DRAFT
SAGA-RD

Steve Fisher and Antony Wilson
Rutherford Appleton Laboratory, UK

Version: 1

November 15, 2011

SAGA API Extension: Information System Navigator API

Status of This Document

This document provides information to the grid community, proposing a standard for an extension to the Simple API for Grid Applications (SAGA). As such it depends upon the SAGA Core API Specification [4]. This document is intended to be used as input to the definition of language specific bindings for this API extension, and as reference for implementers of these language bindings. Distribution of this document is unlimited.

Abstract

This document specifies a Information System Navigator API extension to the Simple API for Grid Applications (SAGA), a high level, application-oriented API for grid application development. This Information System Navigator API is motivated by a number of Use Cases collected by the OGF SAGA Research Group in GFD.70 [5], and by requirements derived from these Use Cases, as specified in GFD.71 [6]. Though motivated by the need to allow users to find additional information about services to that available via the SAGA Service Discovery API it is not dependent upon the Service Discovery API and is applicable to any information system that can be mapped to an entity relationship model.

Contents

1 Introduction

The SAGA Information System Navigator API provides a mechanism to retrieve data from any information model that can be mapped to an entity relationship structure.

One specific use of this API is to provide more detailed information about a service than can be obtained with the `saga::sd` package, [3]. Having selected a service with the `saga::sd::discoverer` API, this API provides the means to traverse the information model and retrieve data published about that service. Alternatively it is possible to start with a selected entity type rather than a service. An optional filter can be used to restrict the results returned. An example from the SAGA use cases [4] is UC 1: Core Grid integrated toolkit, section 7 [4], requires resource selection based on information that is not available from the `saga::sd` package in particular they say:

> "Probably we can think of not only taking into account static information (CPU power, memory) but also dynamic (load level, network congestion...)"

The `saga::sd` package is independent of the data model and will work equally well, for example, with GLUE 1 [1] and GLUE 2 [2]. However more complex queries require that the data model is exposed; this is why the Information System Navigator (ISN) API is needed.

The ISN API can also be used to examine application oriented information rather than information relating to the available services and resources.

This API extension is tailored to provide exactly this functionality, at the same time keeping coherence with the SAGA Core API look & feel, and keeping other Grid related boundary conditions (in particular middleware abstraction and authentication/authorization) in mind.

1.1 Notational Conventions

In structure, notation and conventions, this documents follows those of the SAGA Core API specification [4], unless noted otherwise.

1.2 Security Considerations

As the SAGA API is to be implemented on different types of Grid (and non-Grid) middleware, it does not specify a single security model, but rather provides hooks to interface to various security models – see the documentation of the saga::context class in the SAGA Core API specification [4] for details. A SAGA implementation is

considered secure if and only if it fully supports (i.e. implements) the security models of the middleware layers it builds upon, and neither provides any (intentional or unintentional) means to bypass these security models, nor weakens these security models' policies in any way.

2 SAGA Information System Navigator API

2.1 Introduction

The ISN API provides a mechanism to retrieve data from any information model that can be mapped to an entity relationship structure. It is not restricted to information about services. The schema of the information model is exposed to and must be understood by the user.

It is expected that this ISN API will make use of various information systems. The quality of the information returned will depend upon the quality of the data in the back-end system or systems.

The syntax chosen is similar to that used in the `saga::sd` package, however there is no other connection between the ISN and SD packages.

2.1.1 Information Model

This API can be used to navigate any information system that can be represented as an entity relationship model; this includes the GLUE 1 [1] and GLUE 2 [2] information models. The information models supported is dependent upon the implementation.

2.1.2 Classes

The SAGA Information System Navigator API consists of an `entity_data_set` class which contains a set of `entity_data` objects.

The `entity_data_set` class has three methods: `get_data`, `list_related_entity_names` and `get_related_entities`. The `get_data` method returns a list of objects of the `entity_data` class, with each `entity_data` object representing an instance of an entity as described in the GLUE entity relationship model. The `list_related_entity_names` method returns a list of names of entities for use with the `get_related_entities` method, where the names represent the entities, in the entity relationship model, that can be navigated to from the current entity. The `get_related_entities` method returns an object of the `entity_data_set` class, filtered according to a specified filter.

