Energistics
Common Technical Architecture
Overview Guide

For Version 2 (or higher) of RESQML, WITSML, and PRODML

CTA Overview	The Energistics Common Technical Architecture (CTA) is the foundation technology that supports all of Energistics domain data-exchange standards, such as WITSML, RESQML and PRODML.
Version of Standard	2.x This document describes general concepts for Energistics standards on the Energistics 2.0 architecture. As such, it refers to multiple standards and/or components, which are each versioned separately.
Version of Document	1.1
Date published	17 June 2019
Prepared by	Energistics
Abstract	An overview of standards, components, structures and resources for the Energistics CTA.
Document type	Overview guide
Language	U.S. English
Keywords:	standards, energy, data, information

Amendment History			
Standard Version	**Document Version**	**Date**	**Comment**
NA	1.1	17 June 2019	• Title page: Clarified that this document is for general concepts for Energistics architecture v2.0+. • Updated screen shots/content in Chapter 4, for Energistics common package v2.2, in conjunction with the publication of PRODML v2.1. • Minor edits to improve readability, etc. (no intended behavior changes)

Table of Contents

1 Introduction

This guide provides a brief introduction to the Energistics domain data-transfer standards—RESQML, WITSML, and PRODML—and an overview of the Energistics Common Technical Architecture (CTA) and available resources. The CTA was developed to provide a common technical foundation of shared resources to make it easier and more efficient to implement Energistics domain data-transfer standards.

For more information about a particular domain standard, consult the technical usage guide for that standard.

1.1 What are Energistics Data Transfer Standards?

Energistics has three flagship data-transfer standards, for the three main domains of upstream oil and gas:

- **RESQML**, for reservoir life cycle, from initial structural modeling, to simulation, through production surveillance.
- **WITSML** for the well and related data and activities, such as drilling, completions, interventions, logging, etc.
- **PRODML** for optimizing producing oil and gas wells, with a focus from the reservoir-wellbore boundary to the custody transfer point.

The core of each standard is a set of XSD files (XML schemas) that define the main objects, artefacts, data, and metadata used in each domain. For example, RESQML defines faults, horizons, stratigraphy, models, etc.; WITSML defines wells, wellbores, logs, mud logs, related equipment, etc.; PRODML defines product volumes, reports, related equipment, etc.

The general purpose of these standards is the same: to facilitate transfer of data between the many software applications, systems, and technology used in the extended upstream workflow.

Developers implement the XSD schemas (and related technology) into software so that the software can read and write data in the industry-defined open format (in addition to the software's native format)—thereby making it possible for different applications, from different vendors, to "transfer" or "share" data. For example, if WITSML-enabled Vendor A software saves a file in WITSML format, then WITSML-enabled Vendor B software can read the WITSML file created by Vendor A.

All Energistics standards and related resources are freely available to download from our website: https://www.energistics.org/download-standards/.

1.2 What is Energistics Common Technical Architecture (CTA)?

The Energistics Common Technical Architecture (CTA) is a set of technology, standards, and best practices that provides a common foundation of shared resources for use across Energistics domain standards (**Figure 1–1**). The purpose of the CTA is to make it easier to implement one or all of Energistics domain standards. Ultimately the CTA improves the interoperability and efficiency of software and other technologies that implement the Energistics domain standards.

Where possible, the Energistics CTA leverages existing related standards, and, as necessary, tailors these standards to meet the needs of upstream oil and gas and the Energistics community.

The CTA is realized in these main ways:

- A set of data object schemas that are shared by the domain standards.
- Libraries to implement functionality (when available). These libraries may be from other standards organizations that Energistics standards leverage (for example, HDF5, OPC, etc.).
- Specifications that describe how CTA components work.

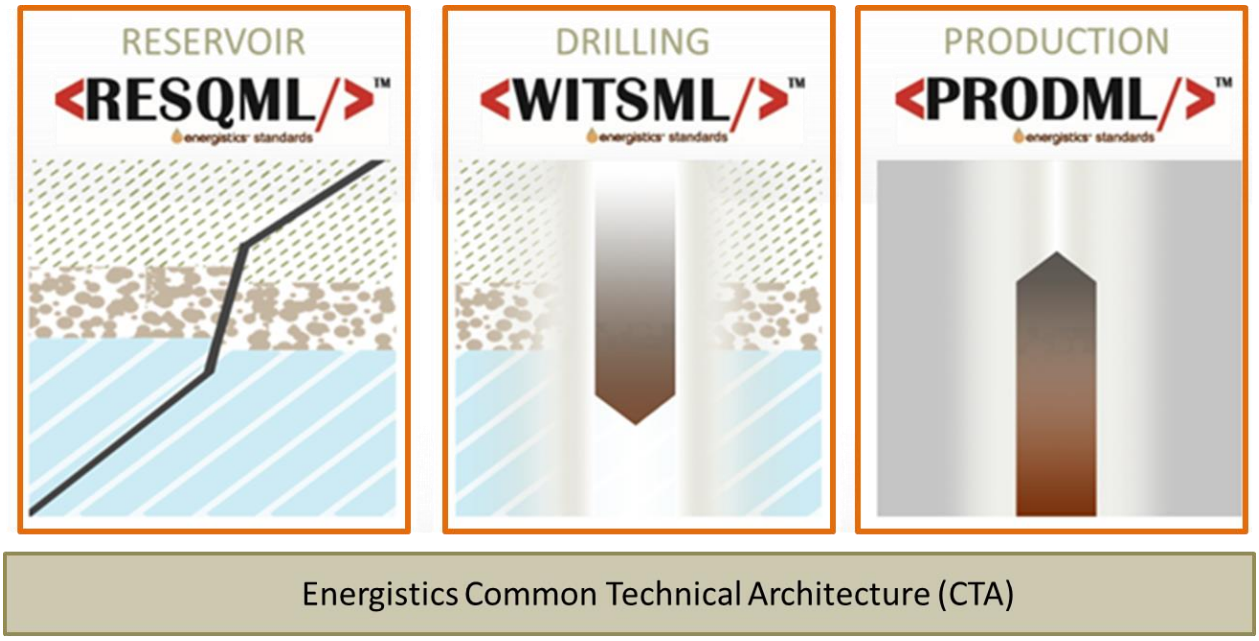

Figure 1–1. Energistics domain standards rest on a shared foundation of the Energistics Common Technical Architecture (CTA). This guide provides an overview of the CTA.

1.3 Audience, Purpose and Scope

This document:

- Is for information technology (IT) professionals—programmers, developers, architects and others—who are implementing one or more of the Energistics domain standards into a software package or other relevant technology.
- Provides an overview of the standards, components, and resources that comprise the Energistics CTA.

1.3.1 Audience Assumptions

This guide assumes that the reader has a good general understanding of programming and XML, and a basic understanding of the exploration and production (E&P) domains and related workflows.

1.4 Resource Set

Each of the Energistics domain standards—RESQML, WITSML or PRODML---is a set of XSD files (XML schemas) and other resources freely available to download and use from the Energistics website. To download a domain standard, go to https://www.energistics.org/download-standards/.

1.4.1 What do You "Get" When You Download an Energistics Domain Standard?

Standards are downloaded as .zip files. When you download any one of the domain standards, the download package contains the items listed below, which are further explained in this section. As an example, **Figure 1**–2 shows images of the content in the zip file when you download WITSML.

- **Common Technical Architecture components** (For available resources, see Section 1.4.2 (page 8)):
  - **Energistics *common* package of schemas**. The *common* package is standardized across all Energistics domain standards; when you download a version of RESQML, WITSML or PRODML, you get the appropriate version of the *common* package. Some common data objects are mandatory (for example, AbstractObject, ObjectReference, objects related to units of measure (UOM), etc.). Other data objects are optional, available for use if wanted (for example, the Data Assurance and Activity Model objects).

  - **XMI file (UML Model)**. The UML data model that developers and architects can explore for better understanding of data objects, definitions, organization, and relationships, in context. An XMI file is a format that can be imported by any UML data modeling tool.
  - **Documentation and Specifications for CTA**. Specifications and related documentation for components of the CTA as described in Section 1.4.2.

- **Domain-specific components** (For available resources, see Section 1.4.3 (page 10)):
  - **Data object schemas**. The schemas are bundled into packages, which have been organized based on that domains data model. In addition to the domain objects, each ML has its own "common" data object package for components shared across each domain (i.e., ResqmlCommon, WitsmlCommon, and ProdmlCommon).
  - **XMI file (UML Model)**. The UML data model that developers and architects can explore for better understanding of data objects, definitions, organization, and relationships, in context. An XMI file is a format that can be imported by any UML data modeling tool.
  - **Documentation**. ML-specific documentation as defined in Section 1.4.3 (page 10).

