Energistics
Common Technical Architecture
Overview Guide v2.3

CTA Overview	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: RESQML, WITSML, and PRODML The CTA is composed of Energistics *common* (EML), the set of data objects shared by all of the domain standards as well as other technologies and specifications adopted and/or developed by Energistics.
Version of Standard	2.3
Version of Document	1.0
Date published	May 16, 2022
Prepared by	Energistics
Abstract	This guide provides an overview of key concepts and functionality that comprise the CTA.
Document type	Overview guide of the
Language	U.S. English
Keywords:	standards, energy, data, information, process, reservoir model, shared earth model

Amendment History			
Standard Version	**Document Version**	**Date**	**Comment**
2.3	1.0	May 16, 2022	For a summary of changes, see *Energistics_CTA_v2.3RC_Release_Notes.pdf*.

Table of Contents

1 Introduction

This guide provides a brief introduction to the Energistics 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 data-transfer standards.

For more information about a particular domain standard, consult the documentation for that standard.

1.1 What are Energistics Data-transfer Standards?

Energistics has three flagship domain data-transfer standards, which correlate to the three main domains of upstream oil and gas:

- **RESQML** is for the reservoir life cycle, from initial structural earth modeling, to reservoir models, through production surveillance.
- **WITSML** covers the well and well construction, including drilling, completions, interventions, logging, etc.
- **PRODML** is for optimizing producing oil and gas wells, with a focus on activities that occur from the reservoir-wellbore boundary to the custody transfer point, which includes production monitoring, optimization, and reporting.

The core of each standard is a set of XML schemas that define the main data objects, artefacts, data, and metadata used in that domain. (Energistics has also begun the work to define data objects using JSON.) 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 the transfer of data between the many software applications, systems, and technology used in the extended upstream workflow. Developers implement the XML schemas (and related technology) into software so that the software can read and write data in this 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.

All Energistics standards and related resources are freely available 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 common 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.

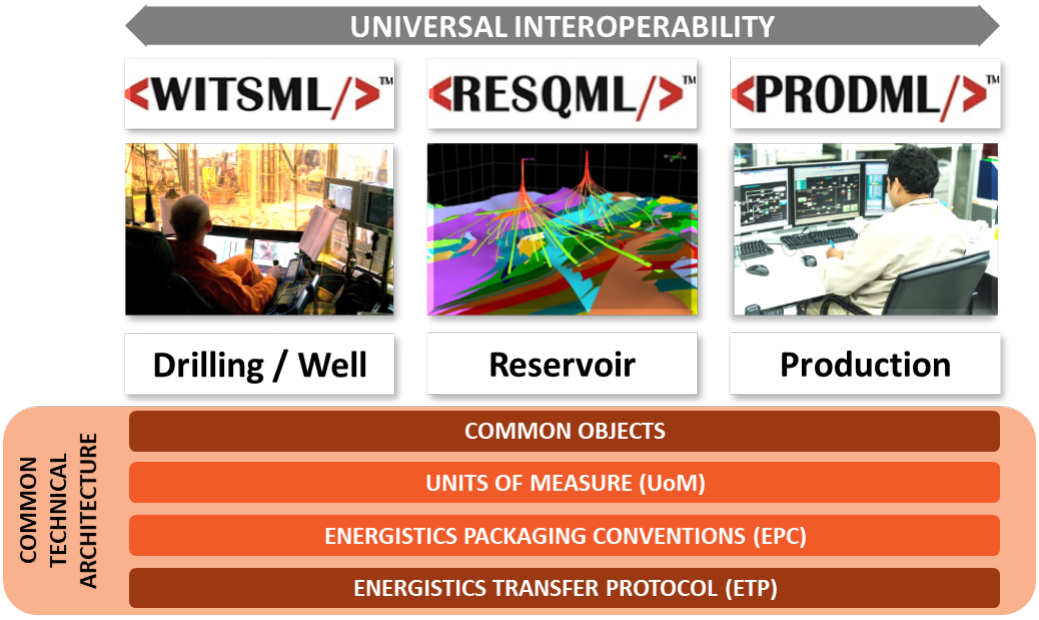


Figure 1-1. Energistics domain standards rest on a shared foundation of the Energistics Common Technical Architecture (CTA). This guide provides an overview of the CTA.

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 Resources in the Download Package

NOTE: All Energistics standards are freely available for download from the website; for more information see https://www.energistics.org/download-standards/.

With the final publication of these versions of the Energistics standards, you can choose from these options on how you want to download the standards:

- **The "traditional" ML-specific, where you download either WITSML, PRODML or RESQML with its appropriate version of Energistics *common*.** The current releases of the domain standards all use Energistics *common* v2.3. The download packages available are:
  - WITSML_v2.1.zip
  - PRODML_v2.2.zip
  - RESQML_v2.2.zip
- **All Energistics domain standards bundled together into a single download package, with the shared version of Energistics *common* v2.3.** (**NOTE:** This is how the standards were package for the public review that ran from Dec 2021 to March 2022.)
  - The download package is energyML.zip.

The download package includes a file named *Download_Package_ReadMe_File.pdf*. It contains a detailed list of Energistics CTA resources (schemas, documentation, etc.), which includes the resources included in the download package as well as the main resources available from the Energistics website (standards download page).

NOTE: Energistics uses a UML data modeling tool to specify the data models for its domain standards and Energistics common. Scripts in the UML tool are then used to produce the set of XSD files (schemas).

In many cases, UML diagrams are used in the documentation. Additionally, the UML data model itself is included in the download package. Energistics saves the UML model as an XMI file, a format that can be imported by any UML data modeling tool.

1.4.1 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.

1.4.1.1 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: Document-Internal	In general, 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. The ETP Specification v1.2 is the ONLY Energistics document that DOES use formatted text to indicate links.

1.5 OSDU Integration

As one of the first "non-oil-company" members of OSDU, Energistics and its members have been working with the OSDU Forum to ensure that Energistics standards are best leveraged in the OSDU standard and data platform. Additionally, with the current release of the Energistics standards, much work has been done to accommodate the OSDU data model in Energistics standards. For example, in cases where it made sense, elements and attributes in the OSDU model have been added to existing Energistics data objects. In cases where no appropriate Energistics data object existed, data objects named something like "OSDU Integration" have been added.

2 CTA Overview

Section 2.1 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). For maximum efficiency and interoperability, the Energistics community always tries to use existing information technology/industry standards. For example, data objects are based on XML and XSD schemas.
- **Energistics Specifications** (Section 2.3) are based on or leverage existing standards and/or industry best practices, but have been tailored to meet the specific needs of upstream oil and gas and related data-transfer.

NOTE: For a complete list of resources for the Energistics CTA, see *energyML_Download_Package_READ_ME.pdf* in the energyML download package, explained in Section 1.4.

2.1 CTA: Main Components and What They Do

Each Energistics domain standard (RESQML, WITSML, and PRODML) has its own set of schemas for the domain-specific data objects it defines. Each domain standard leverages components of the Energistics Common Technical Architecture (CTA).

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). Energistics standards are designed using the Unified Modeling Language (UML) implemented using a product called *Enterprise Architect (EA)*. EA is also used to produce the schemas and some documentation. For more information, see Section 2.2.1.

- **Energistics *common* schemas.** The Energistics *common* folder contains a set of schemas that is standardized across all Energistics domain standards. 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 data 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 Energistics *common* data objects, see the *Energistics common Technical Reference Guide v2.3*.
- **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 the list of and summary of Energistics specifications, see Section 2.3.
- **Information technology (IT) standards.** Energistics standard's leverage existing IT standards for various purposes. For context, Section 2.2 provides a brief overview of some key standards. For a list of the main IT standards, see Appendix: Standards Used by Energistics. Some examples of IT standards and how they are used include:
  - 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).

- 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 some of the key information technology (IT) standards used in or as part of the Energistics CTA. For the complete list of standards, see Appendix: Standards Used by Energistics.

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 from Sparx Systems (https://sparxsystems.com/). 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

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.

NOTE: Additionally, Energistics is working through how to define data objects using JSON. Both XML and JSON are expected to be supported in the future.

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 data object is stored as an XML file, formatted according to the XSD.

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.

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

2.3 Energistics Specifications

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

- For more information, see the referenced specification, which is available from the Energistics website. For more information, see Section 1.4.
- For a succinct list of the industry/IT standards and links to relevant websites, see Appendix: Standards Used by Energistics.

2.3.1 Energistics Identifier Specification (v5.0)

NOTE: Version 5.0 of this specification is a breaking change from the previous published version; it applies to the latest version of the domain standards published as release candidates in December 2021.

The concepts of business objects, data object, asset, component, and feature identifiers are pervasive within WITSML, PRODML, and RESQML. The *Energistics Identifier Specification*:

- Defines key terminology and concepts that are important to the identification of business objects, data objects, and dataspaces when used in Energistics standards.
- Specifies the syntax and semantics of data object and dataspace identifiers, as used in the Energistics Transfer Protocol (ETP).
- Specifies the syntax and semantics of data object references (DORs), an Energistics mechanism that allows a data object to reference other data objects and of data object component references (DOCRs), an Energistics mechanism that allows a data object to a reference a particular component in another data object.

When implementing any of the domain standards (with our without ETP), the syntax and rules specified in the *Energistics Identifier Specification* MUST be observed.

2.3.2 Energistics Unit of Measure Standard (v1.0)

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 transfer, 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*.

2.3.3 Energistics Transfer Protocol (ETP) (v1.2)

Energistics Transfer Protocol (ETP) is a communication protocol API that enables the efficient transfer of data between two software applications (endpoints), which includes real-time streaming. 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 in the Energistics family of data standards, which includes: WITSML (well/drilling), RESQML (earth/reservoir modeling), PRODML (production), and EML (the data objects defined in Energistics *common*). Initially designed to be the API for WITSML v2.0, ETP is now part of the Energistics Common Technical Architecture (CTA).

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 leverages existing IT standards and best practices—such as WebSocket communication protocol, JSON schemas to define messages, a subset of Avro for serialization, and appropriate security protocols to authorize connections.

2.3.4 Energistics Packaging Conventions (EPC) (v1.2)

NOTE: Version 1.2 of this specification is a breaking change from the previous published version; it applies to the latest version of the domain standards published as release candidates in December 2021.

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.

NOTE: Version v1.2 is a breaking change from the previous version of EPC.

2.3.5 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 the *Energistics common v2.3 Technical Reference Guide*.

2.3.6 Practical Well Log Standard (PWLS) (v3.0)

The Practical Well Log Standard (PWLS), which is stewarded for the industry by Energistics, categorizes the marketing names for logging tools and the obscure mnemonics used for the measurements, using plain English. PWLS provides an industry-agreed list of logging tool classes and a hierarchy of measurement properties and applies all known mnemonics to them.

PWLS is implemented in the Energistics domain data models (version 2.0+)—WITSML, RESQML and PRODML—through the Property Kind Dictionary (PropertyKindDictionary data object), which contains all known property kinds (element name = *PropertyKind*). The PropertyKindDictionary is published as part of Energistics *common* (namespace = EML).

Additionally, WITSML v2.1 provides new functionality around PWLS. For more information, see the WITSML v2.1 readme file (included with the WITSML download).

For more information about property kinds and PWLS, see Section 3.9.

3 Key Concepts

This chapter explains key concepts common across all Energistics standards. Domain-specific concepts are explained in the respective ML usage guide. For more information about the data objects referred to in this chapter, see the *Energistics common Technical Reference Guide v2.3*.

3.1 Data Object

NOTES:

- For all information related to data objects and identification, see the *Energistics Identifier Specification v5.0*.
- For information about AbstractObject, see the *Energistics common Technical Reference Guide v2.3*.

A **data object** defined by an Energistics specification is a valid document of the specified format (XML, JSON, other), which conforms to one of the schemas specified in the Energistics namespace (WITSML, RESQML, PRODML or EML (for Energistics *common*)) and inherits from AbstractObject, which is defined in Energistics *common*.

To ensure consistency, all Energistics data objects inherit from AbstractObject. For example, AbstractObject contains all required elements, such as UUID, schema version and EIP metadata citation.

NOTE: Informally within Energistics, a "data object" has also been referred to as a "top-level object" or "top-level element".

3.1.1 Data Object Identification and Traceability

- For identification, each Energistics data object must have a UUID.
- For traceability, Energistics uses EIP metadata, which includes information such as when a data object was created and updated and by what software. This information is included on the Citation element in AbstractObject.

For information on how the UUID and EIP metadata can be used together to manage data object identity during extended data transfer during an interactive session, see Appendix: Example Using UUIDs and EIP Metadata for Traceability to Help Manage Different Versions of a Data Object.

3.2 Units of Measure

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 transfer, 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*.

3.3 Specifying Relationships between Data Objects

In many upstream workflows the relationships between data objects are important. Energistics standards use several mechanisms that allow relationships between data objects to be specified. These include:

- XML construct of ByValue containment (which can be seen in the relevant UML diagrams).
- Energistics-defined mechanisms (data objects) in Energistics *common*, which include:
  - data object reference (DOR) (DataObjectReference). NOTE: The DOR design has changed in Energistics *common* v2.3. EXAMPLES: In WITSML, DORs are used to associate wellbores to their "parent" well; In RESQML, DORs are used to create relationships between earth modeling data objects--faults, horizons, geobodies, grids, properties, etc.--to define models (not just independent objects).

- data object component reference (DOCR) (DataObjectComponentReference), which makes it possible for a data object to reference a specific component in another data object (EXAMPLE: a specific trajectory station in a trajectory). NOTE: The DOCR is new in Energistics *common* v2.3.

For information on the definition and usage rules for DORs and DOCRs, see the *Energistics Identifier Specification v5.0*.

For information on how DORs may be used to navigate a data model using, see Section 3.16.1. Specifically for information on the "direction" of DORs, see Section 3.16.1.1. "Direction" of DORs, refers to which data object "points to" or "holds" the DOR, as shown in the next example from RESQML.