The `entity_data` class implements the `saga::attributes` interface giving *ReadOnly* access to all the key names and values in the `entity_data` object.

2.2 Specification

```
package saga.isn {

   class entity_data_set: implements saga::object
                          implements saga::async
   {
      CONSTRUCTOR           (in  session session,
                             in  url     url = "",
                             in  string  model,
                             in  string  entity_name,
                             in  string  filter,
                             out entity_data_set eds);

      DESTRUCTOR            (in  entity_data_set eds);

      get_data              (out array<entity_data> ed);

      get_related_entities (in  string related_name,
                            out entity_data_set eds);

      get_related_entities (in  string related_name,
                            in  string filter,
                            out entity_data_set eds);

      list_related_entity_names (out array<string>
                                 entities);
   }

   class entity_data: implements saga::object
                      implements saga::async
                      implements saga::attributes
   {
      // no CONSTRUCTOR
      DESTRUCTOR (in entity_data ed);
      // Attributes (extensible):
      //
      // no attributes pre-defined
   }
}
```

2.3 Specification Details

This API will typically use some underlying information system. It may try to use an underlying information system but not be able to access it. If no result can be returned because of information system or other internal problems, an implementation SHOULD throw the `NoSuccess` exception.

`class entity_data_set`

The `entity_data_set` provides the means to navigate around the information model from a selected entity and gives access to the `entity_data` objects.

Navigation consists of moving from entity to entity within an information model, as expressed in the GLUE entity relationship model. A list of possible navigation steps from an `entity_data_set` object is returned by the `list_related_entity_names` method. Navigation to a set of related entities is achieved with the `get_related_entities` method, which returns a new `entity_data_set` object.

In order to restrict the number of `entity_data` objects returned in the `entity_data_set` object, a filter may be used with the `get_related_entities` method. The filter MUST only include attributes from the related entity and it will be applied to the related entities.

Both the constructor for the `entity_data_set` and the `list_related_entity_names` methods take a filter string as an argument. This filter string is used to restrict the set of entities returned. The filter MUST only include attributes from the named entity for the constructor or from the related entity for the `list_related_entity_names` method. The filter strings uses SQL92 syntax as if it were part of a WHERE clause acting to select from a single table. SQL92 has been chosen because it is widely known and has the desired expressive power. Multi-valued attributes are treated as a set of values.

Only the following operators are permitted in expressions not involving multi-valued attributes: IN, LIKE, AND, OR, NOT, =, >=, >, <=, <, <>, in addition to column names, parentheses, column values as single quoted strings, numeric values and the comma. For a multi-valued attribute, the name of the attribute MUST have the keyword ALL or ANY immediately before it, unless comparison with a set literal is intended. For each part of the expression, the attribute name MUST precede the literal value. An implementation SHOULD try to give an informative error message if the filter string does not conform.

The LIKE operator matches string patterns:

'%xyz' matches all entries with trailing xyz

'xyz%' matches all entries with leading xyz

'%xyz%' matches all entries with xyz being a substring

The ESCAPE keyword can be used with LIKE as defined by SQL92.

Column names are not case sensitive but values are.

No use-case has been identified for the operators `>=`, `>`, `<=`, `<` to be applied to strings. An implementation wishing to support these comparison operators on strings MUST select a collation sequence. Alternatively, an implementation CAN treat all string comparisons as true, or reject them as invalid SQL.

