GWD-R
OGSA Naming WG
Grid File System WG

Mark Morgan, University of Virginia
Andrew Grimshaw, University of Virginia
Osamu Tatebe, University of Tsukuba
December 30, 2009

RNS Specification 1.1

Status of This Document

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

Obsoletes

This document obsoletes GFD-101 **[RNS1.0]**.

Copyright Notice

Trademark

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.

Contents

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

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 **[RNS1.0]**. 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 that 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-UML 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 that will normatively describe the details as they pertain to their respective OGSA Basic Profile.

1.2 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” 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.

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/>).

[2] In order to give normative 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.

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/2009/12/rns

1.4 Entry Correspondence

Throughout this document we refer to RNS entries as "corresponding" with one another. Unless otherwise specified, that correspondence is defined to be exact string equality between the entry names of the two entries in question. No comparison between Endpoint References or metadata documents is implied.

1.5 Supports RNS Metadata

Most metadata associated with RNS entries is arbitrary and left to other specifications to profile. However, throughout this document we will refer to one well-known metadata element, the rns:supports-rns element. This XML metadata element is defined as follows and indicates (either from the client perspective, or from the RNS service perspective) whether or not the associated endpoint supports the RNS port type (is a directory).

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

The rns:supportType is an XML enumeration consisting of the values {true, false, unknown} indicating respectively that the endpoint does support RNS, does not support RNS, or it is unknown whether or not the endpoint supports RNS. A value of unknown is equivalent to not specifying the supports-rns element at all.

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.

The remainder of this section consists of pseudo-UML diagrams describing various types and interfaces that make up the RNS 1.1 specification. First, we describe types used in the RNS 1.1 port type operations and then we conclude with a description of the RNS 1.1 port type itself.

2.1 RNS Types

Five types are used in the RNS 1.1 port type's operations. These include a type that binds together a complete RNS entry (consisting of the entry's name, its endpoint address, and any

[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.

associated metadata) as well as a response type that allows for either an RNSEntry or a fault to be returned. The third is a convenience type for associating two names together (a mapping of an old entry name to a new entry name used during the rename operation). Fourth, we give the description of a Metadata binding type that associates an entry's name with its metadata. This type is used in the RNS 1.1 port type's setMetadata operation. Finally, we give a type that binds together a number of RNSEntry values with an iterator endpoint (the two items together forming the results for issuing a lookup command as described later).

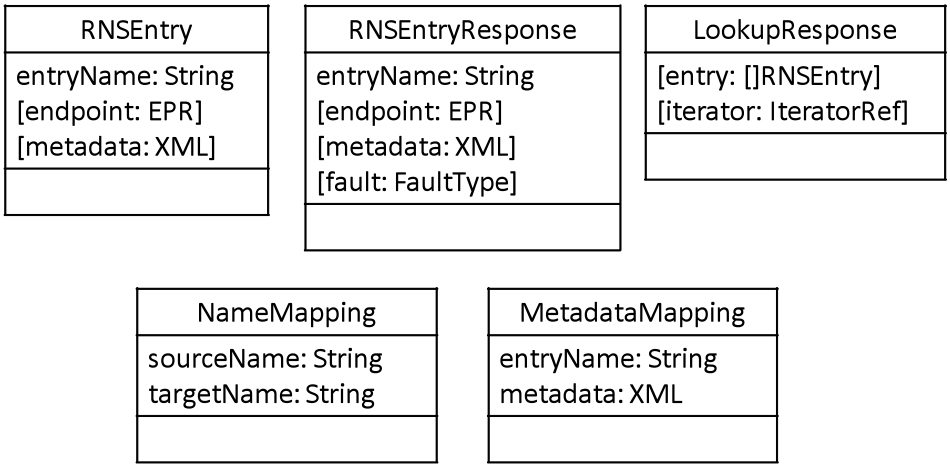

Figure 1 Pseudo-UML for RNS 1.1 Data Types

2.1.1 RNSEntry

The RNSEntry data type is used to describe the full set of values associated with a given entry inside of an RNS directory. Depending on use, various fields within this data type may be omitted as described in the RNS port type operation definitions.

2.1.2 RNSEntryResponse

The RNSEntryResponse data type combines all of the fields of an RNSEntry with a fault type. When implemented, the entryName field should always be set and either the endpoint and metadata fields should be set, or the fault field should be set. This type is used as a result type for batch Add, Rename, setMetadata, and Remove operations to allow for the caller to distinguish between faults for individual elements of the batch operation.

2.1.3 NameMapping

The NameMapping data type is used by the rename operation of the RNS port type to indicate a one-to-one mapping from the old name of an RNS entry to the new desired name. This mapping exists only to facilitate batch operations in the RNS port type.

