WITSML Technical Usage Guide

For WITSML v2.0

WITSML Overview	WITSML is the data exchange standard for specifying and exchanging data for wells and well-related operations and objects, such as drilling, logging and mud logging. Version 2 is a significant change in underlying technology architecture and data model design from the last published version, 1.4.1.1.
Version of Standard	2.0
Abstract	This guide list and describes data objects used in the current version. For each object the intent is to provide: an overview of its business use, design, intended usage, and examples.
Prepared by	Energistics and the WITSML SIG
Date published	11 November 2016
Document type	Usage guide
Keywords:	standards, energy, data, information, process, wells, logs,

Document Information	
DOCUMENT VERSION	1.0
Date	11 November 2016
Language	U.S. English

Amendment History			
Std Version/ Doc Version	**Date**	**Comment**	**By**
2.0 / 1.0	11 Nov 2016	Version 2.0 of WITSML published with the Energistics Common Technical Architecture and other redesign of data objects.	Energistics and WITSML SIG

Table of Contents

1 Introduction

1.1 What is WITSML?

WITSML is data exchange standard for specifying and exchanging data for wells and well-related operations and objects, such as drilling, logging and mud logging. The initial focus of WITSML was for the right-time, seamless flow of well data between operators and service companies to speed and enhance decision making. While this is still a key goal, use of WITSML has expanded, for example, some operators use if for interoperability among their systems.

WITSML has been developed by a global consortium of operators, service companies, software vendors, and government agencies under the umbrella of Energistics.

Beginning with v2.0, WITSML leverages the Energistics Common Technical Architecture (CTA), which delivers a common technology foundation for the Energistics domain standards—which includes WITSML, PRODML, and RESQML. This common architecture makes it easier for companies to implement the standards. It also makes it easier to share data "across" the standards. For example, RESQML and PRODML can leverage well data in WITSML.

The last published version of WITSML, v1.4.1.1, specified a set of 27 data object definitions and a Web services specification.

In v2.0, main changes include:

- A revised set of data objects. For a list, see Table 1 (page 12).
- Replacement of the Web services with the Energistics Transfer Protocol (ETP), which is part of the Energistics CTA.

This guide describes each WITSML data object, including its business use and an overview of its design.

1.1.1 How Does WITSML Work?

WITSML consists of a set of XML schemas (XSD files) and other standards-based technology that define a set of standard-format data objects that represent the objects and data required and produced for well-related operations. Developers implement the schemas into software packages. Software that has implemented WITSML can read and write the standard format, which effectively allows them to "exchange" data.

Data "exchange" can happen in several ways:

- **Streaming**. With the Energistics Transfer Protocol (ETP) many WITSML data objects can be streamed, which may occur to collect data (e.g., to store on a server) or shared between WITSML-enabled software.
  - With ETP, streaming can occur
- **Client-Server**. WITSML servers store data that is accessed by WITSML-enabled client applications.
- **Static Transfer**. While not the preferred method for the 24/7 pace of drilling and well-related operations, it is possible to simply email someone a WITSML document.

1.1.2 Must I Implement All WITSML Data Objects?

The set of data objects that compose WITSML represent real-world objects in the drilling, completions, and workover lifecycle. However, the number of objects implemented is completely up to you and your organization, based on specific needs.

Some objects can be considered mandatory based on requirements of industry workflows. For example, operations reports (cement, stimulation job, drilling, and fluids reports) only make sense if you know for what well/wellbore the data was produced. So all WITSML implementations typically include well and wellbore data objects.

However, if you are a service company with a focused a role—for example, mud logging—then your WITSML implementations may only include the data objects required for your operations.

1.2 Audience, Purpose and Scope

Chapter 1 serves as a high-level overview of WITSML v2.0 and is intended for all audiences (business and technical).

The remainder of the content is for information technology (IT) professionals—programmers, developers, architects and others—who are implementing WITSML into a software package or other relevant technology.

