GFD-Unknown
Category: Unknown
OGSA Naming WG

Editor: Mark Morgan, University of Virginia
27 June 2008
http://forge.gridforum.org/projects/ogsa-naming-wg

RNS Specification 1.1

Status of This Memo

This memo provides information to the Grid community on a simple higher-level naming grid service that allows users to associate human-readable names to WS-Addressing **[WS-Addressing]** Endpoint Reference Types. Distribution is unlimited.

Copyright Notice

Trademarks

OGSA is a trademark of the Open Grid Forum.

Abstract

In their 2002 book, “Distributed Systems: Principles and Paradigms”, Andrew Tannenbaum and Martin van Steen describe in great detail the properties, function, and benefit of naming schemes in distributed systems **[Tannenbaum]**. Specifically, they describe a typical three-layer naming scheme whereby human readable names map to location-independent names or identifiers, which in turn map to location-dependent addresses. This three-tiered approach is instrumental in providing both usability for clients, as well as many of the classic distributed systems “transparencies” like fault and location transparency. WS-Naming **[WS-Naming]** provides the mapping between location-independent names (in the form of *EndpointIdentifiers*) and location-dependent addresses (i.e., WS-Addressing EPRs). In this specification we describe the Resource Namespace Service (RNS), a grid port type that allows clients to manipulate and retrieve mappings from human-readable strings to WS-Addressing Endpoint Reference Types[1], thus providing the higher level mapping described by Tannenbaum and van Steen.

1 Of course, since WS-Naming compliant endpoints are merely extensions on WS-Addressing EPRs, RNS can map to either.

Table of Contents

1 Introduction....4
1.1 Outline for this Document....4
1.2 Terminology....4
1.3 Namespaces....5
2 RNS Port Type....5
2.1 RNS Interface....5
2.1.1 RNS elementCount Property....6
2.1.2 RNS createTime Property....6
2.1.3 RNS accessTime Property....6
2.1.4 RNS modificationTime Property....6
2.1.5 RNS readable Property....6
2.1.6 RNS writeable Property....7
2.2 RNS add Operation....7
2.2.1 RNS add....7
2.2.2 Example SOAP Encoding of the add Message Exchange....9
2.3 RNS lookup Operation....11
2.3.1 RNS lookup....12
2.3.2 Example SOAP Encoding of the lookup Message Exchange....13
2.4 RNS remove Operation....15
2.4.1 RNS remove....15
2.4.2 Example SOAP Encoding of the remove Message Exchange....17
2.5 RNS rename Operation....18
2.5.1 RNS rename....18
2.5.2 Example SOAP Encoding of the rename Message Exchange....20
2.6 RNS setMetadata Operation....21
2.6.1 RNS setMetadata....21
2.6.2 Example SOAP Encoding of the setMetadata Message Exchange....23
3 RNS Properties....24
4 Recommended RNS Entry Metadata....25
5 Faults and Failures....25
5.1 Available Faults and Failures....26
5.2 Message Exchange Failures for RNS....26
5.2.1 Add....26
5.2.2 lookup....26
5.2.3 rename....26
5.2.4 remove....27
5.2.5 setMetadata....27
6 Security Considerations....27
7 Author Information....27
8 Glossary....27
9 Intellectual Property Statement....27
10 References....28

Appendix A: Differences Between RNS 1.0 and RNS 1.1.....30

1 Introduction

In their 2002 book, “Distributed Systems: Principles and Paradigms”, Andrew Tannenbaum and Martin van Steen describe in great detail the properties, function, and benefit of naming schemes in distributed systems **[Tannenbaum]**. Specifically, they describe a typical three-layer naming scheme whereby human readable names map to location-independent names or identifiers, which in turn map to location-dependent addresses. This three-tiered approach is instrumental in providing both usability for clients, as well as many of the classic distributed systems “transparencies” like fault and location transparency. WS-Naming **[WS-Naming]** provides the mapping between location-independent names (in the form of *EndpointIdentifiers*) and location-dependent addresses (i.e., WS-Addressing EPRs). In this specification we describe the Resource Namespace Service (RNS), a grid port type that allows clients to manipulate and retrieve mappings from human-readable strings to WS-Addressing Endpoint Reference Types, thus providing the higher level mapping described by Tannenbaum and van Steen.