3.3.1 Example of DOR Usage from RESQML

In RESQML, the relationships between features, interpretations, representations and properties (informally referred to as the “FIRP” knowledge hierarchy) are constructed using DORs. **Figure 3-1** shows an example, which is further explained in the text below.

This 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.

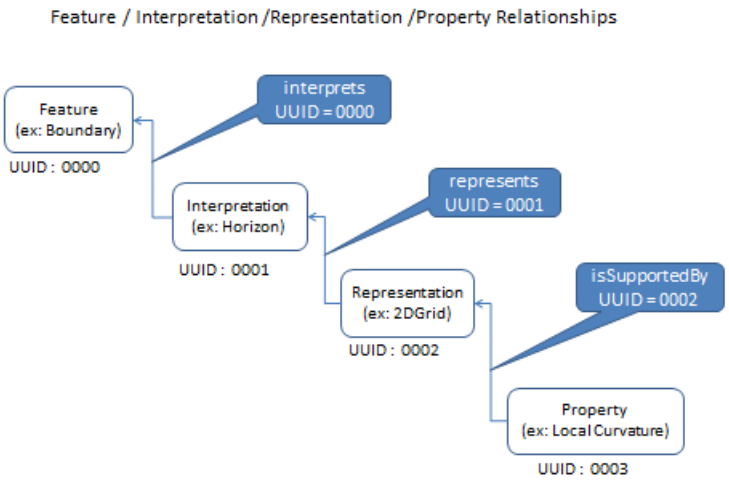


Figure 3-1. Example of relationships in RESQML for FIRP. Property "is supported by" (or provides values for) a representation; a representation "represents" an interpretation; an interpretation "interprets" a feature.

Each data object (except the feature) has a data object reference. 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 3-2 shows the addition of several "children", which include a new interpretation, representation and property.

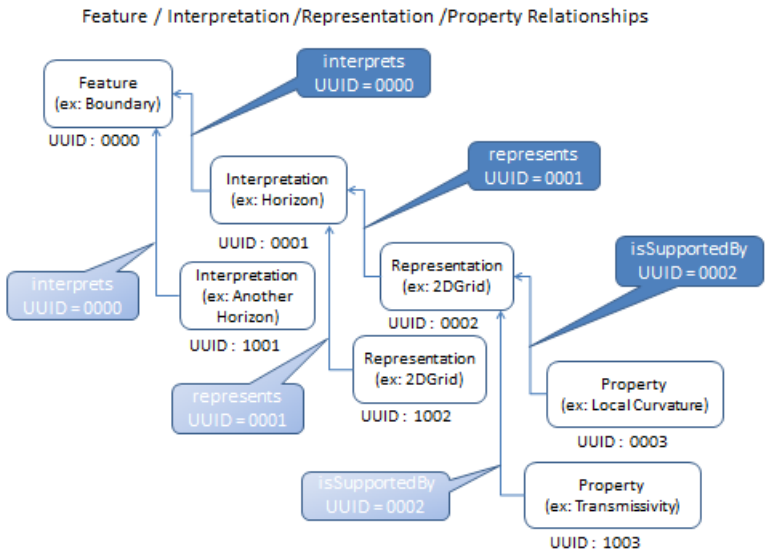


Figure 3-2. The previous example is extended to show multiple children with the addition of: another interpretation of the boundary feature (UUID 1001), another representation of the horizon interpretation (UUID 2002), and another property on the first representation (UUID 1003).

3.4 Aggregate

The Aggregate data object (Figure 3-3) makes it possible to aggregate an arbitrary set of data objects in a single XML document (file). This object is NOT INTENDED for use within an ML (e.g. a WITSML) data store, even though it is constructed similarly to the standard data object pattern. The anticipated normal usage is for collecting an aggregate of object messages for transport outside the context of an ML store.

Here's an example use case from WITSML that this data object supports: You want to email someone a well, its 2 associated wellbores, a trajectory, a BHA run, 3 logs, and 2 operations reports. Using the Aggregate data object, you can group all of these data objects together as one XML file and attach it to the email, instead of attaching 10 files (one each for the 10 data objects listed).

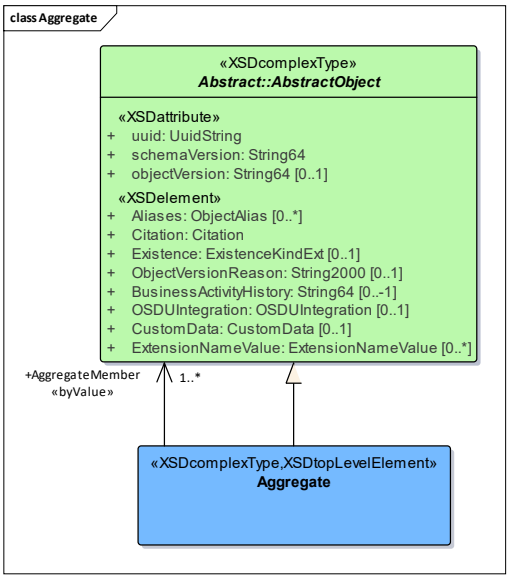


Figure 3-3. Aggregate data object.

3.5 Grouping Data Objects into Collections

A collection is a new concept in Energistics *common* v2.3. Specifically a DataObjectCollection (**Figure 3-4**) is itself an Energistics data object that allows you to create a grouping of other data objects and that grouping can be persisted. A collection may also be a collection of other collections. A collection is not a list; that is, the order does not matter.

The idea for this data object came from RESQML, where many types of data objects are grouped together for various aspects of earth modeling. The concept was generalized and implemented in Energistics *common*, so that it could be used by any of the Energistics domain standards.

A benefit of a collection is that it allows to create these groupings and enrich it over time without changing any of the data objects in that collection. You can also specify a collection kind (CollectionKind) as shown in the figure.

Rules for data object collections (see also the definitions in the *Energistics common Technical Reference Guide v2.3*):

- The associations in the collection are between the data objects in the collection and the collection data object itself.
  - The data objects within the collection may be homogenous or heterogeneous.
  - The data objects within the collection may be related to each other, which would be specified using the Energistics data object references (DataObjectReference) or ByValue containment, according to relationship rules defined in the Energistics domain data models, for specific data objects.
- The relationship of the data objects in the collection to the collection data object are specified and managed in SingleClassAssociation.
  - Using this association class (instead of data object references from the data object to the collection) means that creating associations does NOT modify data objects; that is, it does NOT modify the collection nor the data objects associated to the collection.
  - The SingleClassAssociation also makes it possible to optimize operations like cascading deletes (EXAMPLE: Makes it possible to implement cascading deletes for a collection).
- Use the CollectionsToDataObjectsAssociationSet to associate collections into a collection.

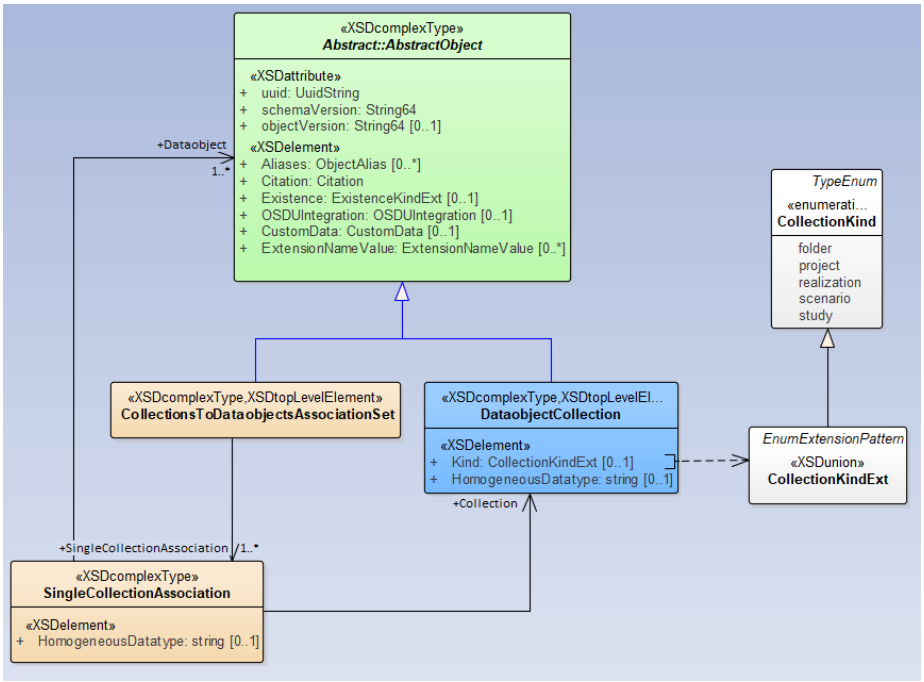


Figure 3-4. DataObjectCollection UML diagram.

3.6 Attaching a File to a Data Object

The Attachment data object (**Figure 3-5**) allows you to attach a file to any Energistics data object. The attachment can be any kind of file (i.e., spreadsheet, PDF, TIFF, etc.) and it may be attached to the data object or to a specific relevant index within a data object. For example, it the Attachment is to a wellbore, it may be attached to a specific measured depth. If the data object is a grid (or other relevant representation from RESQML) a file may be attached to one of the allowed indexable elements shown in the figure.

The file content is contained inside the Attachment data object, where you specify the file type (MIME type), the file name or optionally a URI (if appropriate). The file content must use Base64 binary encoding.

The Attachment data object was originally defined in WITSML but now has been removed. The design has been improved to support more use cases and moved to Energistics *common* for use by all the domain standards.

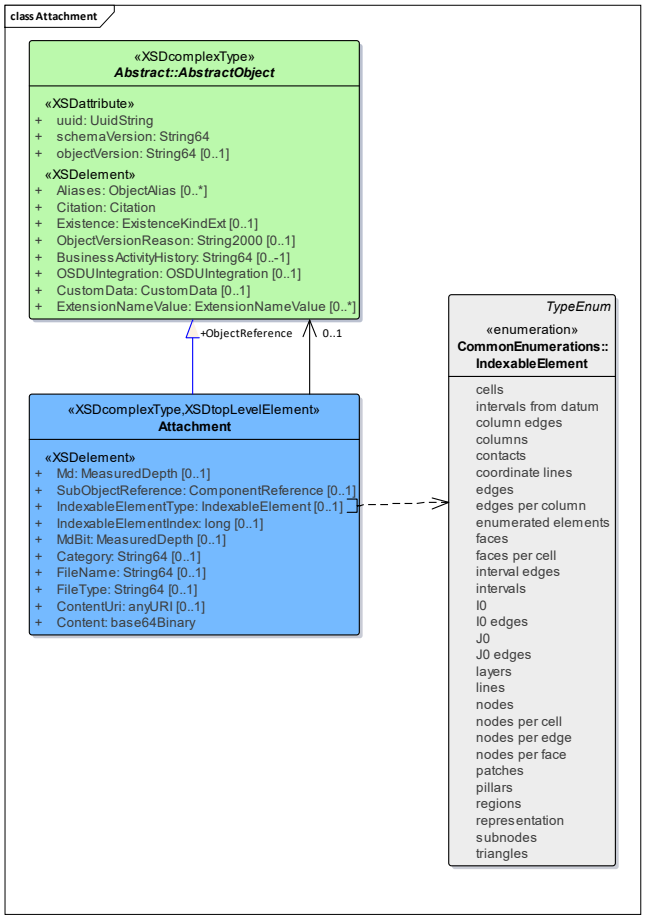


Figure 3-5. Attachment data object.

3.7 Column-Based Table

A column-based table (ColumnBasedTable, **Figure 3-6**) allows the exchange of tables, where the values are arranged against columns that are defined by PropertyKind, UOM and Facet. EXAMPLES: KrPc table and facies tables.

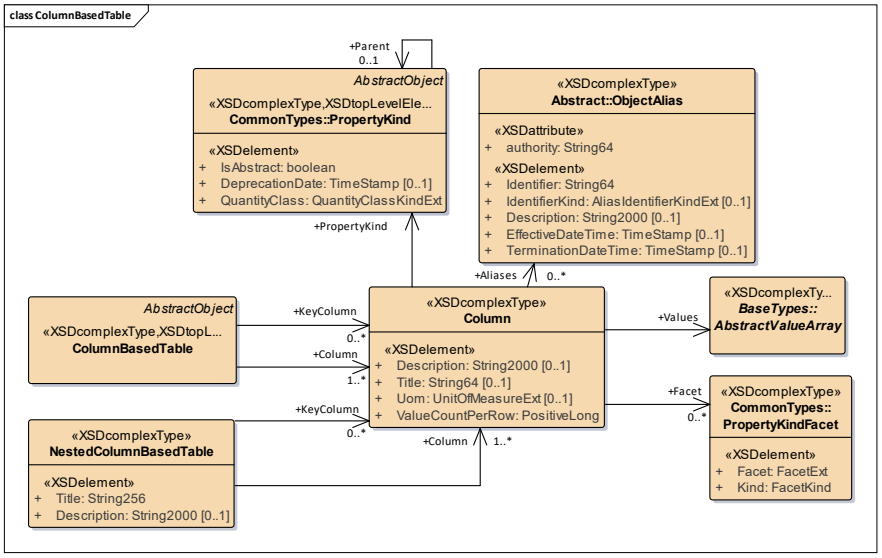


Figure 3-6. ColumnBasedTable data object.

A ColumnBasedTable can be one of two types; either a ColumnBasedTable or a NestedColumnBasedTable. Each column (Column) in the table must be defined. Some columns are "keys" used to identify the row in a table. Other columns are values, which could be arrays or single scalar numbers. Each column is a particular property of data.

3.8 Simple XML Arrays

New data types have been added in BaseTypes package (see the ArrayTypes diagram) that allow the exchange of very simply arrays in XML, without having to use HDF5 (which is how complex arrays of data must be exchanged in Energistics domain standards; see Section 2.2.2.2).

