Energistics Identifier Specification

Overview	For use with all Energistics standards.
Version	3.0
Abstract	This document describes Energistics Identifiers.
Prepared by	Energistics and the Common Technical Architecture Team
Date published	10 May 2014
Document type	Specification
Keywords:	standards, energy, data, information, identifiers

Document Information	
DOCUMENT VERSION	1.0
Date	10 May 2014
Language	U.S. English

Amendment History			
Version	**Date**	**Comment**	**By**
2.1	22 Nov 2011	Made a separate specification independent of the GDA Specification.	Energistics and the PRODML SIG
3.0	10 May 2014	Revised and re-worded for EnergyML and ETP.	Energistics and the ETP Work Group

Table of Contents

1 Overview

1.1 Document Aims

This document describes the syntax and semantics of data object identifiers as used within the Energistics family of data exchange standards. The concepts of data object, asset, component, and feature identifiers are pervasive within the WITSML, PRODML, and RESQML specifications. This document is part of the Energistics Common Technical Architecture that defines the basis for all of the new Energistics standards. In a non-normative sense, this document also attempts to explain the relationship between the common technical architecture and older versions of the standards.

1.2 Terminology and Basic Concepts

The term *currently* as used in this document means 'at the time of publication of this document'.

1.3 Style Guidelines

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. (http://www.ietf.org/rfc/rfc2119.txt). To distinguish between server and client requirements, these key words SHALL be prefaced with SERVER and/or CLIENT.

2 Energistics Data Object

An **Energistics Data Object Schema** is a complete, high-level schema from one of the EnergyML families. By convention, these schemas are usually represented in an XSD file that begins with the string “obj_”, and having an XSD type name that begins with obj_. In EnergyML, there is usually a single root element defined in each schema, though this is not a requirement of XML schemas. A **Data Object** is a single instance document for one of these schemas. This specification is primarily concerned with standards and guidelines for naming, or identifying these data objects, references within a data object to another data object, and references between a data object and other systems or naming authorities.

3 Energistics Identifier

An Energistics Identifier may be either a UUID or an Energistics URI.

4 Universally Unique Identifier (UUID)

Within the context of any Energistics specification, a UUID (as defined by [RFC 4122]) identifies a unique Data Object. The UUID may or may not be generated or ‘owned’ by a server, so long as it conforms to the standard for a UUID. A UUID is more precisely an array 128 unsigned bytes, and can be printed and serialized in various ways. When serialized, Energistics specifications always expect the UUID to appear in ‘Microsoft Registry Format’ which looks like this:

```
f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

A UUID can also be represented as a special form of URI (as described below).

```
urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
```

A UUID by itself should only be used where the underlying data object types are known, are described in metadata or can be discovered via something like the Shared Asset Model service.

4.1 UUID and the AbstractDataObject

Earlier revisions of Energistics schemas specified the use of a unique identifier, but were not specific in requiring the use of a uuid, and often allowed a form of compound identity for some data objects, particularly in WITSML. From the release of commonv2 (shared common schema elements) forward, all Energistics XML Data Objects will inherit from AbstractDataObject, which specifies a required UUID as an attribute of the data object. The following XML snippet demonstrates this, with the yellow highlighted text. Note that this UUID is assigned by the agent or system that generates the document, and is assumed to be the identity as understood by that system.

```
<?xml version="1.0" encoding="UTF-8"?>
<resqml:WellboreFeature
   xmlns=http://www.energistics.org/energyml/data/commonv2
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xmlns:resqml=http://www.energistics.org/energyml/data/resqmlv2
   schemaVersion="v2.0.0.20140608" uuid="F2889E68-3DB3-459C-ABDD-1FBB5C1DC5BF">
   <Citation>
      <Title>Wellbore A-2</Title>
      <Originator>William McKenzie</Originator>
      <Creation>2001-12-17T09:30:47Z</Creation>
      <Format>companyA/MyStructuralModeler</Format>
   </Citation>
   <!--  other elements -->
</resqml:WellboreFeature>
```

4.2 Object References

It is often necessary for one data object to make a reference to another, without including all of the detailed information about the referenced object. In this instance, although the uuid is sufficient to uniquely identify the object, we include additional fields about the referenced object. These can be derived from the grey-highlighted fields in the above figure. In the XML snippet below, we can see how a wellbore interpretation references it parent well feature.

