caBIG_logo_w_tm

***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 The Metadata 5

Chapter 2 The Grid API 6

Exceptions 6

NamingAuthorityConfigurationFault 6

InvalidIdentifierFault 6

NamingAuthoritySecurityFault 6

InvalidIdentifierValuesFault 6

createIdentifier 6

resolveIdentifier 7

# The 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…> |

# The 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

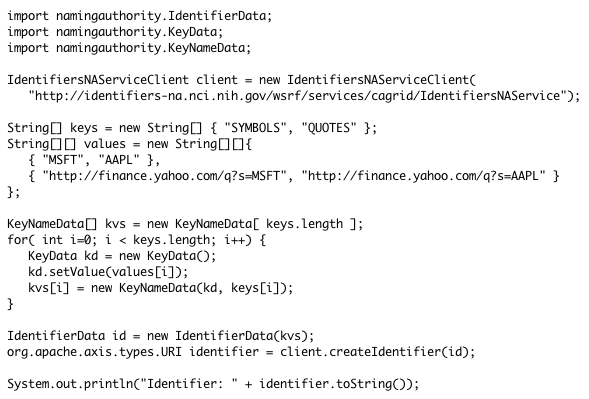
*URI* **createIdentifier**(*IdentifierData*);

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

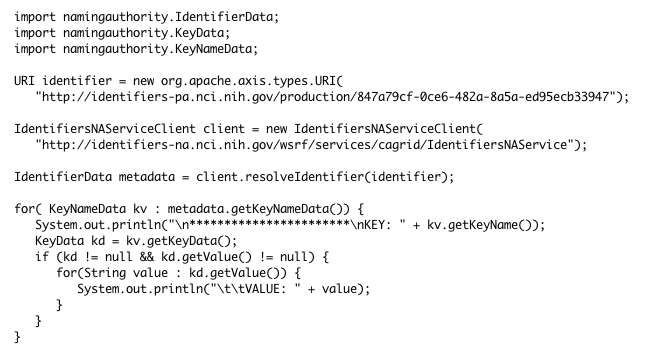
*IdentifierData* **resolveIdentifier**(*URI*);

This methods accepts an identifier and returns the associated metadata.

Exceptions:

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

Example:



## deleteKeys

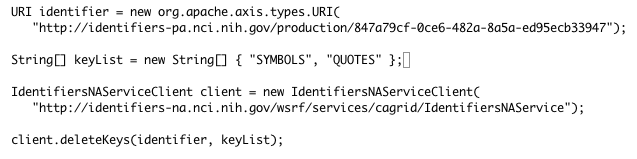
*void* **deleteKeys**(*URI* identifier, *String*[] keyNames);

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*

Example:



TBC