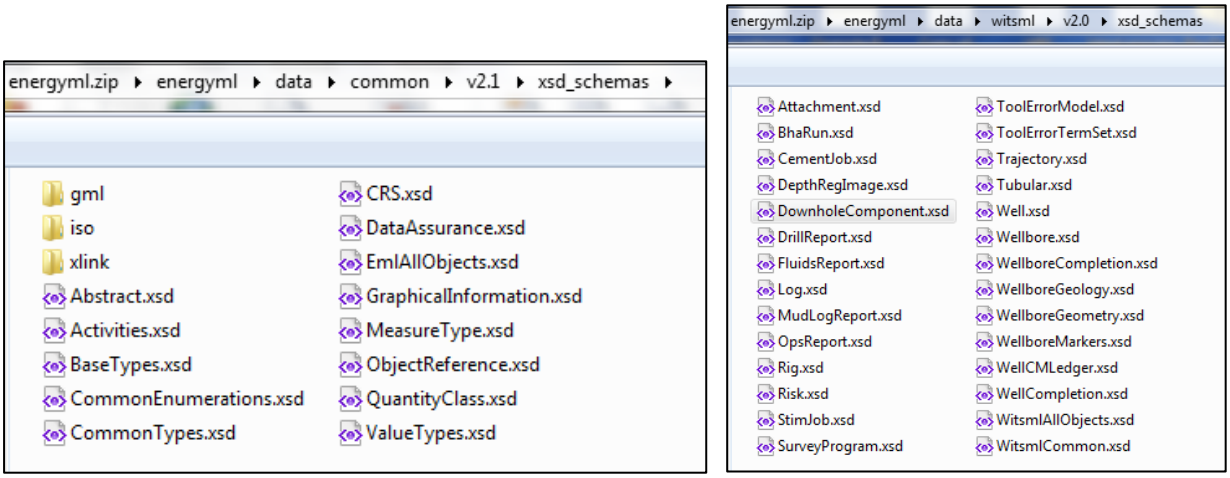

Figure 1–2. The standards are downloaded as a .zip file from the Energistics website. The figure shows (left) content of Energistics common and (right) the WITSML. Each standard has a consistent folder structure (see top of each image).

1.4.2 CTA Resources

This table lists CTA resources for use with Energistics domain standards. These resources are typically included in a standards download, unless otherwise specified in the table.

	Resource/Document	Description
1.	Energistics common XSD files	Schemas (XSD files) for Energistics data objects that can be shared by WITSML, PRODML, and RESQML, which are contained in the folder named *common* and are further described in this document. NOTE: The correct version of Energistics *common* is Included as part of the package when you download one of the Energistics domain standards (WITSML, PRODML, and RESQML).
2.	UML Data Model (XMI file)	The UML data model that developers and architects can explore for better understanding of data objects, definitions, organization, and relationships, in context. Used to generate the XSD files and technical reference documents.

	Resource/Document	Description
		NOTE: Objects in the CTA are in the folder named *common* and included in the UML for each Energistics domain standard. Energistics saves the UML model as an XMI file, a format that can be imported by any UML data modeling tool.
3.	*Energistics Common Technical Architecture Overview Guide* (this document)	Provides an overview of the components that comprise the Energistics CTA.
4.	*Energistics common Technical Reference Guide*	Lists and defines packages, data objects, elements, and relationships for the objects in the CTA *common* folder. Produced from the common UML package from which Energistics *common* XSDs are produced.
5.	*Energistics Packaging Conventions (EPC) Specification* *(Not included in WITSML download.)*	Specifies the Energistics Packaging Conventions (EPC), which is the set of practices to store multiple files as a single entity for data transfer; this single entity is referred to as an "EPC file" (or sometimes, an "Energistics package"). EPC is an implementation of the Open Packaging Conventions (OPC), a container-file technology standard.
6.	*Energy Industry Profile of ISO 19115-1 (EIP)* *(Must be downloaded separately:* https://www.energistics.org/download-standards/*)*	An open, non-proprietary exchange standard for metadata used to document information resources, and in particular resources referenced to a geographic location, e.g., geospatial datasets and web services, physical resources with associated location, or mapping, interpretation, and modeling datasets. The EIP is an ISO Conformance Level 1 profile of the widely adopted international standards ISO 19115-1:2014 which provides XML implementation guidance with reference to ISO Technical Specification 19115-3:2016.
7.	*Energistics Identifier Specification*	Describes the syntax and semantics of data object identifiers used in Energistics data exchange standards, which include UUIDs and URIs, and object reference.
8.	*Energistics Unit of Measure Standard* (Must be downloaded separately: https://www.energistics.org/download-standards/)	A dictionary, grammar specification, and related documentation, which provide a consistent way to define, exchange, and convert between different units of measure. All Energistics standards (PRODML, WITSML, PRODML, etc.) must use this dictionary; other industry groups are also using it. Key data objects and components of the UOM specification are implemented in Energistics *common*.
9.	*Energistics Transfer Protocol (ETP)* *(Must be downloaded separately:* https://www.energistics.org/download-standards/*)*	A data-exchange specification that enables the efficient transfer of real-time data between applications. Specifically envisioned and designed to meet the unique needs of the upstream oil and gas industry and, more specifically, to facilitate the exchange of data for Energistics domain data standards (RESQML, WITSML, and PRODML).

1.4.3 ML Resources

The download of an Energistics domain standard includes these ML-specific resources.

	Resource/Document	Description
1.	XSD files	Schemas (XSD files) for the domain data objects, which are contained in the folder with the domain name (per the example in Figure 1–2 (right)). NOTE: The Energistics *common* folder is Included as part of the package when you download any of the Energistics domain standards.
2.	UML Data Model (XMI file)	The UML data model that developers and architects can explore for better understanding of data objects, definitions, organization, and relationships, in context. Used to generate the XSD files and technical reference documents. NOTE: Energistics saves the UML model as an XMI file, a format that can be imported by any UML data modeling tool.
3.	*<ML> Technical Usage Guide*	Describes the data model, related key concepts, and other details to help develops understand and implement the schemas.
4.	*<ML> Technical Reference Guide*	Lists and defines packages, data objects, elements, and relationships for the objects in the data model. Produced from the UML model package from which the XSD files are produced.
5.	*<ML> Business Overview Guide*	Provides the business case for the domain standard and lists available use cases. (Currently only RESQML has this document.)

1.4.4 Documentation

Energistics is committed to providing quality documentation to help people understand, adopt, and implement its standards. As uptake of the standards increases, lessons learned, best practices, and other relevant information will be captured and incorporated into the documentation. Updated versions of the documentation will be published as they become available.

1.4.4.1 Conventions

This document uses the conventions listed in the following table.

	Document/Resource	Description
1.	Key words	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.	Business Rules	Some mandatory behaviors cannot be implemented in schemas and are specified as business rules as shown in the example below. BUSINESS RULE: Array length is the number of cells in the grid or the blocked well.
3.	Document Hyperlinks: Internal	Though no special text-formatting convention is used, all section, page and figure numbers in Energistics documents are hyperlinks. The table of contents is also hyperlinked.

2 CTA Overview

The first section of this chapter provides an overview of the CTA, identifying the main components, purpose of each, and how they work together.

Subsequent sections list and define the component standards that comprise the CTA, which are presented in these groups:

- **Information Technology Standards** (Section 2.2 (page 12)) are used as published from their respective standards organizations. These are underlying and development process standards, for example, XML, HDF5, and UML.
- **Energistics Standards** (Section 2.3 (page 13)) are based on existing standards and/or industry best practices, but have been tailored to meet the specific needs of upstream oil and gas and related data exchange.

2.1 CTA: Main Components and What They Do

Each Energistics domain standard (RESQML, WITSML, and PRODML has its own set of schemas, which include a <domain>common package (e.g., WitsmlCommon) of schemas for consistency across each ML. The underlying technology to define the schemas (XSD files) for the objects, artefacts, data, and metadata is XML, with HDF5 used for large numeric arrays (see Section 2.2.2). Each instance of a top-level data object must be identified by a universally unique identifier (UUID) (see Section 2.2.3).

Each domain ML leverages components of the Energistics CTA.

Energistics standards are designed using the Unified Modeling Language (UML), which is also used to produce the schemas and some documentation. For more information, see Section 2.2.1.

- **Energistics *common* schemas.** The *common* package is standardized across all Energistics domain standards; for each of the MLs, the same *common* package is included in the download. Like the Energistics domain schemas, these schemas are also XML XSD files. Data object schemas can be considered in these categories:
  - Mandatory (for example, AbstractObject, ObjectReference, objects related to units of measure (UOM), etc.).
  - Optional, available for use if wanted (for example, the Data Assurance and Activity Model objects).
  - Objects defined by Energistics specifications (see next bullet), which may be optional or mandatory, depending on the specification and domain ML.

  For more information on the *common* data objects, see Chapter 4.
- **Energistics specifications** describe objects and behaviors for handling mandatory and optional functionality across domains. For example, units of measure, metadata, and object identification are mandatory. Other standards, such as packaging objects together for exchange, are optional or ML-specific. Related data objects are implemented in the Energistics *common* schemas. The specs describe additional behavior requirements. For more information on the standards on which these specifications are based, see Section 2.3. For the complete list of CTA resources, see Section 1.4.2.
  - **Energistics Transfer Protocol (ETP)** is the Energistics specification that serves as the new application programming interface (API) for all Energistics domain MLs. Initially designed to replace the WITSML SOAP API, ETP is based on the WebSocket protocol. It delivers real-time streaming capabilities and is being expanded to provide CRUD (create, read, update and delete) capabilities. For more information, see Section 2.3.1.
- **Information technology (IT) standards.** Energistics standard's leverage existing IT standards (see Section 2.2) for various purposes. For example:
  - The Unified Modeling Language (UML) is used to develop the data model and produce the schemas and some documentation.
  - XML is used to define the data object schemas (XSD) and instances of data (XML files).

- UUID (as specified by RFC 4122 from the IETF) is used to uniquely identify an instance of a data object.
- HDF5 is used when needed as a companion to the XML data object to store large numeric data sets.

2.2 Information Technology Standards

This section lists and describes the information technology (IT) standards used in or as part of the Energistics CTA.

