NORDUGRID

NORDUGRID-TECH-21

12/5/2009

ARC Information System

Documentation and developer's guide

*

*@

Contents

Chapter 1

Design Overview

Chapter 2

ISIS

2.1 Registration handling

2.1.1 Functionality

The functionality of ISIS is made of two parts. On the one hand they are working as ordinary Web Services, and on the other hand maintain a peer-to-peer network - `ISIS cloud`.

Main functionality of ISIS service visible from outside of ISIS cloud is to accept registration and provide collected information to clients. For that ISIS implements operations described in following section. The single ISIS service accepts Registration Records pushed to it by other services (including ISIS services too) and stores them in local XML database. Stored records can be queried by clients using mandatory and service-specific attributes for selection criteria.

In case of multiple ISIS services they form a peer-to-peer network. Inside that cloud Registration Records are propagated between services in such a way that all ISIS instances continiously try to synchronise their databases.

2.1.2 Interface

Operation Register

Input

- **Header**
  - **RequesterID** Identifier of the client.
  - **MessageGenerationTime** Time when following set of RegEntry was generated. There may be multiple RegEntry elements.
- **RegEntry**
  - **SrcAdv**
    - **Type** Type of service being registered. This element is opaque string for now. There shall be service types defined later.
    - **EPR** Endpoint Reference of service being registered in terms of WS-Addressing.
    - **SSPair** Set of key/value pairs representing service specific information.
  - **MetaSrcAdv**
    - **ServiceID** Globally unique and persistent identifier of the service.
    - **GenTime** (Generation Time) The actual timestamp of information (called wake_up_time in the pseudo code below)
    - **Expiration** Validity period of Service Advertisement record.

Output

Fault Optional element describing fault which occured while performing registration. If missing registration succeeded.

Faults

none No specific faults are defined

This operation is usually called by service which wants to register it's presence in ISIS. This message consists of one or more **Registration Entry** and at most one **Registration Header**. The **Registration Entry**(RegEntry) contains a **Service Advertisement**(SrcAdv) and a corresponding **Service Advertisement Metadata**(MetaSrcAdv). This structure is shown on Figure 2.1.

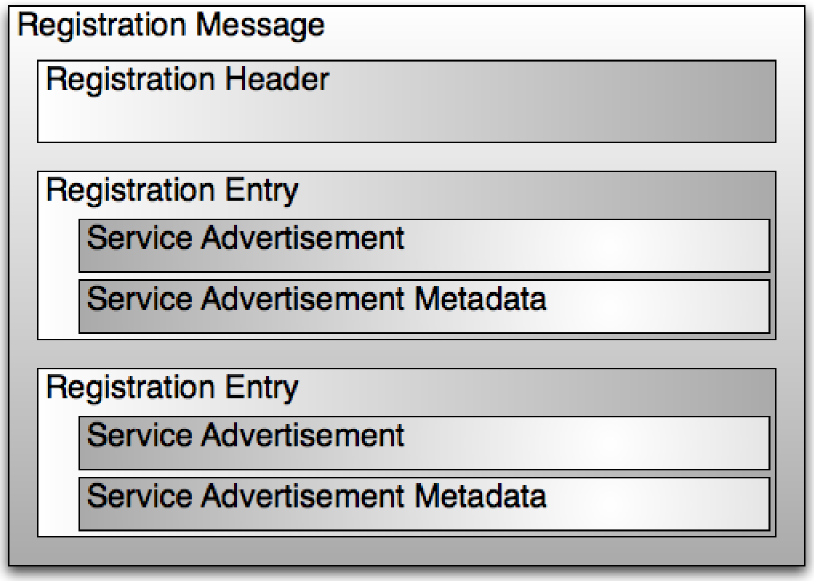

Figure 2.1: Embedded structure of Registration Message

Service must supply mandatory information. Among those the `Endpoint Reference` is used to contact service - only required element is contact URL of service. `Type` specifies kind of service and is used to find out functionality nad interface of service. `ServiceID` is used to distinguish between registered services and to deal with case of service changing it's contact URL. For more information about mandatory and optional information see Section 3.1

As result of this operation new Registration Record is stored inside ISIS internal database and eventually propagated to other ISISes. If registration record of same ID already existed it will be renewed.