What constitutes a "simple array" is not specifically defined; implementers/users should use their best judgement. New data types include:

3.9 Property Model/PWLS

The Practical Well Log Standard (PWLS) categorizes the obscure mnemonics used for oilfield data and relates them to the marketing names for logging tools that make those measurements using plain English. PWLS provides an industry-agreed list of logging tool classes and a hierarchy of measurement properties and applies all known mnemonics to them.

In general, the main goal of PWLS is to support the classification of well log property measurement data—usually referred to as *curves* or *channels*—that are commonly used by oil and gas companies, and to use this classification as a tool to support general queries over large populations of channels.

The data model in PWLS makes it possible to do queries such as "give me all the gamma ray logs" and have a store return all gamma ray logs (channels), from all vendors, regardless of the variety of vendor mnemonics used to identify gamma ray data.

As the basis for this categorization of channel data PWLS provides an industry-agreed upon standard hierarchical list of property names, which are used for consistency and validation of those names.

PWLS is integrated into Energistics domain data transfer standards WITSML, RESQML and PRODML, through Energistics *common* as the Property model (**Figure 3-7**) in the CommonTypes class. Each PWLS curve is defined as a property (PropertyKind). The Property Kind Dictionary (see Figure 3-7**)** is a container XML file of all the properties in PWLS. The Property Kind Dictionary is included in the energyML download file in the folder, v2.3→common→ancillary→ PropertyKindDictionary.xls (for more information see the energyML download package read-me file).

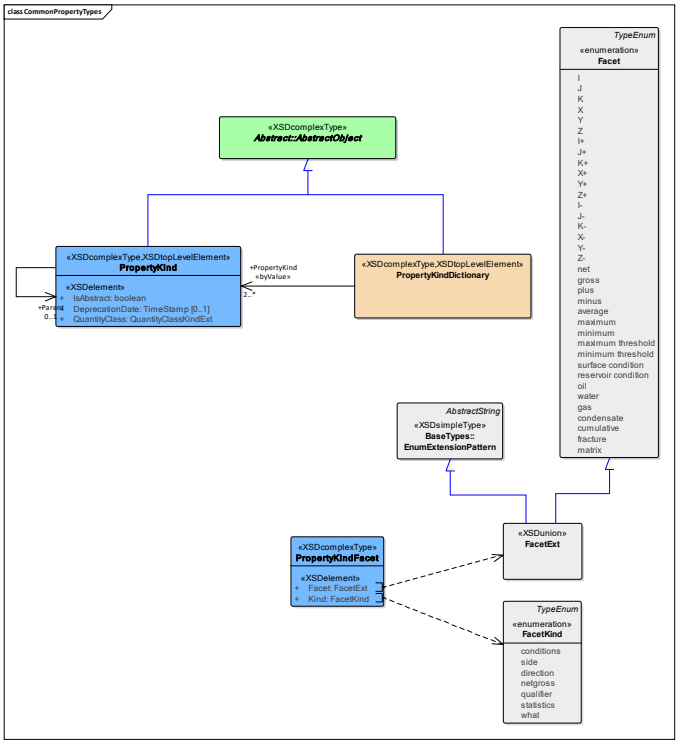


Figure 3-7. Energistics Property model.

The availability of the Property Kind Dictionary supports many possible implementation use case. For example, in the RESQML data model, representations (such as grids) can be used to specify specific property values for supported property kinds. So the properties can reference a property kind in the Property Kind Dictionary.

WITSML v2.1 specifies a new PWLS class that leverages the property, curve and tool data in PWLS (including 2 new dictionary files for channel kinds and logging tool kinds) to support new workflows. For example, data managers can maintain a library of ChannelKind and LoggingToolKind data objects to track the kinds of tools and channels they have received data for from logging companies.

For more information on the WITSML v2.1 implementation, see the WITSML v2.1 Release Notes included in the WITSML folder of the energyML download package.

3.10 Spatial Location (CRS and Datums)

For oil and gas operations, the accuracy of spatial information is crucial. With the release of Energistics *common* v2.3, coordinate reference systems and datums have been completely redesigned. For information, see Chapter 4.

3.11 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 3.11.1.

NOTE: The Activity model was initially developed by RESQML, but has been promoted to Energistics *common* so it can be used by any of the Energistics ML domain standards.

3.11.1 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.

3.11.1.1 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.

3.11.1.2 Data Object Organization

Figure 3-8 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.

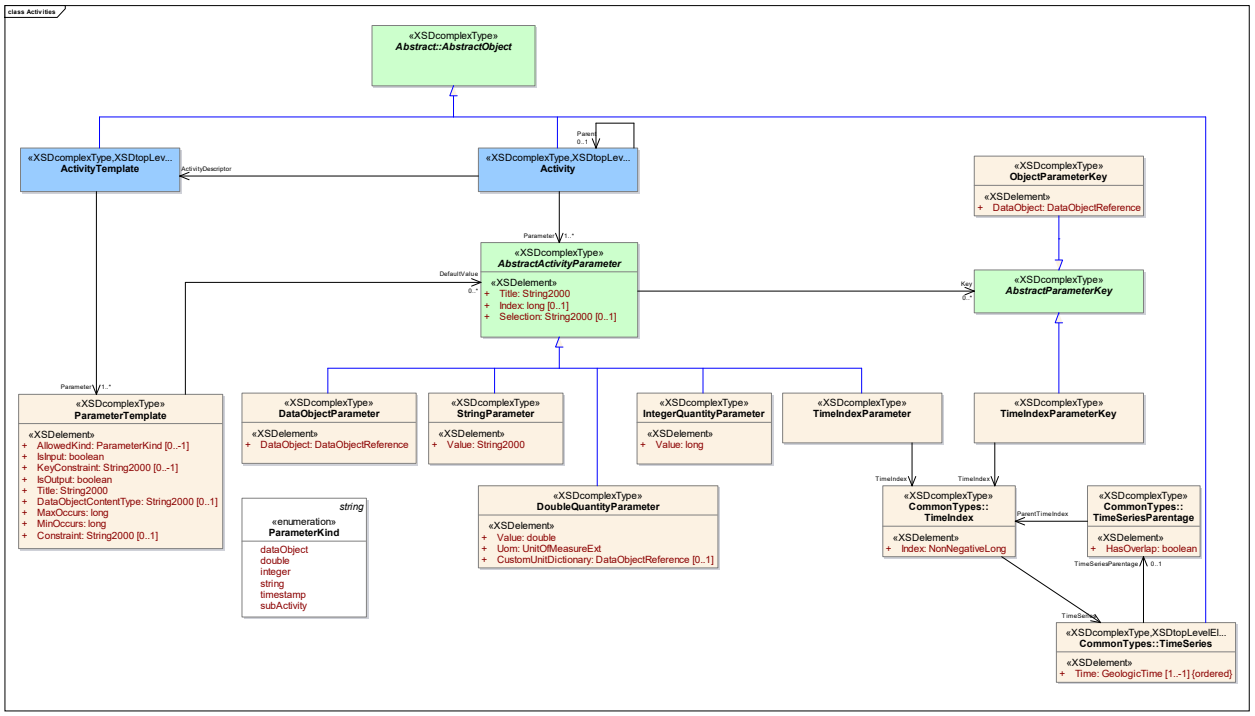


Figure 3-8. UML diagram of activity model data objects, which are described below.

3.11.1.2.1 Activity Template

An activity template provides the semantics of the activity. The *Title* (or name) provides the type of activity, for example, GenericCreationActivity (**Figure 3-9**). It also contains a list of parameter templates, which describe each potential parameter along with its role in the activity.

3.11.1.2.1.1 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.

3.11.1.2.1.2 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*.

3.11.1.2.2 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.

3.11.1.2.2.1 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”.

3.11.1.2.3 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.

3.11.1.3 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.

3.11.1.4 RESQML Example

This example (**Figure 3-9**) 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.

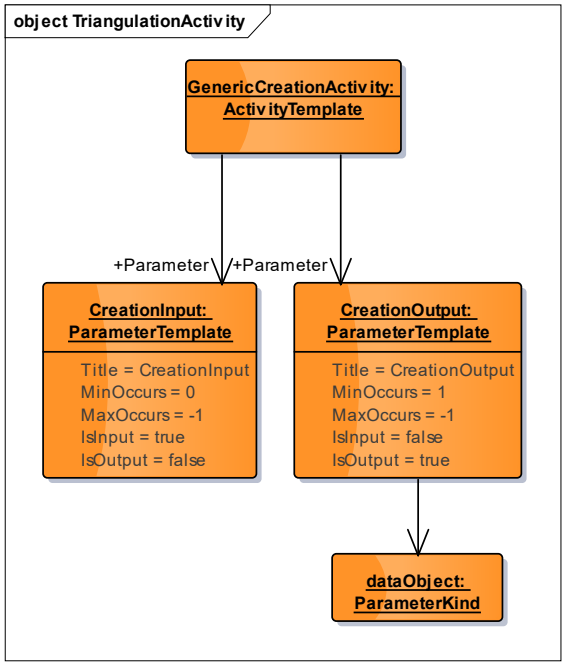


Figure 3-9. Template for a generic "creation activity" which can be used to create any data object.

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 3-10**, we created a triangulated representation based on 2D grid representation.

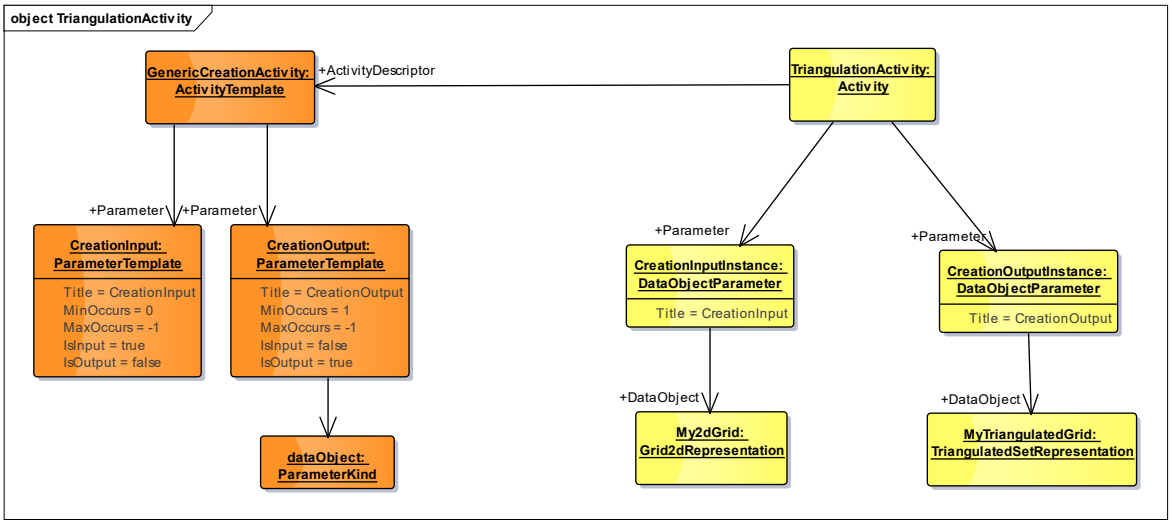


Figure 3-10. Example for a triangulated representation based on a 2D grid.

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).

3.12 Data Assurance

The business case for data assurance points back to the old adage “garbage in, garbage out”. 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 can also list which rules within the policy failed, resulting in a negative conformance.

Figure 3-11 shows a UML diagram of the data assurance record and related data objects, which are described below.

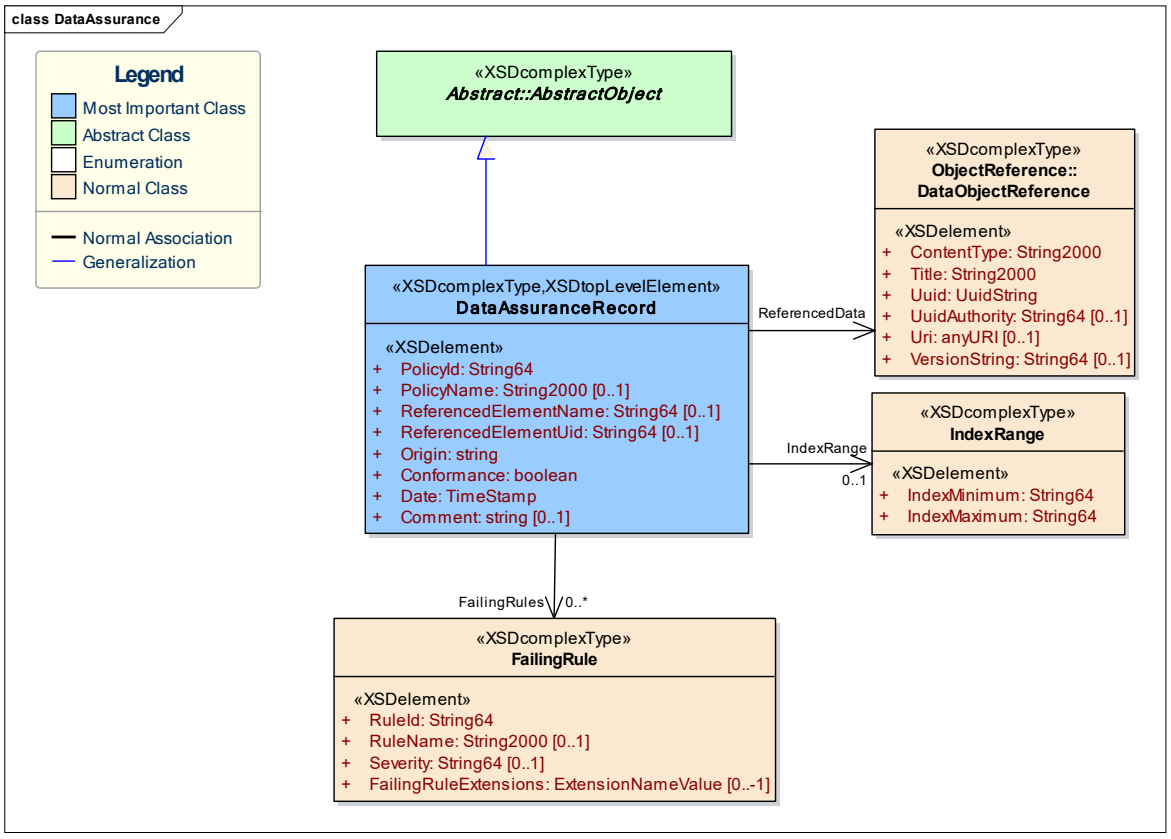


Figure 3-11. UML diagram for the data assurance record object and associated objects.

