



NextGRID Naming & Addressing Use Cases and Requirements

| | | |
|-----------------|----------------|-------|
| Editors: | Stephen Davey | UEDIN |
| | David Snelling | FLE |
| | | |

| Date | Author | Comments | Version | Status |
|--------------|----------------|-----------------|---------|--------|
| Aug 24, 2006 | Stephen Davey | Initial Version | 0.1 | Draft |
| Dec 4, 2006 | Stephen Davey | First draft | 0.2 | Draft |
| Feb 26, 2008 | David Snelling | Final QA | 1.0 | Final |



| | |
|---|-----------|
| 1 INTRODUCTION..... | 3 |
| 1.1 Naming Profile Overview | 3 |
| 1.2 Naming & Addressing Background | 4 |
| 1.3 Relevant Specifications..... | 5 |
| 1.4 Terminology | 5 |
| 1.5 Relationship between Names and Addresses | 6 |
| 2 NEXTGRID USE CASES AND COMPONENTS..... | 7 |
| 3 GENERAL NAMING & ADDRESSING REQUIREMENTS | 10 |
| 4 REFERENCES..... | 12 |



1 Introduction

This document presents the NextGRID Naming & Addressing use cases and requirements that accompany the NextGRID Naming Profile 1.0 and the Addressing parts of the NextGRID Basic Profile 1.0.

The NextGRID Generalized Specifications aim at capturing NextGRID architectural concepts in a set of composable profiles. These profiles are specified in such a way that they could be implemented in terms of other well known specifications. While overall consistency is achieved at the conceptual level, and captured through the motivating use cases accompanying each specification, the implementation in terms of other specifications may not be consistent between different profiles. Thus each profile defines an implementable realisation of the underlying concept, but implementers of the full NextGRID architecture may need to support multiple competing underlying specifications.

1.1 Naming Profile Overview

The NextGRID Naming Profile is intended for use when implementing naming services that are in line with the concepts of NextGRID [1]. It mandates the adherence to a certain set of specifications and clarifies their use. A service implementation that uses those specifications in a manner conformant with the Profile may be said to be an “implementation of the NextGRID Naming Profile 1.0” or, informally, to be a “NextGRID Naming Service.”

The primary issues addressed in the NextGRID Naming Profile and the Addressing parts of the NextGRID Basic Profile [2] are as follows:

- Addressing & Endpoint References
- Use of End Point Identifiers in addresses
- Use of Resolvers in addresses
- Resolution of an End Point Identifier
- Resolution of a Renewable Reference
- Creation of End Point Identifiers
- Registering End Point Identifiers
- Registering Human-Oriented Names
- Management of Naming Authorities



1.2 Naming & Addressing Background

Within the NextGRID Architecture a large number of constructs, such as services, resources, databases, results, etc. require uniquely identifying entities; naming provides that unique identifying function. Some examples of the purposes that a name may have are for:

- Logging & auditing
- Tracking
- Storing
- Identifying (uniquely) - by clients and by services
- Persistence
- Searching
- Cataloguing
- Provenance

From an architectural point of view, the ability to attach names, where possible both human readable and globally unique, to data resources is of key importance. It enhances the readability of grid applications and commands, provides flexibility of use and configuration of applications, and overall enhances the user experience.

The OGSA work on naming recognises three levels of name: human-oriented, abstract names, and addresses. From the OGF OGSA glossary [3]:

“Name” – is an attribute used to identify an entity. In OGSA naming, there are three types of names: human-oriented names, abstract names, and addresses.

“Human-oriented name” - is based on a naming scheme that is designed to be easily interpreted by humans (e.g. human-readable and human-parsable)¹. These are also sometimes referred to as “Contextualised Names”.

“Abstract name” - is a persistent name suitable for machine processing that does not necessarily contain location information. Abstract names are bound to addresses.

“Address” - specifies the location of an entity. This is typically defined as an End Point Reference (EPR), which is more than just a URL.

And additionally,

“Resolution” – Name resolution is the mapping of human-oriented names to abstract names, which are then mapped to some form of address.