2.1.4 MetadataMapping

The MetadataMapping data type is used by the setMetadata operation of the RNS port type to indicate a one-to-one mapping between an entry name and the metadata document that a caller wishes to set for that operation. This mapping exists only to facilitate batch operations in the RNS port type.

2.1.5 LookupResponse

The LookupResponse type is used as the result type from an RNS lookup operation. This type combines a number of 0 or more RNSEntry elements with an optional endpoint for an iterator service. These two fields can be combined in any way that that RNS service wishes (the implied meaning being that if any RNSEntry elements are returned, the constitute the first n such elements matching the lookup request and that if an iterator is returned it can be used to retrieve the remaining elements).

2.2 RNS Interface

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

RNS
elementCount: unsignedLong createTime: dateTime accessTime: dateTime modificationTime: dateTime readable: boolean writable: boolean
add(entry: []RNSEntry): []RNSEntryResponse lookup(entryName: []String): LookupResponse remove(entryName: []String): []RNSEntryResponse rename(entry: []NameMapping): []RNSEntryResponse setMetadata(entry: []MetadataMapping): []RNSEntryResponse

Figure 2 RNS Port Type Pseudo-UML

2.2.1 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.

2.2.2 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].

2.2.3 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].

[4] Unless otherwise stated, any reference to timestamps used within this document specifically refers 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.

2.2.4 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].

2.2.5 RNS readable Property

The readable property is a required property that 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.

2.2.6 RNS writable Property

The writable property is a required property that an RNS implementation MUST use to advertise whether or not a given RNS resource is writable (can respond to an add, rename, remove, or setMetadata 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 writable 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).

2.3 RNS add Operation

The add operation is used by clients who wish to add new entries to an RNS map. This operation assumes that the list of RNS entries given as parameters contain, for each element, an entry name that must be unique within that RNS directory and optional entry endpoint and metadata information. If a given entry is requested without an entry endpoint then 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, then an XML document describing desired metadata MAY also be given indicating initial metadata to set on the mapping. A client may include as a metadata element the well-known XML element rns:supports-rns as described above indicating explicitly whether or not the endpoint included is an RNS compliant resource or not (if this metadata value is absent, the RNS service MAY choose to attempt to ascertain whether or not the target endpoint is an RNS endpoint via other external mechanisms). 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 its own metadata to the metadata document. The RNS resource MUST respond to an add request message with an addResponse message.

2.3.1 RNS add

The format of the add Message is:

```
...
<rns:add>
        <rns:entry entry-name=”rns:EntryNameType”>
                <rns:endpoint>
                        wsa:EndpointReferenceType
                </rns:endpoint> ?
                <rns:metadata> {any} * </rns:metadata> ?
        </rns:entry> +
</rns:add>
...
```

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

/rns:entry
: This element describes exactly one entry to be added to the rns map. The cardinality of this element is [1, ∞).

/rns:entry@entry-name
: This required property gives the name for an 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/rns: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/rns: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 aside from the supports-rns entry if it exists. 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 RNS implementation MUST respond with an addResponse message (or a fault) containing the same number of RNSEntry elements as the request message contained. Further, the response message MUST indicate a one-to-one correspondence between requested entries and response entries. The response to the add message is a message of the form:

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

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

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

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

[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:entry-response/rns:endpoint
This element gives the WS-Addressing EndpointReferenceType for the entry indicated. If the fault element is included, this element MUST be omitted.

/rns:entry-response/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. If the fault element is included, this element MUST be omitted.

/rns:entry-response/rns:fault
This element is an optional element that, in the case that adding this entry failed, gives a fault document describing the failure. Because the format of this element depends on the profile specific rendering, we give no further details in this document. Rendering documents MUST define the format of this element normatively. If either the endpoint or the metadata elements is include, then this element MUST be omitted.

2.3.2 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/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/add
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
```

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

2.4 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, an RNS service resource MUST respond with a lookupResponse

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/addResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
```

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).

2.4.1 RNS lookup

The format of the lookup Message is:

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

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

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

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

/rns:entry-response/rns:endpoint
: This element gives the WS-Addressing EndpointReferenceType for the entry indicated. If the fault element is included, this element MUST be omitted.

/rns:entry-response/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. If the fault element is included, this element MUST be omitted.

/rns:entry-response/rns:fault

This element is an optional element that, in the case that obtaining this entry failed, gives a fault document describing the failure. Because the format of this element depends on the profile specific rendering, we give no further details in this document. Rendering documents MUST define the format of this element normatively. If either the endpoint or the metadata elements is include, then this element MUST be omitted.

/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.