This specification is based off of version 1.0 of the RNS specification. Most changes from that original document are syntactic in nature and the general behavior of RNS remains unchanged. Version 1.1 was created to simplify the interface slightly and to clear up ambiguities that arose as part of implementation and testing.

1.1 Outline for this Document

The remainder of this document will be organized as follows. First, we will present a high level overview of the port type we recommend for the RNS specification. We will follow this with sections which drill down into the details of the port type. Born of the necessity to support potentially numerous OGSA Basic Profiles (of which at the time of this document's writing, only one such profile exists – the OGSA WSRF Basic Profile 1.0 **[WSRFProfileDoc]**), explicit WSDL cannot be given, but a pseudo-schema for the port types will be indicated where applicable[2]. Finally, we will summarize the information in this document and wrap up with information about security considerations, author information, and glossary terms. Accompanying this document will be a number of *Profile Rendering* documents which will normatively describe the details as they pertain to their respective OGSA Basic Profile.

1.2 Terminology

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

In addition to the terms introduced in **[RFC2119]**, additional terms commonly used in this document are defined in the Glossary in the back.

2 In order to give normaitve specifications for various port types and in light of this requirement that OGSA specifications are referent to basic profiles of a diverse nature, it seems obvious that any specification will need to be accompanied by various *rendering* documents which will describe normatively how to map the basic port types to the various profiles.

When describing abstract data models, this specification uses the notational convention used by the **[XML Infoset]**.

When describing concrete XML schemas, 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 **[Xpath]** notation (e.g., /x:MyHeader/x:SomeProperty@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/>).

1.3 Namespaces

The following namespaces are used in this document:

Prefix	Namespace
s11	http://schemas.xmlsoap.org/soap/envelope
xsd	http://www.w3.org/2001/XMLSchema
wsa	http://www.w3.org/2005/08/addressing
iterator	http://schemas.ogf.org/ws-iterator/2008/06/iterator
rns	http://schemas.ogf.org/rns/2008/06/rns

2 RNS Port Type

The RNS port type consists of five operations that give clients the ability to query and manipulate the contents of an RNS directory[3]. Associated with each entry are arbitrary XML documents containing a mixture of user settable information (presumably referring to the associated entry though this restriction is not explicitly stated) as well as RNS required and RNS implementation dependent metadata.

2.1 RNS Interface

The RNS port allows clients to manipulate mappings of human readable names to WS-Addressing endoints. The RNS interface is conceptually defined as follows:

3 Because RNS in many ways is the grid equivalent of a file system directory, we will often refer to RNS resources as such throughout this document.

RNS
elementCount: unsignedLong createTime: dateTime accessTime: dateTime modificationTime: dateTime readable: boolean writeable: boolean
add(entryName: String , [entryEndpoint: EPR], [entryMetadata: XML]): RNSEntry lookup([entryName: String]): LookupResults remove(entryName: String): RNSEntry rename(oldEntryName: String, newEntryName: String): RNSEntry setMetadata(String entryName, XML newMetadata): RNSEntry

2.1.1 RNS elementCount Property

The elementCount property is a required property describing the total number of elements contained within this RNS resource. This property has a cardinality of exactly 1.

2.1.2 RNS createTime Property

The createTime property is an optional property which an RNS implementation SHOULD advertise indicating the time[4] at which the RNS resource was created. This property has a cardinality of [0, 1].

2.1.3 RNS accessTime Property

The accessTime property is an optional property which an RNS implementation SHOULD advertise indicating the time at which the RNS resource was last accessed. This property has a cardinality of [0, 1].

2.1.4 RNS modificationTime Property

The modificationTime property is an optional property which an RNS implementation SHOULD advertise indicating the time at which the RNS resource was last modified. This property has a cardinality of [0, 1].

2.1.5 RNS readable Property

4 Unless otherwise stated, any reference to timestamps used within this document specifically refer to the timestamp as indicated by the host machine on which the resource resides or was created. No attempt and implementing a global clock is suggested or indicated by this specification.

The readable property is a required property which an RNS implementation MUST use to advertise whether or not a given RNS resource is readable (can respond to a lookup operation). This property has a cardinality of exactly 1. This property is not an indication of security. An RNS implementation may advertise that it is readable without the client having permission to read it. All this property is meant to do is declare whether or not reads on this RNS resource have a meaning. In all likely-hood, this property will always be true as an RNS directory which cannot be read makes little sense, but the property is included for symmetry with the writable property and there seemed little reason to limit the possibility of a write only RNS resource.