```
<?xml version="1.0" encoding="UTF-8"?>
<resqml:WellboreInterpretation
   xmlns=http://www.energistics.org/energyml/data/commonv2
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xmlns:resqml=http://www.energistics.org/energyml/data/resqmlv2
   schemaVersion=" v2.0.0.20140608"
   uuid="00000000-0000-0000-0000-000000000000">
   <Citation>
      <Title>a</Title>
      <Originator>a</Originator>
      <Creation>2001-12-17T09:30:47Z</Creation>
      <Format>a</Format>
   </Citation>
   <resqml:Domain>depth</resqml:Domain>
   <resqml:Interprets xsi:type="eml:DataObjectReference">
      <ContentType>application/x-resqml+xml;version=2.0.0.20140608;type=obj_WellboreFeature
      </ContentType>
      <Title>Wellbore A-2</Title>
      <UUID>F2889E68-3DB3-459C-ABDD-1FBB5C1DC5BF</UUID>
      <UuidAuthority>witsml.acme.com</UuidAuthority>
      <VersionString>2014-09-11T10:46:30Z</VersionString>
   </resqml:Interprets>
   <resqml:IsDrilled>true</resqml:IsDrilled>
</resqml:WellboreInterpretation>
```

Some comments on the above reference:

1. The UUID is required and is the uuid attribute of any AbstractDataObject-derived class.
2. The Title is required and is intended to provide a human-readable name for the object. It is the equivalent of the nameWell and nameWellbore fields in older versions of WITSML. There is no assumption that the title is unique.

3. The actual type of the reference object is DataObjectReference, and it is located in the ObjectReference package of commonv2.
4. The ContentType field is required, and indicates the XML type of the referenced object. Using the information in this field, it is easy to reference objects from a different ML (i.e. RESQML referencing a WITSML object) and of a specific version. If the **application/x-resqml+xml;version=2.0;** portion of the ContentType is missing, it considered to be the same as the content type of the enclosing document. So, in the above example, the ContentType could just contain **type=obj_WellboreFeature**, as long as the enclosing document was also a resqml v2.0 document.
5. Similarly, the **UuidAuthority** is optional, and if missing is assumed to be the same as the authority for the enclosing document. In the case of a SOAP or ETP service, we can usually assume that the authority for a given object is the server/client with which we are communicating. However, by including this value, it is possible for one object to refer to an object on another server. Note that the authority may look like a web address, but it is not one. It is up to clients and servers to arrange for the mapping from authority to an address.
6. The VersionString is optional, and is used to provide a version of this specific object instance. This is performance optimization feature that allows clients to effectively cache potentially very large objects from a server and only request a new one when changes have been made. In the example, a date time string has been used as the version, but this is entirely up to the user. From a client's perspective, the only thing that matters is if the string is different. Also note that this feature was conceived for RESQML, where many transfers occur by disk. In the context of an ETP session, the use of the Notify protocol would obviate the need for this value.

5 Energistics URI

The Energistics URI also provides a mechanism to identify an asset using compound identity, location in a hierarchy, or identifiers other than UUIDs. Two things are important to remember about URIs:

1. They are identifiers not and not necessarily locators. Because of their format, and the way identifiers are constructed it is a common mistake to think that they point to an actual resource on the web (a URL). While it is possible for a URI to be URL, this is not always true and no assumption should be made that it is true. They are simply ways of identifying things.
2. It is expected and normal for a given real-world object of interest to have multiple URIs.

5.1 Scheme

An Energistics URI is any URI that uses the Energistics URI **scheme**. Practically, this means a sequence of characters that begins with the string "eml:". The scheme-specific string that follows this prefix uniquely defines a data object or component. It has the form:

```
eml://authority/path
```

The ABNF syntax (as defined by [RFC 2234]) for the above is:

```
emlUri = "eml:" [ "//" authority ] "/" path
```

As with any URI, the scheme is not case sensitive.

5.2 Authority

This refers to the authority portion of an Energistics URI. Conceptually, its purpose is to identify an authoritative source for information about an Energistics URI. Although this value may look like a host name and may indeed be a host name, there is no assumption or requirement that it is an actual host or the address of an actual web service of any sort or that it can be directly de-referenced as a web address. In general, it is assumed that either configuration files or a discovery web service (such as the Shared Asset Model) would provide mapping between a named authority and other service endpoints.

If an authority is present, it must begin with the character sequence "//" and is separated from the rest of the hierarchy part by a single "/". It is possible for an Energistics URI to have no authority part at all. There is no reason that identifiers without an authority could not work well within a single installation if all cooperating services understood the meaning of the identifiers being passed. For instance, a simple sensor might use a system of local, un-qualified names for each measurement channel, such as Temperature, Pressure, etc. as its URIs, with no reference to a global authority.

It is recommended, but not required, that the general host/subnet naming conventions of the DNS be used in identifying authorities with increasing universality. For example:

```
sam
sam.aberdeen
sam.aberdeen.company.com
```

or, for a WITSML store

```
witsml141
witsml141.bigfoot.chevron.com
```

As with any URI, the authority is not case sensitive.

5.2.1 Well Known Authorities

The authority can also be used to identify a well-known "naming system" associated with the identifier that is present in the URI. The intent is to allow for some flexibility and extensibility in this portion of the string. It is expected that vendors, operators, and standards bodies (e.g. Energistics) will agree on well-known naming systems as appropriate. These may be private or public and would be published as appropriate. The list of these naming systems is expected to be relatively short, relatively stable, and easily mapped to generic functionality in software components.

The following names are reserved for use to designate specific well known authorities and naming systems.

api[1]

```
eml://api/well(10-digit-code)
eml://api/wellbore(12-digit-code)
eml://api/completion(14-digit-code)
```

witsml141

```
   eml://witsml141/well(uid)
   eml://witsml141/well(uid)/wellbore(uid)
```

Except for the above, this specification does not include any particular standards or conventions for the actual names of authorities. They are entirely application and system-defined.

5.3 Path

Within the Energistics URI scheme the path string is that portion which follows the authority (as defined by [RFC 3986]). This portion of the string is designed to be extensible to handle operator, vendor and industry-specific specializations of the naming syntax. This portion of the URI is really the meat of the identifier and consists of two sub-parts which may recur. Path portions begin with and are separated by a single forward slash "/". A leading slash is required following the scheme name (or authority). The ABNF syntax of the path is as follows:

path = path-part *("/" path-part)

[1] *API Bulletin D12A - The API Well Number and Standard State and County Numeric Codes Including Offshore Waters.* American Petroleum Institute. 1979.

path-part = component-type [“(” identifier “)”]

identifier = 1*(unreserved / pct-encoded)

Where `authority`, `unreserved` and `pct-encoded` are defined by [RFC 3986].

Unlike the scheme and authority, the path of a URI is case sensitive.

5.3.1 Identifier

The actual component name or id of the component is contained within parenthesis. The identifier is not required in scenarios where the identity is fully defined by the type. For example, within the context of a wellbore there is only one bottomhole so an identifier is not needed. And there would generally only be one pressure meter at the bottomhole.