The `get_related_entities` operation is overloaded: the last parameter, the `filter,` may be omitted.

```
- CONSTRUCTOR
  Purpose:   create a new entity_data_set object
  Format:    CONSTRUCTOR  (in session  session,
                           in url      url = "",
                           in string   model,
                           in string   entity_name,
                           in string   filter,
                           out entity_data_set eds);
  Inputs:    session:      optional session handle
             url:          optional URL to guide the
                           implementation
             model:        the name of the information model
             entity_name:  name of the entity to navigate to
             filter:       filter for filtering entities,
                           may be null
  Outputs:   eds:          new entity_data_set object
  Throws:    AuthorizationFailed
             AuthenticationFailed
             BadParameter
             DoesNotExist
             NoSuccess
             NotImplemented
             Timeout
  Notes:     - the url specified as in input parameter is to
               assist the implementation to locate the
               underlying information system such that it
               can be queried.
             - if the url is syntactically valid, but no
```

```
                service can be contacted at that URL, a
                'DoesNotExist' exception is thrown.
              - the semantics for the other exceptions is as
                outlined in the SAGA Core API specification
              - note that the session parameter is optional,
                as described in the SAGA Core API
                specification, section 3.5.2. Also Section
                2.2.2 of the same document applies to
                url and its default value.
              - the model name used should be consistent
                across implementations. For GLUE information
                models the names SHOULD be “glue1” and
                “glue2”.

- DESTRUCTOR
  Purpose:   destructor for entity_data_set object
  Format:    DESTRUCTOR   (in entity_data_set eds);
  Inputs:    eds:          object to be destroyed
  Outputs:   -
  Throws:    -
  Notes:     -

- get_data
  Purpose:   returns a set of entity_data objects
  Format:    get_data     (out array<entity_data> ed);
  Inputs:    -
  Outputs:   ed:           a set of entity_data objects
                           associated with this entity
  Throws:    NoSuccess
             Timeout
  Notes:     -

- get_related_entities
  Purpose:   returns an entity_data_set object for the given
             entity name and matching the filter string
  Format:    get_related_entities (in string related_name,
                                   in string filter,
                                   out entity_data_set eds);
  Inputs:    related_name:   name of the related entity to
                             navigate to
             filter:         optional filter for filtering
                             related entities
  Outputs:   eds:            entity data set matching the
                             specified filter string
  Throws:    BadParameter
             DoesNotExist
             NoSuccess
             Timeout
  Notes:     - the filter MUST only include attributes from
               the related entity. There is a special
               case where there is a self relationship
               between entities, i.e. "AdminDomain" in GLUE
```

```
                2, in such cases the keywords "up" and "down"
                should be used in place of the name of the
                related entity to navigate to. The directions
                "up" and "down" are implementation dependent.
              - the last parameter, the filter, may be
                omitted.
              - if the related_name is syntactically valid,
                but is unknown as a relation to the original
                entity a 'DoesNotExist' exception is thrown.

- list_related_entity_names
   Purpose:   returns a set of names of those entities that
              may be navigated to, from this entity_data_set
   Format:    list_related_entity_names (out array<string>
                                         entities);
   Inputs:    -
   Outputs:   entities: a list of names of related entities
   Throws:    NoSuccess
              Timeout
   Notes:     - this is a schema operation
              - there is a special case where there is a
                self relationship between entities, i.e.
                "AdminDomain" in GLUE 2, in such cases the
                keywords "up" and "down" will be included
                as appropriate.
```

---

class entity_data

The `entity_data` class provides read access to the data of an entity. This class implements the `saga::attributes` interface and offers getter methods for the user to read key/value pairs. Access to the keys and values is through the `saga::attributes` interface. The class provides no other methods. This class has no CONSTRUCTOR, it can only be accessed via an `entity_data_set` object.

```
- DESTRUCTOR
   Purpose:   destructor for entity_data object
   Format:    DESTRUCTOR   (in entity_data ed);
   Inputs:    ed:           object to be destroyed
   Outputs:   -
   Throws:    -
   Notes:     -
```