¹ In fact, nearly all abstract names are easily read and interpreted by humans, but they are not always very user-friendly. So perhaps a better definition of a human-oriented name is one that is short, user-friendly and easy to generate by a human (rather than a hexadecimal string generated by a machine).



A human-oriented name, such as path name, is mapped to an abstract name. An abstract name is then dynamically mapped to an address. It is this address that allows messages and operations to be directed at the named entity.

Note that in the latest WS-Naming specification the word “abstract name” is no longer used and is more commonly replaced by “**Endpoint Identifier**”.

1.3 Relevant Specifications

WS-Addressing

The lowest level of naming is the notion of an endpoint name. The NextGRID Architecture uses a WS-Addressing endpoint reference (EPR) to refer to a specific grid endpoint/resource. However, endpoints can be highly dynamic in time and space and a naming system based solely on this element would be very difficult. Therefore, addresses should also include some sort of (ideally) global identifier and run-time naming resolution. WS-Naming provides such a mechanism for rebinding EPRs, as well as a mechanism for including abstract names within EPRs [4].

WS-Naming

WS-Naming is a profile on top of the WS-Addressing specification, where additional optional elements EndpointIdentifier, EndpointIdentifierResolver and ReferenceResolver may be included in the WS-Addressing Endpoint Reference. Note that a WS-Name may specify a resolving service or their syntax may imply a resolving service. This name resolution service provides the mapping between a stale EPR or an abstract name and a newer EPR (or EPRs) [4].

Resource Namespace Service (RNS)

At the topmost level are “human-oriented names” and they are typically the primary interface for users and applications. These names are not guaranteed to be either unique or static. Mappings between the human names and EPRs are maintained and accessed by the RNS services. These EPRs may be WS-Names if they include abstract names (i.e. they conform to the WS-Naming specification), or they may simply be addresses. RNS addresses the need to access resources within a grid environment by way of a universal name that ultimately resolves to a meaningful address. It is also intended to facilitate namespace services for a wide variety of Grid applications [5].

1.4 Terminology

EPR - A Web Service Endpoint Reference as defined in [6]. In Web Services Resource Framework (WSRF) [7] an EPR conveys the information needed to identify or reference a stateful resource.

EPI - A Web Service Endpoint Identifier, which is an IRI the resource. An Endpoint Identifier is a name that uniquely identifies the resource.

EPR-Minter - The EPR-minter is the entity, typically part of the server-side runtime environment, responsible for creating an EPR to refer to the web service endpoint for a resource. The EPR-minter has enough knowledge of the server-side context to ensure that messages based on this EPR arrive at the intended resource.

EPN - A Web Service Endpoint Name, which is an implementation of an EPI.

IRI - Internationalized Resource Identifier, RFC 3987 [8].