2.2.1 Data Modeling with UML and EA

The Unified Modeling Language™ (UML®) is a general-purpose modeling language used to design software and business process systems. Energistics implements UML using Enterprise Architect (EA), a data modeling software tool. The UML model has these uses:

- **Schema generation.** The schemas (XSD files) that developers use to implement Energistics standards into a software package are automatically generated from the EA model.
- **Visualization and communication**. Developers can explore the class diagrams to get an understanding of organization and relationships, and drill down on objects to get definitions in context. The UML model is save as an XMI file—a format that can be imported by any UML data modeling tool—which is included in the package when you download a standard from the Energistics website.
- **Documentation.** For convenience, the content of the UML model is also produced in a technical reference guide, with the objects organized alphabetically within the main EA packages.

2.2.2 File Formats: XML and HDF5

Energistics data objects are defined using XML, which is used because of its portability and ability for humans (as well as computers) to read and understand it. However, XML is not very efficient at handling large volumes of numerical or array data, so for this purpose Energistics uses the Hierarchical Data Format, version 5 (HDF5). Energistics has a standard pattern to provide a reference from the XML data object to associated HDF5 data.

2.2.2.1 XML

Each Energistics data object is defined by an XML schema definition (XSD) file, each of which is generated from the UML data model. For example, objects such as wells, grids, equipment, reports, etc. are defined by XSDs. Each instance of a data object is stored as an XML file.

Where possible, Energistics has established common design patterns, common types, and reference data which are implemented in the CTA or the individual domain standards (as appropriate).

These common patterns provide a rich set of integrated data objects for cross-domain workflows and make it possible for domain standards to share objects (instead of duplicating them). For example, WITSML defines wells and welllbores, which may be used by RESQML and PRODML.

2.2.2.2 Hierarchical Data Format 5 (HDF5)

HDF5 is a data model, a set of open file formats, and libraries designed to store and organize large amounts of data for improved speed and efficiency of data processing. Specifically, HDF5 provides:

- Machine/architecture-independent "binary" format (supported on Windows, Linux, etc. APIs are available in C++, Java, and .NET).
- Built-in data compression.
- Hyper-slabbing of array data so that sub-arrays may be extracted without reading the entire data file.

Example of Energistics use of HDF5:

- RESQML uses it for storage and retrieval of geometry and property data and multi-million cell models.

- PRODML Distributed Acoustic Sensing (DAS) data objects uses it for the huge arrays of both raw and processed data associated with DAS.

NOTE: HDF5 implementation on Windows can crash if the file system is full or fails. Before writing HDF5 files, check available space in the file system. The HDF5 group has been engaged, but no workarounds or solutions are available now.

2.2.2.3 Thread Safe Issues with HDF5

HDF5 on Windows is only thread safe if the thread option is built and then only if using the C language bindings (hdf5.dll).

All of the binding libraries for HDF5 bottom out using the C library. On Windows, only the stand-alone C library has locking, and their CMake options document that use of the C++ bindings library with locking version of the C library (1.8.6 and later versions) is not supported. Behavior of Hdf5DotNet with locking library is unknown. Behavior of other language bindings (Python, FORTRAN) is unknown.

2.2.2.4 For More Information on HDF5

For more information on HDF, including available tools and tutorials, see the HDF Group website at: http://www.hdfgroup.org/HDF5/. The HDFView tool is especially useful for visualizing and understanding the data stored in an HDF5 file.

- Additional links:
  - https://support.hdfgroup.org/HDF5/doc/
  - https://support.hdfgroup.org/HDF5/doc/UG/HDF5_Users_Guide-Responsive%20HTML5/index.html#t=HDF5_Users_Guide%2FDataModelAndFileStructure%2FThe_HDF5_Data_Model_and_File_Structure.htm
  - https://support.hdfgroup.org/HDF5/doc/UG/HDF5_Users_Guide-Responsive%20HTML5/index.html#t=HDF5_Users_Guide%2FLibraryAndProgrammingModel%2FThe_HDF5_Library_and_Programming_Model.htm

2.2.3 Universally Unique Identifiers (UUIDs)

To manipulate and exchange data objects independently, each instance of an Energistics data object requires a universally unique identifier (UUID).

Energistics uses UUID standard RFC 4122 from the Internet Engineering Task Force (IETF) (https://tools.ietf.org/html/rfc4122).

According to the abstract of the RFC: "This specification defines a Uniform Resource Name namespace for UUIDs (Universally Unique IDentifier), also known as GUIDs (Globally Unique IDentifier). A UUID is 128 bits long, and can guarantee uniqueness across space and time."

For UUIDs, Energistics standards are case-insensitive.

For more information about use of UUIDs by Energistics, see the *Energistics Identifier Specification*.

2.3 Energistics Standards

The standards listed in this section are based on existing industry standards and/or best practices, but have been tailored by the Energistics community to meet the specific needs of upstream oil and gas and data exchange.

- For a complete list of these specifications and other resources, see Section 1.4.2 (page 8).
- For a succinct list of the standards and links to relevant websites, see Appendix A (page 31).

2.3.1 Energistics Transfer Protocol (ETP)

Energistics Transport Protocol (ETP) is a new data exchange specification that enables the efficient transfer of real-time data between applications. ETP has been specifically envisioned and designed to

meet the unique needs of the upstream oil and gas industry and, more specifically, to facilitate the exchange of data for Energistics domain standards (RESQML, WITSML, and PRODML).

Initially designed to replace the SOAP API for WITSML, initial use cases were centered on moving real-time drilling and logging data between applications: from sensor to acquisition systems, to aggregators, to replication, and to client systems.

ETP is a series of feature notification mechanism, so data receivers do not have to poll for data and can receive new data as soon as they are available from a data provider. To achieve maximum use of the limited bandwidth available in upstream operations and to avoid blocking remote procedure calls, ETP is based on the asynchronous exchange of messages, which is fundamentally different from and far more efficient than the request/response pattern of the previous SOAP/HTTP.

ETP has been designed in a modular fashion, as a set of sub-protocols. Each of these protocols is designed to support a specific set of data workflows. The goal for ETP is for complete conformance (i.e., no optional behavior) at a sub-protocol level. In this sense, sub-protocols are similar to interfaces in object-oriented programming; they define a small, closely-defined unit of behavior that must be implemented.

Sub-protocols are also linked to styles of message transfer, message size, and structure. For example: there is one protocol for transferring real-time channel-oriented data; another protocol for transferring static business objects (wells, wellbores, reports, earth model elements, etc.) as XML strings; and another protocol for transferring large, heterogeneous, binary arrays (such as properties on a reservoir grid).

ETP uses the WebSocket communication protocol, JSON schemas to define messages, a subset of Avro for serialization, and JWT authentication.

2.3.2 Energistics Packaging Conventions (EPC)

EPC is an implementation of the Open Packaging Conventions (OPC), a widely used container-file technology that allows multiple types of files to be bundled together into a single package. Built on the widely used ZIP file structure and originally created by Microsoft, OPC is now an open standard supported by these standards organizations:

- Ecma International (http://www.ecma-international.org/publications/standards/Ecma-376.htm)
- *ISO/IEC 29500-2:2012*, which has 4 parts, which are all freely available at this link (near bottom of the page) (http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html).

An EPC file (or package) is a ZIP file, which may be “unzipped” to view its components. When implemented as part of an Energistics standard, the zipping/unzipping is done using the OPC libraries (per the *EPC Specification*).

However, any software tool that can read a ZIP file can be used to unzip and see the contents of an EPC file (file extension *.epc*). To open an EPC file with a ZIP tool, do the following: Select the EPC file, right-click with the mouse, and select the command to "Open with", and then choose your ZIP reader tool.

- For more information on how EPC works, see the *Energistics Packaging Conventions Specification*.
- For specific requirements and usage of EPC by individual domain standards, see the individual domain usage guides.

2.3.3 Coordinate Reference System (CRS)

Energistics domain standards use both global (projected) and local coordinate reference systems, which are implemented using data objects in Energistics *common*. For more information, see Section 4.7.

2.3.4 Energy Industry Profile of ISO Metadata Standards

For data object identification and traceability, Energistics standards uses key metadata, such as when a data object was created and updated and by what software. This and other key metadata are specified according to the *Energy Industry Profile (EIP) of ISO 19115-1:2014.* This EIP is an open, non-proprietary exchange standard for metadata used to document information resources, and in particular resources

referenced to a geographic location, e.g., geospatial datasets and web services, physical resources with associated location, or mapping, interpretation, and modeling datasets.

The EIP is an ISO Conformance Level 1 profile of the widely adopted international standards ISO 19115-1:2014 which provides XML implementation guidance with reference to ISO Technical Specification 19115-3:2016.

The goals of the EIP are to:

- Realize metadata standards and guidelines that enable stakeholders in the energy industry ("the community") to effectively and efficiently discover, evaluate, and retrieve a diversity of information resources from widely distributed repositories and collections.
- Support both proprietary data management needs and exchange of data between and within organizations.
- Leverage existing standards to encourage adoption within the community and integration into the business and exploit existing organizational resources needed for governance and long-term maintenance.

Implementation of the EIP into Energistics data objects is included in the current version of the schemas, for example, as the Citation element on the Abstract data object. For more information, see Section 4.1.

2.3.5 Energistics Unit of Measure Standard

The *Energistics Unit of Measure Standard* (*UOM Standard*) is a set of resources that defines a standard unit of measure (UOM) dictionary to promote consistent usage, data exchange, and unit conversions. The set includes the base Energistics Unit of Measure Dictionary and related documentation for creating, implementing, and maintaining a UOM dictionary that is patterned after the Energistics dictionary.

All implementations of Energistics standards must adhere to the UOM Standard, which is implemented in Energistics *common*.