2.1.6 RNS writeable Property

The writeable property is a required property which an RNS implementation MUST use to advertise whether or not a given RNS resource is writeable (can respond to an add, rename, remove, or setMetadata operation). This property has a cardinality of excatly 1. This property is not an indication of security. An RNS implementation may advertise that is is writeable without the client having permission to write it. All this property is meant to do is declare whether or not writes to an RNS resource have a meaning. For example, consider the case of an RNS resource representing all of the users in a NIS service. That RNS could be used as a means of naming those users in a read-only fashion (writes obviously would not be permitted in the general case).

2.2 RNS add Operation

The add operation is used by clients who wish to add a new entry to an RNS map. This operation can be requested without an entry endpoint in which case the client is indicating a desire to have a new RNS entry created whose EPR refers to a newly created RNS resource (a sub-directory if you will). If a resource endpoint is given in the request message, then an XML document describing desired metadata MAY also be given indicating initial metadata to set on the mapping. The absence of metadata in the add operation neither prohibits the setting of metadata in the future using the setMetadata operation, nor does it necessarily indicate that an entry will have no metadata associated with it as an RNS implementation is free to (and in some cases required to) add it's own metadata to the metadata document. The RNS resource MUST respond to an add request message with an addResponse message.

2.2.1 RNS add

The format of the add Message is:

```
...
<rns:add>
        <rns:entry-name> rns:entryNameType </rns:entryName>
        <rns:entry-endpoint> wsa:EndpointReferenceType </rns:entry-endpoint> ?
        <rns:entry-metadata> {any} * </rns:entry-metadata> ?
</rns:add>
...
```

The components of the add message are further described as follows:

/rns:entry-name

This element gives the name for the RNS entry. This element is essentially of type string but is restricted so as to not contain any unprintable characters nor the forward slash (/) character.

/rns:entry-endpoint

This element indicates the endpoint address (WS-Addressing EPR) of an existing endpoint[5] that the client wishes for the new RNS entry to refer to. This element is optional and if not included the RNS implementation MUST create a new RNS resource to add using the given name. An RNS implementation MAY use any service provider to create that RNS endpoint though it is generally assumed that the same service provider as the target RNS resource is used.

/rns:entry-metadata

This element indicates 0 or more arbitrary XML documents that the client wishes to associate with the given RNS entry. The only restriction to this metadata is that no element in the document can be in the **rns** namespace. The RNS specification reserves the right to use that namespace to indicate elements of metadata that are determined by the specification, and by the implementations thereof.

The response to the add message is a message of the form:

5 Actually, the endpoint need not “exist” in any real sense of the word. We merely imply here that it is not the RNS implementation's responsibility to create a new endpoint.

```
...
<rns:addResponse>
        <rns:entry name=" rns:EntryNameType ">
                <rns:endpoint> wsa:EndpointReferenceType </rns:endpoint>
                <rns:metadata>
                        <rns:supports-rns value=" rns:supportType "/>
                        {any} *
                </rns:metadata>
        </rns:entry>
</rns:addResponse>
...
```

The components of the addResponse message are further described as follows:

/rns:entry
This element represents a single entry within an RNS resource.

/rns:entry@name
This property indicates the name that the entry possesses within the target RNS resource.

/rns:entry/rns:endpoint
This element gives the WS-Addressing EndpointReferenceType for the entry indicated.

/rns:metadata
This element indicates any metadata associated with the given RNS entry. Most of the contents of this element are arbitrary set by either the client or the implementation. One element however is recommended and is described later in this document in the section on metadata.

2.2.2 Example SOAP Encoding of the add Message Exchange

The following is a non-normative example of an add request message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11=”http://www.w3.org/2003/05/soap-envelope”
        xmlns:rns=”http://schemas.ogf.org/rns/2008/06/rns”
        xmlns:wsa=”http://www.w3.org/2005/08/addressing”>
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/add
                </wsa:Action>
                <wsa:To s11:mustUnderstand=”1”>
                        http://tempuri.org/rns-source
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:add>
                        <rns:entry-name>MyEntry</rns:entryName>
                        <rns:entry-endpoint>
                                <wsa:Address>http://tempuri.org/entry</wsa:Address>
                        </rns:entry-endpoint>
                        <rns:entry-metadata>
                                <arbitrary-xml-data/>
                        </rns:entry-metadata>
                </rns:add>
        </s11:Body>
</s11:Envelope>
```

The following is a non-normative example of an add response message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/addResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:addResponse>
                        <rns:entry name="MyEntry">
                                <rns:endpoint>
                                        <wsa:Address>http://tempuri.org/entry</wsa:Address>
                                </rns:endpoint>
                                <rns:metadata>
                                        <rns:supports-rns value="false"/>
                                        <arbitrary-xml-data/>
                                </rns:metadata>
                        </rns:entry>
                </rns:addResponse>
        </s11:Body>
</s11:Envelope>
```