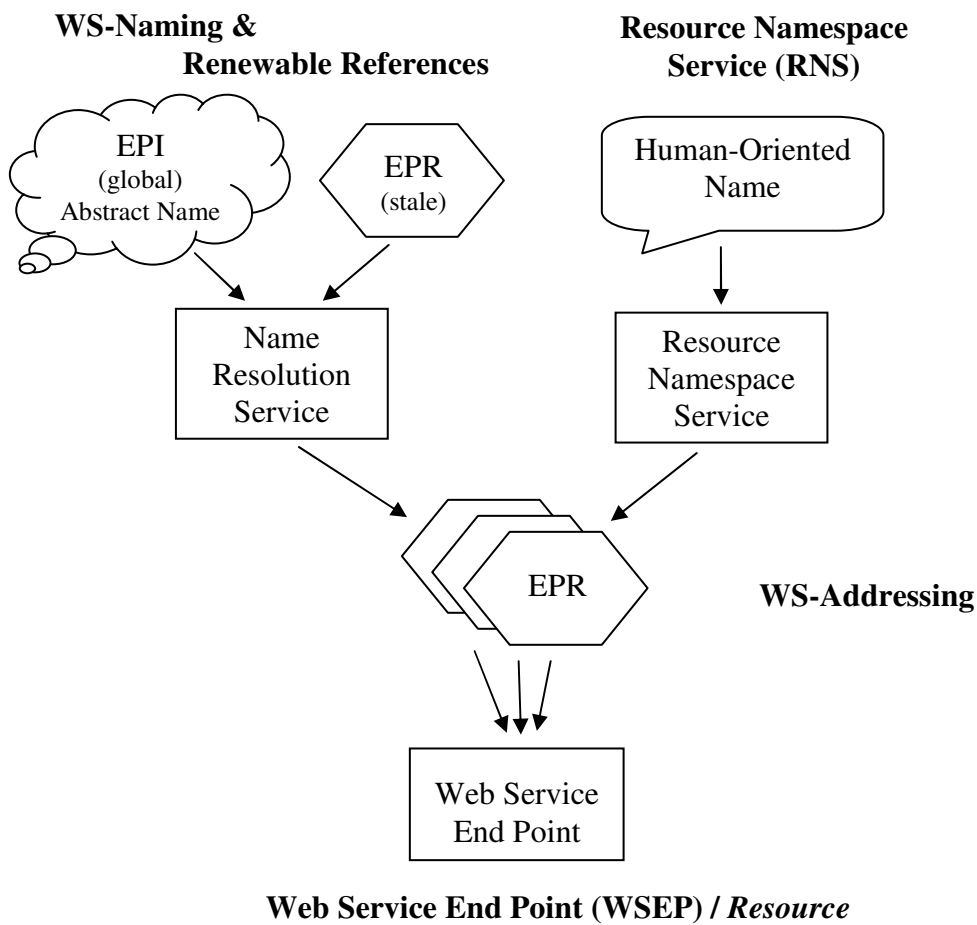
RenewableReference - An EPR to a stateful WSEP that holds the EPR for another endpoint.

URI - Universal Resource Identifier RFC 2396 [9].

WSEP - A Web Service Endpoint; sometimes referred to simply as a resource.

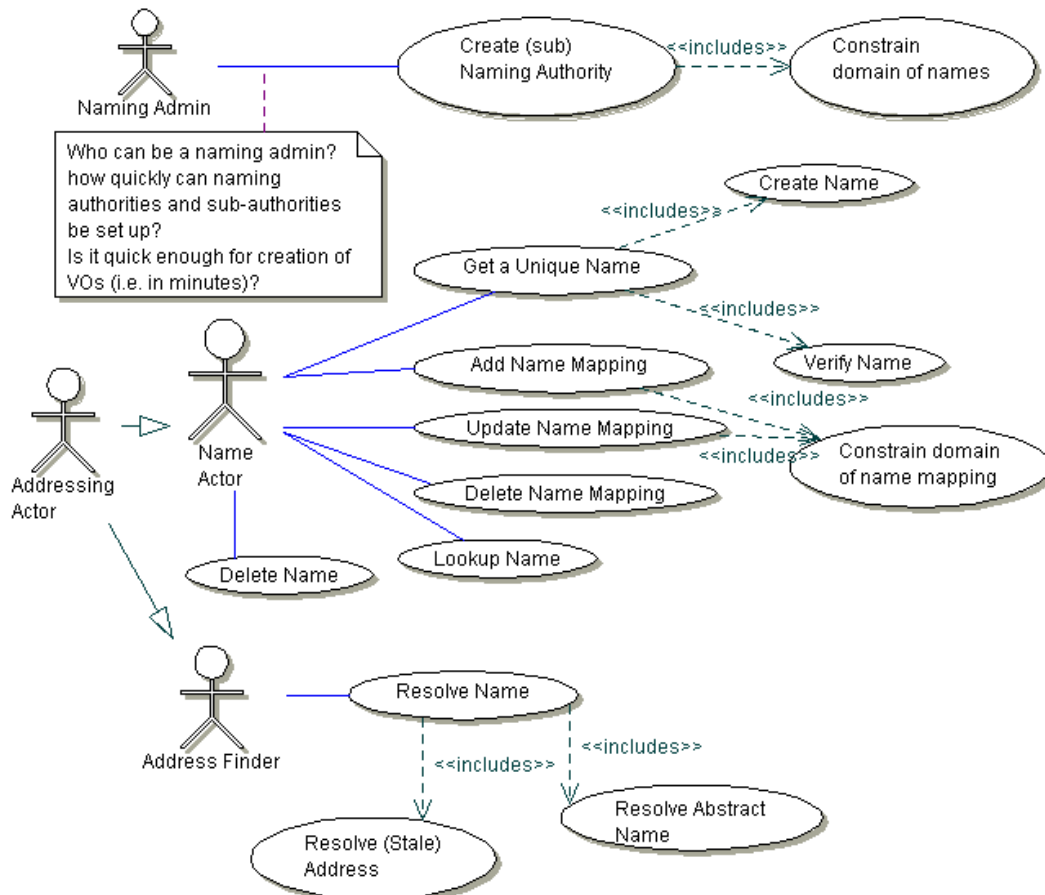
1.5 Relationship between Names and Addresses

