



NextGRID Registry Use Cases

| | | |
|-----------------|------------------|---------------|
| Editors: | Peer Hasselmeyer | NEC |
| | Mike Surrige | IT Innovation |
| | Philipp Wieder | FZJ |

| Date | Author | Comments | Version | Status |
|------------|-------------|-------------------|---------|---------|
| 15/08/2006 | ph | first draft | 0.1 | draft |
| 09/03/2007 | ph | first release | 0.5 | release |
| 17/05/2007 | V. Li | Final | 0.6 | Draft |
| 09/01/2008 | ph | minor corrections | 0.7 | Draft |
| 05/02/2008 | D. Snelling | Final QA | 1.0 | Final |



- 1 INTRODUCTION..... 3**
- 1.1 Registration Data 3
- 1.2 Registry Communication 3

- 2 USE CASES..... 3**
- 2.1 Register 4
- 2.2 Query 4
- 2.3 Remove 5
- 2.4 Update..... 5

1 Introduction

This document describes use cases of the NextGRID service registry (hereafter, “the registry”). Although there are different types of registries, this document is currently specifically aimed at registries storing service information, hence “service registry”.

A service registry is a component that facilitates the discovery of services in a distributed system. It acts as a mediator between service publisher and consumer. It collects advertisements, namely “service registrations”, from service publishers and passes them on to clients on request. The benefits of a service registry are twofold. On one hand, it relieves publishers the burden of announcing service availability to each and every potential client. On the other hand, it relieves service consumers the burden from manually configuring software agents with the endpoints of services.

The use cases introduced in this document describe interaction with the registry. Before introducing the actual use cases, some assumptions and observations that apply to all use cases are discussed in the following sections.

1.1 Registration Data

A service registration consists of a service endpoint and a service description. The format of the endpoint depends on the technology used to communicate with the service. An example is a Uniform Resource Locator (URL). Descriptions can contain arbitrary data that often depends – at least partly – on the domain of the service’s functionality.

The endpoint of the service must not necessarily be the final access point of the service. It can just as well present a factory that creates the actual service on demand. It can also be a proxy that forwards requests to the actual service. The actual service might not yet exist as service access might be negotiated for a future point in time. In that case, the binding between the proxy and the service might be established later on.

The description of a service can contain arbitrary data. The actual description used depends in most cases on the purpose of the service and the environment it is used. A registry should be able to cope with any kind of data here and not put unnecessary restrictions on it. In particular, the combination of data with different schemas and from different sources should be possible.

1.2 Registry Communication

All use cases require some communication of an actor with the registry. Every actor needs to know the protocol spoken by the registry and its endpoint. In all use cases, the actors are assumed to know how to communicate with the registry. This usually requires some a priori agreements and possibly some deployment and/or run-time configuration.

2 Use Cases

The basic use cases for a registry – register and query – were already mentioned in the previous chapter. Some more use cases usually apply to a registry. An overview of the registry use cases is given in Figure 1. The individual use cases are described in more detail in the following sections.

NextGRID Registry Use Cases

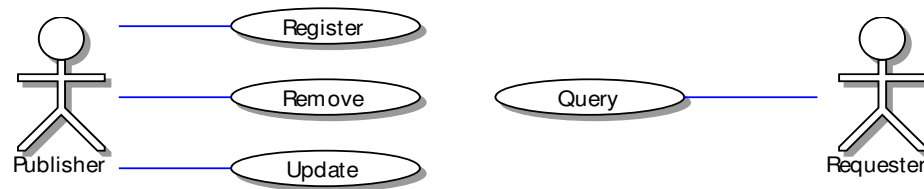


Figure 1: Registry Use Cases

The use cases involve the registry and one additional actor that triggers the respective functions. There are two actors in the use cases, the service publisher and the service requester. Both kinds of actors are specific roles in the registry use cases. The roles are often assumed by other actors in more general service provisioning scenarios.

The service publisher can be a service provider. But this is not mandated. Services can just as well be published, updated, or removed on behalf of the provider by a third party. For example, an intermediary could copy (parts of) a registry’s content to another registry. To the second registry, the intermediary acts in the role of a service publisher.

The service requester can be a service consumer. But this is not mandated. Services can just as well be requested on behalf of the consumer by a third party, e.g. a local service cache.

2.1 Register

A service publisher wants to make a service known to potential consumers. One way of doing so is to contact a service registry, store (“register”) information about the service there, and have the registry distribute that information to clients.

To perform the registration successfully, the service publisher must provide information about the service to the registry. The information contains the endpoint of the service and a further description. When a registry receives a registration request, it takes the associated data and stores it in its database of known (“registered”) services.

Service registries are autonomous entities. They have no inherent obligation of storing somebody’s registration. If, for some reason (e.g. access restrictions or storage problems), a registry rejects a service registration, it must return an error message to the publisher. In that case, the registration database will not contain the service that was being registered.

2.2 Query

A service requester is looking for a particular service. It contacts a service registry, sends a query for that service to the registry, and receives a list of matching service registrations.

To perform the query successfully, the service requester must communicate a suitable query to the registry. To formulate such a query, the requester needs to know what query languages the registry understands. The possible query languages can either be established a priori or determined at run-time. In addition, the requester must be able to understand the results returned by the registry. The successful result should consist all the services in the registration database that match the query. Each returned matching service should contain the service registration data, including the service’s endpoint and the service’s description. If there are no matching services in the registration database, an empty result is returned.

If, for some reason (e.g. access restrictions), a registry rejects a service query, an error message must be returned to the requester.

2.3 Remove

If, for example, a publisher doesn't want one or more service registrations to be publicly available, the publisher can remove the service's registration from the registry.

To successfully achieve this, the publisher must unambiguously identify the entry to be removed. The registry then removes the entry from its database.

In case the identified registration does not exist in the registry's database, the registry does nothing. If, for some reason (e.g. access restrictions), the registry is unable or unwilling to remove the identified entry, it must inform the publisher by issuing an appropriate error message.

2.4 Update

In some cases, the information of a service may have changed; this requires a service publisher to update the existing service registration with the updated service information.

To update an existing registration, the publisher must unambiguously identify the entry to be updated and specify the new data that is to be stored, including the service's endpoint and the service's description. The registry then updates the registration in its database accordingly.

If the identified registration does not exist in the registry's database, the registry must inform the publisher by issuing an appropriate error message. If for some reason (e.g. access restrictions), the registry is unable or unwilling to update the identified registration, it must inform the publisher by issuing an appropriate error message.