2.3.6 Energistics Identifier Specification

The *Energistics Identifier Specification* describes the syntax and semantics of data object identifiers used in the Energistics family of data exchange standards, which includes universally unique identifiers (UUID) and Uniform Resource Identifiers (URI). The concepts of data object, asset, component, and feature identifiers are pervasive within the WITSML, PRODML, and RESQML specifications. Because Energistics is focused on exchanging data between software applications, having a standard way to identify data objects is crucial.

3 Key Concepts

This chapter explains key concepts common across all Energistics standards. Domain-specific concepts are explained in the respective ML usage guide.

3.1 Top-Level 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. Energistics refers to these as 'top-level data-objects', which inherit from AbstractObject and, by definition, each has a UUID and citation metadata (based on the *Energistics EIP)*.

For more information, see the *Energistics Identifier Spec*.

3.2 Specifying Relationships between Data Objects

Energistics domain standards use the standard XML hierarchical relationship, for example, where one data object is contained inside another. However, the nature of E&P data and workflows means there are many possible hierarchies or relationships that cannot be modeled in a single hierarchy.

To address these relationships, the CTA includes a data-object reference, which is explained in Section 4.11.1 (page 28) and the *Energistics Identifier Spec*.

Additionally, relationships are specified in the context of an Energistics Package, which is explained in the *EPC Specification*.

3.3 Activity Model

The purpose of the activity model is to capture:

- The activities (tasks or actions) that occurred to create and edit a subsurface model.
- How the activities relate to the data being exchanged.

Each top-level Energistics data object has metadata about its own history, for example, the date that it was created and last edited and by whom or what software. The purpose of the activity model is to provide additional context about "why" and "how" objects were created and changed, and the dependencies between the different elements of a model.

Energistics has basic mechanisms for defining activities and referencing affected data object(s). For more information, see Section 4.2 (page 20).

3.4 Data Assurance

The data assurance record declares conformance with a pre-defined data assurance policy of any data object being transferred using Energistics standards. The policies themselves do not need to be transferred along with the data (to do so would mean repeating the same policy definitions tens of thousands of times in a typical data transfer).

The data assurance record carries the policy ID of a policy (presumably a policy name or a number known to the receiver), a yes-or-no statement indicating conformance with the policy, the name of the person or software agent that determined conformance with the policy, and the date on which the conformance was determined. In addition, the data assurance record can also list which rules within the policy failed, resulting in a negative conformance.

For more information, see Section 4.8 (page 25).

4 The Energistics Common Package

A key component of the Energistics Common Technical Architecture (CTA) is the Energistics *common* folder, which contains data objects shared by all Energistics domain standards. Many of these data objects implement requirements from other Energistics specifications. For example, the design of the AbstractObject in *common* implements requirements from these Energistics specs: Identifier, EIP, and Unit of Measure.

For more information, see the *Energistics common Technical Reference Guide*.

NOTE: While the ultimate goal is to have all the latest published versions of the Energistics domain standards on the same version of Energistics common, the reality of different development and release schedules has prevented that from happening. When you download a version of a domain standard, that download includes the correct version of Energistics *common* for that version of the domain standard.

Additionally, each domain standard has its own common folder (i.e., WitsmlCommon, ResqmlCommon and ProdmlCommon), which provides consistency required for that particular domain. So domain data objects inherit from their domain common, and then from Energistics *common*.

Figure 4–1 shows the UML model of the *common* package, which is used to generate the XSD files. This chapter is organized by the order of the packages in the figure.

This chapter describes how the data objects in Energistics *common* package are intended to work when a domain standard is implemented in software. Classes and attributes defined in the UML model are converted into XML elements, types, and attributes in the resulting XML schema (XSD file), from where programmers can use proxy generators to create classes in their development environments. Because behavior is not specifiable in XML, the operations part of the class boxes is not used, and the UML model does not hold methods.

The resulting XSDs for Energistics *common* are included when you download any of the Energistics domain standards.

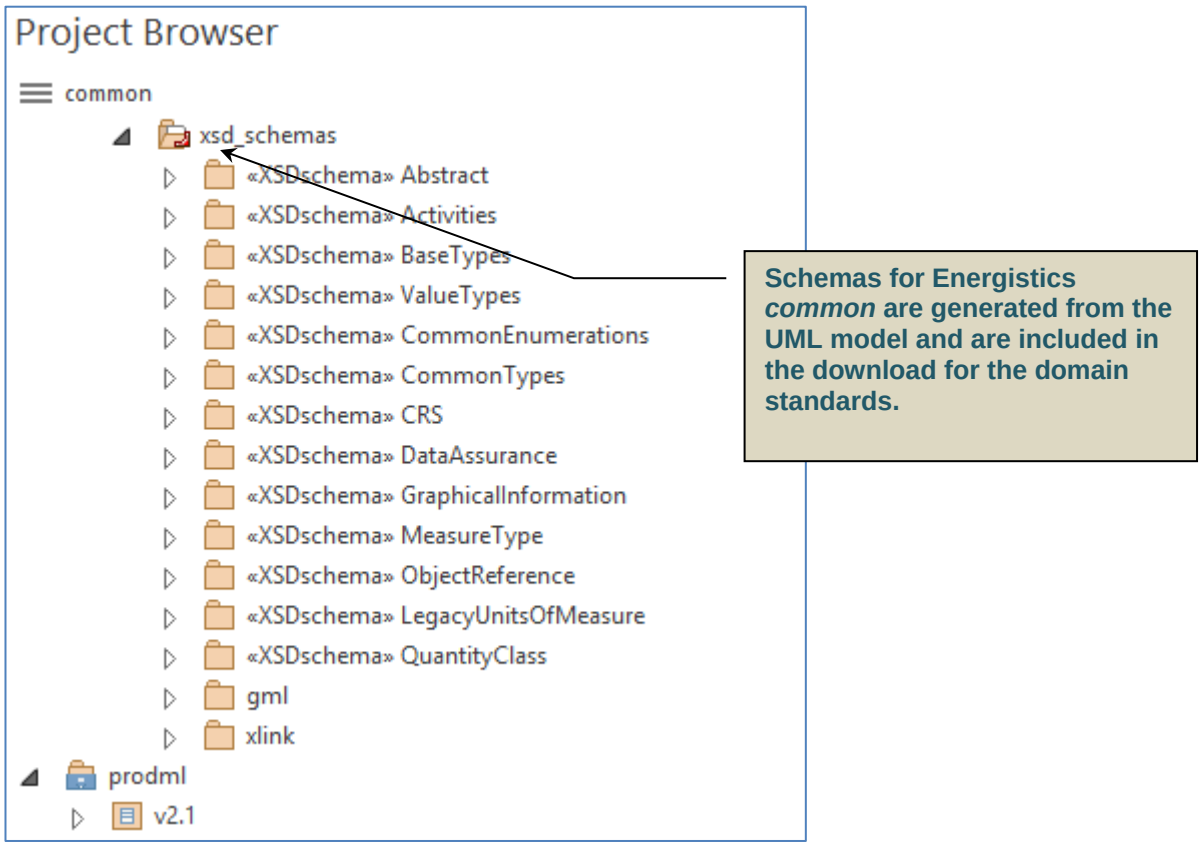

Figure 4–1. Each Energistics domain standard references the Energistics *common* folder, which contains shared data objects of the Energistics CTA. This screen shot shows *common* with the PRODML model.

4.1 Abstract

Data objects in the abstract package (**Figure 4**–2) are used as the roots of all global elements in Energistics' XML schemas.

Each domain standard inherits first from its domain common (as described above) and then from Energistics *common*. The functionality provided from the root-most object—AbstractObject—includes aliasing, simple extensibility, and summary authorship metadata to provide some object traceability.

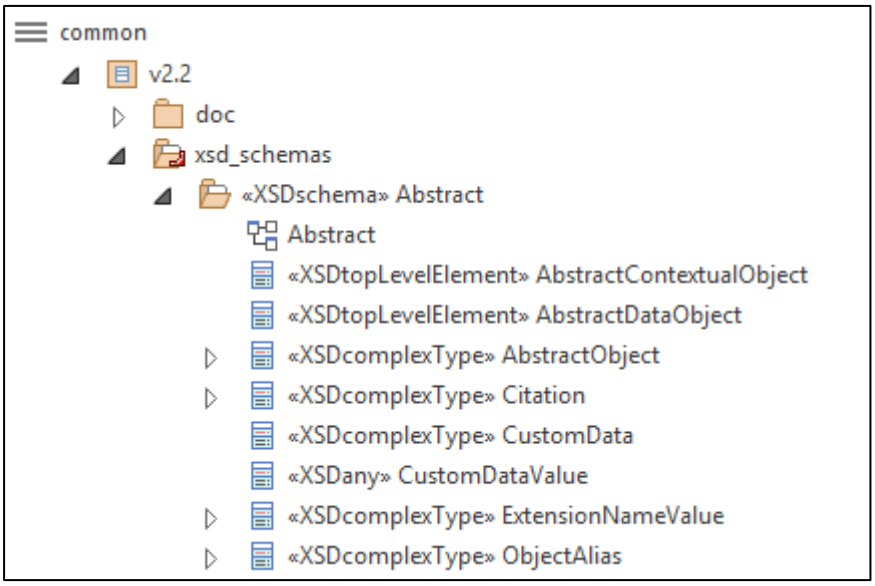

Figure 4–2. Abstract Class: data object class and elements (Energistics Common Technical Architecture).

4.1.1 AbstractObject

AbstractObject is the parent class for all top-level elements across Energistics standards.

AbstractDataObject is the substitution group for normative data objects.

AbstractContextualObject is the substitution group for contextual data objects.