The following diagram shows the relationship between Names, Addresses and Web Service Endpoints.



2 NextGRID Use Cases and Components

Naming - Use case Diagram



There are 3 Naming use cases that are being considered:

- Creating a Naming Authority.
- Getting a unique name and storing it.
- Resolving a name.

For the “Creating a Naming Authority” use case there is just one step:

1. Create the Naming Authority (or sub-Authority). This step will also include checking if the actor is allowed to perform this action, and also whether the requested Naming Authority follows the correct syntax etc.



This use case comes from the use of the Handle.net software [10] and syntax in which every handle consists of two parts: its naming authority, otherwise known as its prefix, and a unique local name under the naming authority, otherwise known as its suffix:

`<Handle> ::= <Handle Naming Authority> "/" <Handle Local Name>`

Naming authorities under the Handle System are defined in a hierarchical fashion resembling a tree structure. The parent namespace and its child namespaces may be served by different handle services, and they may or may not share any administration privileges.

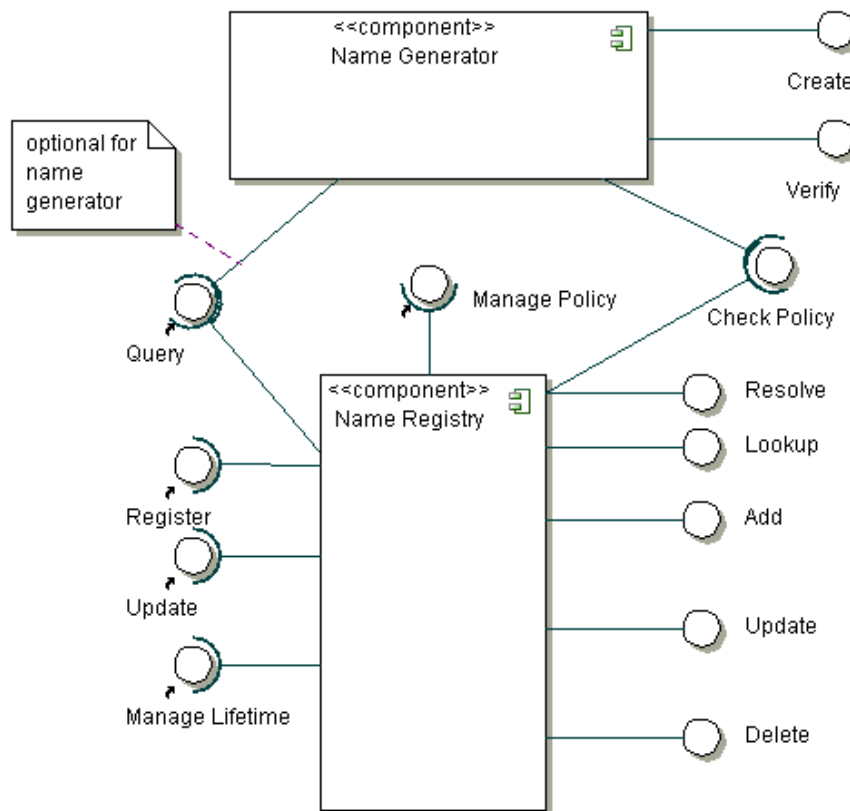
For the “Getting a unique name and storing it” use case there are several steps:

1. Firstly the actor needs to get a unique name. This step has 2 sub-steps:
 - a) If they have not provided a name themselves then a name needs to be created.
 - b) In both cases this new name needs to be verified as being unique (either globally if that is possible/required, or within the naming service or just the namespace).
2. This new name and a mapping to one or more values (such as addresses) can be added. This step will also include checking if the actor is allowed to perform this action, and also whether the requested mapping meets the constraints that have been imposed.
3. The actor may want to update one or more of the name mappings. Again the constraints will need to be checked.
4. The actor may want to delete one or more of the name mappings for a given name.
5. The actor can lookup a name and obtain the current name mappings (if they are allowed).
6. The actor can delete a name and all of the associated name mappings (if they are allowed).

For the “Resolving a name” use case there are 2 possible steps:

1. The address finder may want to resolve a name (normally referred to as an abstract name) and obtain the associated address or addresses.
2. The address finder may want to resolve a stale address. He would need to provide the name and the known stale addresses and would be returned zero or more additional addresses which may or may not also be stale (invalid).

Naming – Component Diagram



Created by Borland® Together® Designer Community Edition

From the 3 Naming use cases that are detailed above, 2 naming components have been identified:

1. Name Generator component.
2. Name Registry component.

From the “Creating a Naming Authority” use case and the getting a unique name step of the second use case, the Name Generator component will need to have the following interfaces:

- Create
- Verify

For the checking of the domain of names constraints, the following interface is also required:

- Check Policy, from the NextGRID Security profile [11].

In addition the following interface is optional:

- Query, from the NextGRID Registry profile [12].



From the “Getting a unique name and storing it” use case and the “Resolving a name” use case, the Name Registry component will need to have the following interfaces:

- Add
- Delete
- Update
- Lookup
- Resolve

These first 4 interfaces map onto the common Registry interfaces, see [12]:

- Register
- Manage Lifetime
- Update
- Query

The fifth interface, “Resolve”, is defined by the WS-Naming specification [4].

3 General Naming & Addressing Requirements

There are a number of requirements on a naming scheme. For example, it should be autonomous, scalable, distributed, secure, reliable, trusted, and have global scope. In addition it is desirable that the naming scheme (and name resolution service) should be fast, efficient, extensible and adaptable to international language standards. It should also be remembered that there will be the requirement to name data that is being generated on the fly, as well as data that has already been materialised and stored. A naming scheme that is not practical to use and does not include these properties is less likely to gain widespread use.

The OGF WS-Naming Working Group document “Web Service Endpoint Identification and Resolution: Use Cases and Requirements Version 1.0” [13] also describes a number of use cases:

Resource Mobility. Resources may need to be relocated and this can typically also require a change in the physical network address and/or access protocol.

Assertion Target. In policy languages a stable resource identifier is frequently needed as the target of a policy expression, either as an identifier for the subject or for the object.

Resource Attributes. When enforced policy-rule expressions rely on resource attributes parties expect that their resource representations resolve to the same attribute bindings.

Resource Reference Consistency. Clients often use the same resources many times and over an extended period. Any ambiguity about the resource to which a given reference will refer to in the future will result in increased administration costs.

Resource Metadata Caching. A client (or, more generally, a resource consumer) may cache the location of (or other metadata about) one or more resources. A persistent name that can be compared with other names on the client greatly facilitates this process.



Audit Label. Resource identifiers are often required for audit entries in various clients, services, and intermediates. In many cases, the name of a resource may need to persist much longer than the resource itself.