2.3 RNS lookup Operation

The lookup operation is used by clients who want to obtain an entry (or all entries) within a given target RNS directory. Clients can indicate either a specific entry to retrieve, or the desire to retrieve all entries of an RNS. The service endpoint is free to respond with either a list of entries in the SOAP response message, the EPR of an iterator (the exact specification of this iterator is *profile rendering* dependent and hence not defined in this document), or both a list of entries and an iterator. If an iterator is not returned, then the list of entries included in the message MUST represent all available entries that match the lookup request operation. If both a list of entries and an iterator are returned, then the iterator MUST contain all matching entries from the lookup request that are not part of the returned entries (I.e., the list of returned entries and the entries contained in the iterator are disjoint, and the union set of all their entries constitutes all matching entries from the RNS). Finally, a RNS service

resource MUST respond with a lookupResponse message containing neither a list of entries, nor an iterator in the case when a lookup message is sent with no entry name to match and when the RNS resource is empty (if a lookup entry name is given, the RNS resource must either respond with the valid entry, or must fault as indicated later in this document).

2.3.1 RNS lookup

The format of the lookup Message is:

```
...
<rns:lookup>
        <rns:entry-name> rns:entryNameType </rns:entryName> ?
</rns:lookup>
...
```

The components of the lookup message are further described as follows:

/rns:entry-name

This optional element gives the name for the RNS entry which the caller wishes to look up. This element is essentially of type string but is restricted so as to not contain any unprintable characters nor the forward slash (/) character. If this element is missing or null, then the caller is asking for a list of all contained entries.

The response to the lookup message is a message of the form:

```
...
<rns:lookupResponse>
        <rns:entry name=" rns:EntryNameType ">
                <rns:endpoint> wsa:EndpointReferenceType </rns:endpoint>
                <rns:metadata>
                        <rns:supports-rns value=" rns:supportType "/>
                        {any} *
                </rns:metadata>
        </rns:entry> *
        <rns:iterator> wsa:EndpointReferenceType </rns:iterator> ?
</rns:lookupResponse>
...
```

The components of the lookupResponse message are further described as follows:

/rns:entry
: This element represents a single entry within an RNS resource.

/rns:entry@name
: This property indicates the name that the entry possesses within the target RNS resource.

/rns:entry/rns:endpoint
: This element gives the WS-Addressing EndpointReferenceType for the entry indicated.

/rns:metadata
: This element indicates any metadata associated with the given RNS entry. Most of the contents of this element are arbitrary set by either the client or the implementation. One element however is recommended and is described later in this document in the section on metadata.

/rns:iterator
: This element indicates the WS-Addressing EndpointReferenceType for an iterator which can be used by a client to acquire the remainder (those entries not given directly in the message) of the entries matching the user's query.

2.3.2 Example SOAP Encoding of the lookup Message Exchange

The following is a non-normative example of an lookup request message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11=”http://www.w3.org/2003/05/soap-envelope”
        xmlns:rns=”http://schemas.ogf.org/rns/2008/06/rns”
        xmlns:wsa=”http://www.w3.org/2005/08/addressing”>
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/lookup
                </wsa:Action>
                <wsa:To s11:mustUnderstand=”1”>
                        http://tempuri.org/rns-source
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:lookup>
                        <rns:entry-name>MyEntry</rns:entryName>
                </rns:lookup>
        </s11:Body>
</s11:Envelope>
```

The following is a non-normative example of an lookup response message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/lookupResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:lookupResponse>
                        <rns:entry name="MyEntry">
                                <rns:endpoint>
                                        <wsa:Address>http://tempuri.org/entry</wsa:Address>
                                </rns:endpoint>
                                <rns:metadata>
                                        <rns:supports-rns value="false"/>
                                        <arbitrary-xml-data/>
                                </rns:metadata>
                        </rns:entry>
                        <rns:iterator>
                                <wsa:Address>http://tempuri.org/iterator</wsa:Address>
                        </rns:iterator>
                </rns:lookupResponse>
        </s11:Body>
</s11:Envelope>
```

2.4 RNS remove Operation

The remove operation is used by clients who want permanently remove an entry from an RNS resource. The RNS service MUST respond to a remove request message with a removeResponse message.

