Energistics Identifier Specification

Overview	For use with all Energistics standards.
Version of Spec	4.0
Abstract	This document describes the syntax and semantics of data object identifiers as used Energistics data-exchange standards.
Prepared by	Energistics and the Common Technical Architecture Team
Date published	16 Oct 2017
Document type	Specification
Keywords:	standards, energy, data, information, identifiers

Document Information	
DOCUMENT VERSION	1.2
Date	16 October 2017
Language	U.S. English

Amendment History				
Standard Version	**Doc Version**	**Date**	**Comment**	**By**
2.1	1.0	22 Nov 2011	Made a separate specification independent of the GDA Specification.	Energistics and the PRODML SIG
3.0	1.0	10 May 2014	Revised and re-worded for EnergyML and Energistics Transfer Protocol (ETP).	Energistics and the ETP Work Group
4.0	1.0	20 Oct 2016	Updates to the specification for the Energistics Common Technical Architecture.	Energistics and the ETP Work Group
4.0	1.1	23 Aug 2017	Clarified definitions of contentType and examples in Chapter 4. Restructured the chapter to add discrete sections.	Energistics
4.0	1.2	16 Oct 2017	Minor error corrections to code examples discovered during SIG meeting/review of WITSML on ETP implementation specifications. Removed “Chapter 7 WITSML and Compound Identifiers” because this content is now covered in the WITSML for ETP implementation specifications.	Energistics and WITSML SIG

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 (CTA) 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 CTA and older versions of the standards.

1.2 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).

2 Energistics Data Object

An **Energistics data object schema** is a complete, high-level schema from one of the Energistics domain specifications (RESQML, WITSML, PRODML). These schemas are represented in XSD files as global (i.e., root level) XML elements. For Energistics standards, there is a single root element across all schemas (AbstractObject), though this is not a general requirement of XML schemas.

A **data object** is a single-instance document described by one of these schemas. This specification is primarily concerned with standards and guidelines for names (or identification) of 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 an Energistics Content Type, a universally unique identifier (UUID) or an Energistics Uniform Resource Identifier (URI). Content Types are used to identify *types* of data objects. UUIDs identify a *single instance* of a data object, and URIs can identify *one or more instances* of a data object.

4 Energistics Content Type

An Energistics content type is a string that identifies a specific type of data object from the Energistics schemas (domain standards (aka MLs) and Energistics *common*). It follows the definition in Section 5.1 of RFC 2045 (https://www.ietf.org/rfc/rfc2045.txt).

NOTE: While we use the format and syntax of the RFC, none of the Energistics Content Types are formally registered as mime types.

4.1 Format Details

Energistics requires use of these IETF-optional parameters:

- "version=" is the schema version of the object. If an object itself is versioned, this is NOT the version of the object; it is the version of the schema to which the object conforms.
- "type=" to indicate an object type. As a special case (use is noted in Energistics specifications where this usage can be applied) **type=*** can be used to specify that all objects in a schema version are supported.

The general form of the content type is:

```
application/x-{schemaFamily}+xml;version={schemaVersion};type={topLevelObjectType}
```

Where:

- schemaFamily is the Energistics domain standards where the data object is defined: i.e., *witsml*, *prodml*, or *resqml* OR, if the data object is defined in Energistics *common*, the schemaFamily is *x-eml*.
- schemaVersion is exactly the version attribute of the associated data object schema.
- topLevelObjectType is the xml element name of the top level node of the associated data object schema OR the name of the associated XSD type. NOTE: Some earlier ML versions use an obj_ prefix as part of the element name. ML-specific documentation or specifications indicate how to handle the obj_ prefix.

4.1.1 Case Sensitivity

Referring to the format described above:

- The only part of the content type that is case-sensitive is the topLevelObjectType because it must be the name exactly as specified in WITSML, PRODML, RESQML or Energistics *common*.
- All other parts of this string are case-insensitive.

4.2 Example Content Types for Data Objects

The following are examples of well and wellbore objects from WITSML 2.0; the well test object from PRODML 2.0; all objects in RESQML 2.0.1; and a top-level object (TLO), activity, from Energistics *common* v2.1.