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 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. The relationship between a data object and a data assurance record is established using an Energistics data object reference (DataObjectReference) (which is describe in Section 3.3).

Note that Energistics standards do not determine whether the data is good or bad, nor do they carry information indicating that the data is good or bad. The data object carries information indicating whether the data in the transfer conforms 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.

3.12.1 WITSML Data Assurance Use Cases

The WITSML SIG did initial development of the data assurance data 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.

3.13 Graphical Information

The GraphicalInformation package in Energistics common contains the abstract data object from which any detailed graphical information models can be defined. Currently, only RESQML (v2.2) defines a graphical information model (set of data objects). For more information, see the *RESQML Technical Usage Guide*.

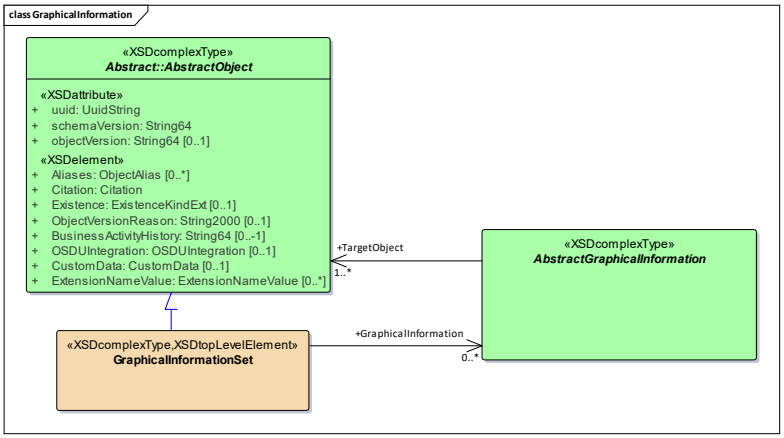


Figure 3-12. Graphical Information package.

3.14 Custom Data

While Energistics data models are detailed and rich, many organizations have data that is not included in the standard data model. Thus, Energistics provides mechanisms to add custom data, which are shown in **Figure 3-13**.

GENERAL RULE FOR CUSTOM DATA: The receiving endpoint MUST accept the custom data. It MAY issue a warning (e.g., "I don't know what this is.") but it must accept it.

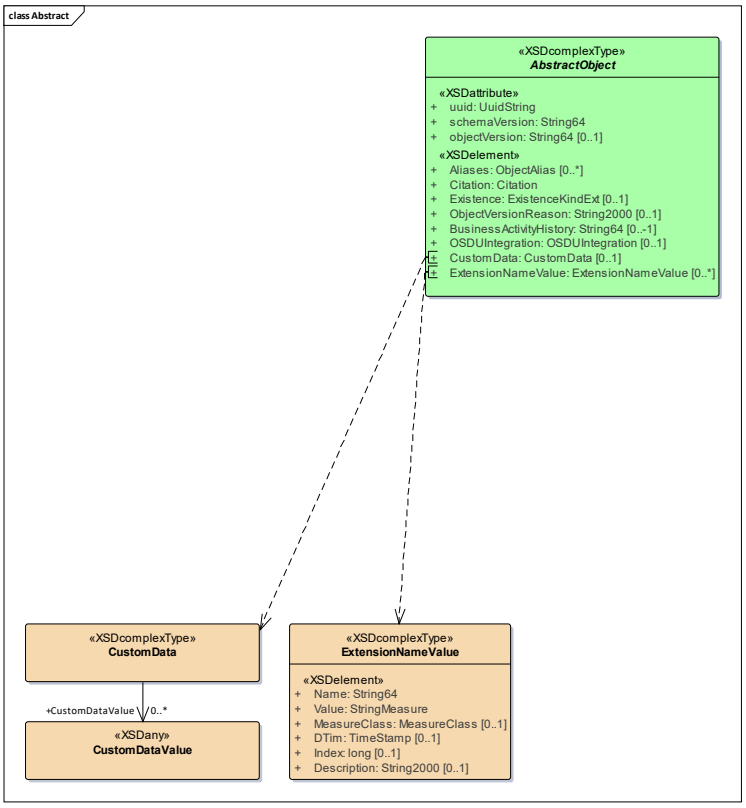


Figure 3-13. Energistics provides mechanisms to add custom data including CustomData and ExtensionNameValue elements (diagram elements from the Abstract diagram in the Abstract class).

CustomData allows you to add custom data fields to existing data objects using name:value pairs. ExtensionNameValue allows you to extend standard ML domain "named" extensions without having to modify the schema.

NOTE: If you want to reference a data object through data that is not part of an Energistics data model (i.e., data objects that are defined by CustomData), you should use a DOR block inside CustomData. You SHOULD NOT use extra metadata to specify the relationships.

Throughout the data model are constructs that allow you to extend enumerated lists, as the example shown in **Figure 3-14**.

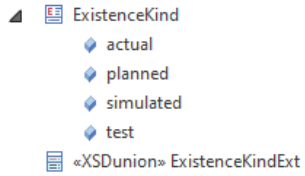


Figure 3-14. The ExistenceKindExt allows an implementation to add custom values to an enumerated list. This construct is used throughout the Energistics data models where extensions are allowed.

3.15 Jagged Array Object

Originally developed by RESQML, the jagged array object has now been moved to Energistics common (BaseTypes) so it is available for all MLs. This “jagged array” construction is used to store irregular array data. This type of a data structure appears in many programming languages where it is sometimes called a “list of lists” or an “array of arrays”. This construction uses a pair of arrays. The “elements” of the array stores all of the values while the “cumulative length” stores the offsets. Specifically, the offset is the cumulative count of elements to the end of that portion of the array. It is implicit that the offset to the beginning of the first element is zero. The differences in offsets may also be used to determine the length of each elemental array.

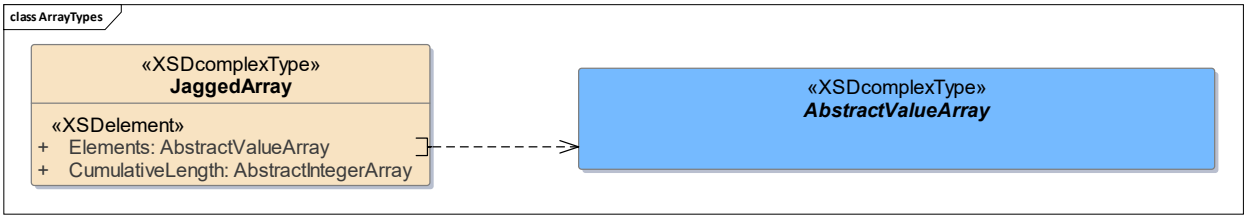


Figure 3-15. Jagged array object.

For example, to store the following three arrays as a jagged array:

(a b c)
(d e f g)
(h)
Elements = (a b c d e f g h)
Cumulative Length = (3 7 8)

3.16 Data Model as a Graph

NOTE: This content was originally published in the *ETP Specification v1.2*. It has been replicated here as a general concept that people implementing Energistics standards with ETP should be aware of. It also explains how data object references (Section 0) can be used and understood.

Discovery (Protocol 3) in ETP v1.2 has been developed to work with data models as graphs. This section provides a general definition of a graph and how it works, and identifies key concepts, which are used as inputs for a customer to formulate a discovery request.

IMPORTANT! When understood and used properly, these inputs allow customers to specify precisely the desired set of objects in a single request, thereby reducing traffic on the wire. Conversely, if the graph concepts are not understood and not used properly, related operations will be highly inefficient.

A graph is a mathematical structure used to model pairwise relations between objects (https://en.m.wikipedia.org/wiki/Graph_theory). In this context, a graph is made up of *nodes* (which are also called *points* or *vertices*) and the *lines* (also called *links* or *edges*) that connect the nodes (Figure 3-16).

In some instances, the "direction" of the lines in the graph (which node points to another node) is not relevant; these are referred to as *undirected graphs* (Figure 3-16, left). In other graphs, the direction is important; these are referred to as *directed graphs* (Figure 3-16, right).

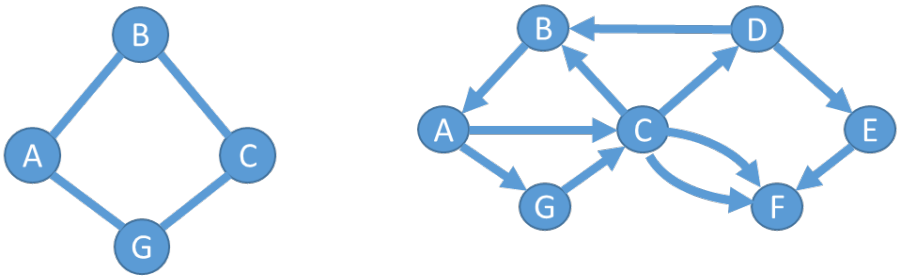


Figure 3-16. Examples of graphs: left image is an undirected graph and right image is a directed graph.

3.16.1 Energistics Data Models

For Energistics:

- Nodes represent data objects in a data model (WITSML, RESQML, PRODML or EML (i.e., Energistics *common*) (For the definition of data object, see the *Energistics Identifier Specification v5.0.*).
- Lines (directed links between nodes) represent relationships between those data objects. A data object can have multiple distinct references to other data objects (as specified in the various domain models).

For example, a wellbore may reference the well it is in and may reference multiple channels of data and/or channel sets of data collected about the wellbore. Or a 3D grid may reference hundreds of properties, and reference the faults and horizons used to derive the grid structure.

In some instances, a graph may have an obvious structure, for example a graph can be a tree or hierarchy, such as the traditional well, wellbore, log hierarchy. But in other cases, such as earth and reservoir modeling, objects may be in many-to-many relationships, so often there is no obvious "structure" or pattern (beyond the relationships). Figure 3-17 shows how a set of data objects and the relationships among them form a directed multigraph.

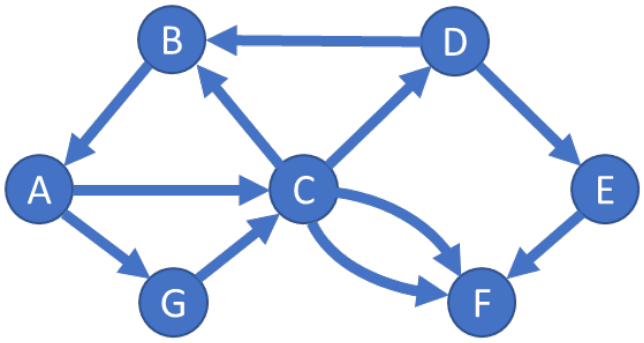


Figure 3-17. A set of data objects and the relationships among them form a directed multigraph.

3.16.1.1 Links and Relationships: Sources and Targets

The "direction" of links in some Energistics data models is relevant and is used in the Discovery protocol. Figure 3-18 shows a directed link between nodes A and C, which represents a relationship between data object A and data object C.

Figure 3-18. Node C is the "target" of the directed link from A to C; node A is the "source" of the directed link from A to C.

In graph theory terminology:

- Data object A is the “source” of the directed link from A to C.
- Data object C is the “target” of the directed link from A to C.
- By extension, data object A is the “source” of the relationship between A and C.

Relationships in Energistics data models are specified using 2 main constructs:

- eml:DataObjectReference (DOR)
- ByValue containment

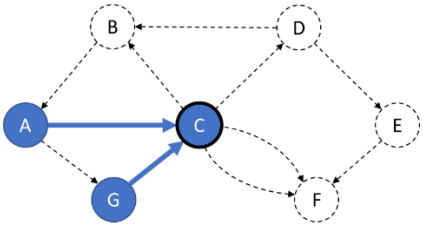
In general, these rules apply for specifying sources and targets:

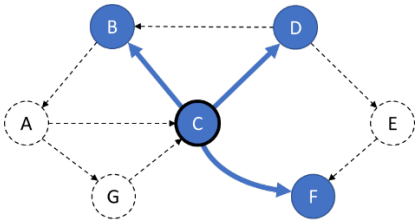
- For DORs, the data object that contains the DOR is the source, and the object that it "points to" is the target. **RECOMMENDATION:** DORs should be to one or more data objects in the SAME dataspace. Technically, DORs do not prohibit referencing a data object in another dataspace (i.e., a URI to the data object can be specified). However, some aspects of ETP functionality (e.g., notifications) may not work as designed.
- For ByValue containment, a contained object is the source, and the "container" is the target.

NOTES:

1. The “source” of a relationship between two data objects may be ML-specific.
2. For more information on these topics, see the relevant ML's ETP implementation specification.

Figure 3-19 is a more complex example of sources and targets relative to node C.





Sources are nodes with directed links **to** C.
Nodes A and G are sources of C.
C is the target of these relationships.

Targets are nodes with directed links **from** C.
Nodes B, D and F are targets of C.
C is the source of these relationships

Figure 3-19. More examples of targets and sources relative to node C.

