caBIG_logo_w_tm

***Users and Developers Guide v. 1.0***

**caGRID**

**Identifier Framework**

|  |  |
| --- | --- |
| ***Contacts and Support*** | |
| Calixto Melean (Developer) | Calixto.Melean@osumc.edu |
| Scott Oster (Architect) | Scott.Oster@inventrio.com |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Revision History*** | | | |
| **Version** | **Date** | **Author** | **Changes** |
| **0.1** | 2010-07-20 | **Calixto Melean** | **Initial Draft** |

Chapter 1 Identifier Metadata 5

Chapter 2 Naming Authority Grid API 6

Exceptions 6

NamingAuthorityConfigurationFault 6

InvalidIdentifierFault 6

NamingAuthoritySecurityFault 6

InvalidIdentifierValuesFault 6

createIdentifier 6

resolveIdentifier 7

deleteKeys 8

createKeys 9

replaceKeyValues 10

getKeyNames 11

getKeyData 12

Chapter 3 Naming Authority Web Interface 13

Home Page and HTML Responses 13

XML Responses 14

Retrieving Naming Authority Configuration 15

# Identifier Metadata

caGrid provides a framework for globally identifying objects in the grid. The identifier is essentially a forever globally unique name for the data-object such that it can be unambiguously used to refer to the data from different application contexts.

Metadata is information that can be attached to the identifier. It’s any information that describes the object being identified. Typically, it would also be information that can be used to locate and/or retrieve the target data object.

When a deployment of identifiers is being planned, an important decision to be made is what the metadata is going to be.

A typical example is the identification of data objects accessible by a caGrid data service. The framework’s *identifiers-client* project has a built-in profile that enables the retrieval of such objects. This profile requires the existence of a *CQL query string* and an *End Point Reference* in the identifier metadata.

Metadata is represented in the framework in the form of key/value pairs. Where the key names the piece of relevant metadata, and value is the value associated with the metadata key. For example:

|  |  |
| --- | --- |
| **Metadata Key** | **Metadata Value** |
| EPR | <ns1:EndpointRerefence…> |
| CQL | <CQLQuery…> |

# Naming Authority Grid API

The identifiers framework provides a standard analytical grid service. This API enables the creation and maintenance of identifiers.

## Exceptions

The following exceptions can be thrown by one or more methods described in the sections that follow.

### NamingAuthorityConfigurationFault

The target naming authority is not running correctly. A configuration issue exists.

### InvalidIdentifierFault

The provided identifier does not exist.

### NamingAuthoritySecurityFault

The requesting user (grid identity) is not authorized to perform the requested operation.

### InvalidIdentifierValuesFault

The provided metadata is invalid (e.g., a non-null key array with empty key strings).

## createIdentifier

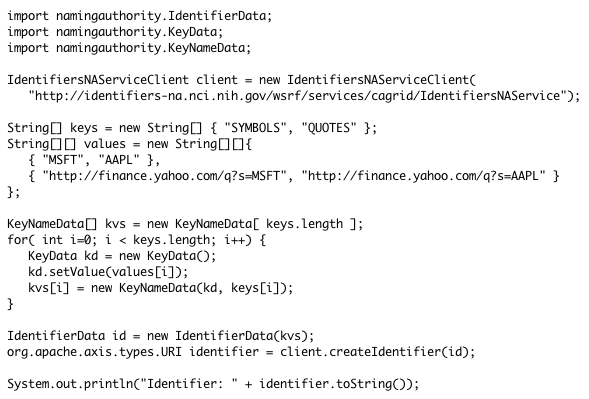
*URI* **createIdentifier**(*IdentifierData*);

This method is used to create an identifier. Input metadata (*IdentifierData*) is optional. Metadata can also be added to the identifier later using other available methods. The output is the newly created identifier URI.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*
* *InvalidIdentifierValuesFault*

Example:



## resolveIdentifier

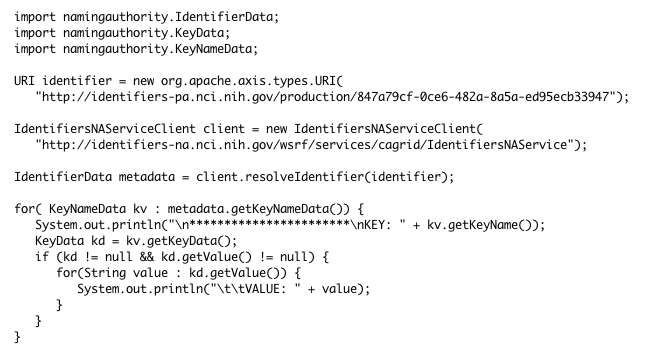
*IdentifierData* **resolveIdentifier**(*URI*);

This methods accepts an identifier and returns the associated metadata.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*

Example:



## deleteKeys

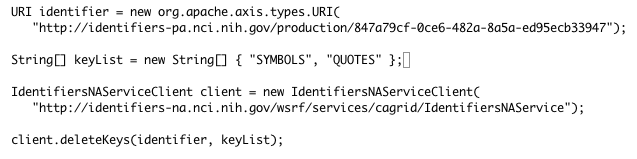
*void* **deleteKeys**(*URI* identifier, *String*[] keyNames);

This method accepts an identifier and a list of metadata key names. It deletes the specified key names from the identifier metadata.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*
* *InvalidIdentifierValuesFault*
  + *No keys were provided*
  + *Specified key(s) does not exist*

Example:



## createKeys

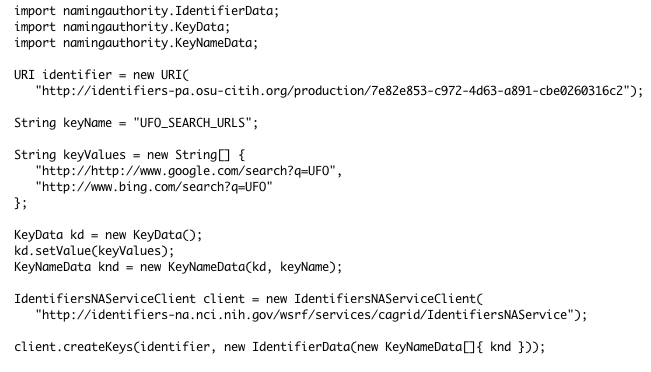
*void* **createKeys**(*URI*, *IdentifierData*);

This method is used to add new metadata keys (and their associated values) to an existing identifiers. It accepts an identifier URI and the IdentifierData structure containing the new keys and data to be added to the provided identifiers.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*
* *InvalidIdentifierValuesFault*
  + *No keys were provided*
  + *A key with the provided name already exists*

Example:



## replaceKeyValues

*void* **replaceKeyValues**(*URI*, *IdentifierValues*);

This method is used to replace the values currently assigned to the specified keys with a new set of values. Old previous values are discarded. It accepts the identifier URI and the new data.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*
* *InvalidIdentifierValuesFault*
  + *No keys were provided*
  + *One or more of the specified keys does not exist*

Example:



## getKeyNames

*String*[] **getKeyNames**(*URI*);

This method is used to retrieve the metadata key names associated with the provided input identifier. The values are not returned.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*

Example:



## getKeyData

*KeyNameData* **getKeyData**(*URI identifier, String keyName*);

This method takes an identifier and a metadata key name, and return the associated metadata value.

Exceptions:

* *NamingAuthorityConfigurationFault*
* *InvalidIdentifierFault*
* *NamingAuthoritySecurityFault*
* *InvalidIdentifierValuesFault*
  + *The key name specified does not exist.*

Example:



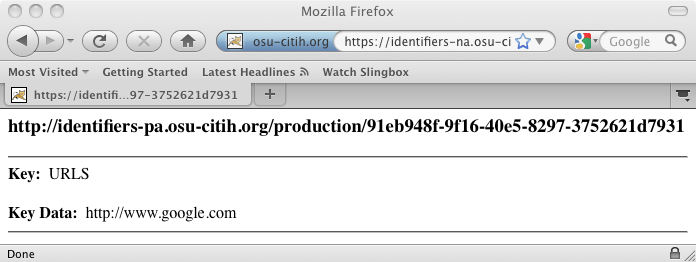
# Naming Authority Web Interface

The identifiers framework deploys a web application, whose main purpose is to enable resolution of identifiers via HTTP. When an identifier URI is “*followed*” (e.g., entered into a web browser), the web application resolves the identifier and returns the corresponding metadata.