2.4.1 RNS remove

The format of the remove Message is:

```
...
<rns:remove>
        <rns:entry-name> rns:entryNameType </rns:entryName>
</rns:remove>
...
```

The components of the lookup message are further described as follows:

/rns:entry-name
: This element gives the name for the RNS entry which the caller wishes to remove. This element is essentially of type string but is restricted so as to not contain any unprintable characters nor the forward slash (/) character.

The response to the remove message is a message of the form:

```
...
<rns:removeResponse>
        <rns:entry name=" rns:EntryNameType ">
                <rns:endpoint> wsa:EndpointReferenceType </rns:endpoint>
                <rns:metadata>
                        <rns:supports-rns value=" rns:supportType "/>
                        {any} *
                </rns:metadata>
        </rns:entry>
</rns:removeResponse>
...
```

The components of the removeResponse message are further described as follows:

/rns:entry
: This element represents a single entry formerly within an RNS resource.

/rns:entry@name
: This property indicates the name that the entry possessed within the target RNS resource.

/rns:entry/rns:endpoint
: This element gives the WS-Addressing EndpointReferenceType for the entry removed.

/rns:metadata

This element indicates any metadata associated with the given RNS entry. Most of the contents of this element are arbitrary set by either the client or the implementation. One element however is mandatory and is described later in this document in the section on metadata.

2.4.2 Example SOAP Encoding of the remove Message Exchange

The following is a non-normative example of an remove request message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/remove
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:remove>
                        <rns:entry-name>MyEntry</rns:entryName>
                </rns:remove>
        </s11:Body>
</s11:Envelope>
```

The following is a non-normative example of an remove response message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/removeResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:removeResponse>
                        <rns:entry name="MyEntry">
                                <rns:endpoint>
                                        <wsa:Address>http://tempuri.org/entry</wsa:Address>
                                </rns:endpoint>
                                <rns:metadata>
                                        <rns:supports-rns value="false"/>
                                        <arbitrary-xml-data/>
                                </rns:metadata>
                        </rns:entry>
                </rns:removeResponse>
        </s11:Body>
</s11:Envelope>
```

2.5 RNS rename Operation

The rename operation is used by clients who want permanently rename an entry within an RNS resource. The rename operation can only be used by clients to change the name of an existing entry within an RNS directory to another name within the same RNS directory. The RNS service MUST respond to a rename request message with a renameResponse message.

2.5.1 RNS rename

The format of the rename Message is:

```
...
<rns:rename>
        <rns:old-entry-name> rns:entryNameType </rns:old-entry-name>
        <rns:new-entry-name> rns:entryNameType </rns:new-entry-name>
</rns:rename>
...
```

The components of the rename message are further described as follows:

/rns:old-entry-name
: This element gives the original name for the RNS entry which the caller wishes to rename. This element is essentially of type string but is restricted so as to not contain any unprintable characters nor the forward slash (/) character.

/rns:new-entry-name
: This element gives the new name for the RNS entry to which the caller wishes to rename the old entry. This element is essentially of type string but is restricted so as to not contain any unprintable characters nor the forward slash (/) character.

The response to the rename message is a message of the form:

```
...
<rns:renameResponse>
        <rns:entry name=" rns:EntryNameType ">
                <rns:endpoint> wsa:EndpointReferenceType </rns:endpoint>
                <rns:metadata>
                        <rns:supports-rns value=" rns:supportType "/>
                        {any} *
                </rns:metadata>
        </rns:entry>
</rns:renameResponse>
...
```

The components of the renameResponse message are further described as follows:

/rns:entry
: This element represents a single entry within an RNS resource.

/rns:entry@name
This property indicates the new name that the entry possesses after the rename operation.

/rns:entry/rns:endpoint
This element gives the WS-Addressing EndpointReferenceType for the entry renamed.

/rns:metadata
This element indicates any metadata associated with the given RNS entry. Most of the contents of this element are arbitrary set by either the client or the implementation. One element however is mandatory and is described later in this document in the section on metadata.

