10/3/2010

ISIS Information Indexing Service

Technical handbook

Ivan Marton[*]

[*]martoni@niif.hu

Contents

1 Introduction

2 Components

2.1 Registrant

The Registrant is active module of the HED (Hosting Environment Daemon) which is bound to a set of ISIS's. 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 ISIS's belonging to the same network.

If the administrator wants to use more then one network for registering his services then more then one Registrant have to be instanced for this purpose. In this case, the default Registrant will be used for registering every services that isn't configured in a different way.

The registration can be done once or periodically. It based on the configuration of the Registrant as default behavior but can be overwritten for every service separately.

The Registrant is doing the message aggregation of every service belongs to it if it's possible. The algorithm of the Registrant is following:

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)
```

There is a minimum value of the period defined. If the settings is less then 2 minutes then it will be replaced with this value. Before sending the information received from the services (**Service Advertisement**, see Section 2.2), the Registrant extend them some additional data (**Service Advertisement Metadata**). This part of the message contains the following information:

- GenTime: (Generation Time) The actual timestamp (called wake_up_time in the pseudo code)
- Expiration: (The time of Service Advertisement's expiration.) Generation Time + Renewal period length.
- Source: The Endpoint of the Service
- Status: Status of the Service Advertisement.

A **Registration Entry** contains a **Service Advertisement** and a corresponding **Service Advertisement Metadata**.

The **Registration Message** is the one that will be sent to an ISIS service. This message is consists of more **Registration Entry** and at most **Registration Header**. This embedded structure can be shown on Figure 2.

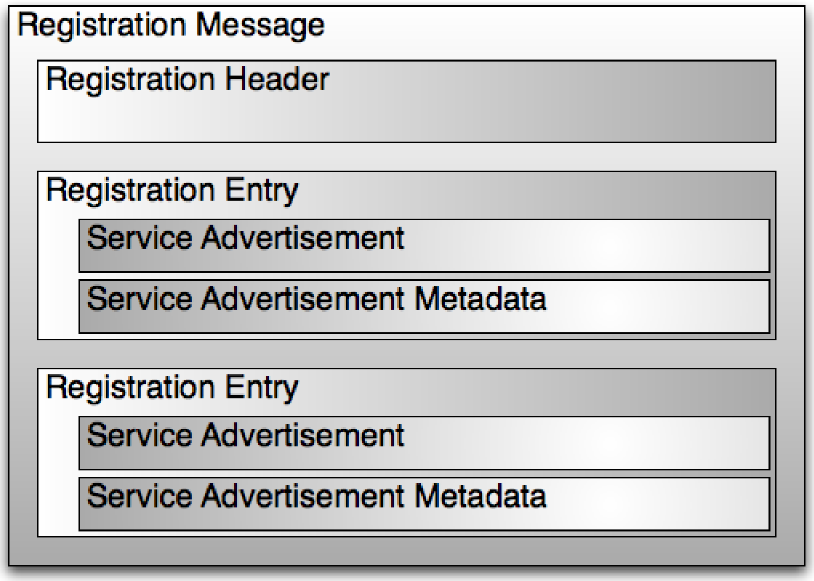

Figure 1: Embedded structure of Registration Message

2.2 Services

The service developers have to ensure that the services are able to provide the necessary information about themselves. It can be easily done by implementing the proper interface of the Arc::Service class. It means to fill the RegistrationCollector function with the necessary code parts to assemble the always up-to-date status information about the service and anything else wants to be advertised. This information package is called **Service Advertisement**.

The **Service Advertisement** can contains any information wants to advertised but the mandatory elements are the following attributes these one have to be present:

- Service ID: A globally unique identifier.
- Service Type: The Glue2 type of service.
- Endpoint URL: The URL where the service can be achieved.

2.3 ISIS Service

The ISISs perform two completely different function. On the one hand they are working as ordinary HED services, and on the other hand maintain a peer-to-peer network between each other.

2.3.1 Service functions

Every ISIS service has two different interfaces:

- WSRF interface
- service specific interface

The WSRF interface returns the service specific information expect of the stored database. (The Property Document doesn't contains the Activities that are the Registration Entries in this case.) The service specific interface is for retrieve these data. This interface has 5 operation defined:

- Register

  The register operation is accessed by the Registrant. See Section 3.1 for further details.
- RemoveRegistration

  This operation is used for requesting the removal of a list of zero or more Registration Entries stored in the Information System. The attribute of this operation is a list of Service IDs. Every entry belonging to any of the given Service ID will be removed.
- GetRegistrationStatuses

  This operation is used for requesting the current status(es) of zero or more Registration Entries previously pushed into the system by the Register operation. The operation receives a list of Service ID as its input and returns a list of ¡ServiceID, Status¿ pairs.