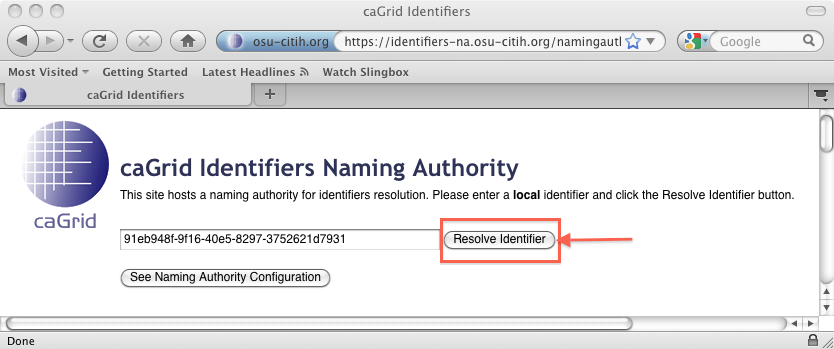
## Home Page and HTML Responses

When the web application detects a web browser, or any client that includes “*text/html*” in the *ACCEPT* HTTP request header, the response is prepared in HTML format.

For example, entering an identifier such as *http://identifiers-pa.osu-citih.org/production/91eb948f-9f16-40e5-8297-3752621d7931* in a web browser, produces the following output:



The naming authority also provides a simple home page that can be used to enter a local identifier for resolution. Simply remove the trailing *NamingAuthorityService* from the naming authority web application end point (e.g., https://identifiers-na.osu-citih.org/namingauthority):

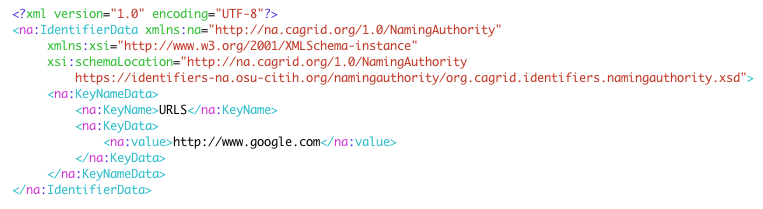


Enter a local identifier and click *Resolve Identifier*.

## XML Responses

When an HTTP client sets the *Accept* HTTP request header to “*application/xml*”, the response is returned in XML format.

For example, resolving an identifier such as *http://identifiers-pa.osu-citih.org/production/91eb948f-9f16-40e5-8297-3752621d7931* would produce a response like:

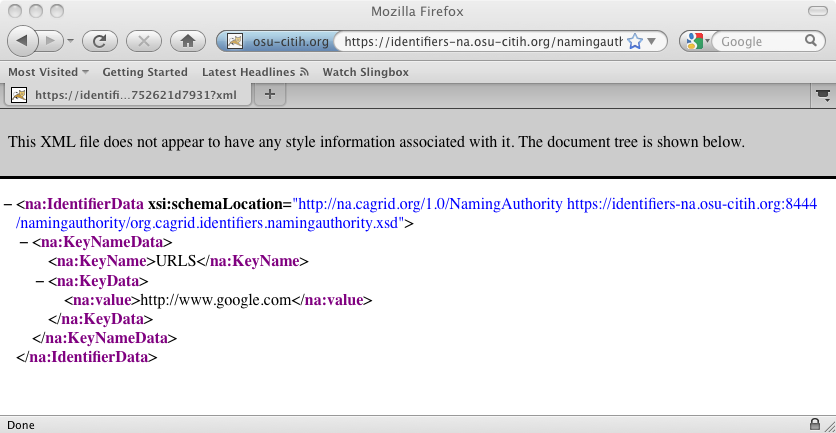


The identifiers client project (covered later) leverages this functionality and provides utility methods that can be used to resolve identifiers via HTTP and convert the response to java objects.