2.5.2 Example SOAP Encoding of the rename Message Exchange

The following is a non-normative example of an rename request message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/rename
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:rename>
                        <rns:old-entry-name>MyEntry</rns:old-entry-name>
                        <rns:new-entry-name>MyEntry.2</rns:new-entry-name>
                </rns:rename>
        </s11:Body>
</s11:Envelope>
```

The following is a non-normative example of an rename response message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/renameResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:renameResponse>
                        <rns:entry name="MyEntry.2">
                                <rns:endpoint>
                                        <wsa:Address>http://tempuri.org/entry</wsa:Address>
                                </rns:endpoint>
                                <rns:metadata>
                                        <rns:supports-rns value="false"/>
                                        <arbitrary-xml-data/>
                                </rns:metadata>
                        </rns:entry>
                </rns:renameResponse>
        </s11:Body>
</s11:Envelope>
```

2.6 RNS setMetadata Operation

The setMetadata operation is used by clients who want to replace the user-defined metadata for an entry within an RNS service resource. This operation is only guaranteed to modify the user-defined metadata within that resource. All metadata maintained by the RNS service itself (whether required or implementation specific) is maintained at the discretion of that service implementation. The RNS service MUST respond to a setMetadata request message with a setMetadataResponse message.

2.6.1 RNS setMetadata

The format of the setMetadata Message is:

```
...
<rns:setMetadata>
        <rns:entry-name> rns:entryNameType </rns:entry-name>
        <rns:metadata>
                {any} *
        </rns:metadata>
</rns:setMetadata>
...
```

The components of the setMetadata message are further described as follows:

/rns:entry-name

This element gives the name for the RNS entry for which the caller wishes to replace metadata. This element is essentially of type string but is restricted so as to not contain any unprintable characters nor the forward slash (/) character.

/rns:metadata

This element contains an arbitrary set of XML documents representing the new metadata that the caller wishes to store with the given endpoint.

The response to the setMetadata message is a message of the form:

```
...
<rns:setMetadataResponse>
        <rns:entry name=" rns:EntryNameType ">
                <rns:endpoint> wsa:EndpointReferenceType </rns:endpoint>
                <rns:metadata>
                        <rns:supports-rns value=" rns:supportType "/>
                        {any} *
                </rns:metadata>
        </rns:entry>
</rns:setMetadata esponse>
...
```

The components of the setMetadata esponse message are further described as follows:

/rns:entry

This element represents a single entry within an RNS resource.

/rns:entry@name
This property indicates the name of the entry for which metadata was modified.

/rns:entry/rns:endpoint
This element gives the WS-Addressing EndpointReferenceType for the entry whose metadata was modified.

/rns:metadata
This element indicates the metadata associated with the given RNS entry after the update is applied. Most of the contents of this element are arbitrary set by either the client or the implementation. One element however is mandatory and is described later in this document in the section on metadata.

2.6.2 Example SOAP Encoding of the setMetadata Message Exchange

The following is a non-normative example of a setMetadata request message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/setMetadata
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:setMetadata>
                        <rns:entry-name>MyEntry</rns:entry-name>
                        <rns:metadata>
                                <new-arbitrary-xml-data/>
                        </rns:metadata>
                </rns:setMetadata>
        </s11:Body>
</s11:Envelope>
```

The following is a non-normative example of an setMetadata response message using **[SOAP1.1]**:

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2008/06/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2008/06/rns/setMetadataResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
        </s11:Header>

        <s11:Body>
                <rns:setMetadataResponse>
                        <rns:entry name="MyEntry">
                                <rns:endpoint>
                                        <wsa:Address>http://tempuri.org/entry</wsa:Address>
                                </rns:endpoint>
                                <rns:metadata>
                                        <rns:supports-rns value="false"/>
                                        <new-arbitrary-xml-data/>
                                </rns:metadata>
                        </rns:entry>
                </rns:setMetadataResponse>
        </s11:Body>
</s11:Envelope>
```

3 RNS Properties[6]

The following table indicates the properties that RNS resources contain. The *Requirement Level* entry in the table describes whether a RNS resource MUST have the property, SHOULD have the property, or MAY have the property (as per **[RFC2119]**). This list is by no means exhaustive and implementors are free to add their own properties as they see fit.