1.3 Resource Set: What's Included When You Download WITSML

WITSML is a set of XML schemas (XSD files) and other technologies freely available to download and use from the Energistics website. When you download WITSML, you get a zip file structured as shown in Figure 1-1, which contains the resources listed in the tables below in these 2 main groups:

- **WITSML-specific** (Section 1.3.1). The schemas and documentation specific to WITSML. Note in Figure 1-1 (left) (red square):
  - WitsmlAllObjects.xsd is schema that includes all other schemas in shown.
  - The WitsmlCommon schema includes objects shared by other WITSML data objects.
- **Energistics Common Technical Architecture** (Section 1.3.2) (in the *common->v2.1* folder) Components of the Energistics Common Technical Architecture (CTA), which is a set of specifications, schemas, and technologies shared by all Energistics domain standards. The components required for WITSML are included in the WITSML download.

To download the latest version of this standard, visit http://www.energistics.org/drilling-completions-interventions/witsml-standards/current-standards

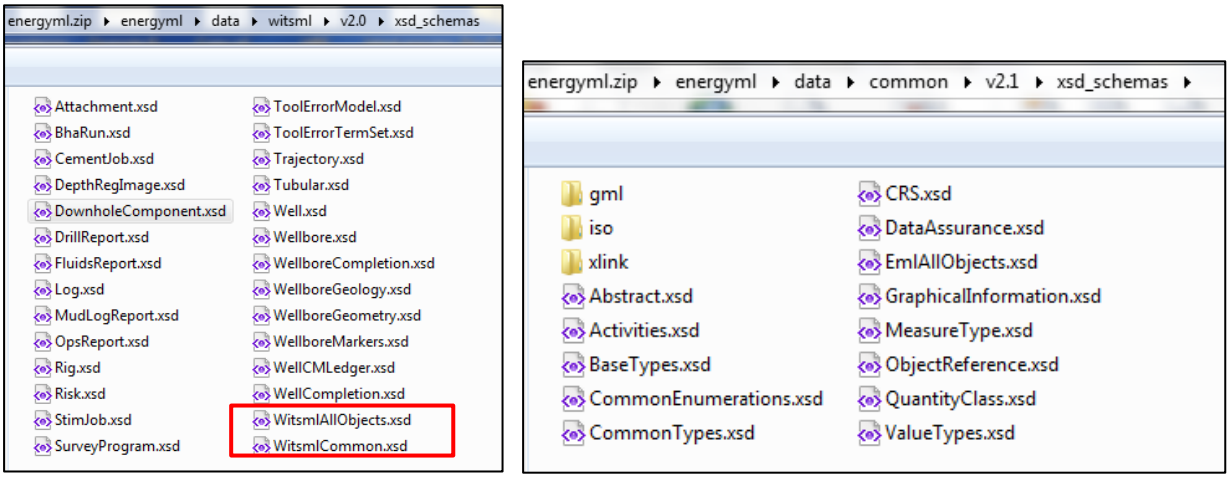

Figure 1–1. You download the WITSML standard as zip file from the Energistics website. It contains the resources described in the two tables below. The figure shows (left) the WITSML schemas (xsd files) and (right) the Energistics common schemas.

1.3.1 WITSML Resources

	Resource / Document	Description
1.	WITSML:XSD files	The WITSML package includes a readme file that details the contents of the download package.
2.	WITSML UML Data Model	The entire UML data model that developers and architects can explore for better understanding of data objects, definitions, organization, and relationships. Energistics saves the UML model as an XMI file, a format that can be imported by any UML data modeling tool.
3.	*WITSML Technical Usage Guide (This document: START HERE)*	Detailed explanation of key concepts, data objects, and architecture intended for software/IT professionals.
4.	*WITSML Technical Reference Guide*	Generated from the UML model, this guide lists and defines data objects and elements in the WITSML model.
5.	*WITSML v2.0 over ETP Implementation Specification* *COMING SOON*	Describes how to implement the WITSML v2.0 data model with the Energistics Transfer Protocol (ETP).