2.4 Examples

This C++ example shows, using a possible C++ binding, how the SAGA information system navigator is used to get data about selected sites. For this example we use the information model “glue1” and select the entity “Site”. To restrict the sites returned by the query the filter “Description='LCG Site'” is used, where “Description” is an attribute of the “Site” entity. An `entity_data_set` object is returned in response to the query. This object contains a set of `entity_data`, with each `entity_data` relating to details about an individual site. The second example shows how to extract the data for each site.

```
#include "saga/saga.hpp"
int main(int argc, char *argv[])
{
  std::string model = "glue1";
  std::string entity_name = "Site";
  std::string filter = "Description='LCG Site'";
  try
  {
     // Create an EntityDataSet
     saga::isn::entity_data_set eds(model, entity_name,
        filter);
     std::cout << "Selected " << eds.get_entity_count()
        << " sites" << std::endl;
     std::vector<std::string>
        rel =   eds.list_related_entity_names();
     std::vector<std::string>::const_iterator iter;
     std::vector<std::string>::const_iterator
        endIter = rel.end();
     std::cout << "Related Enities:"<< std::endl;
     for ( iter = rel.begin(); iter != endIter; ++iter )
     {
        std::cout << "    " << *iter << std::endl;
     }
  }
  catch ( saga::exception& e )
  {
     std::cerr << "ERROR: " << e.get_message() << std::endl;
     exit(1);
  }
  return 0;
}
```

In order to examine the contents of the data associated with an entity add the following after line 24 of the previous example:

```
1  // Extract the data set
2  std::vector<saga::isn::entity_data> data_set =
3     eds.get_data();
4  std::vector<saga::isn::entity_data>::const_iterator
5     dataIter;
6  std::vector<saga::isn::entity_data>::const_iterator
7  endIter = data_set.end();
8  for ( dataIter = data_set.begin();
9        dataIter != endIter;
10       ++dataIter )
11 {
12    std::vector<std::string> attribNames =
13       dataIter>list_attributes();
14    std::vector<std::string>::const_iterator attribNamesIter;
15    std::vector<std::string>::const_iterator
16       attribNamesEnd = attribNames.end();
17    for ( attribNamesIter = attribNames.begin();
18          attribNamesIter != attribNamesEnd;
19          ++attribNamesIter )
20    {
21       if ( !dataIter>attribute_is_vector(*attribNamesIter) )
22       {
23          std::string attribValue =
24             dataIter>get_attribute(*attribNamesIter);
25          std::cout << *attribNamesIter << ": "
26             << attribValue << std::endl;
27       }
28       else
29       {
30          std::vector<std::string> attribValues =
31             dataIter>get_vector_attribute(*attribNamesIter);
32          std::vector<std::string>::const_iterator
33             attribValuesIter;
34          std::vector<std::string>::const_iterator
35          attribValuesEnd = attribValues.end();
36          for ( attribValuesIter = attribValues.begin();
37                attribValuesIter != attribValuesEnd;
38                ++attribValuesIter )
39          {
40             std::cout << *attribNamesIter << ": "
41                << *attribValuesIter << std::endl;
42          }
43       }
44    }
45    std::cout << std::endl;
46 }
```

3 Intellectual Property Issues

3.1 Contributors

This document is the result of the joint efforts of several contributors. The authors listed here and on the title page are those committed to taking permanent stewardship for this document. They can be contacted in the future for inquiries about this document.

Antony Wilson
antony.wilson@stfc.ac.uk
Rutherford Appleton Lab
Harwell Science &
Innovation Campus
DIDCOT
OX11 0QX
UK

Steve Fisher
dr.s.m.fisher@gmail.com
Rutherford Appleton Lab
Harwell Science &
Innovation Campus
DIDCOT
OX11 0QX
UK

3.2 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.

3.3 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.

3.4 Full Copyright Notice

References

[1] S. Andreozzi et al. GLUE Schema Specification version 1.3. https://forge.gridforum.org/sf/go/doc14185?nav=1, 2007.

[2] S. Andreozzi et al. GLUE Schema Specification version 2.0. http://www.ogf.org/documents/GFD.147.pdf, 2009.

[3] S. Fisher, A. Wilson and A. Paventhan. SAGA API Extension: Service Discovery API. Grid Forum Document GFD-R-P.144, 2009. Open Grid Forum.

[4] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and C. Smith. A Simple API for Grid Applications (SAGA). Grid Forum Document GFD.90, 2008. Open Grid Forum.

[5] A. Merzky and S. Jha. A Collection of Use Cases for a Simple API for Grid Applications. Grid Forum Document GFD.70, 2006. Global Grid Forum.

[6] A. Merzky and S. Jha. A Requirements Analysis for a Simple API for Grid Applications. Grid Forum Document GFD.71, 2006. Global Grid Forum.