The OGF WS-Naming Working Group document then derives requirements from these use cases:

- It is important that any solution is consistent with current tooling and does not require incompatible changes to existing tooling.
- In order to support the metadata caching use case, it is necessary that a client be able to compare the EPRs of two resources and test them for equivalence.
- Messages to an identified (named) resource must always either be delivered to only that resource or fail to be delivered to any resource. Equivalently, each message must contain sufficient information to uniquely identify its destination within any context in which it is processed.
 - The profile should define desirable uniqueness properties associated with an EPR (see “*Unambiguous Web Service Endpoint Profile*” [4]).
 - The profile should optionally define EPR creation that provides the desirable uniqueness properties associated with the `wsa:Address` element of the EPR (see “*Web Service Endpoint Address Identifier Profile*” [4]).
- In order to support the “audit and the policy enforcement at intermediaries” use cases, it must be possible for a resource to ensure that all messages, subject to audit, carry a resource identifier that can be used to link the message to the resource.
 - The profile should define an *endpoint identifier* that can be embedded in an EPR to provide an alternative, more stable name for that endpoint (see “*Web Service Endpoint Name Specification*” [4]).
- Given an EPR of a mobile resource, the client must be able to find a valid EPR to that resource based on information available in the former EPR.
 - The profile should specify *resolution services* that allow for resolution of the most current endpoint’s EPR in case the old EPR fails, or if only an endpoint identifier is available (see “*Endpoint Reference Resolution Specification*” [4]).



4 References

- [1] D. Snelling, M. Fisher, A. Basermann, F. Wray, P. Wieder and M. SurrIDGE, *NextGRID Vision and Architecture White Paper V5*, v18, NextGRID Project, 17 May 2007, http://www.nextgrid.org/download/publications/NextGRID_Architecture_White_Paper.pdf
- [2] V. Li and D. Snelling (Eds.), *NextGRID Basic Profile*, v1.0, January 2008, http://www.nextgrid.org/GS/management_systems/basic/NextGRID_basic_profile.pdf
- [3] J. Treadwell, *Open Grid Services Architecture: Glossary of Terms*, v1.6, GFD.106, Open Grid Forum, 25 January 2005, <http://www.ogf.org/documents/GFD.120.pdf>
- [4] A. Grimshaw and D. Snelling, *WS-Naming Specification*, GFD.109, July 3, 2007, <http://www.ogf.org/documents/GFD.109.pdf>
- [5] M. Pereira, O. Tatebe, L. Luan and T. Anderson, *Resource Namespace Service Specification*, OGF.101, May 2007, <http://www.ogf.org/documents/GFD.101.pdf>
- [6] D. Box and F. Curbera (Ed.), *Web Services Addressing 1.0 – Core (WS-Addressing)*, W3C Last Call, 31 March 2005, <http://www.w3.org/TR/2005/WD-ws-addr-core-20050331>
- [7] I. Foster, T. Maguire and D. Snelling, *OGSA WSRF Basic Profile 1.0*. GFD.72, GF 2006, <http://www.ogf.org/documents/GFD.72.pdf>
- [8] M. Duerst and M. Suignard, *RFC 3987 – Internationalized Resource Identifiers (IRIs)*, Internet Engineering Task Force, January 2005, <http://www.ietf.org/rfc/rfc3987>
- [9] T. Berners-Lee, R. Fielding and L. Masinter, *RFC 2396 – Uniform Resource Identifiers (URI): Generic Syntax*, Internet Engineering Task Force, August 1998, <http://www.ietf.org/rfc/rfc2396>
- [10] S. Sun, L. Lannom, B. Boesch, *RFC 3650 – Handle System Overview*, Network Working Group, The Corporation for National Research Initiatives, November 2003, <ftp://ftp.rfc-editor.org/in-notes/rfc3650.txt>
- [11] J. Claessens and M. SurrIDGE (Eds.), *NextGRID Security Profile*, v1.0, February 2008, http://www.nextgrid.org/GS/management_systems/security/NextGRID_security_profile.pdf
- [12] P. Hasselmeyer, M. SurrIDGE, and P. Wieder, *NextGRID Registry Profile*, v1.0, January 2008, http://www.nextgrid.org/GS/management_systems/registrt/NextGRID_registry_profile.pdf
- [13] F. Siebenlist and D. Snelling, *Web Service Endpoint Identification and Resolution: Use Cases and Requirements Version 1.0*, OGSA Working Group, Global Grid Forum, 21 April 2006, <https://forge.gridforum.org/sf/docman/do/downloadDocument/projects.ogsa-naming-wg/docman.root/doc6867>