">
        <wsdl:part name="parameters" element="tns:UnsetAttributesResponse"/>
    </wsdl:message>
    <wsdl:message name="InternalErrorFaultMessage">
        <wsdl:part name="fault" element="tns:InternalErrorFault"/>
    </wsdl:message>
    <wsdl:message name="RemoveInputMessage">
        <wsdl:part name="parameters" element="tns:Remove"/>
    </wsdl:message>
    <wsdl:message name="DefineAttributesOutputMessage">
        <wsdl:part name="parameters" element="tns:DefineAttributesResponse"/>
    </wsdl:message>
    <wsdl:message name="RemoveOutputMessage">
        <wsdl:part name="parameters" element="tns:RemoveResponse"/>
    </wsdl:message>
    <wsdl:portType name="NameMapperPortType" wsrp:ResourceProperties="tns:NameMapperResourceProperties">
        <wsdl:operation name="Register">
            <wsdl:input message="tns:RegisterInputMessage"/>
            <wsdl:output message="tns:RegisterOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="Remove">
            <wsdl:input message="tns:RemoveInputMessage"/>
            <wsdl:output message="tns:RemoveOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="Rename">
            <wsdl:input message="tns:RenameInputMessage"/>
            <wsdl:output message="tns:RenameOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="SetAttributes">
            <wsdl:input message="tns:SetAttributesInputMessage"/>
            <wsdl:output message="tns:SetAttributesOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="UnsetAttributes">
```

```
            <wsdl:input message="tns:UnsetAttributesInputMessage"/>
            <wsdl:output message="tns:UnsetAttributesOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="DefineAttributes">
            <wsdl:input message="tns:DefineAttributesInputMessage"/>
            <wsdl:output message="tns:DefineAttributesOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="UndefineAttributes">
            <wsdl:input message="tns:UndefineAttributesInputMessage"/>
            <wsdl:output message="tns:UndefineAttributesOutputMessage"/>
            <wsdl:fault name="fault" message="tns:NameMapperFaultMessage"/>
        </wsdl:operation>
        <wsdl:operation name="QueryResourceProperties">
            <wsdl:input name="QueryResourcePropertiesRequest" message="wsrpw:QueryResourcePropertiesRequest"
wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties/QueryResourceProperties"/>
            <wsdl:output name="QueryResourcePropertiesResponse" message="wsrpw:QueryResourcePropertiesResponse"
wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties/QueryResourcePropertiesResponse"/>
            <wsdl:fault name="InvalidResourcePropertyQNameFault" message="wsrpw:InvalidResourcePropertyQNameFault"/>
            <wsdl:fault name="InvalidQueryExpressionFault" message="wsrpw:InvalidQueryExpressionFault"/>
            <wsdl:fault name="QueryEvaluationErrorFault" message="wsrpw:QueryEvaluationErrorFault"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrpw:ResourceUnknownFault"/>
            <wsdl:fault name="UnknownQueryExpressionDialectFault" message="wsrpw:UnknownQueryExpressionDialectFault"/>
        </wsdl:operation>
        <wsdl:operation name="GetMultipleResourceProperties">
            <wsdl:input name="GetMultipleResourcePropertiesRequest"
message="wsrpw:GetMultipleResourcePropertiesRequest" wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
ResourceProperties/GetMultipleResourceProperties"/>
            <wsdl:output name="GetMultipleResourcePropertiesResponse"
message="wsrpw:GetMultipleResourcePropertiesResponse" wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
ResourceProperties/GetMultipleResourcePropertiesResponse"/>
            <wsdl:fault name="InvalidResourcePropertyQNameFault" message="wsrpw:InvalidResourcePropertyQNameFault"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrpw:ResourceUnknownFault"/>
        </wsdl:operation>
        <wsdl:operation name="GetResourceProperty">
            <wsdl:input name="GetResourcePropertyRequest" message="wsrpw:GetResourcePropertyRequest"
wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties/GetResourceProperty"/>
            <wsdl:output name="GetResourcePropertyResponse" message="wsrpw:GetResourcePropertyResponse"
wsa:Action="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties/GetResourcePropertyResponse"/>
            <wsdl:fault name="InvalidResourcePropertyQNameFault" message="wsrpw:InvalidResourcePropertyQNameFault"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrpw:ResourceUnknownFault"/>
```

```
        </wsdl:operation>
        <wsdl:operation name="Subscribe">
            <wsdl:input message="wsntw:SubscribeRequest" wsa:Action="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-
BaseNotification/Subscribe"/>
            <wsdl:output message="wsntw:SubscribeResponse" wsa:Action="http://docs.oasis-open.org/wsn/2004/06/wsn-WS-
BaseNotification/SubscribeResponse"/>
            <wsdl:fault name="TopicNotSupportedFault" message="wsntw:TopicNotSupportedFault"/>
            <wsdl:fault name="InvalidTopicExpressionFault" message="wsntw:InvalidTopicExpressionFault"/>
            <wsdl:fault name="SubscribeCreationFailedFault" message="wsntw:SubscribeCreationFailedFault"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsntw:ResourceUnknownFault"/>
            <wsdl:fault name="TopicPathDialectUnknownFault" message="wsntw:TopicPathDialectUnknownFault"/>
        </wsdl:operation>
        <wsdl:operation name="GetCurrentMessage">
            <wsdl:input message="wsntw:GetCurrentMessageRequest" wsa:Action="http://docs.oasis-
open.org/wsn/2004/06/wsn-WS-BaseNotification/GetCurrentMessage"/>
            <wsdl:output message="wsntw:GetCurrentMessageResponse" wsa:Action="http://docs.oasis-
open.org/wsn/2004/06/wsn-WS-BaseNotification/GetCurrentMessageResponse"/>
            <wsdl:fault name="TopicNotSupportedFault" message="wsntw:TopicNotSupportedFault"/>
            <wsdl:fault name="InvalidTopicExpressionFault" message="wsntw:InvalidTopicExpressionFault"/>
            <wsdl:fault name="NoCurrentMessageOnTopicFault" message="wsntw:NoCurrentMessageOnTopicFault"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsntw:ResourceUnknownFault"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```