- application/x-witsml+xml;version=2.0;type=Well
- application/x-witsml+xml;version=2.0;type=Wellbore
- application/x-prodml+xml;version=2.0;type=WellTest
- application/x-resqml+xml;version=2.0.1;type=*
- application/x-eml+xml;version=2.1;type=Activity

4.3 Usage of Content Types in the Common Technical Architecture

ContentTypes are found in the Common Technical Architecture, in these places:

- DataObjectReference, as described in this document (see Section 5.2).
- Resource structure of the Energistics Transfer Protocol (ETP).
- ServerCapabilities record of the ETP.
- Energistics Packaging Conventions (EPC) to identify the contents of parts in a package file.

5 Universally Unique Identifier (UUID)

Within the context of any Energistics specification, a UUID as defined by RFC 4122 (https://tools.ietf.org/html/rfc4122) 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 of 16 unsigned bytes (or a single 128 bit unsigned integer), 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 using some kind of service.

5.1 UUID and the AbstractObject

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 Energistics *common* (shared common schema elements) forward, all Energistics XML data objects inherit from AbstractObject, 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" 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>
```

5.2 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, additional fields about the referenced object are included. These fields can be derived from the highlighted fields in the above figure. The XML snippet below shows how a wellbore interpretation references its parent well feature in RESQML.

```
<?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"
   uuid="79D94D70-401D-4E6D-B18A-BC744EE74E03">
   <Citation>
      <Title>Wellbore A-2</Title>
      <Originator>William McKenzie</Originator>
      <Creation>2001-12-17T09:30:47Z</Creation>
      <Format>companyA/MyStructuralModeler</Format>
   </Citation>
   <resqml:Domain>depth</resqml:Domain>
   <resqml:InterpretedFeature xsi:type="eml:DataObjectReference">
      <ContentType>application/x-resqml+xml;version=2.0;type=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:InterpretedFeature>
   <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 AbstractObject-derived class.
2. The Title is required and is intended to provide a human-readable name for the data 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 Energistics *common*.
4. The required ContentType field indicates the XML type of the referenced data object. The information in this field makes easy to reference data 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 is considered to be the same as the content type of the enclosing document. So, in the above example, the ContentType could just contain **type=WellboreFeature**, as long as the enclosing document was also a RESQML v2.0+ document.
5. The **UuidAuthority** is optional, and, similarly, if missing is assumed to be the same as the authority for the enclosing document. In the case of a Simple Object Access Protocol (SOAP) or Energistics Transfer Protocol (ETP) service, you can usually assume that the authority for a given object is the server/client with which you are communicating. However, by including this value, it is possible for one data object to refer to a data 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 designate a version of this specific data object instance. Use of VersionString is a performance optimization feature that allows clients to effectively cache potentially very large data objects from a server and only request a new one when changes have been made. In the example above, a date-time string has been used as the version, but use of VersionString is entirely up to the person/team implementing this spec. 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 eliminate the need for this value.

6 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 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, it is not always true and do not assume that it is true. URIs are simply ways of uniquely identifying things.
2. It is expected and normal for a given real-world object of interest to have multiple URIs.

6.1 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 (https://tools.ietf.org/html/rfc2234) for the above is:

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

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

6.2 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, assume that either configuration files or a discovery Web service 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. If all cooperating services understood the meanings of the identifiers being passed, identifiers without authorities should work well within a single installation. For example, 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 you use the general host/subnet naming conventions of the DNS to identify authorities with increasing universality. For example, for a WITSML store:

```
witsml14
witsml14.bigfoot.chevron.com
```

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

6.2.1 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)
```

witsml14

```
eml://witsml14/well(uid)
eml://witsml14/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.

6.3 Path

Within the Energistics URI scheme the path string is that portion that follows the authority as defined by RFC 3986 (https://tools.ietf.org/html/rfc3986). To handle operator-, vendor-, and industry-specific specializations of the naming syntax, this portion of the string is designed to be extensible. This portion of the URI is the main 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)

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.

6.3.1 Identifier

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

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

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

6.3.2 Component-type

The source of facility types in the URI is ML-specific; for example, the ReportingFacility enumeration in the PRODML data schema. This list can be augmented at a site or installation using the standard mechanism in PRODML and WITSML. The following list of types existed at the time this specification was published:

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.

6.3.2.1 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 using something like a shared asset model (SAM) service.

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