This operation is also used by ISIS services to propagate Registration Records inside ISIS cloud.

Operation RemoveRegistrations

Input

MessageGenerationtime

ServiceID Multiple identifiers of services, whose records has to be removed.

Output

RemoveRegistrationResponseElement

- **ServiceID** Identifier of service whose record was not removed
- **Fault** Description of failure reason

Faults

none No specific faults are defined

This operation is used to explicitely requesting the removal of zero or more Registration Entries associated with specified ServiceID values stored in the Information System. If corresponding record does not exist it's identifier will be present in response message together with corresponding Fault element.

Operation GetISISList

Input

none

Output

EPR Multiple Endpoint References of known ISIS services

Faults

none No specific faults are defined

In response to this operation the EndpointReferences to all known ISIS services are returned. The operation is used for obtaining a list of known ISIS instances from any particular ISIS. Clients can then use the obtained list to run direct queries against the ISIS instances. The operation is provided for fault tolerance and for providing optional performance boost. This operation returns the known peer-to-peer neighbors. If client is interested in all ISIS instances they can be obtained using the Query operation because they are ordinary services registered into the Information System.

Operation Query

Input

QueryString XPath query expression

Output

any Result of query

Faults

none No specific faults are defined

This operation allows any XPath queries to be performed on stored Registration Records. The records are treated as merged in one XML document with each record being equivalent to `RegEntry` element of `Register` operation. In response all elements produced by XPath query are returned. The purpose of this operation is to make it possible to obtain any kind of information related to the Indexing Database.

2.2 Peer-to-Peer

2.2.1 Functionality

2.2.2 Interface

2.3 Authorization

2.3.1 Client Authorzation

To ensure information stored in ISIS cloud can't be tampered and only available to proper clients following authorization framework is implemented. All actions perfored by ISIS clients are divided into three following groups:

- Operations initiated by other ISISes in the cloud. Those include:
  - Register with Registration Message containing information not about contacting client
  - RemoveRegistrations with request to remove Registration Message representing not contacting client
  - Connect

  Those operations may cause uncontrollable changes in colleccted information and must be granted only for higly trusted entities like ISISes themselves.
- Operations initiated by the Services registering to Information System. Those are:
  - Register with Registration Message containing information about contacting client
  - RemoveRegistrations with request to remove Registration Message representing contacting client
- Operations which are allowed for any liable client of particular Grid infrastructure.
  - Query
  - GetISISList

2.3.2 Information Authorzation

2. From storage/access location point of view information is divided

Chapter 3

Service

3.1 Information generation

The service developers have to ensure that the services are providing the necessary information about themselves. This is done by implementing the subclass of the Arc::Service class - the RegistrationCollector function has to provide up-to-date status information about the service and anything else it wants to be advertised. This information package is called **Service Advertisement**. The **Service Advertisement** can contain any information service wants to advertise but the mandatory elements have to be always present:

- Service ID: A globally unique identifier of the service.
- Service Type: The Glue2 type of service.
- Endpoint URL: The URL where the service can be contacted provided as part of EPR element.

Because there may be multiple registration processes running in parallel it is important to ensure that implementation RegistrationCollector is thread safe or there are internal locks implemented.

3.2 Registration

The registration of service is carried out by internal module called Registrant. The Registrant is active module of the HED (Hosting Environment Daemon) which is bound to a set of ISISes. In practice, the configuration part of the Registrant contains exactly one ISIS to bind, and the Registrant will collect the necessary information about the other ISISes belonging to the same network.