1.3.2 Energistics Common Technical Architecture Resources

These resources are included in a standards download, unless otherwise specified in the table.

	Resource / Document	Description
1.	Energistics common XSD files	Schemas (XSD files) for Energistics CTA data objects, which are contained in the folder named *common* and are further described in this document. NOTE: The *common* folder is Included as part of the package when you download any of the Energistics domain standards.
2.	UML Data Model (XMI file)	The UML data model that developers and architects can explore for better understanding of data objects, definitions, organization, and relationships, in context. Used to generate the XSD files and technical reference documents. NOTE: Objects in the CTA are in the folder named *common* and included in the UML for each Energistics domain standard. Energistics saves the UML model as an XMI file, a format that can be imported by any UML data modeling tool.
3.	*Energistics Common Technical Architecture Overview Guide (For CTA, start here.)*	Provides an overview of the components that comprise the Energistics CTA.
4.	*Energistics common Technical Reference Guide*	Lists and defines packages, data objects, elements, and relationships for the objects in the CTA *common* folder. Produced from the common UML package from which Energistics *common* XSDs are produced.
5.	*Energistics Packaging Conventions (EPC) Specification*	Specifies the Energistics Packaging Conventions (EPC), which is the set of practices to store multiple files as a single entity for data transfer; this single entity is referred to as an “EPC file” (or sometimes, an “Energistics package”). EPC is an implementation of the Open Packaging Conventions (OPC), a container-file technology standard.

	Resource / Document	Description
6.	*Energy Industry Profile of ISO 19115-1 (EIP)*	An open, non-proprietary exchange standard for metadata used to document information resources, and in particular resources referenced to a geographic location, e.g., geospatial datasets and web services, physical resources with associated location, or mapping, interpretation, and modeling datasets. The EIP is an ISO Conformance Level 1 profile of the widely adopted international standards ISO 19115-1:2014 which provides XML implementation guidance with reference to ISO Technical Specification 19115-3:2016.
7.	*Energistics Identifier Spec*	Describes the syntax and semantics of data object identifiers used in Energistics data exchange standards, which include UUIDs and URIs, and object reference.
8.	*Energistics Unit of Measure Standard* (Must be downloaded separately: http://www.energistics.org/asset-data-management/unit-of-measure-standard)	A dictionary, grammar specification, and related documentation, which provide a consistent way to define, exchange, and convert between different units of measure. All Energistics standards (PRODML, WITSML, PRODML, etc.) must use this dictionary; other industry groups are also using it. Key data objects and components of the UOM spec are implemented in Energistics *common*.
9.	*Energistics Transfer Protocol (ETP)* *(Must be downloaded separately: http://www.energistics.org/standards-portfolio/energistics-transfer-protocol)*	A data-exchange specification that enables the efficient transfer of real-time data between applications. Specifically envisioned and designed to meet the unique needs of the upstream oil and gas industry and, more specifically, to facilitate the exchange of data for Energistics domain data standards (RESQML, WITSML, and PRODML).

1.3.3 Documentation Updates

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 Overview of the New v2.0 Design

Many existing WITSML data objects have been redesigned and a few unused data objects have been removed. Use of the Energistics CTA was the basis for the main v2.0 design changes. In addition, some data objects were redesigned to address data modeling issues in v1.4.1. Another significant change: the Energistics Transfer Protocol replaces the previous Web services.

1.4.1 Energistics Common Technical Architecture (CTA)

One of the goals of digital oilfield technology is to break down domain silos in E&P workflows. In support of this goal, Energistics is working to better harmonize its domain-based standards, including WITSML, RESQML (earth modeling), and PRODML (production operations).