3.16.1.2 Types of Relationships: Primary and Secondary

Energistics defines these types of relationships between data objects:

- **Primary**: Primary relationships are the “most important” relationships between data objects. For example, primary relationships may be used to organize or group data objects, such as organizing Channels into ChannelSets or organizing ChannelSets into Logs. In some cases, all relationships between data objects are important so all relationships are primary. Common characteristics of a primary relationship:
  - One end of the relationship is mandatory; that is, one object cannot exist (as a data object in the system) without the other. In the above example: A ChannelSet cannot exist without at least 1 Channel.
  - Relationships where one data object "contains" one or more other data objects (i.e., a by-value relationship), indicated with the ByValue construct in XML, such as ChannelSets containing Channels.
- **Secondary**: Secondary relationships are “less important” relationships between data objects and may provide additional contextual information about a data object to improve understanding. For example, the reference from a Channel to a Wellbore. Common characteristics of a secondary relationship:

  - Both ends of the relationships are optional.
- **GENERAL RULE:** ML-specific rules determine whether a relationship is primary or secondary. The ML rules to apply are determined by one of the data objects in the relationship.
  - In the case of relationships based on DORs, the type of relationship is determined by the ML of the data object that has the DOR.
  - In the case of by-value relationships, the relationship type is determined by the ML of the data object containing other data objects by value.

    EXAMPLES:
    A WITSML v2.0 ChannelSet contains WITSML v2.0 Channels by value and a ChannelSet has a DOR pointing to a WITSML Wellbore. So the WITSML rules determine whether the relationships between ChannelSet and Wellbore and ChannelSet and Channel are primary or secondary.
    A RESQML v2.0.1 obj_WellboreFeature has a DOR pointing to a WITSML Wellbore, so the RESQML rules determine whether the relationship between an obj_WellboreFeature and a Wellbore is primary or secondary.
- **NOTES:**
  1) For the ML-specific rules on which relationships are primary and which are secondary, see the ML-specific ETP implementation guide.
  2) Future versions of the ML data models will label the type or relationship (Primary or Secondary).

4 Spatial Location (CRS and Datums)

The oil and gas industry is fundamentally a spatial industry. The high-level goal: Locate oil and gas reserves below the earth's surface (sometimes miles below) and determine the best place on the earth's surface to place facilities and equipment (e.g., wells, offshore platforms, etc.) to most efficiently and economically bring those reserves to the surface. This means that the industry is concerned with location both:

- On the earth's surface.
- Below the earth's surface, with one particularly important use case being a location along a wellbore.

For oil and gas operations, accuracy of spatial information is crucial.

The Energistics data model for spatial location is based on models and best practices from the geodesy discipline, and best practices and adjustments defined and required for upstream oil and gas operations and Energistics data transfers.

This chapter provides an overview of the Energistics data model design and how it works. For the list of data objects and definitions, see the *Energistics common v2.3 Technical Reference Guide*.

NOTE: The UML diagrams are large and detailed, so won't easily fit on a page. For easy reference in this chapter, relevant portions of diagrams are included in the text. To easily see an entire diagram, use the XMI file that was delivered in the energyML download package. The diagrams referenced below are in the common→doc folder.

NOTE: In previous versions, Energistics standards supported use of OpenGIS® Geography Markup Language Encoding Standard (GML). That support has now been deprecated.

4.1 Terminology

This section provides some simple definitions to start. Subsequent sections in this chapter further explain how they are used in the data model.

Term	Definition
coordinate reference system (CRS)	A CRS is a coordinate-based local, regional or global system used to locate geographical entities or manmade structures on or in relation to the earth's surface or subsurface. A CRS commonly defines a specific map projection, as well as transformations between different CRSs. From: https://en.wikipedia.org/wiki/Spatial_reference_system A CRS refers to the way in which spatial data that represent the earth's surface (which is round/3 dimensional) is flattened so that you can "draw" them on a 2D surface. However, because each method uses a different (sometimes) mathematical approach to perform the flattening, it results in different coordinate system grids. These approaches to flattening the data are specifically designed to optimize the accuracy of the data in terms of length and area. From: https://www.earthdatascience.org/courses/earth-analytics/spatial-data-r/intro-to-coordinate-reference-systems/ For more information, see Section 4.4.
position	A place where something is located or has been placed/installed. For example, a well surface location is a position. For more information, see Section 4.3.
reference point or datum	A designated position from which the position of other objects/artefacts are specified. EXAMPLE: On an offshore platform or in pad drilling, a surveyed point may be identified (such as a corner of the platform or drill pad) and the position of wells (or other equipment) are given relative to that reference point; for example, 10 ft north and 30 ft east of the reference point.

	For more information, see Section 4.4.

4.2 Data Model Overview

The spatial data model is organized into these main UML packages (and thus schemas):

- Datum, where positions (see Section 4.3) and reference points (see Section 4.4) are defined.
- CRS, where CRSs are defined (see Section 4.5).
- BaseTypes (see Section 4.6). This package contains the common re-usable structures and types for commonly used by EnergyML schemas. It includes some designed to specify depth locations based on requirements and experience of the Energistics community.

This data model has been designed to incorporate relevant spatial models and best practices from the geodesy discipline and oil and gas operations and experience. It can accommodate many different use cases:

- Most common use cases:
  - Specify surface locations using positions.
  - For measured depth (MD), use the Energistics defined type in BaseTypes.
- Define positions and reference points whose meaning are defined in a related CRS.
- Convey positional uncertainty of reference points and origins of local CRSs.
- Other use cases not explicitly discussed here.

For both positions and CRS, the model supports the Public Land Survey System (PLSS), a system developed and used in North American that describes position and reference positions in sections, townships, and ranges. PLLS appears on relevant diagrams (figures in this section). Additionally, in the CRS package of the UML model, there is a separate PLLS diagram.

Like the overall Energistics data model, this part of the model makes us of abstract data objects, which makes implementing the standards easier for developers.

4.3 Positions

As defined in Section 4.1, a position is a place where something is located or has been placed/installed. For example, a well surface location is a position.

The Energistics data model supports the kinds of positions shown in **Figure 4-1**—which are 2D positions, compound positions, and 3D positions, each of which is described below in this section. All figures in this section are from the Positions diagram in the UML model.

Each position is also associated with a CRS, which is explained in Section 4.5. That section also provides some explanation about the various CRS models included in the Energistics data model (i.e. types and characteristics of 2D and 3D models).

NOTE: The figures in this section are from the Positions diagram. To easily see the entire diagram, use the XMI file that was delivered in the energyML download package. This diagram is in the common→doc folder.

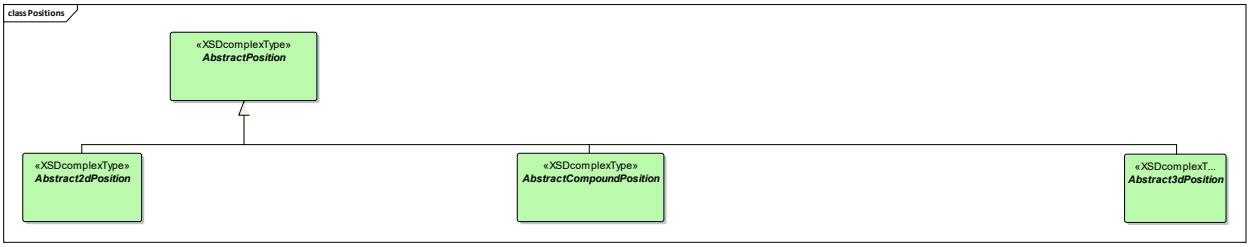


Figure 4-1. The Position data model supports three high-level kinds of positions: 2D, compound, and 3D.

4.3.1 2D Positions

Figure 4-2 shows the main kinds of 2D positons:

- Geographic2dPosition (Figure 4-2, right) recognizes the earth is a sphere and specifies a position using latitude and longitude.
- Cartesian 2D position (AbstractCartesian2dPosition, Figure 4-2, green box). To accommodate all possible schemes, the coordinates are generically labeled Coordinate1 and Coordinate2. The meaning of C1 and C2 (e.g., east and north, or x and y) are available through the definition of the CRS that the position references (see bottom of the figure). Cartesian 2D positions can be:
  - Projected2dPosition, which is what a land surveyor provides.
  - LocalEngineering2dPosition.
  - Public Land Survey System (PLLS) position, as described in Section 4.2.

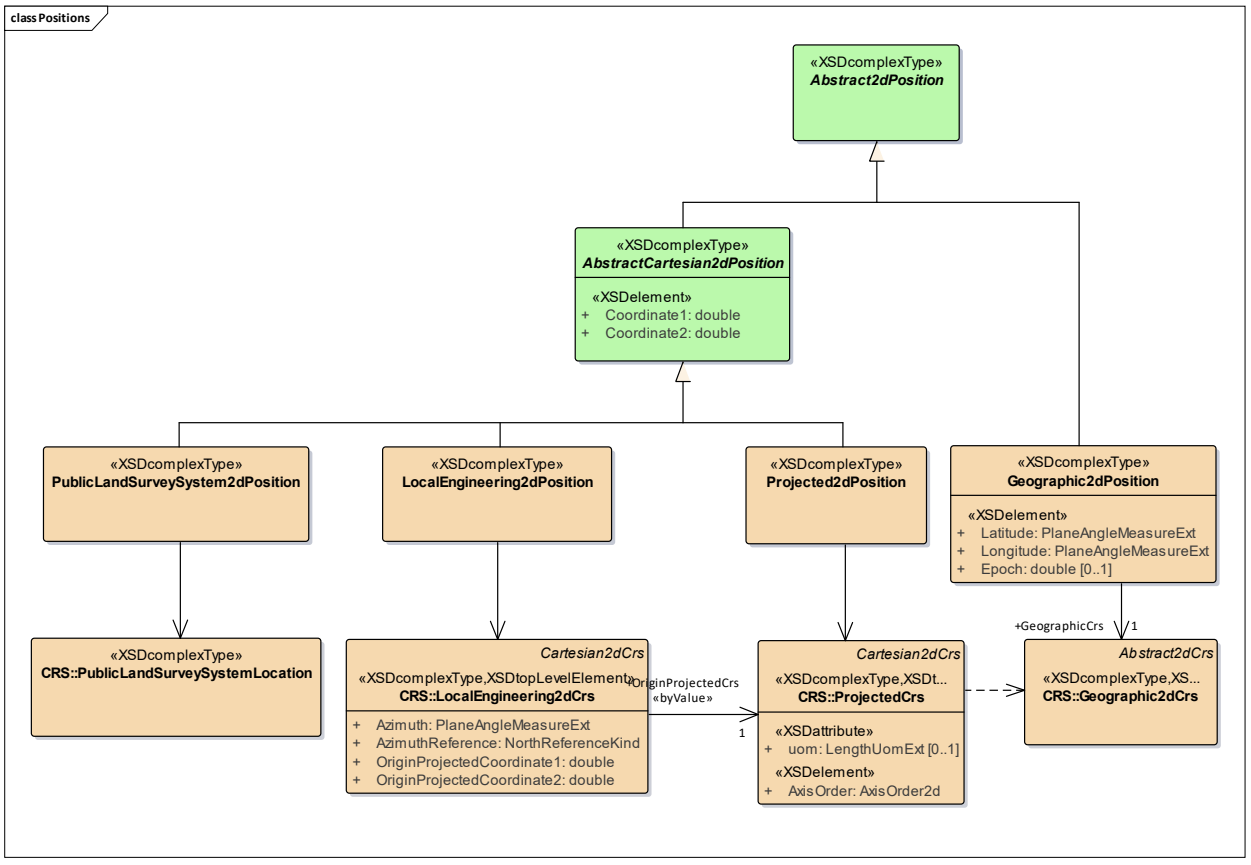


Figure 4-2. 2D positions include 2 types: Geographic2dPostion and AbstractCartesian2dPosition, which can be PublicLandSurveySystem2dPosition, LocalEngineering2dPosition, and Projected2dPosition. Note the references to CRS diagrams (lowest boxes), which shows that the meaning of the position is defined in a referenced CRS.

4.3.2 3D Positions

A 3D position specifies a position in 3D space. **Figure 4-3** shows the types of which are 3D positions supported.

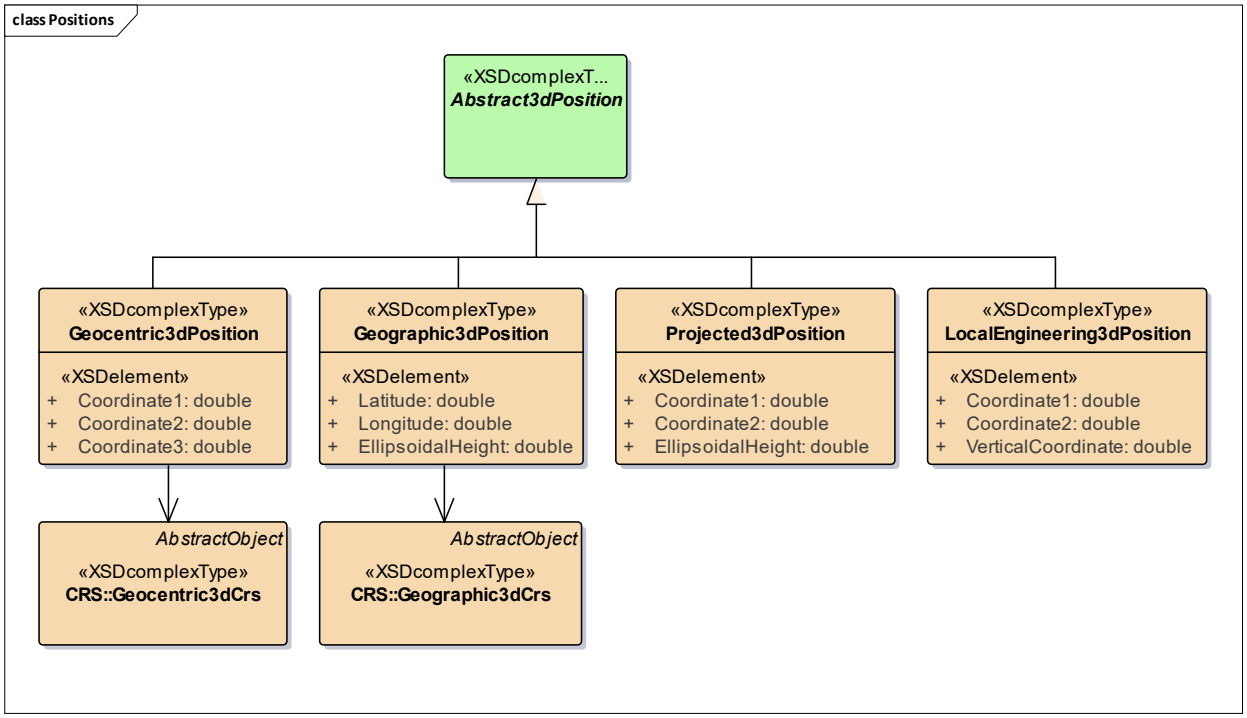


Figure 4-3. 3D positions can be of the types shown in the figure. Note the references to CRS diagrams (lowest boxes), which shows that the meaning of the position is defined in a referenced CRS.