The XML response conforms to the schema *org.cagrid.identifiers.namingauthority.xsd*, which is available from the naming authority project, and can also be downloaded from the naming authority web application (e.g., *https://identifiers-na.nci.nih.gov/namingauthority/org.cagrid.identifiers.namingauthority.xsd*).

The naming authority web application supports an alternative way for clients to resolve identifiers to XML by adding ***“?xml***” to the identifier URI. This is useful when the ACCEPT HTTP header can not be set for some reason (e.g., a human using a web browser wishing to see XML, instead of HTML).

For example, entering an identifier such as *http://identifiers-pa.osu-citih.org/production/91eb948f-9f16-40e5-8297-3752621d7931?xml* in a web browser, produces the following output:

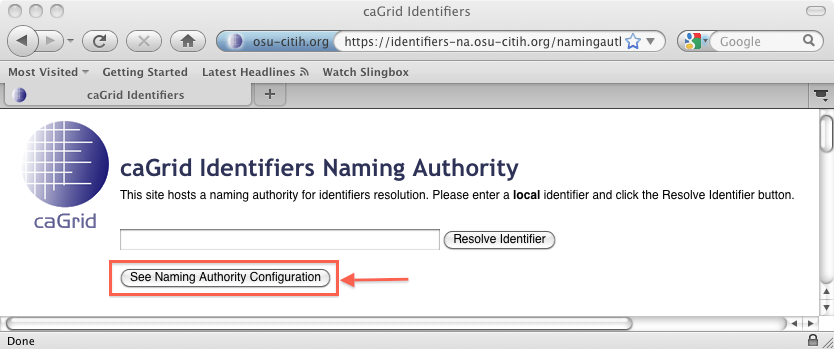


## Retrieving Naming Authority Configuration

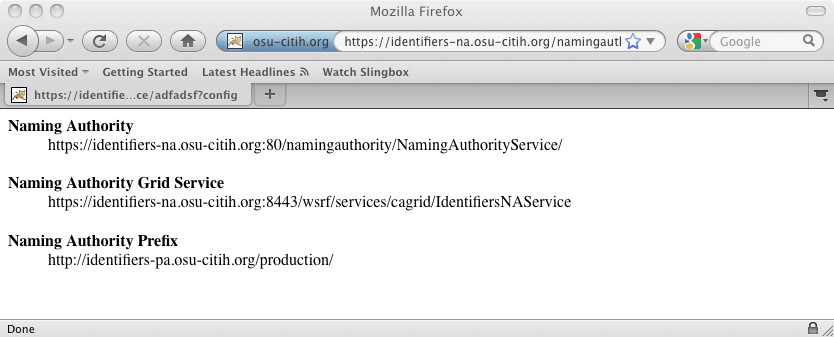
The naming authority web application makes available some of its configuration settings via HTTP. Simply add “***?config***” to any identifier URI or to the web application end point. For example:

* http://identifiers-pa.nci.nih.gov/production/7e82e853-c972-4d63-a891-cbe0260316c2**?config**
* https://identifiers-na.nci.nih.gov/namingauthority/NamingAuthorityService**/?config**

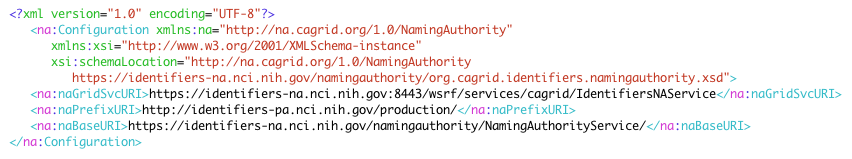
Alternatively, the *See Naming Authority Configuration* button can be used on the naming authority home page.



And the response:



As before, client programs can request XML by setting the *ACCEPT* request header to “*application/xml*”.



The identifiers client project (covered later) leverages this functionality to discover the grid service end point for a given identifier URI.

The XML response conforms to the schema *org.cagrid.identifiers.namingauthority.xsd*, which is available from the naming authority project, and can also be downloaded from the naming authority web application (e.g., *https://identifiers-na.nci.nih.gov/namingauthority/org.cagrid.identifiers.namingauthority.xsd*).