Energistics is also moving towards sharing resources and establishing processes across its standards where this approach makes sense, for example, with coordinate reference systems, units of measure, and file packaging conventions. These shared resources are referred to as the Energistics Common Technical Architecture (CTA).

For a list of the main components of the Energistics CTA used by WITSML, see Section 1.3.2 (page 9). For an overview of CTA, see Section 2.1 (page 14).

1.4.2 ETP Replaces v1.4.1 Web Services

Energistics Transport Protocol (ETP) is a new data-exchange specification developed by the Energistics community. It is a component of the Energistics CTA.

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. It enables the efficient transfer of real-time data between applications.

One of the goals of ETP is to replace TCP/IP WITS level 0 data transfers with a more efficient and simple-to-implement alternative.

The three main initial use cases for ETP and WITSML are to move real-time data between applications, including:

- Transfer from a wellsite provider to a WITSML store (server)
- Transfer of data from WITSML store to WITSML store (replication)
- Transfer of data from WITSML store to client applications

ETP defines a publish/subscribe mechanism so that data receivers do not have to poll for data and can receive new data as soon as they are available from a data provider.

ETP is being expanded beyond real-time data transfer to include functionality for data discovery and historical data queries. ETP is the underlying protocol for WITSML v2.0.

1.4.3 Version 2.0 Design Approach

The move to a common architecture and harmonization of Energistics' three flagship standards were the main drivers for the v2.0 redesign. However, the WITSML Special Interest Group (SIG) also recognized that this major change was an opportunity to improve individual object's design or to remove data objects that were not being used.

For the list of WITSML data objects and summary of changes, see Table 1 (page 12).

The basic design approach was:

- At a minimum, the v1.4.1.1 data objects that were in use were migrated to the v2.0 design.
- If a working group of SIG members was available to redesign an object, then the SIG worked to do that. For example, some of the objects that had major redesigns include: log, mud log, stim job, and cement job.
- If the SIG was certain that a data object was not being used, it was deprecated/removed from WITSML.

1.4.3.1 Conventions No Longer Supported

The Energistics v2.0 architecture eliminates the need for some common conventions used in schemas of previous versions of WITSML. Conventions that are NO LONGER USED include:

- **obj**_ and **cs**_ designations. For information on top-level data objects (previously designated with obj_), see Section 3.2 (page 18).
- **Plural objects**. Previously, each object had a plural object container (e.g., wells contained well).
- **XSD:choice.**

1.4.4 Data Object Mapping: v1.4.1.1 to v2.0

Table 1 lists the names of WITSML v1.4.1.1 data objects, corresponding name in v2.0 (some data object names changed) and a brief summary of the changes in 2.0)

Table 1 WITSML Data Object Mapping: v1.4.1.1 to v2.0

V1.4.1.1 Data Object Name	V2.0 Data Object Name	Summary of Change
Attachment	<same as v1.4.1.1>	Converted to v2.0 design.
BhaRun	<same as v1.4.1.1>	Converted to v2.0 design.
CementJob	<same as v1.4.1.1>	Redesigned. Added Cement Job Evaluation
	Cement Job Evaluation	New
ConvCore		Deprecated/removed
CoordinateReferenceSystem		Moved to Energistics CTA (common folder)
	Data Assurance	New. Part of Energistics CTA, but important use for WITSML
DepthRegImage	<same as v1.4.1.1>	Standalone object for use with v1.4.1.1; Converted to v2.0 design.
DrillReport	<same as v1.4.1.1>	Converted to v2.0 design.
FluidsReport	<same as v1.4.1.1>	Converted to v2.0 design.
FormationMarker	Wellbore Markers	Converted to v2.0 design.
Log	<same as v1.4.1.1>	Redesigned
MudLog	WellboreGeology	Redesigned
	MudLogReport	New
OpsReport	<same as v1.4.1.1>	Converted to v2.0 design.
Rig	<same as v1.4.1.1>	Converted to v2.0 design.
	RigUtilization	New. The v1.4.1.1 Rig object was split into Rig and Rig Utilization.
Risk	<same as v1.4.1.1>	Converted to v2.0 design.
SidewallCore		Deprecated/removed
StimJob	<same as v1.4.1.1>	Redesigned
SurveyProgram	<same as v1.4.1.1>	Converted to v2.0 design
Target		Deprecated/removed
ToolErrorModel	<same as v1.4.1.1>	Converted to v2.0 design.
ToolErrorTermSet	<same as v1.4.1.1>	Converted to v2.0 design.
Trajectory	<same as v1.4.1.1>	Converted to v2.0 design.
Tubular	<same as v1.4.1.1>	Converted to v2.0 design.
Well	<same as v1.4.1.1>	Converted to v2.0 design.
Wellbore	<same as v1.4.1.1>	Converted to v2.0 design.
WbGeometry	WellboreGeometry	Converted to v2.0 design.