To register services to more than one ISIS network multiple Registrant instances has to be configured. In this case, the default Registrant will be used for registering every services unless configured expilicitely. The registration of service can be done once or periodically based either on the configuration of the Registrant or overwritten for every service separately. The Registrant is also performing message aggregation of all services linked to it if possible. The simplified algorithm of the Registrant is presented below.

Registrant - pseudo algorithm

```
// Initialize phase
Read the configuration and store the information about the services in a list
do { // Cyclic phase in a different Thread
 wake_up_time = now();
 messages = null;
  if ( 0 < count(service where service.next_run <= wake_up_time)) {
    foreach( service where service.next_run <= wake_up_time) {
      messages.add(service.RegistrationCollector);
```

```
        service.next_run = wake_up_time + service.period;
      }
      if (0 < count(messages)) {
        sent_message = assemble message with headers(messages);
        send(sent_message);
      }
    } else {
      sleep(min(service.next_run) - now());
    }
  } while(true)
```

Current implementation does not allow value of the period to be less than 2 minutes.

Service provides **Service Advertisement** part of information sent to ISIS (see Section 2.1.2). Before sending this information the Registrant extends it with additional data (**Service Advertisement Metadata**).

An example layout of services is shown on Figure 3.1. In this configuration example there are two Registrants configured in one HED container for three services. The *Registrant A* is the default Registrant and the services configured in the following way:

- Service 1: There is no Registrant configured so the default one will register it.
- Service 2: The *Registrant A* is configured explicitly.
- Service 3: The *Registrant B* is configured explicitly.

So the Service 1 and 2 are handled both by *Registrant A* and Service 3 by *Registrant B*. In the first step each Registrant performs information collection from all assigned services sequentially. Registrants themselves are executing in parallel. For Figure 3.1 the approximate sequence of information collection is (*A1*-*A2*, *B1*-*B2*) followed by (*A3*, *A4*). After collection the aggregated information is sent to the corresponding ISISes independently by each other Registrant (*A5*, *B5*).

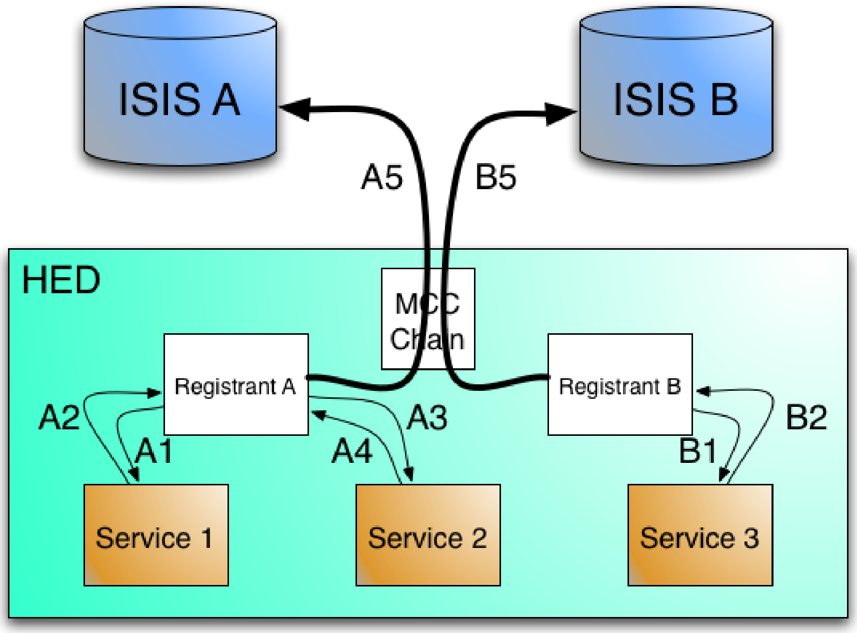

Figure 3.1: Overview of the registration process

This registration operation is done once during the start-up phase and periodically according to (per service) configured periods.

3.3 Configuration