# Client

The identifiers framework includes an *identifiers-client* project that serves two purposes:

* Aid users in resolving identifiers using both the http and grid interface
* Provide an extensible framework to aid in retrieving data objects using the identifier metadata resulting from the resolution process. This is accomplished by *plugging in* retrieval profiles into the framework.

## Resolution

The ***Resolver*** class provides utility methods to resolve identifiers, hiding some of the complexity present in the underlying framework. For example, when the resolution via HTTP is used, *Resolver* automatically de-serializes the XML returned by the naming authority into naming authority domain java objects that can then be used the API client.

### resolveHttp

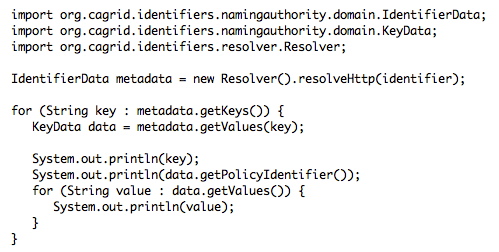
*IdentifierData* **resolveHttp**( *URI identifier* );

This method resolves the input identifier and returns the corresponding metadata. Since identifiers are live URLs, they are simply followed (http get) to retrieve their metadata in XML format from the naming authority. The XML is then de-serialized into naming authority domain objects (*IdentifierData*).

Exceptions:

* *HttpException*
  + *Unexpected HTTP error was encountered*
* *NamingAuthorityConfigurationException*
  + *Naming authority reports a configuration error*
  + *Failed to de-serialize naming authority response*
* *NamingAuthoritySecurityException*
  + *The identifier can not be resolved by anonymous users.*
* *InvalidIdentifierException*
  + *The identifier does not exist*

Example:



Since initializing a *Resolver* object is expensive, it is recommended to create the object only once when multiple identifiers are going be resolved.

### resolveHttp (GSI)

*IdentifierData* **resolveHttp**( *URI identifier*, *GlobusCredential* *credentials* );

This method resolves the input identifier using the provided globus credentials and returns the corresponding metadata. This is useful when the naming authority is running a secure deployment where identifiers may not be viewed by every one.

The implementation uses the *Globus Grid Security Infrastructure* (GSI) API (*GSIHttpURLConnection*) to target the naming authority directly. The naming authority end point is discovered by first retrieving the naming authority configuration using the input identifier URI, as explained in the *Retrieving Naming Authority Configuration* section. The reason for this is that the GSI API does not support HTTP redirects, which are used by our deployment with a prefix authority (PURL).

As before, the response XML is de-serialized into naming authority domain objects (*IdentifierData*).

Exceptions:

* *HttpException*
  + *Unexpected HTTP error was encountered*
* *NamingAuthorityConfigurationException*
  + *Naming authority reports a configuration error*
  + *Failed to de-serialize naming authority response*
* *NamingAuthoritySecurityException*
  + *The identifier can not be resolved by anonymous users.*
* *InvalidIdentifierException*
  + *The identifier does not exist*

Example:

### resolveGrid

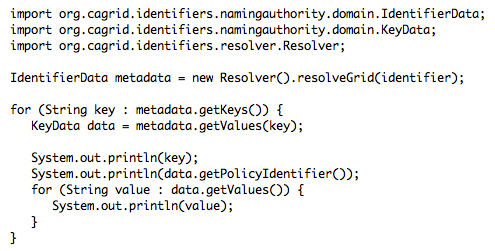
*IdentifierData* **resolveGrid**( *URI identifier* );

This method resolves the input identifier using the naming authority grid service interface (*resolveIdentifier*) and returns the corresponding metadata. The grid service end point is discovered by first retrieving the naming authority configuration using the input identifier URI, as explained in *Retrieving Naming Authority Configuration* section.

Exceptions:

* *HttpException*
  + *Unexpected HTTP error was encountered*
* *NamingAuthorityConfigurationException*
  + *Naming authority reports a configuration error*
* *NamingAuthoritySecurityException*
  + *The identifier can not be resolved by anonymous users.*
* *InvalidIdentifierException*
  + *The identifier does not exist*

Example:



Since initializing a *Resolver* object is expensive, it is recommended to create the object only once when multiple identifiers are going be resolved.

## Retrieval

TBC