4.1.1.1 SchemaVersion

Because a data transfer could potentially include XML objects from different versions of the standards, every top-level object must specify the version of the schema to which it conforms. It does this by setting the schemaVersion attribute inherited from AbstractObject.

4.1.1.2 Identifiers

Each top-level object in Energistics standards that conforms to the CTA is identified by its UUID attribute, which must be implemented as a GUID in the XML document. All objects inherit the UUID attribute from AbstractObject. For more information, see the *Energistics Identifier Specification*.

4.1.2 Citation

Use the citation data object to optionally add simple authorship metadata to any Energistics object. The citation data object uses attributes like title, originator, editor, last update, etc. from the *Energy Industry Profile of ISO 19115-1 (EIP).*

4.1.3 CustomData

XML objects can be extended at runtime using the custom data element. The custom data value elements are untyped in XML (using xs:any) so any type of data can be included at runtime.

4.1.4 ObjectAlias

Use the aliases attribute to create multiple aliases for any object instance. Note that an authority is required for each alias.

4.2 Activities

The activities data objects allow you to define activities and reference affected data object(s). Its purpose is to capture:

- The activities (tasks or actions) that occurred to create and edit an object, for example, a subsurface model in RESQML.
- How the activities relate to the data being exchanged.

4.2.1 How it Works

To describe an activity requires two parts; you must specify:

- **An activity template,** which is a general descriptions of possible activity types. A template is a semantic description of what the activity is about and the types of parameters that could be involved in the activity.

  For example, we can specify a template to describe the creation of any data object (as described above). The mandatory output for the template is one or more new data objects. The possible inputs are unlimited so that the template can accommodate the needs (potential complexity) of any data object in an earth model.

  A template may be very generic (for example, if the exporting software does not capture detailed descriptions of activities). Or the templates may be very detailed and precise in providing semantic information about the parameters involved in the activity.

- **An instance of an activity**. The instance describes an activity that has actually occurred. Each instance is associated with a template, which provides its semantics.

  As part of a RESQML transfer, a “writer” (software creating data for transfer) must include the most current templates along with the activity instances for the data objects contained in the data transfer. A “reader” uses the templates to understand the activity instances.

4.2.2 Data Object Organization

Figure 4–3 is a UML diagram of the activity model; the model elements are described below. These data objects can be found in the Energistics common package of the UML model.

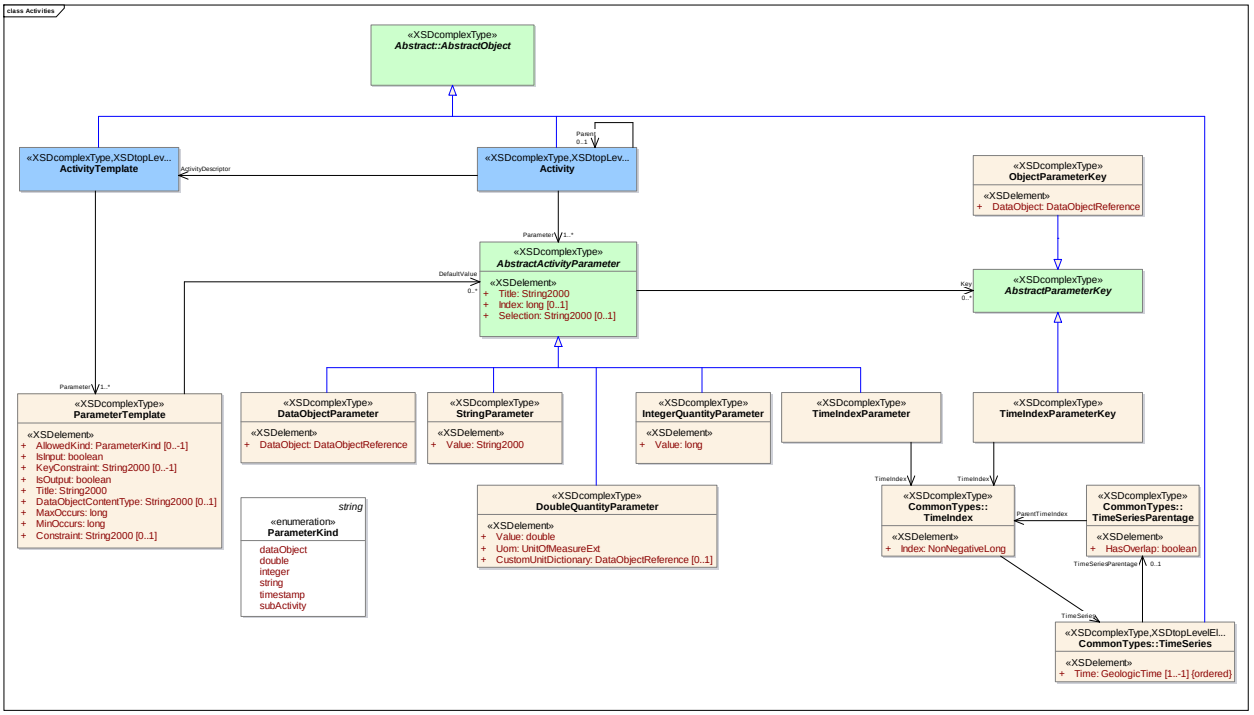

Figure 4–3. UML diagram of activity model data objects, which are described below.

4.2.2.1 Activity Template

An activity template provides the semantics of the activity. The *Title* (or name) provides the type of activity, for example, “GenericCreationActivity” (**Figure 4**–4). It also contains a list of parameter templates, which describe each potential parameter along with its role in the activity.

4.2.2.2 Parameter Template

For each parameter in the activity, describe its:

- Role provided by the parameter *Title.*
- Types associated with this parameter.
  - *AllowedKind* (optional) indicates the possible kinds for this parameter. See SubActivity below.
  - *DataObjectContentType* is used when the kind is limited to data objects and can also restrict the allowed object types.
- Use as input and/or output, based on *IsInput* and *IsOutput* information.
- Cardinality based on *MinOccurs* and *MaxOccurs* information (which are mandatory (1..-1) where -1 means infinite).
- Default value of the parameter.
- Additional constraint provided in free text form and targeted to be human readable.

4.2.2.2.1 SubActivity

When inside an activity, a parameter type is itself an activity and is a sub-activity of the main activity. This nested approach means we can create trees of activities, where complex process can be captured as an aggregation of smaller activities. In this case, the *AllowedKind* for the parameter is *subActivity*.

4.2.2.3 Activity

The activity object describes an activity that has actually occurred, which provides actual values to the parameters. An activity is a single implementation of the template it is associated to.

An activity contains:

- A link to the template it is instantiating.
- A list of actual parameter values.
- An optional parent activity, when the activity is a sub-activity.

4.2.2.3.1 Activity Parameters

Each parameter is represented by different objects according to the type of value it represents: DataObjectParameter, FloatingPointQuantityParameter, StringParameter, IntegerQuantity, TimeIndexParameter.

A parameter either: 1) stores a value or 2) references another object, for example, the DataObjectParameter. Its *Title* must match the *Title* of the corresponding *ParameterTemplate*.

To provide an optional textual description about the way the values have been selected for this parameter, use *Selection*. For example: “All wells” or “Porosity with maximum values greater than 0.05”.

4.2.2.4 Parameters with Multiple Values

When the cardinality of the associated *ParameterTemplate* is greater than one, then you must provide one value for each cardinality. Each value is provided as an individual parameter and each parameter of this collection must be individualized by one of these methods:

- an *Index*, used when the collection is the equivalent of a list.
- a *Key*, used when the collection is the equivalent of a map. The *Key* can also be a time index or a reference to an object.

4.2.3 RESQML Use Cases

The activity model was initially developed by the RESQML SIG, so the initial use cases are for RESQML.

Currently the main focus of the additional activity information is the software user: the goal is to capture human-readable information to provide users with more context, to help them make better decisions. (If developers can use the new information for software automation, that use is an added benefit, not the main purpose of the current version.)

Some examples of what the activity model can capture:

- Information indicating that a horizon surface is the result of an interpretation and then a “fit to well marker” process involving a limited set of wells. It can describe the two activities and the parameters used in this process, including the seismic volume on which the interpretation occurred for the interpretation and wells, and markers used in the “fit”.
- Information indicating that a reservoir grid property is the result of a geo-statistical simulation involving a limited set of well logs. It can describe the simulation and its input/output, including the well log properties, the simulation type, and numerical parameter values.

4.2.4 RESQML Example

This example (**Figure 4**–4) shows how to create an activity template named “GenericCreationActivity” which can be used to describe the creation of one or more data objects. The example also shows how to create an instance of the generic creation activity, in this case, a triangulated representation based on a 2D grid representation.

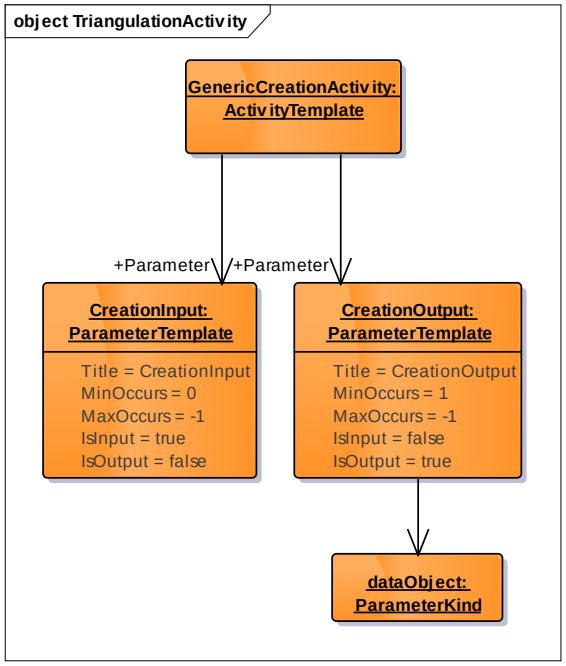

Figure 4–4. Template for a generic "creation activity" which can be used to create any data object.