4.3.3 Compound Position

A compound position is a proper 2D position (2 coordinates) plus a vertical distance about or below the surface location. Most oil and gas data uses compound positions. **Figure 4-4** shows the kinds of compound positions supported.

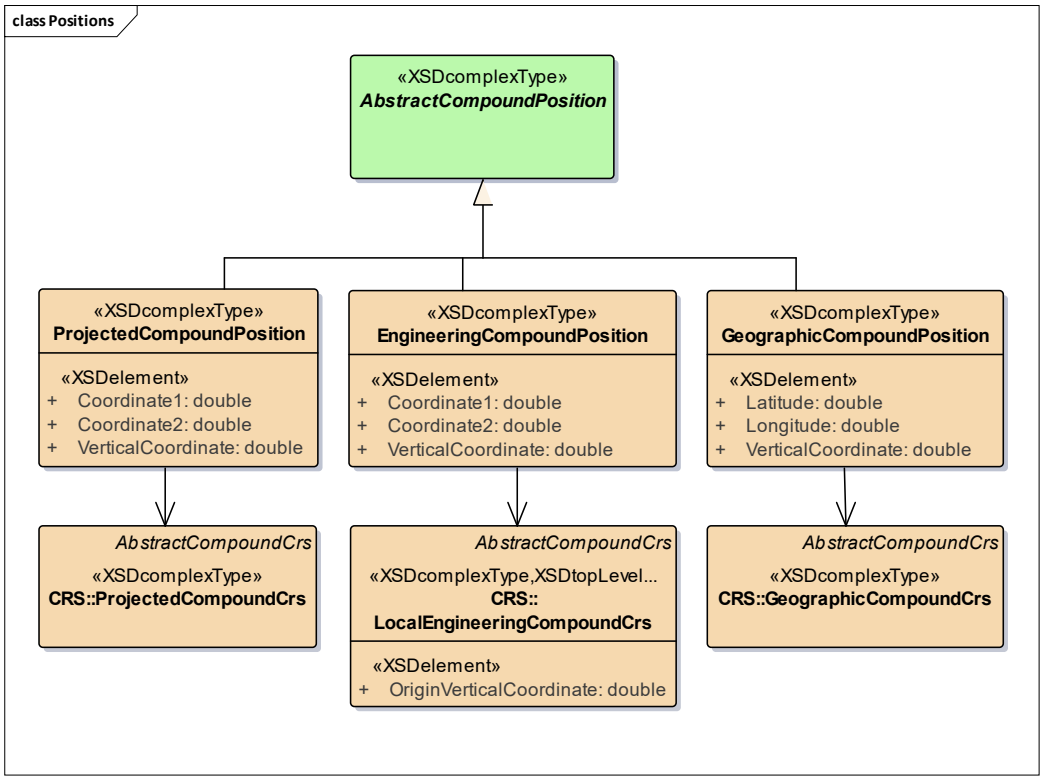


Figure 4-4. Compound positions can be of the type shown in the figure: projected, engineering or geographic.

4.4 Reference Points (Datums)

As defined in Section 4.1, a reference point is a designated position from which the position of other objects/artefacts are specified. EXAMPLE: On an offshore platform or in pad drilling, a surveyed point may be identified (such as a corner of the platform or drill pad) and the position of wells (or other equipment) are given relative to that reference point; for example, 10 ft north and 30 ft east of the reference point.

Reference points are designed and can work similarly to positions (EXAMPLE: A reference point can be defined in the context of a CRS, as shown in **Figure 4-7)**. This section explains the design differences unique to reference points. The main design differences are:

- Reference points can specify a kind (**Figure 4-5**).
- Reference points can recursively reference other reference points (**Figure 4-6**).

NOTE: The figures in this section are from the ReferencePoints diagram. To easily see the entire diagram, use the XMI file that was delivered in the energyML download package. This diagram is in the common→doc folder.

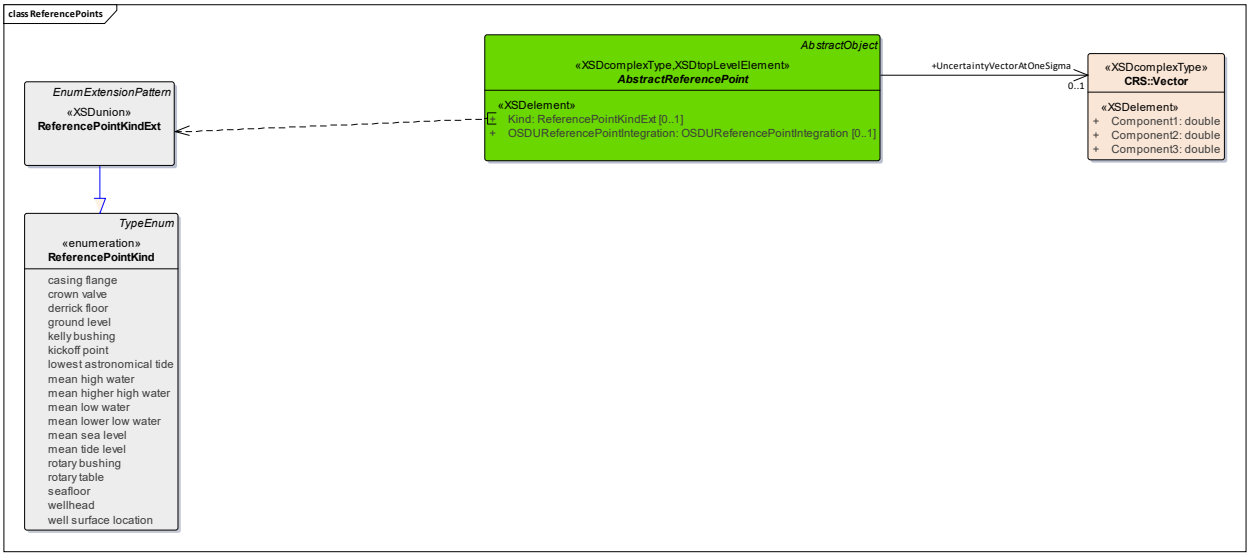


Figure 4-5. Reference points can specify a kind, which are listed in the ReferencePointKind enumeration shown above. This list is also extensible so organizations can specify their own.

Figure 4-6 shows how reference points can reference other reference points, recursively as necessary, creating a "chain" of reference points. Additionally, you can specify a reference point in wellbore.

EXAMPLE: This is a typical use case for recursive reference points (RP) in oil and gas operations. A land surveyor provides a latitude/longitude or x-y location and an elevation above sea level of a well surface location. That is the "starting" RP for a well. However, in drilling and well operations, the most commonly used reference points are either the drill floor of Kelly bushing (KB). So with this model, a user can specify the drill floor or KB as an RP, relative to the "starting" RP, and then other positions can be specified in reference to that drill floor of KB reference point.

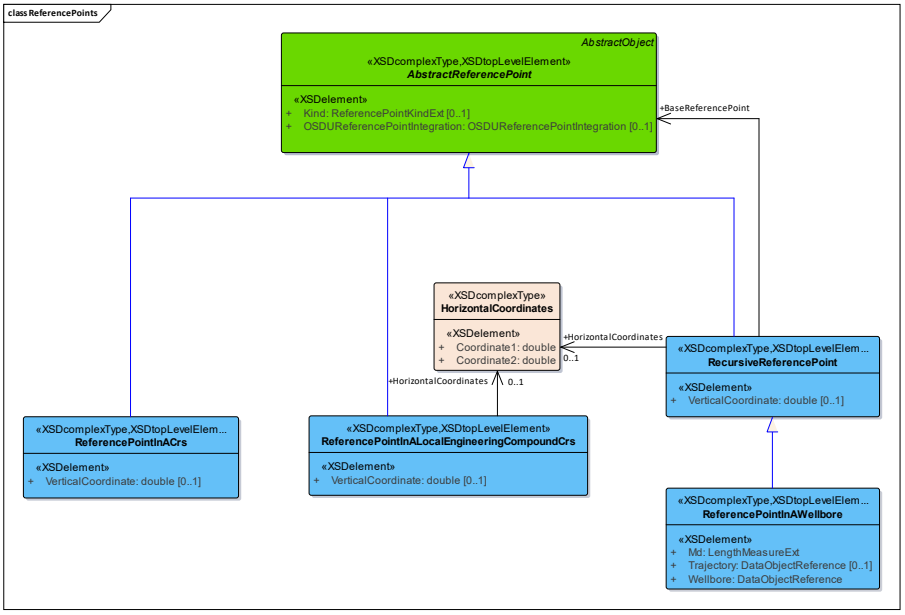


Figure 4-6. The model design makes it possible for reference points to reference other reference points, recursively as necessary.

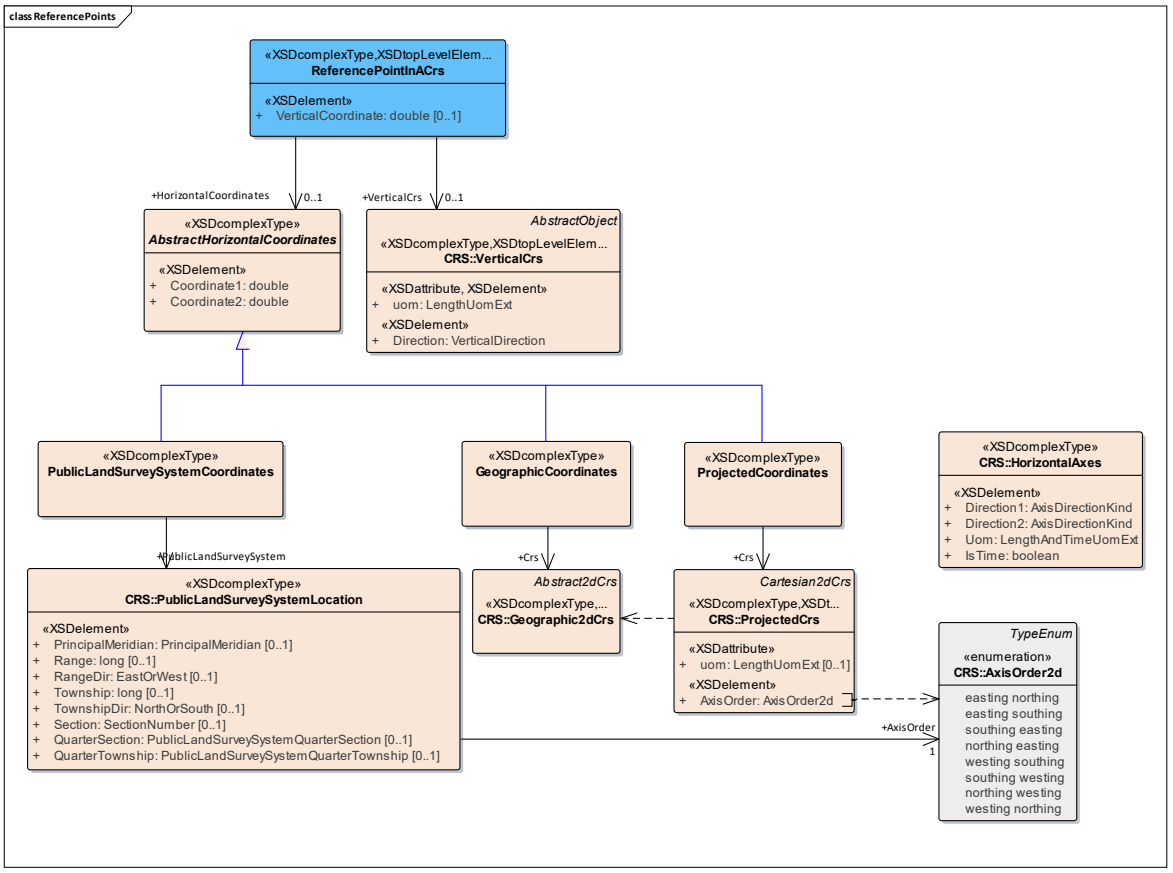


Figure 4-7. A reference point can be defined in the context of a CRS, similarly to how positions are defined.

4.5 CRS

Section 4.1 provides some high-level definitions of coordinate reference system (CRS).

The main use case for a CRS in the context of an Energistics data transfer is to provide the greater context for defining and understanding positions and reference points. As such, the Energistics data model design of the CRS parallels that of the position model describe in Section 4.3.

Additionally organizations may have other use cases for CRS, beyond what is explained here but that can be supported by the current design.

Figure 4-8 shows the main kinds of CRS, which include:

- 3D, which includes:
  - Geographic (Geographic3dCrs) is similar to a 2D CRS (i.e., it assumes a model of the earth that is ellipsoid/not). This model provides a lat/long position and an elevation/height above the model of the earth).
  - Geocentric (Geocentric3dCrs) is a system of x, y, z coordinates, where the origin is the center of the earth. Some of these models are gravitational and some are based on true knowledge of the rotational center of the earth.
- 2D CRSs are described in Section 4.5.1
- Vertical CRS (not shown in the figure) is used to specify the details for a vertical datum (reference point) and is described in Section 4.5.2.

The model also provides options for describing details of each of these CRSs, which include, EPSG codes, well known text, local authority or unknown. See Section 4.5.3.