- GetISISList

  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 you are interested in every known ISIS then they can be queried with the Query operation because they are ordinary services registered into the Information System.
- Query The operation is used for request any kind of queries related to the Indexing Database. For providing as much flexibility as possible any XQuery searches can be executed. This operation is accessed by clients. For an example and further details see Section 3.2.

2.3.2 Peer-to-Peer functions

2.4 Client

The client part of the information service is for querying the data stored in the database of the ISIS services. This is implemented as a part of the ARCLib and using the ISIS service's interface for the various data queries. For further details see Section 3.2.

3 Operations

The different operations are performed by different Information system components. These cases are described below in the dedicated subsections.

3.1 Registration

The registration operation is done by the Registrants and described in Section 2.1. An example layout can be shown on Figure 2. In this configuration there is two different Registrant configured in one HED for three services. Let the *Registrant A* 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.

In the sketched case the Service 1 and 2 are handled equivalently by *Registrant A* and Service 3 by *Registrant B*.

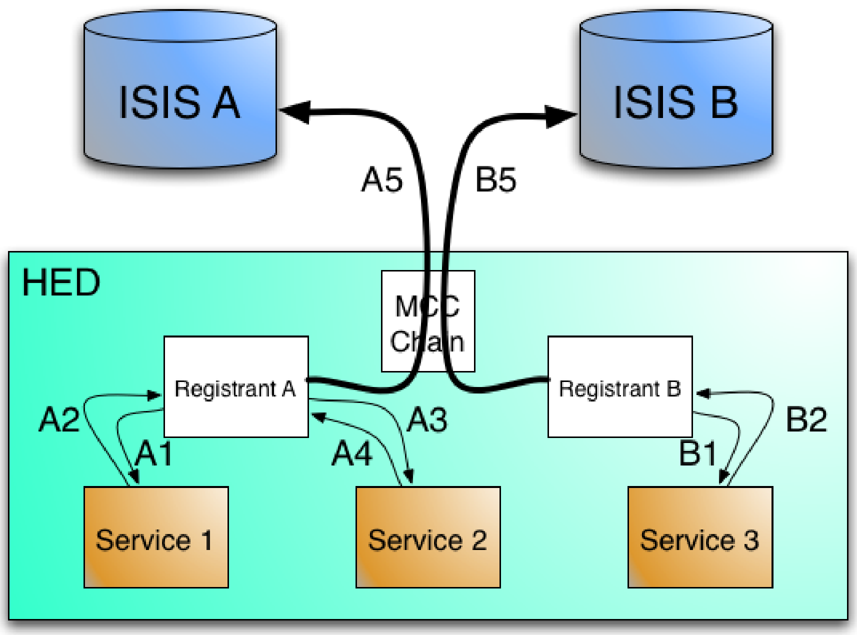

Figure 2: Overview of the registration operation

In the first step the information collection from the first handled services is done parallel (*A1*-*A2*, *B1*-*B2*). The retrieved data are stored and other services are queried if there is any (*A3*, *A4*). Finally, the aggregated information is sent to the corresponding ISISs independently from each other (*A5*, *B5*).

This registration operation is done once during the start-up phase and cyclic later by right of the (per service) configured periods. This periodical operation follows the algorithm described in Section 2.1, and do every possible message aggregation.

3.2 Query data

3.3 Configuration

The configuration of the different modules can be done through the ARC Configuration file. The format of these entries is shown by the relevant XSD Schema files.

Registrant.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:registrant="http://www.nordugrid.org/schemas/RegistrantConfig/2008"
  xmlns:arcConfig="http://www.nordugrid.org/schemas/ArcConfig/2007"
  targetNamespace="http://www.nordugrid.org/schemas/RegistrantConfig/2008"
  elementFormDefault="qualified">

    <xsd:complexType name="Registrant_Type">
        <!-- This element defines configuration of Information
            Registration interface. -->
        <xsd:sequence>
            <!-- Period specifies how often registration to be done in seconds.
                The missing or 0 value means a non period registration. -->
            <xsd:element name="Period" type="xsd:nonNegativeInteger"/>
            <!-- URL specifies contact endpoint of a bootstrap Information
```

```
                Registration service. Further ISIS's adresses will be queried
                from this service. -->
            <xsd:element name="ISIS_URL" type="xsd:string" />
            <!-- Optional KeyPath, CertificatePath, ProxyPath and CACertificatesDir
                are paths to files storing X509 credentials used for establishing
                connections. -->
            <xsd:element name="KeyPath" type="xsd:string" minOccurs="0" />
            <xsd:element name="CertificatePath" type="xsd:string" minOccurs="0" />
            <xsd:element name="ProxyPath" type="xsd:string" minOccurs="0" />
            <xsd:element name="CACertificatesDir" type="xsd:string" minOccurs="0" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="Registrant" type="registrant:Registrant_Type"/>

    <!-- Optional element for Service element overriding the default Period length. -->
    <xsd:element name="Period" type="xsd:nonNegativeInteger"/>

</xsd:schema>
```