The left box shows that this creation activity may have zero (see MinOccurs) to unlimited (see MaxOccurs) creation input parameters (see IsInput and IsOutput).

The right box shows at least one mandatory output (see MinOccurs, MaxOccurs, IsInput and IsOutput) which must be a data object (see the lower right box).

This activity template/description can then be used to describe the specifics of the creation of any data object. For example, in **Figure 4–5**, we created a triangulated representation based on 2D grid representation.

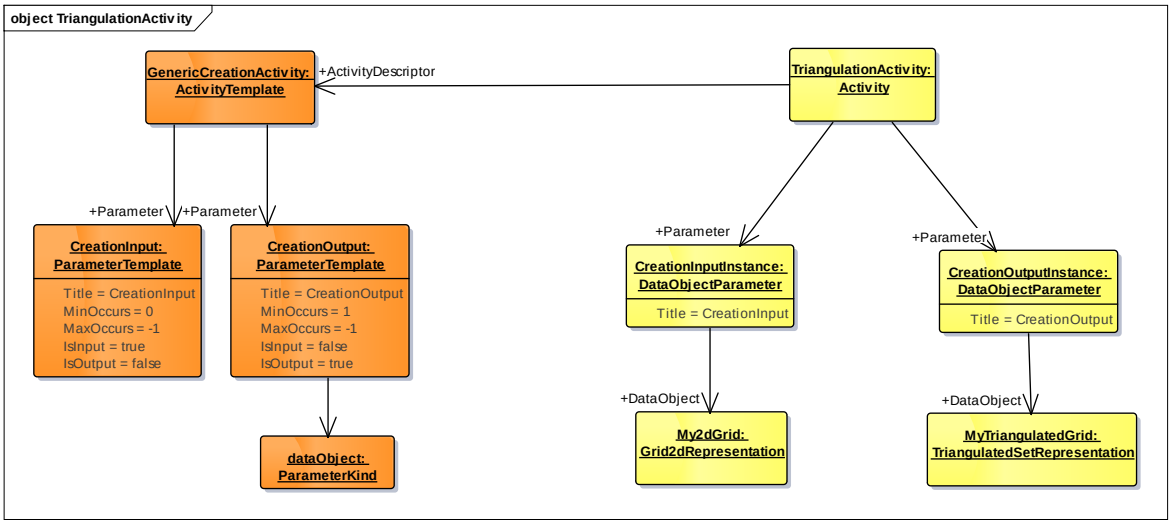

Figure 4–5. Example for a triangulated representation based on a 2D grid.

The triangulation activity (top yellow box) uses the creation activity template/description that we previously defined. This particular triangulation activity indicates that we have created a particular triangulated set representation (called MyTriangulatedGrid) from a particular 2D grid representation (called My2dGrid).

4.3 Base Types

The base types class (**Figure 4–6,** partial list) defines the intended abstract supertypes for the data types shown. Each is defined in the EA model and the technical reference guide. These types are specializations of normal XML schema datatypes with special purposes, for example, specific maximum lengths for string types. They provide consistency and protect consumers of the standard documents from potentially unlimited-length strings appearing in documents. In the case of UUID string, an XML regular expression pattern is applied so XML schema validation could be used to reject an improperly-formed UUID.

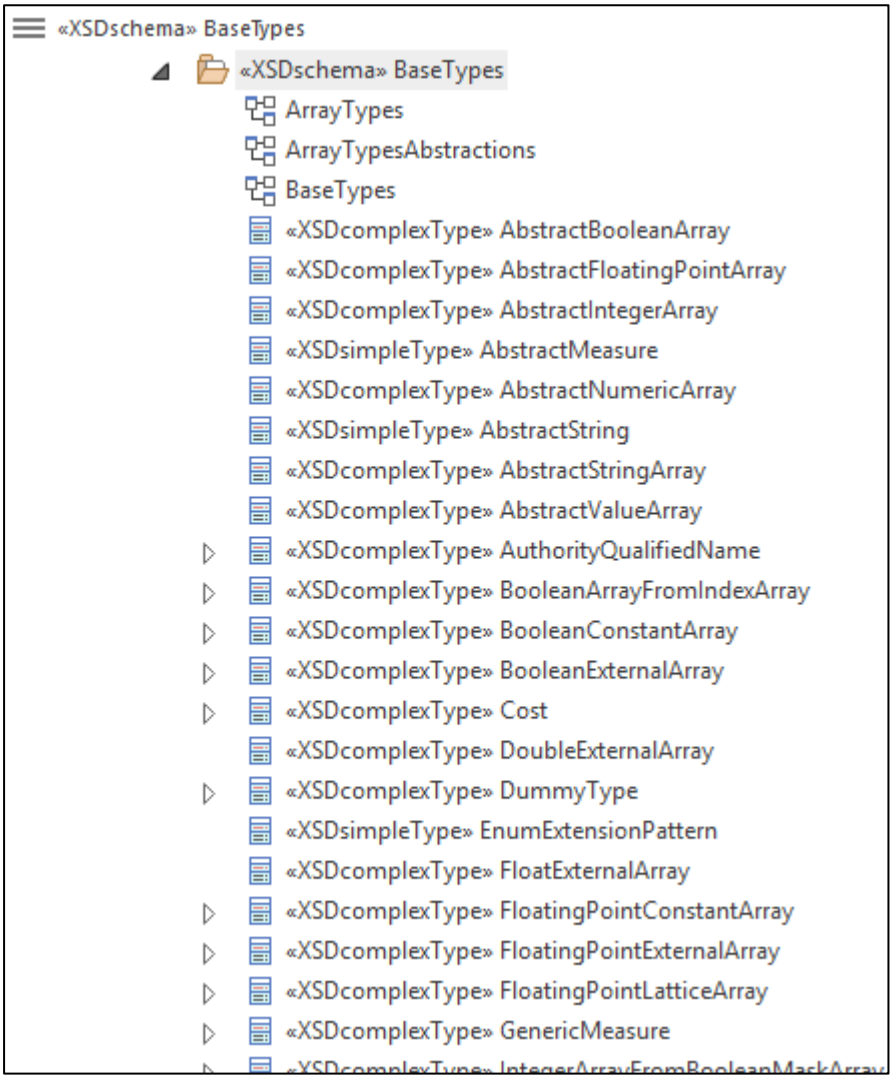

Figure 4–6. BaseTypes class (partial list).

4.4 Value Types

Some measurement values need to have certain measurement conditions included. For example, pressure and temperature can reference a standard condition. Or pressure measurements can be absolute or relative to another pressure (either gauge pressure or another value).
The value types package includes classes to define these types of relationships.

4.5 Common Enumerations

Enumerations available for use by all Energistics domain standards. For the detailed list, see the *CTA Technical Reference Guide* or the UML model.

4.6 Common Types

Common data types available for use by all Energistics domain standards. These types relate to specifying properties and time series. For more information, see the *CTA Technical Reference Guide* (or the UML model).

4.7 CRS

Common has a CRS package, which has these abstract classes: AbstractProjectedCrs, AbstractVerticalCrs, and AbstractGeodeticCrs. Each of these classes has five concrete children:

- A projection identified by an EPSG code.
- A projection not known to the EPSG, which is defined using OGC's Geographic Markup Language (GML).
- A projection that is intentionally obscured to anonymize a dataset. This is identified by a simple unvalidated string.
- A projection identified by a code according to a local authority. This would be used where a company or regulatory regime has chosen not to use EPSG codes.
- A projection identified by well-known text that complies with ISO 19162.

4.7.1 EPSG Codes

An Energistics schema can use EPSG codes for geolocalization as part of defining a CRS. The EPSG database includes geodetic parameters and assigned codes for easy reference to well-known global locations (http://www.epsg-registry.org/).

The EPSG codes database is maintained and published by the Geomatics Committee of the International Association of Oil & Gas Producers (OGP http://www.ogp.org.), which absorbed the now-defunct European Petroleum Survey Group (EPSG) (http://www.epsg.org/). The OGP is considered to be the single global source for positioning advice, guidance, and formats provision for the upstream oil and gas industry.

4.7.2 Geographic Markup Language

The OpenGIS® Geography Markup Language Encoding Standard (GML) is an XML grammar for expressing geographical features from the Open Geospatial Consortium (OGC) http://www.opengeospatial.org/standards/gml.

4.8 Data Assurance

The data assurance record (DataAssuranceRecord) declares conformance with a pre-defined data assurance policy of any data object being transferred using Energistics standards. The policies themselves do not need to be transferred along with the data (to do so would mean repeating the same policy definitions tens of thousands of times in a typical data transfer).

The data assurance record carries the policy ID of a policy (presumably a policy name or a number known to the receiver), a yes-or-no statement indicating conformance with the policy, the name of the person or software agent that determined conformance with the policy (the Origin), and the date on which the conformance was determined. In addition, the data assurance record carries can also list which rules within the policy failed, resulting in a negative conformance.

To "attach" the assurance record to the instance of the data object that it is declaring conformance for, use the data object reference (see Section 4.11 below).