Property	**Requirement Level**	**Description**

6 These resource properties are in addition to any resource properties required or specified by the OGSA WSRF Basic Profile 1.0 on which this specification is dependent.

elementCount	MUST	The total number of elements contained in the target WS-Iterator resource.
createTime	MAY	The time (container relative) at which the RNS resource was created.
accessTime	MAY	The time (container relative) at which the RNS resource was last accessed.
modificationTime	MAY	The time (container relative) at which the RNS resource was last modified.
readable	MUST	A boolean value indicating whether the RNS resource is readable (true) or not (false). Please see section above on property meanings for more detail.
writeable	MUST	A boolean value indicating whether the RNS resource is writeable (true) or not (false). Please see section above on property meanings for more detail.

4 Recommended RNS Entry Metadata

Metadata associated with a given RNS entry is made up of a combination of both user settable metadata, and metadata inserted dynamically by the RNS implementation. Of this dynamic RNS metadata information, one piece of metadata is recommended by this specification. This element indicates whether or not the associated resource is another RNS resource or not. Because it is not always possible to determine whether or not a given target resource implements a given port type (this is guaranteed by some *profile renderings* but not all), a value of unknown is permitted. However, users and implementors should be aware that unknown resource types will likely lead to degraded usability.

The general form of the resource type metadata elements is described as follows:

```
<rns:supports-rns value=" rns:supportType"/>
```

rns:supportType is an enumeration with one of the values {**true, false, unknown**}. If this piece of metadata is absent, then the client should assume that the value is unknown.

5 Faults and Failures

As before with properties, it is not possible to normatively describe the faulting and failure mechanisms for RNS in this document. Instead, in this section, we will non-normatively describe the fault and failure conditions in terms of causes and information available to calling clients and leave it up to the *Profile Rendering* documents to normatively describe the exact syntax for conveying the appropriate information.

5.1 Available Faults and Failures

This section describes every possible fault and failure that is relevant specifically to the RNS port type. Following this section we will indicated every message exchange possible between clients and RNS resources and list for each the faults and failures that that message exchange might generate.

Read Not Permiited Failure	This failure indicates that a read operation (lookup) was attempted on an RNS resource which does not support reads.
Write Not Permitted Failure	This failure indicates that a write operation (add, remove, rename, setMetadata) was attempted on an RNS resource which does not support writes.
RNS Entry Exists Failure	This failure indicates that a client tried to create (either via add, or rename) an entry whose name already existed withing the target RNS resource. This failure MUST include the name of the entry that already existed.
RNS Entry Does Not Exist Failure	This failure indicates that a client tried to reference an entry (via lookup, rename, remove, or setMetadata) an entry which does not exist in the target RNS resource. This failure MUST include the entry name which could not be found.

5.2 Message Exchange Failures for RNS

The following describes what failures can be generated by each RNS message exchange (in addition to failures described by the *profile rendering* and other associated port types and specifications).

5.2.1 Add

- Write Not Permitted Failure
- RNS Entry Exists Failure

5.2.2 lookup

- Read Not Permitted Failure
- RNS Entry Does Not Exist Failure

5.2.3 rename

- Write Not Permitted Failure

- RNS Entry Exists Failure
- RNS Entry Does Not Exist Failure

5.2.4 remove

- Write Not Permitted Failure
- RNS Entry Does Not Exist Failure

5.2.5 setMetadata

- Write Not Permitted Failure
- RNS Entry Does Not Exist Failure

6 Security Considerations

Security is, of course, important for RNS resources. Considering that RNS will likely be used by service port type implementations to manage and return sets or lists of data that might, in their own right, be considered sensitive. However, as always, security is a cross-cutting concern here and as such specification thereof lies outside the purview of this document.

7 Author Information

Editor:
Mark Morgan
University of Virginia, Department of Computer Science
151 Engineer's Way
P.O. Box 400740
Charlottesville, VA. 22904-4740
Phone: +1 (434) 243-2175
E-mail: mmm2a@virginia.edu

8 Glossary

Conceptual Interface An interface which describes the conceptual behavior of a service but which doesn't necessarily reflect the actual parameters and methods that are being received and sent.

9 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights

that might gbe 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 implementors or users of this specification can be obtained from the OGF Secretariat.

The OGF 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 OGF Executive Directory.

Full Copyright Notice

10 References

[WSRFProfileDoc] I. Foster, T. Maguire, D. Snelling, *OGSA WSRF Basic Profile 1.0*, https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-wsrf-basic-profile/en/15, GWS-R (draft-ggf-ogsa-wsrf-basic-profile-021), 6 July 2005.