2.4.2 Example SOAP Encoding of the lookup Message Exchange

The following is a non-normative example of a 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/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/lookup
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
```

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

2.5 RNS remove Operation

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

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/lookupResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
```

removeResponse message.

2.5.1 RNS remove

The format of the remove Message is:

```
...
<rns:remove>
        <rns:entry-name> rns:EntryNameType </rns:entry-name> +
</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 cardinality of this element is [1, ∞).

The removeResponse message contains a list of RNSEntryResponses (one each for each entry that was removed or that the client attempted to remove). The response message must contain entries in a one-to-one correspondence with the requested entry names. The response to the remove message is a message of the form:

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

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

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

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

/rns:entry-response/rns:endpoint
: This element gives the WS-Addressing EndpointReferenceType for the entry removed. If the fault element is included, then the endpoint element MUST be omitted.

/rns:entry-response/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. If the fault element is given, then the metadata element MUST be omitted.

/rns:entry-response/rns:fault
This element is an optional element that, in the case that removing this entry failed, gives a fault document describing the failure. Because the format of this element depends on the profile specific rendering, we give no further details in this document. Rendering documents MUST define the format of this element normatively. If either the endpoint or the metadata elements is include, then this element MUST be omitted.

2.5.2 Example SOAP Encoding of the remove Message Exchange

The following is a non-normative example of a 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/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/remove
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
```

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

2.6 RNS rename Operation

The rename operation is used by clients who want permanently rename a set of entries within an

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/removeResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
```

RNS resource. The rename operation can only be used by clients to change the name of existing entries within an RNS directory to different names within the same RNS directory. The RNS service MUST respond to a rename request message with a renameResponse message.

2.6.1 RNS rename

The format of the rename Message is:

```
...
<rns:rename>
        <rns:rename-request
                source-name="rns:EntryNameType"
                target-name="rns:EntryNameType"/> +
</rns:rename>
...
```

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

/rns:rename-request
: This element, of cardinaity [1, ∞), indicates, for each instance, a single rename operation to perform consisting of both a source and a target entry name.

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

/rns:rename-request@target-name
: This attribute gives the new name for an 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 message for a rename request contains a list of RNSEntryResponses indicating the new values for the renamed elements. As with remove, this response message must contain entries in a one-to-one correspondence with the requested entry names (this time the correspondence is based off of the new or target names of the entry rename operations). The response to the rename message is a message of the form:

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

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

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

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

/rns:entry-response/rns:endpoint
This element gives the WS-Addressing EndpointReferenceType for the entry renamed. If the fault element is given, then the endpoint element MUST be omitted.

/rns:entry-response/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. If the fault element is given then the metadata element MUST be omitted.

/rns:entry-response/rns:fault
This element is an optional element that, in the case that renaming this entry failed, gives a fault document describing the failure. Because the format of this element depends on the profile specific rendering, we give no further details in this document. Rendering documents MUST define the format of this element normatively. If either the endpoint or the metadata elements is include, then this element MUST be omitted.

2.6.2 Example SOAP Encoding of the rename Message Exchange

The following is a non-normative example of a 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/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/rename
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
```

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

2.7 RNS setMetadata Operation

The setMetadata operation is used by clients who want to replace the user-defined metadata for entries within an RNS service resource. This operation is only guaranteed to modify the user-

```
<s11:Envelope
        xmlns:s11="http://www.w3.org/2003/05/soap-envelope"
        xmlns:rns="http://schemas.ogf.org/rns/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/renameResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
```

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.

2.7.1 RNS setMetadata

The format of the setMetadata Message is:

```
...
<rns:setMetadata>
        <rns:set-metadata-request entry-name=”rns:EntryNameType”>
                {any} *
        </rns:set-metadata-request> +
</rns:setMetadata>
...
```

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

/rns:set-metadata-request
: This element gives a single setMetadata request operation. It contains an arbitrary set of XML documents representing the new metadata that the caller wishes to store with the given endpoint.

/rns:set-metadata-request@entry-name
: This attribute 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.

The setMetadata response message contains a set of RNS entries representing the newly formed RNS entries for the target RNS resource after the setMetadata request is processed. The response message MUST contain RNS entries in a one-to-one correspondence with the entries requested in the request message. The response to the setMetadata message is a message of the form:

```
...
<rns:setMetadataResponse>
        <rns:entry-response entry-name=” rns:EntryNameType “>
                <rns:endpoint> wsa:EndpointReferenceType </rns:endpoint> ?
                <rns:metadata>
                        <rns:supports-rns value=” rns:supportType “/>
                        {any} *
                </rns:metadata> ?
                <rns:fault> {fault} </rns:fault> ?
        </rns:entry-response> +
</rns:setMetadataResponse>
```

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

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

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

/rns:entry-response/rns:endpoint
This element gives the WS-Addressing EndpointReferenceType for the entry whose metadata was modified. If the fault element is given, then the endpoint element MUST be omitted.