The business case for data assurance points back to the old adage “garbage in, garbage out”. Energistics *common* provides a mechanism to convey additional data to support each end user’s own data quality objectives. **Figure 4–7** shows a UML diagram of the data assurance record and related data objects, which are described below.

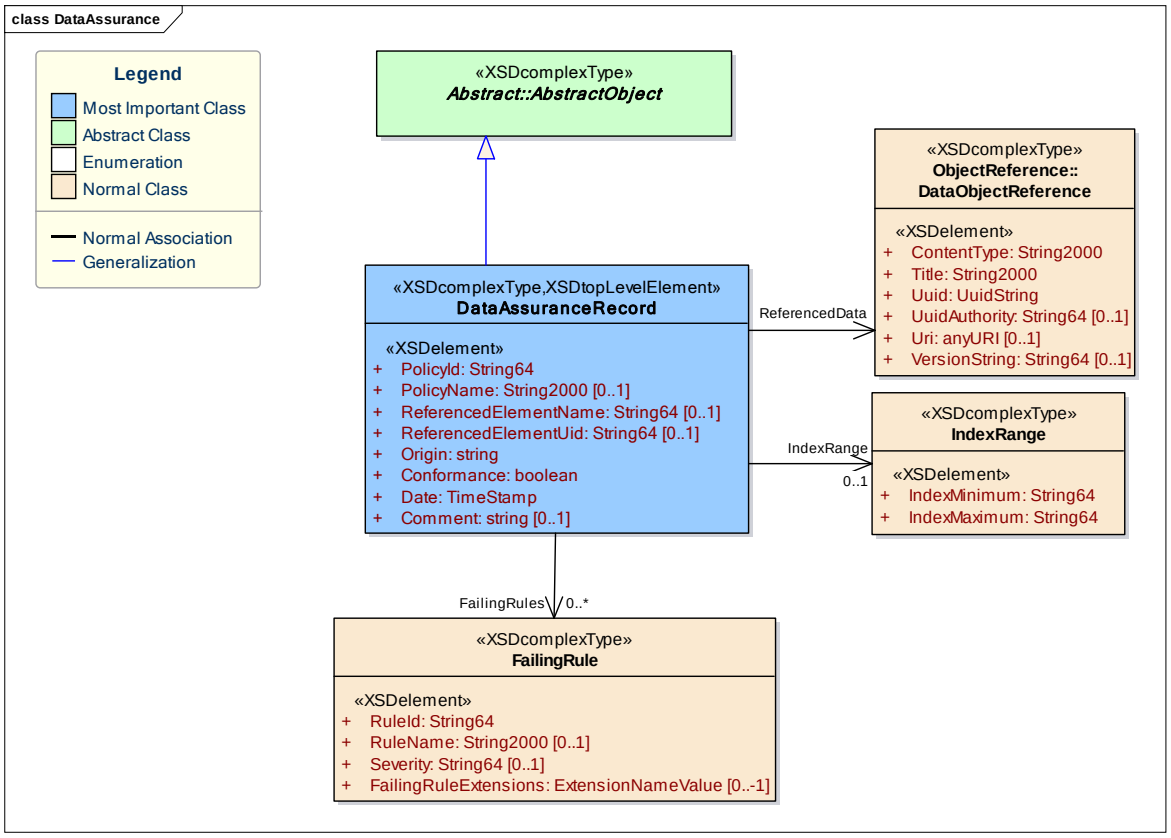

Figure 4–7. UML diagram for the data assurance record object and associated objects.

The main data object is the data assurance record. This XML document declares the conformance with a pre-defined policy of any given piece of data being transferred using an Energistics domain standard. The policies themselves do not need to be transferred along with the data (to do so would mean repeating the same policy definitions tens of thousands of times in a typical data transfer).

The purpose of the arrow pointing to abstract object is to establish a linkage from the data assurance record to any object contained in an Energistics (RESQML, WITSML or PRODML) transfer.

Note that Energistics standards do not determine whether the data are good or bad, nor do they carry information indicating that the data are good or bad. The data object carries information indicating whether the data in the transfer conform to the user’s policies or not. Energistics domain standards will always carry the information, but now the user can decide whether the failure of a portion of the data to conform to a desired policy renders it useless, or if it increases the uncertainty in the result, or if the policy non-conformance doesn’t matter at all.

4.8.1 WITSML Data Assurance Use Cases

The WITSML SIG did initial development of the data assurance object. The following list summarizes the main use cases that drove development.

1. **Support for data validation functionality.** Ability to detect data entries that violate the rules/valid values for a particular sensor/set of sensors so that applications can notify end users of any out-of-range data values.

2. **Provide information about precision.** Ability to provide metadata to consumers with information about the precision of data delivered. WITSML clients will then be able to communicate this information to users.
3. **Provide information about sensor accuracy.** Ability to provide metadata to consumers with information about the accuracy of the sensor providing the data. WITSML clients will then be able to communicate this information to users.
4. **Provide information about sensor calibration.** Ability to provide metadata to consumers with information about the calibration of the sensor providing the data (when, who, what, etc.). WITSML clients will then be able to communicate this information to users.
5. **Support for data quality in real time.** Ability to correct data live and propagate the correction or a warning in the stores/applications that have downloaded the original data.
6. **Support for data validity flag in real time.** Ability to set a live data quality flag and propagate this or a warning of the state in the stores/applications that have downloaded the original data.
7. **Support for data auditability and traceability.** When applicable, the ability to optionally receive auditability and/or traceability data generated from the point of origin to the end user.

4.9 Graphical Information

Package for classes to transfer graphical attributes (e.g., line style, color, etc.) of data objects. For future development.

4.10 Units of Measure

For details about how units of measure work in Energistics standards, see the *Energistics Unit of Measure Standard.*

This section provides an overview of the UOM-related classes in *common*.

4.10.1 Measure Types

A handful of attributes in the EA model (elements in the XML schema) use complex datatypes, which include a validated UOM as part of the attribute. These datatypes are in the measure type package.

4.10.2 Quantity Classes

A quantity class represents a set of units with the same dimension and same underlying measurement concept. For example, length is a quantity class.

Quantity classes are used to constrain items in the data model because the class defines all of the units that are allowed to be used with something that represents a specialization of that class.

4.11 Object Reference

This package defines mechanisms for data objects to reference one another and external files in the context of an EPC file (**Figure 4**–8).

- The data object reference (DataObjectReference) provides a way for instances of top-level Energistics objects to reference one another by their respective UUIDs.

  Examples of usage of data object reference:

  - In WITSML, it is used to create the well/wellbore/<object> hierarchy, where <object> can be a log, wellbore geology, or trajectory station.
  - In RESQML, it is used to create relationships between earth modeling data objects--faults, horizons, geobodies, grids, properties, etc.--to define models (not just independent objects).
- The EPC external part reference (EpcExternalPartReference) provides a reference from an instance of an XML data object in an EPC file to a system file where the actual data for that object is stored.

  Examples of usage of EPC external part reference:

- In RESQML and PRODML-DAS, it's used to reference HDF5 files that store numerical data and arrays associated with an XML data object.
- Could potentially be used to include a seismic survey in an EPC package; i.e., create an XML data object to identify the survey and then point to a SEG-Y file containing the actual survey data.

For more information, see the *Energistics Packaging Conventions (EPC) Specification.*

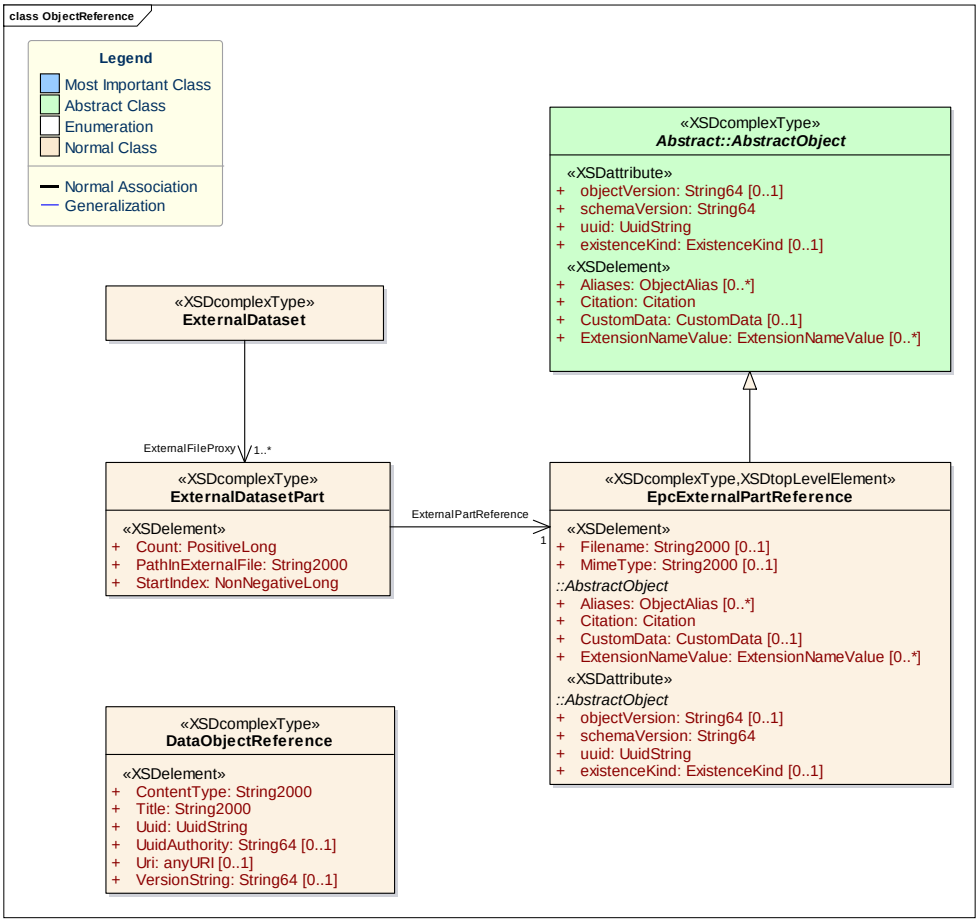

Figure 4–8. UML diagram of ObjectReference.