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2991.txt, IETF RFC 2119, March 1997.

[XML-Infoset] http://www.w3.org/TR/xml-infoset/

[XPATH] http://www.w3.org/TR/xpath

[WS-Addressing] M. Gudgin, M. Hadley, and T. Rogers (ed.), *Web Services Addressing 1.0 – Core (WS-Addressing)*, 9 May 2006, http://www.w3.org/TR/2006/REC-ws-addr-core-20060509

[WS-Enumeration] http://www.w3.org/Submission/WS-Enumeration/

[ByteIOSpec] M. Morgan (ed.), *ByteIO Specification 1.0*, http://www.ggf.org/documents/GFD.88.pdf, GFD.88, 31 October 2006.

[Tannenbaum] Tannenbaum, A. and van Steen, M., *Distributed Systems: Principles and Paradigms*, Prentice Hall, 2002. p. 184-210.

[ByteIOWSRFRend] M. Morgan (ed.), *ByteIO OGSA WSRF Basic Profile Rendering 1.0*, http://www.ggf.org/documents/GFD.87.pdf, GFD.87, 31 October 2006.

[BES] http://www.ggf.org/documents/GFD.108.pdf

[WS-Naming] http://www.ggf.org/documents/GFD.109.pdf

[SOAP1.1] http://www.w3.org/TR/soap11

Appendix A: Differences Between RNS 1.0 and RNS 1.1

While this specification is similar to the RNS 1.0 specification in many ways, there are a few differences between the two specifications that are worth noting in detail. Specifically, four major differences include the elimination of the RNS junction concept, the replacement of a **move** operation with a **rename** operation, the removal of regular expressions in operations, and the inclusion of an iterator.

In the original specification, the notion of a junction was introduced to give RNS clients a resource representation of an RNS entry. This gave them a point against which operations could be applied to acquire metadata about that junction and perform some operations. However, the original specification was not able to seperate junctions from entries clearly and some confusion resulted whereby it was not clear whether or not various WS-Addressing EndpointReferenceTypes were junctions, or entries. Further, a small oversight in the original document made it difficult to acquire the endpoint reference for junctions in all cases. To address these issues, the RNS 1.1 specification completely removes the concept of a junction, leaving RNS implementations as simple mappings from names to values (EPRs). In the process, arbitrary metadata attached to, and included with the listing of, all RNS entries replaces the functionality that formerly came from junctions.

Another change that features in RNS 1.1 is the replacement of the **move** operation with the **rename** operation. This change comes about from the relative difficult in implementing an atomic move operation across address disjoint (and possibly implementation disjoint) RNS service providers. Because RNS is a *posix-like* implementation of a directory structure, clients will tend to perceive move as being atomic while that guarantee cannot in fact be enforced. However, with rename, the operation can easily be made atomic because the operation itself only permits renaming within a given RNS directory (hence, guaranteed to be a rename operation occurring within a single RNS service provider).

Unlike list in the original RNS specification **[RNS]**, this specification no longer allows for arbitrary regular expressions to be passed as parameters to lookup. This decision comes from experience trying to implement the original RNS specification. Since our goal here is to provide a familiar human-readable name mapping to web service endpoints, we must consider the typical human use cases. Users are used to looking things up by either exact name, or file pattern expression – not regular expression. As an implementor, this author ran into problems with the original specification having to do with passing regular expressions on the wire and determining when a user who had typed a string in to search for had meant to type a regular expression, a file pattern, or an exact name. Further, taking as an example POSIX directory IO, it is not common to search for a subset of entries within a directory by pattern except at the user level. When users request such an operation, the Operating System typically lists all the contents and simply picks out the entries that it wants to keep. We have found this same paradigm works equally well in the Grid case despite the potential performance degredation. Further, this particular scheme makes it much easier for clients to do client-side caching of results for future uses.

Finally, experience with RNS 1.0 has shown that the likely-hood of encountering RNS directories

containing tens of thousands, or even millions of entries is high enough that the old way of returning all entries in an RNS directory in a single SOAP message is untenable. To fix this potential problem, the list operation has been modified to permit service implementations to return either collections of entries, or iterators that traverse the entries, or both. With these iterators, service implementors can choose at run-time whether or not the entries are numerous enough to suggest iteration, thus avoiding the problem of unmanageably large result sets in SOAP messages.