/rns:entry-response/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. If the fault element is given, then the metadata element MUST be omitted.

/rns:entry-response/rns:fault
This element is an optional element that, in the case that setting the metadata to this entry failed, gives a fault document describing the failure. Because the format of this element depends on the profile specific rendering, we give no further details in this document. Rendering documents MUST define the format of this element normatively. If either the endpoint or the metadata elements is include, then this element MUST be omitted.

2.7.2 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/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/setMetadata
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://tempuri.org/rns-source
                </wsa:To>
```

The following is a non-normative example of a 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/2009/12/rns"
        xmlns:wsa="http://www.w3.org/2005/08/addressing">
        <s11:Header>
                <wsa:Action>
                        http://schemas.ogf.org/rns/2009/12/rns/setMetadataResponse
                </wsa:Action>
                <wsa:To s11:mustUnderstand="1">
                        http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous
                </wsa:To>
```

3. 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.

The introduction of group operations into the specification greatly enhances the efficiency that can be obtained when using RNS, but also presents a problem when it comes to faulting. The RNS specification defines all operations that take a list of entries MUST respond with a response message or a fault message containing the same number of entries as request message contains. Further, the response message MUST indicate a one-to-one correspondence between requested entries and response entries.

3.1 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 indicate 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 Permitted 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 within 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.

3.2 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). Exceptions listed in *Italics* are those that are returned inside of the RNSEntryResponse elements rather than those that are “thrown” by the service.

3.2.1 add

- Write Not Permitted Failure
- *RNS Entry Exists Failure*

3.2.2 lookup

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

3.2.3 rename

- Write Not Permitted Failure

- *RNS Entry Exists Failure*
- *RNS Entry Does not Exist failure*

3.2.4 remove

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

3.2.5 setMetadata

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

4. 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.

5. Author Information

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

Andrew Grimshaw
University of Virginia, Department of Computer Science
151 Engineer's Way
P.O. Box 400740
Charlottesville, VA. 22904-4740
E-mail: grimshaw@virginia.edu

Osamu Tatebe
University of Tsukuba, Department of Computer Science
1-1-1 Tennodai
Tsukuba, Ibaraki 3058573 Japan
E-mail: tatebe@cs.tsukuba.ac.jp

6. Contributors and Acknowledgements

We greatly acknowledge the contributions made to this document by Hideo Matsuda, Yoshiyuki Kido, Takuya Ishibashi, Yoshiyuki Watase, Go Iwai, Masahiro Nakamura, Fumikazu Konishi, Yusuke Tanimura, Erwin Laure, David Martin and all people who provided constructive and valuable input in the group discussion.

7. Glossary

Entry Correspondence Correspondence between two entries in an RNS operation request and response message pair defined by exact equality of the entry names.

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.

8. Intellectual Property Statement

The OGF 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 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 Director.

9. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF 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.

10. Full Copyright Notice

11. 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), 2005.

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

[XML-Infoset] J. Cowan, Richard Tobin, XML Information Set (Second Edition), http://www.w3.org/TR/xml-infoset/, 2004

[XPATH] J. Clark, S. DeRose, XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath, 1999

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

[WS-Enumeration] J. Alexander, D. Box, L. F. Cabrera, D. Chappell, G. Daniels, C. Kaler, D. Orchard, I. Sedukhin, M. Simek, M. Theimer, Web Services Enumeration (WS-Enumeration), http://www.w3.org/Submission/WS-Enumeration/, 2006

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

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

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

[BES] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, M. Theimer, OGSA® Basic Execution Service Version 1.0, http://www.ogf.org/documents/GFD.108.pdf, GFD.108, 2007

[WS-Naming] A. Grimshaw, D. Snelling, WS-Naming Specification, http://www.ogf.org/documents/GFD.109.pdf, GFD.109, 2007

[RNS1.0] M. Pereira, O. Tatebe, L. Luan, T. Anderson, Resource Namespace Service Specification, http://www.ogf.org/documents/GFD.101.pdf, GFD.101, 2007

[SOAP1.1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/soap11, 2000

Appendix A. Difference 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 separate 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 **[RNS1.0]**, 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 implementer, 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 degradation. Further, this particular scheme makes it much easier for clients to do client-side caching of results for future uses.

Many of the operations in the RNS 1.1 specification have been modified to support group or batch executions to improve potential performance. In these cases, a one-to-one correspondence is generally assumed between elements of the request message and elements of the response message. The only RNS operation for which this is not true is the lookup operation which already was defined with a batch mechanism.

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 implementers 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.