4.11.1 Relationship Mechanisms: Data-Object Reference

The data-object reference is a special mechanism in the schema that is used to specify relationships between and among Energistics data objects. The reference mechanism is specified in the data-object schema and includes:

- The UUID(s) of the data-object being referenced.
- The nature of the reference relationship. For example, one data object may "interpret" another data object or one data object may "represent" another data object. Possible relation types include:
  - Interprets
  - Represents
  - IsSupportedBy
  - isbasedOn

Because these relationships are specific, they are entered as relation names in the UML model in EA.

4.11.2 "Direction" of Data-Object References

The feature/interpretation/representation/properties knowledge hierarchy creates some special considerations for which data object specifies (or "holds") the reference. During the reservoir lifecycle, a feature can have many interpretations, an interpretation can have many representations, and a representation may have many properties. However, an interpretation cannot "know" how many future representations will be created, or their UUIDs. In contrast, when a user creates a representation, the user must know and specify which interpretation it "represents." For this reason, the child data object must specify (or hold) the relationship.

Here are the general rules on direction in data-object references:

- Parent-child:
  - In a one-to-one parent-child relationship, the data-object reference is from the parent to the child.
  - In a one-to-many parent-child relationship, the data-object reference is from the child to the parent.
- *isbasedOn,* where one data object is based on another data object defined at the same level. The data object reference goes from one class to another to express that the source object which is based on a target object needed to collect all the information gathered by the target object.

4.11.3 Example of DOR Usage from RESQML

In RESQML, the relationships between features, interpretations, representations and properties (informally referred to as the “FIRP”) are parent-child relationships held by the child(ren). **Figure 4**–9 shows an example, which is further explained in the text below. As explained in the previous section, the child data objects specify the relationship.

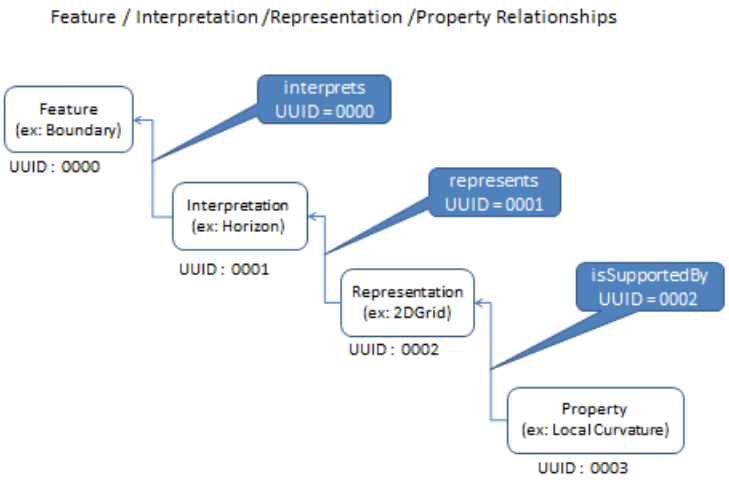

Figure 4–9. Example of relationships in RESQML for FIRP. Property "is supported by" (or provides values for) a representation; a representation "represents" an interpretation; an interpretation "interprets" a feature.

Each data object (except the feature) has a data object reference, which includes the UUID of the data object it references and the type of relationship. The relationships can be described as follows:

- Horizon 1 Interpretation *interprets* a genetic boundary feature of UUID= 0000.
- 2D Grid *represents* a horizon interpretation of UUID= 0001.
- A Local Curvature *isSupportedBy* (i.e., has numeric *values and is described within)* a 2D grid representation of UUID= 0002.

Figure 4–10 shows the addition of several children, which include a new interpretation, representation and property.

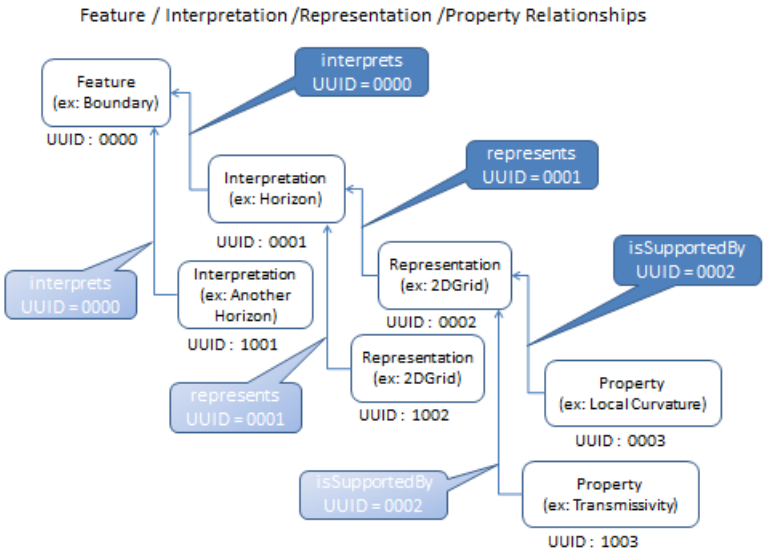

Figure 4–10. The previous example is extended to show multiple children with the addition of: another interpretation of the boundary feature (UUID 1001), another representation of the horizon interpretation (UUID 2002), and another property on the first representation (UUID 1003).

Appendix A. Standards Used by Energistics

The following table lists standards used in Energistics standards and the sponsoring organization for each standard.

Standards/Organization	Description of Use
XML Schema 1.1 XML Schema Part 1: Structures Second Edition http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ W3C- World Wide Web Consortium 28 October 2004	Used to define the schema that constrains the content of a PRODML XML document.
XML Schema Part 2: Datatypes Second Edition http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/ W3C-World Wide Web Consortium	Used to define the schema that constrains the content of a PRODML XML document.
Hierarchical Data Format 5 (HDF5) The HDF Group http://www.hdfgroup.org/	Open file formats and libraries. Designed to store and organize large amounts of array data, and improve speed and efficiency of data processing.
Geographic Markup Language (GML) Open Geospatial Consortium (OGC) http://www.opengeospatial.org/	The OpenGIS® Geography Markup Language Encoding Standard (GML). The Geography Markup Language (GML) is an XML grammar for expressing geographical features.
EPSG Codes International Association of Oil & Gas Producers (OGP) http://www.epsg.org/	The European Petroleum Survey Group (EPSG), the globally recognized experts on geodetic issues, has been absorbed into the Surveying and Position Committee of the International Association of Oil & Gas Producers (OGP), which is now the owner of the EPSG database of Geodetic Parameters and assigned codes. PRODML implementations can use EPSG codes to define a coordinate reference system.
Unified Modeling Language™ (UML®) Object Management Group http://www.uml.org/	UML is a general purpose modeling language, which was designed to provide a standard way to visualize system design. Originally intended for software architecture design, its use has expanded.
Energy Industry Profile of ISO/FDIS 19115-1	The EIP is an ISO Conformance Level 1 profile of the widely adopted international standards ISO 19115-1:2014 which provides XML implementation guidance with reference to ISO Technical Specification 19115-3:2016.
Open Packaging Conventions *Standard ECMA-376* *Office Open XML File Formats* http://www.ecma-international.org/publications/standards/Ecma-376.htm *ISO/IEC 29500-2:2012*	To address the challenges of the multi-file data sets used in upstream oil and gas, Energistics and its members have developed file packaging conventions based on the Open Packaging Conventions (OPC), a widely used container-file technology that allows multiple types of files to be bundled together into a single package. The Energistics Packaging Convention (EPC) is intended for use with all Energistics standards. OPC is supported by the two organizations listed in the left column.

Standards/Organization	Description of Use
Information technology – Document description and processing languages – Office Open XML File Formats – Part 2: Open Packaging Conventions *http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html*	
Internet Engineering Task Force IETF RFC 4122 https://tools.ietf.org/html/rfc4122	IETF RFC 4122 is a standard for defining universally unique identifiers (UUID). According to the abstract of the specification: "This specification defines a Uniform Resource Name namespace for UUIDs (Universally Unique Identifier), also known as GUIDs (Globally Unique Identifier). A UUID is 128 bits long, and can guarantee uniqueness across space and time." For Energistics usage, see *the Energistics Identifier Specification*.
IETF RFC 2234: https://tools.ietf.org/html/rfc2234 IETF RFC 3986: https://tools.ietf.org/html/rfc3986	IETF syntax and serialization specifications referenced and used by the *Energistics Identifier Specification* for Uniform Resource Identifiers (URI).
WebSocket Protocol https://tools.ietf.org/html/rfc6455	From the abstract: The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. It is the underlying protocol for the Energistics Transfer Protocol (ETP).
JSON http://www.json.org/ http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf	JavaScript Object Notation (JSON) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. Optionally, it can be used to encode ETP messages. ETP also supports binary encoding.
Avro Apache Avro specification (http://avro.apache.org/docs/current/spec.html)	The serialization of messages in ETP follows a subset of the Apache Avro specification. Avro is a system for defining schemas and serializing data objects according to those schemas.
JWT https://tools.ietf.org/html/rfc7519	JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. All ETP servers MUST support authentication using JWT.