```
eml://sam.mycompany.com/well(123456)/wellbore(01)/bottomhole/pressure meter
```

5.3.2 Component-type

The source of facility types in the URI is the ReportingFacility enumeration in the PRODML data schema. This list can be augmented at a site or installation using the standard mechanism in PRODML/WITSML. The following list of types existed at the time of the release of this specification:

block valve	bottomhole	casing
choke	cluster	commercial entity
company	completion	compressor
controller	controller -- lift	county
country	field	field - area
field - group	field - part	flowline
flow meter	formation	generator
installation	lease	license
manifold	organizational unit	packer
perforated interval	pipeline	plant - processing
platform	production tubing	pressure meter
pump	processing facility	regulating valve
reservoir	separator	sleeve valve
state	storage	tank
temperature meter	template	terminal
trunkline	tubing head	turbine
valve	well group	well
wellbore	wellhead	zone

The type `unknown` is explicitly forbidden in an Energistics URI.

5.3.2.1 Reserved types

In addition to the component types defined by ReportingFacility in the relevant version of the schema, the following special type is recognized: tag

The `tag` type represents a single identifier which is generally unique within the context of a historian. It should not be used within the context of any other component-type and should only be used where the underlying component types are known or can be discovered via something like a SAM service.

```
eml://sam.myCompany.com/tag/1234567890abcdef
```

5.4 Relative URI

A few special cases exist where an identifier is defined within the context of another identifier. In this scenario, the authority is defined by the parent identifier and the path is relative to the parent path. The ABNF syntax is:

energymlRelativeUri = “eml:” path

For example, the following Energistics URI:

```
eml://sam.mycompany.com/well(123456)/wellbore(01)/bottomhole/pressure meter
```

can be defined as a Relative URI within the context of a Energistics URI for the well.

```
wellbore(01)/bottomhole/pressure meter
```

A Relative URI is a special case and must only be used when the context for the URI is well-known.

6 WITSML & Compound Identifiers

Because the WITSML specification provides many of the data-objects intended to be transported by GDA, it is worthwhile to discuss exactly how WITSML objects will be identified using Energistics URIs.

1. A WITSML Store should be considered an authority, for URI purposes. Consistent with Energistics URI rules, the authority string may resemble or indeed be a host address, but there is no requirement or assumption that it be so. Authorities which refer to a WITSML store MUST begin with the string “witsml141”.
2. WITSML provides for both Uids and Names. Uids values are unique within the server that provided them and are “opaque”. Names may or may not be unique either within the WITSML server or in general. These names may or may not be the same as or similar to other valid Energistics URIs for the data-objects, but when querying for an object in a WITSML store, using the WITSML authority, you must use Uids as Energistics URIs. This does not mean that you cannot query by other names, just that they should not be supplied in Uid member of the query object.
3. WITSML uses the concept of ‘Compound Identity’ where most objects are identified by a hierarchy of several key values, only one of which is truly unique within the server. The others are only unique within the context of the ‘parent’ identifier. In virtually all cases, the hierarchy begins with well, followed by wellbore, followed by one more level (or two in some cases).
4. Considering all the above, the canonical form for the path component of a Energistics URI that represents a WITSML object then is as follows (where the values in curly braces represent the content of corresponding attributes in the data-object schema):

```
eml://witsml141/well(uidWell)/wellbore(uidWellbore)/objectName(uid)
```

where uidWell, uidWellbore and uid represent the values of those attributes and objectName represents the type of the dependent data-object (e.g., trajectory).

7 Aliases in Process Historians

The use of Aliases to define references to measurement points in process historians is somewhat unique to the world of near-real time operations and also deserves some specific treatment in terms of how Energistics URIs should be formed.

Simplistically, an Alias is a logical identifier for a tag. It does not identify anything in and of itself, but rather requires a combination of some other facility plus the alias to resolve to an actual tag or measurement value. In this sense, the use of aliases requires compound identity as described for WITSML objects above. So we might write something like this:

```
eml://sam.mycompany.com/well(123456)/sensor(BottomHolePressure)
```

to refer to a Tag name by its alias. One of the common operations desired of the Shared Asset Model (SAM) is translate between references that use tags and references that use Aliases. The response from a SAM query to do this might look like:

```
eml://osi.mycompany.com/tag(AA_12345_RT56)
```

Example

```
eml://witsml.mycompany.com/well(1231237)
```