Table 1 WITSML Data Object Mapping: v1.4.1.1 to v2.0		
V1.4.1.1 Data Object Name	**V2.0 Data Object Name**	**Summary of Change**
Completion (actually made up of 4 top-level data objects shown near the bottom of the list in Figure 2-2 (page 16)).	<same as v1.4.1.1>	Converted to v2.0 design.

1.5 History

WITSML was initially developed in October 2000 by the WITSML project, an oil industry initiative sponsored by BP and Statoil, as a new standard for drilling information transfer to evolve WITS, an earlier standard. Initial participants included Baker Hughes, Landmark (Halliburton), Schlumberger, and NPSi (as technical advisor). At the completion of WITSML v1.2 in March 2003, Energistics (known then as POSC) accepted custody of WITSML and is managing the support and future evolution of WITSML through the WITSML Special Interest Group (SIG).

WITSML was designed to be a more modern alternative to WITS (Wellsite Information Transfer Standard). Some of the original semantic content of the WITSML data schemas was derived from the WITS specification.

Version 2—based on the Energistics Common Technical Architecture, particularly use of the new Energistics Transfer Protocol (ETP), and the streamlining and redesign of the WITSML data objects—is the next evolution of WITSML.

2 Technical Architecture Overview

Each of Energistics domain standards—RESQML, WITSML and PRODML—leverages components in the Energistics Common Technical Architecture (**Figure 2-1**).