NOTE: The figures in this section are from the CRS diagram. To easily see the entire diagram, use the XMI file that was delivered in the energyML download package. This diagram is in the common→doc folder. Additionally, the XMI file has additional detailed diagrams for CompoundCRS and PublicLandsSurveySystem.

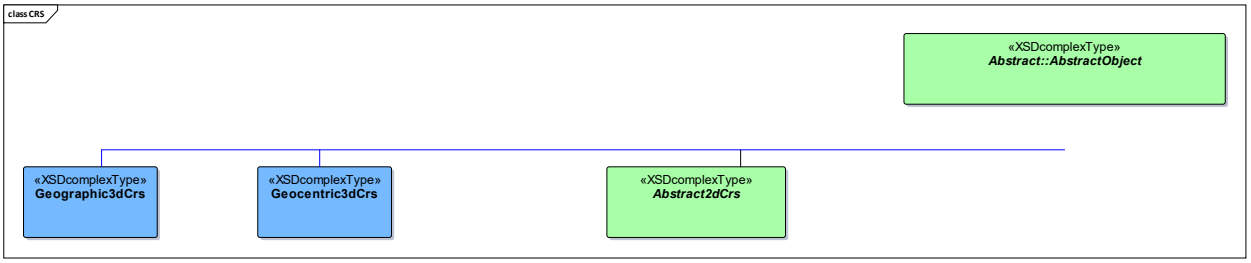


Figure 4-8. A CRS can be of the same types as positions: 3D (geographic or geocentric) and 2D.

4.5.1 2D CRS

The 2D CRSs (Abstract2dCrs) are organized into these main types in.

- Geographic2dCrs (Figure 4-9)
- Cartesian2dCrs (Figure 4-10), which is explained below the figure.

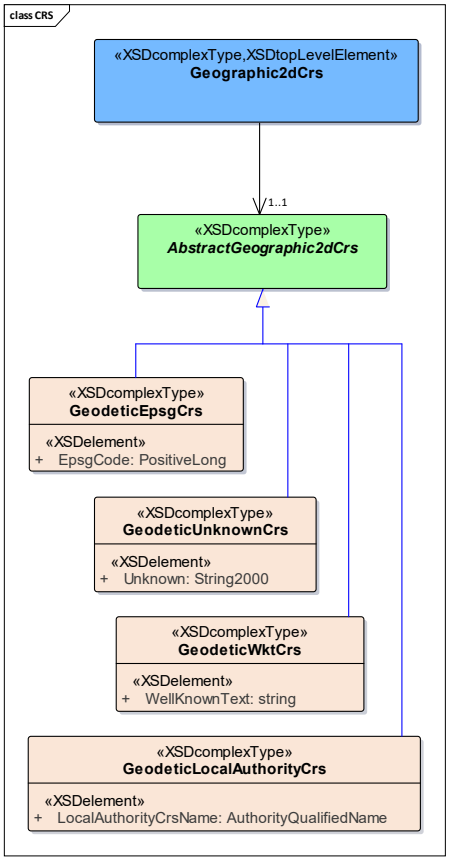


Figure 4-9. Geographic 2D CRS.

Figure 4-10 shows the Cartesian 2D CRSs, which is a projected CRS.

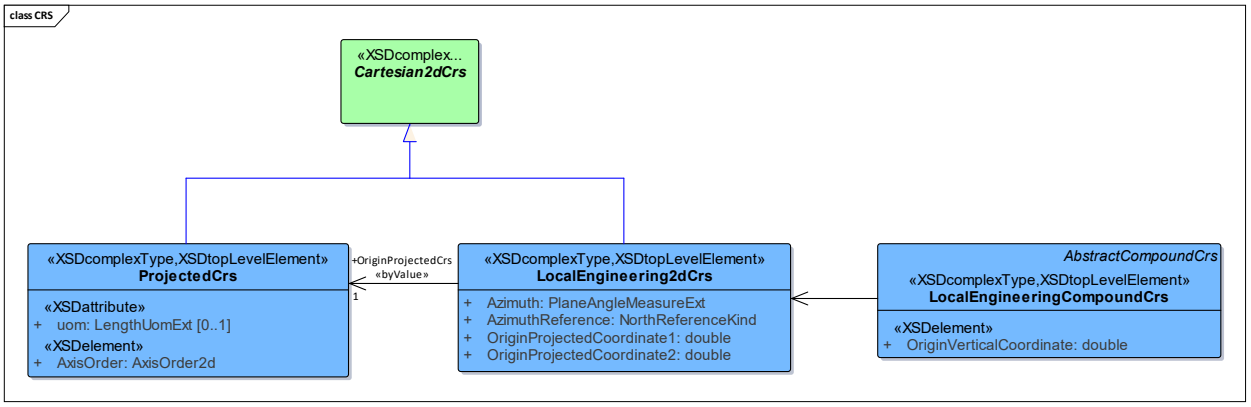


Figure 4-10. Cartesian 2D CRS are organized similarly as 2D positions: projected, local engineering and local engineering compound.

Cartesian 2D CRSs are organized into these 3 styles:

- Projected CRS (ProjectedCrs) (**Figure 4-11**)
- LocalEngineering2dCrs (**Figure 4-12** (page 43)**,** left). Refers to a CRS where there is a surveyed position (e.g., the corner of a platform or drill pad) that serves as a reference point for other positions.
- LocalEngineeringCompoundCrs (**Figure 4-12** (page 43), right). This is 2D CRS, with the addition of the position above or below the 2D position, for example, the KB position above the drill pad.

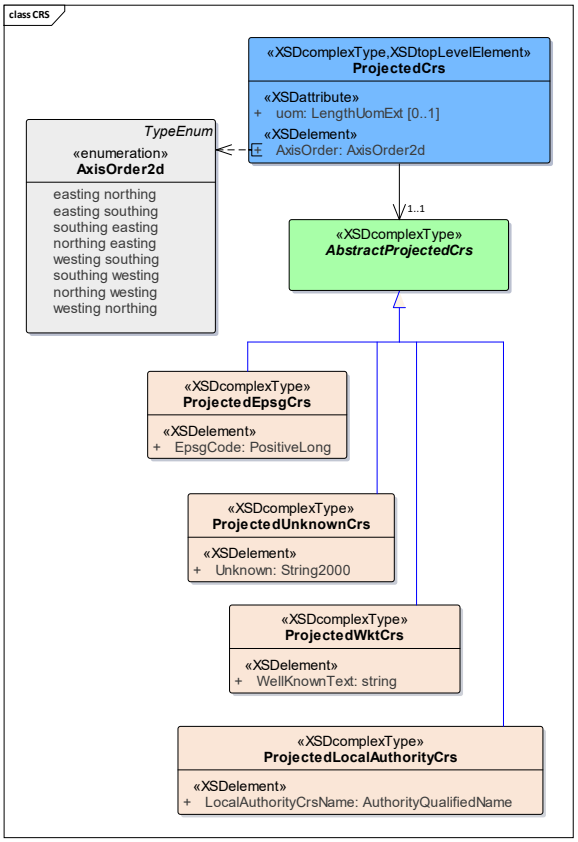


Figure 4-11. Projected 2D CRS.

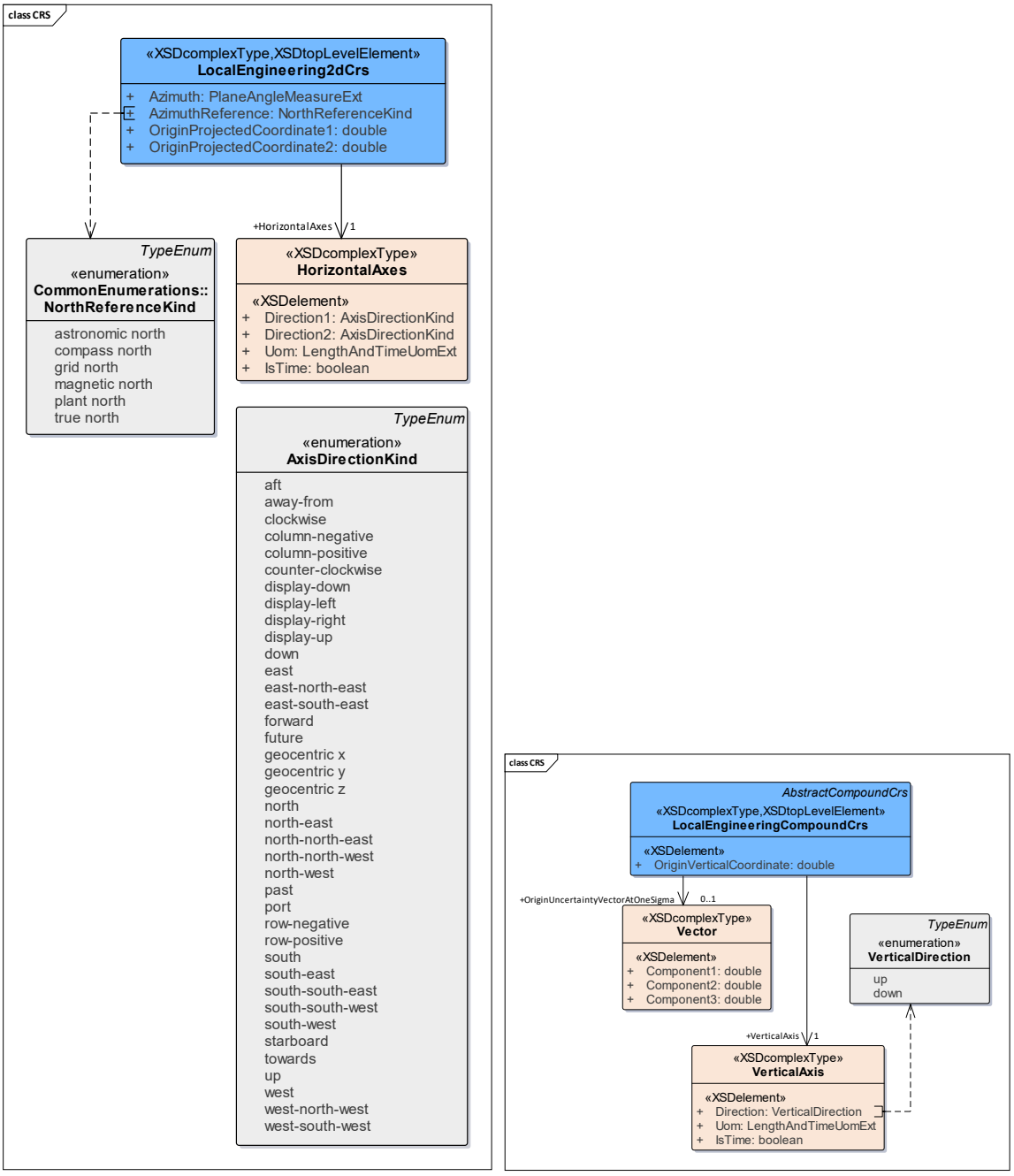


Figure 4-12. Local engineering 2D CRS (left); Local engineering compound CRS (right).

4.5.2 Vertical CRS

The CRS model includes the vertical CRS shown in **Figure 4-13**.

A vertical CRS is a 1D coordinate reference system that uses the earth's gravity field to determine the depth or elevation of a point relative to a reference surface (datum). This reference surface may be a simple geometric shape, like a sphere or an ellipsoid, or may be a more complex model that predicts sea level or other equal-gravity surfaces. A vertical CRS always defines a direction in which values increase—depth below or height above the reference datum surface. True vertical depth (TVD), is a vertical distance from a point in a wellbore to a point on the reference surface. It is calculated based on a measured depth along the wellbore and the azimuth and inclination of that point on the wellbore.

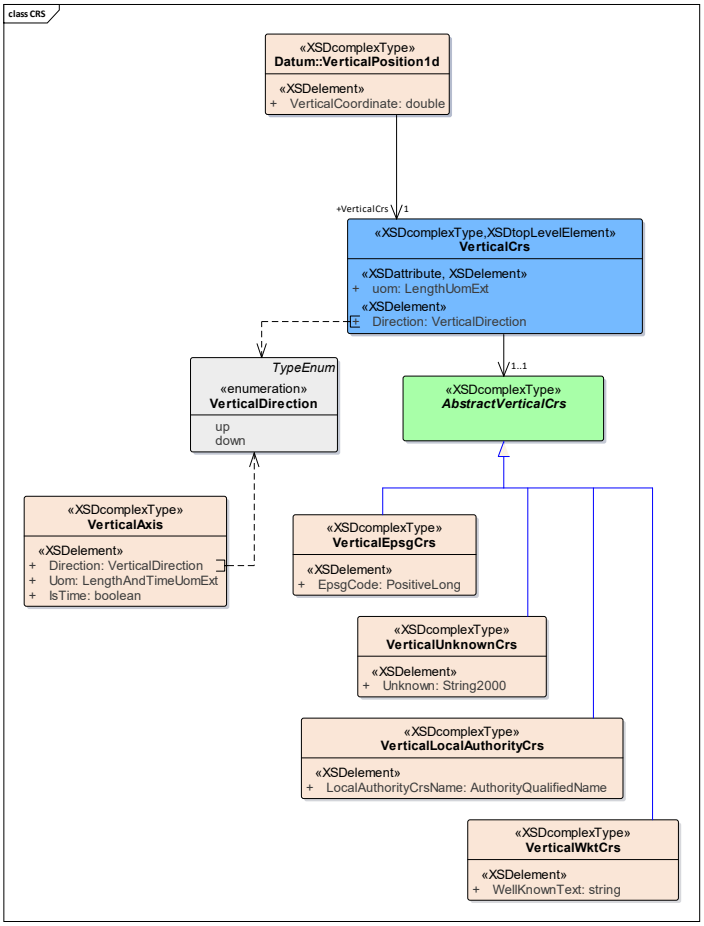


Figure 4-13. Vertical CRS.

There is another kind of coordinate reference system that is not explicitly data-modelled in EnergyML, which is a linear coordinate system. A linear coordinate system uses a distance along a linear feature to determine a location. Linear systems are used for measured depths along a wellbore, distances along pipelines or control lines (like fiber cables), or along cultural features line roads or rivers.

A linear coordinate system becomes a 1D CRS when its datum is defined, that is, identifying the starting point of that linear measurement. As described in Section 4.4, this is done by defining a reference point as the starting location. EnergyML has a pre-defined data type modelled for re-use which carries this distance and the identity of the reference point; it is the MeasuredDepth type.

4.5.3 Ways to Describe a CRS

Additionally the model provides the ways listed below to describe each supported CRS model. To see the specific options for each model, see the figures above (EXAMPLE: Figure 4-13 shows these elements for a vertical CRS.).

- **EPSG codes.** 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.uk/). The OGP is considered to be the single global source for positioning advice, guidance, and formats provision for the upstream oil and gas industry.
- **Well known text.** Some standards exist that use long, detailed strings to describes all the parameters necessary for describing an ellipsoidal model of the earth, or a local model of the earth, for example, ISO 19162. Because the text can be quite long, these types are not widely used. Typically used

where a standard authority, such as EPSG, does not have a code or when you have reason to not choose to use one of standard authorities.

- **Local authority.** Many national governments and oil companies have their own sets of CRS. To specify this requires you to specify the authority (e.g., the organization that specifies the CRS) and the identifier of the CRS from that authority.
- **Unknown**. This element is specified in cases where a user needs to anonymize a location, for example, when giving data to a university or other research organization. Use it to describe the reason a CRS is not being provided.

4.6 Energistics Base Types for Depth, Elevation and Intervals

The BaseTypes package contains the common re-usable structures and types commonly used by all EnergyML Schemas. Figure 4-14 shows the types that may be used to specify depths and elevations. Note that AbstractVerticalDepth is positive down and AbstractElevation is positive up. Intervals (MdInterval and AbstractTvdInterval) may also be specified.

Use of these types requires that a reference point (datum) be specified, as these values are meaningless unless they reference a known position.

For the complete list of all elements defined in BaseTypes, see the *Energistics common v2.3 Technical Reference Guide*.

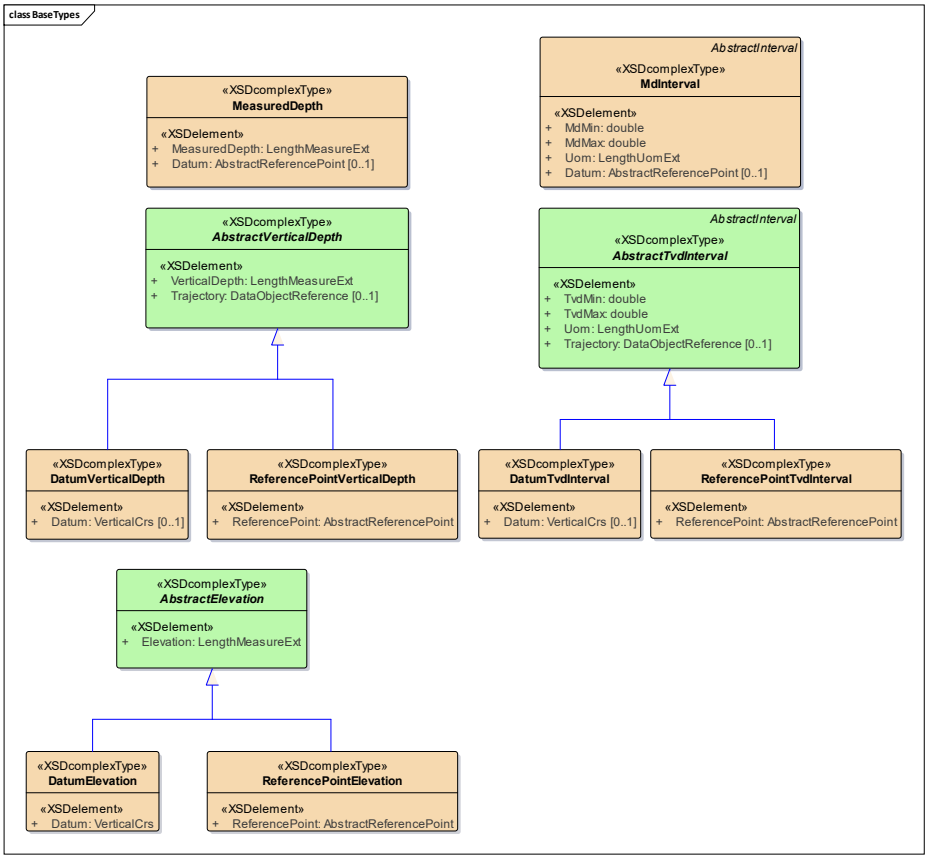


Figure 4-14. Base types that may be used to specify depths and elevations (from the BaseTypes diagram in the BaseTypes UML package).

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.

4.7 Specify Default Datum on a Data Object

The energyML data model allows you to specify a default datum on a data object. For example, in WITSML you can specify a default datum on a well, wellbore and rig. If this data object default value is specified, then it becomes the default datum for any spatial/location attributes specified in that data object.

NOTE: Currently, WITSML is the only data model that uses this feature.

5 Appendix: 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.
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* *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*	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
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.
Security and authorization	For the list of industry standards leveraged for security and authorization, see the *ETP Specification v1.2*.

6 Appendix: Example Using UUIDs and EIP Metadata for Traceability to Help Manage Different Versions of a Data Object

This appendix provides an example from RESQML of how UUIDs and EIP metadata can be used for traceability and data conflict resolution.

EIP metadata can be used during the modeling process (workflow) itself—when RESQML models are frequently updated and transferred between software packages—to help users determine the latest version of a model.

6.1 Overview

In **Figure 6-1. Conflict resolution: frequent transfer in a single session.**, Software A and B represent applications used in consecutive stages of a subsurface workflow. When new information becomes available for a given model in Software A, users of Software B will typically have to reload a new RESQML document containing the entire updated model. If users of Software B already modified the model, they will have to either:

- Discard their modified model and manually add their modifications to the updated/newly imported model, or
- Manually resolve the conflict between these two versions of the same model.

6.2 How it Works

EIP metadata can help tremendously in the process of conflict resolution. Some examples are available below.

The two key pieces of information available for each individual data object are:

- The UUID used to indicate if an entity present in an Energistics package (RESQML data) is already available in the current session of Software B, indicating the potential conflict between two versions of the same entity.
- The "modified" time inside the EIP metadata elements (or the "created" time if the entity has not yet been modified) indicating that the version in New File A is more recent than the edition (or version) of the entity in the Software B model.

After a conflict has been detected, Software B can extract other EIP metadata to help users select one version, either manually or following a common strategy (for example, always overwrite, ignore, keep the most recent one, import as a copy, and so forth).

In the above process, Software B is “kept live” between the initial import of the model and the import after modification. However, the same resolution process can also occur in two different sessions of Software B, for example, if Software B saves the model in another Energistics package or in its own persistence mechanism (e.g., database). IMPORTANT! For this approach to work, the persistence mechanism MUST retain the UUID and EIP metadata.

6.2.1 Conflict Resolution: Frequent Transfer in One Session

Figure 6-1 shows an example workflow for resolving data conflicts resulting from frequent update and transfer of data models during one session.

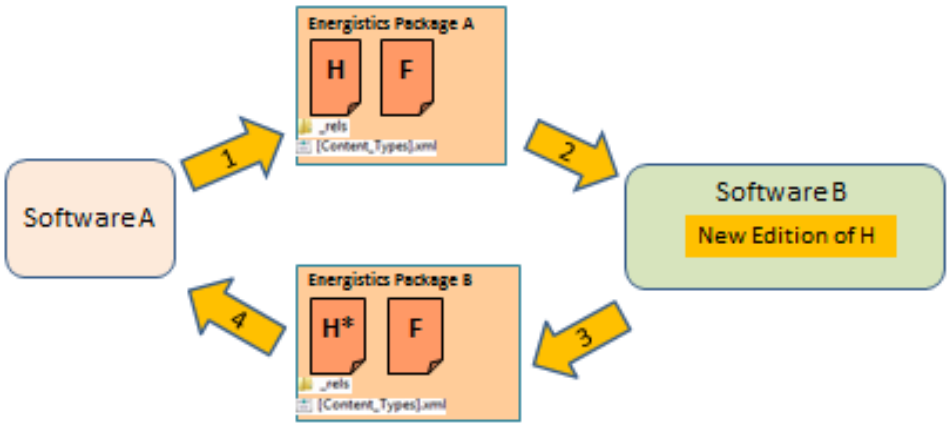


Figure 6-1. Conflict resolution: frequent transfer in a single session.

The scenario is:

1. Software A exports H and F in Energistics Package A.
2. Software B imports Energistics Package A, modifies H.
3. Software B exports Energistics Package B: F + H*, (where H* = edited H).
4. When Software A imports Energistics Package B, it should have enough information to know:
   - F has not been modified, and does not have to be reloaded.
   - H* is an edited version of H; there is enough information to let a user decide how to reconcile the two versions.

6.2.2 Conflict Resolution: Transfer in Multiple Sessions with Persistent Data Store

Figure 6-2 shows an example workflow for resolving data conflicts resulting from updates and transfer during multiple sessions with a persistent data store.

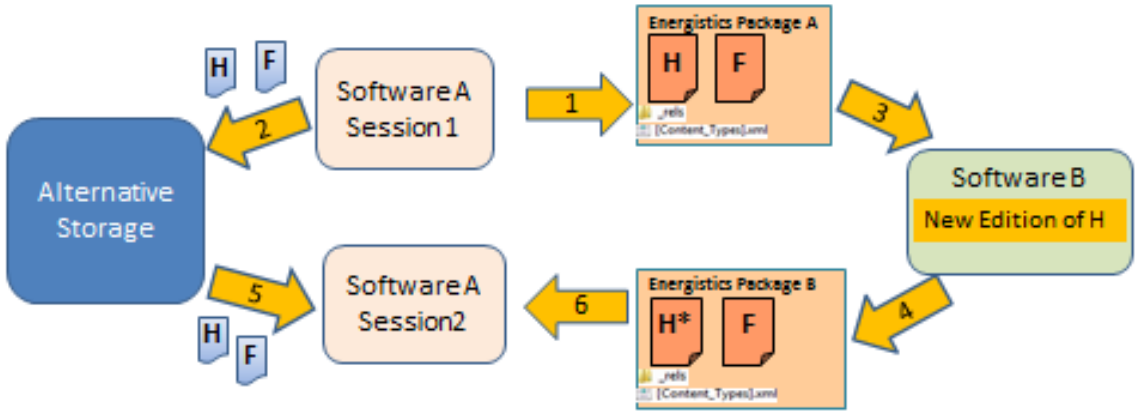


Figure 6-2. Conflict resolution: transfer with alternative persistence.

The scenario is similar to the single-session scenario:

1. Software A exports Energistics Package A.
2. Software A saves (persists) H and F in alternative storage (for example, a database), then Software A Session1 is closed.
3. Software B imports Energistics Package A and edits are made to H, creating H*.
4. Software B exports document Energistics Package B: H* and F (no changes).
5. Software A begins a new session, Session 2, and reloads H and F from alternative storage.
6. When Software A imports Energistics Package B into Session 2, Session2 has enough information to know without using Energistics Package A that:
   - F has not been modified, and does not have to be reloaded.
   - H* is an edited version of H; there is enough information to let a user decide how to reconcile the two versions.

6.2.3 Conflict Resolution Example

Figure 6-3 shows how RESQML with EIP metadata can be implemented in a software package to help resolve data conflicts that arise from frequent update and transfer of data models—a common occurrence in the subsurface workflow.

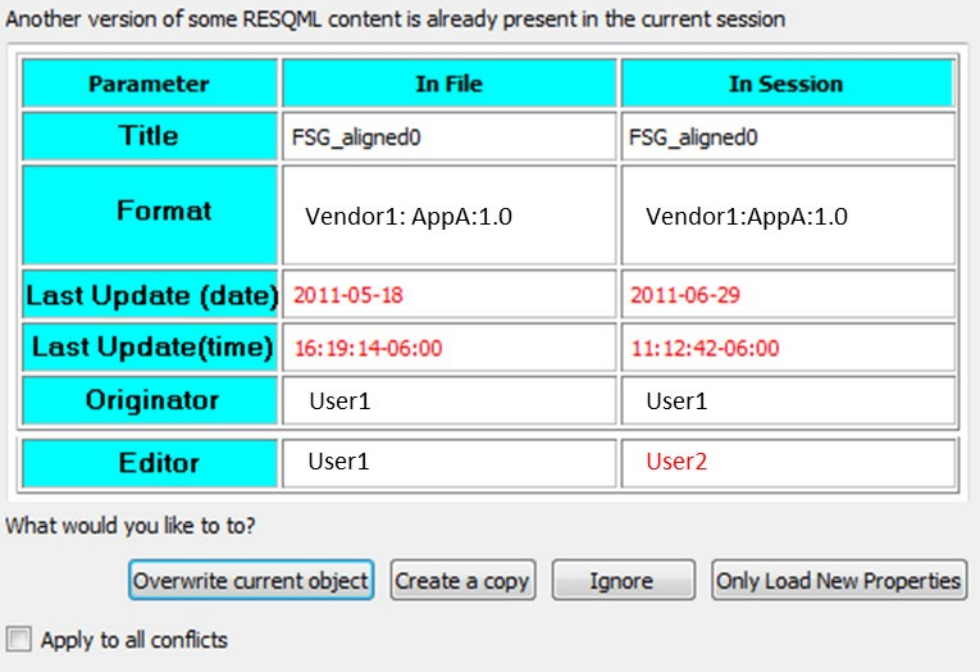


Figure 6-3. Conflict resolution example of how RESQML and EIP metadata can be used in software to resolve data conflicts.

When the same object is present in an Energistics package (RESQML data) and session, a user can **decide what to do based on the differences (red text):**

- Replace the content of the session by the content of the file.
- Create a new copy with the content of the file.
- Do not load the content of the file.