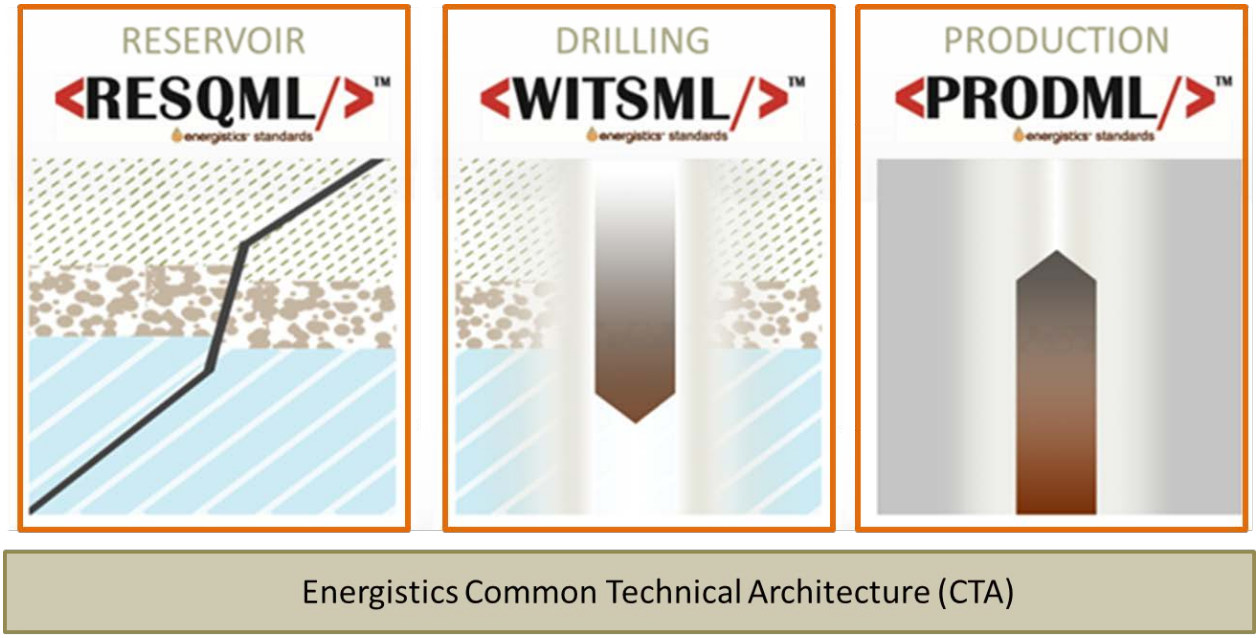

Figure 2–1. Energistics domain standards rest on a shared foundation of the Energistics Common Technical Architecture (CTA).

This chapter presents an overview of the CTA and the WITSML data model.

Related resources:

- For more information on the CTA and its components, see the *Energistics CTA Overview Guide* and other related CTA specifications (as listed in Section 1.3.2).
- For WITSML key concepts, see Chapter 3. Chapters 4 – 11 describe WITSML data objects.

2.1 CTA: Main Components and What They Do

NOTE: For more information about standards used in the CTA, see the *CTA Overview Guide*.

Each Energistics domain standard (RESQML, WITSML, and PRODML) has its own set of schemas, which include a <domain>common package (e.g., WitsmlCommon) of schemas for consistency across each ML (refer to Figure 1-1). 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. Each instance of a top-level data object must be identified by a universally unique identifier (UUID).

Each domain ML leverages components of the Energistics CTA.

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

- **Energistics *common* schemas.** The *common* package is standardized across all Energistics domain standards; for each of the MLs, the same *common* package is included in the download. Like the Energistics domain schemas, these schemas are also XML XSD files. Data object schemas can be considered in these categories:
  - Mandatory (for example, AbstractObject, ObjectReference, objects related to units of measure (UOM), etc.).

  - Optional, available for use if wanted (for example, the Data Assurance and Activity Model objects).
  - Objects defined by Energistics specifications (see next bullet), which may be optional or mandatory, depending on the specification and domain ML.

- **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 complete list of CTA resources, see Section 1.3.2.
  - **Energistics Transfer Protocol (ETP)** is the Energistics spec that serves as the new application programming interface (API) for all Energistics domain MLs. Initially designed to replace the WITSML SOAP API, ETP is based on the WebSocket protocol. It delivers real-time streaming capabilities and is being expanded to provide CRUD (create, read, update and delete) capabilities.
- **Information technology (IT) standards.** Energistics standard's leverage existing IT standards for various purposes. For example:
  - The Unified Modeling Language (UML) is used to develop the data model and produce the schemas and some documentation.
  - XML is used to define the data object schemas (XSD) and instances of data (XML files).
  - UUID (as specified by RFC 4122 from the IETF) is used to uniquely identify an instance of a data object.
  - HDF5 is used when needed as a companion to the XML data object to store large numeric data sets.

2.2 WITSML Data Model Overview

The WITSML UML model is implemented and organized in an Enterprise Architect Project (EAP) file, grouped into packages as shown in **Figure 2-2**. Each of the schema packages contains classes that represent the key data objects. The UML model is also the source for the XSD files (schemas), which are generated by an automated process. The zip file structure of the WITSML download (see Figure 1-1 page 8) reflects the UML file structure.

The UML model is available as part of the WITSML download. It is saved as an XMI file, which can be imported into any data modeling software tool.

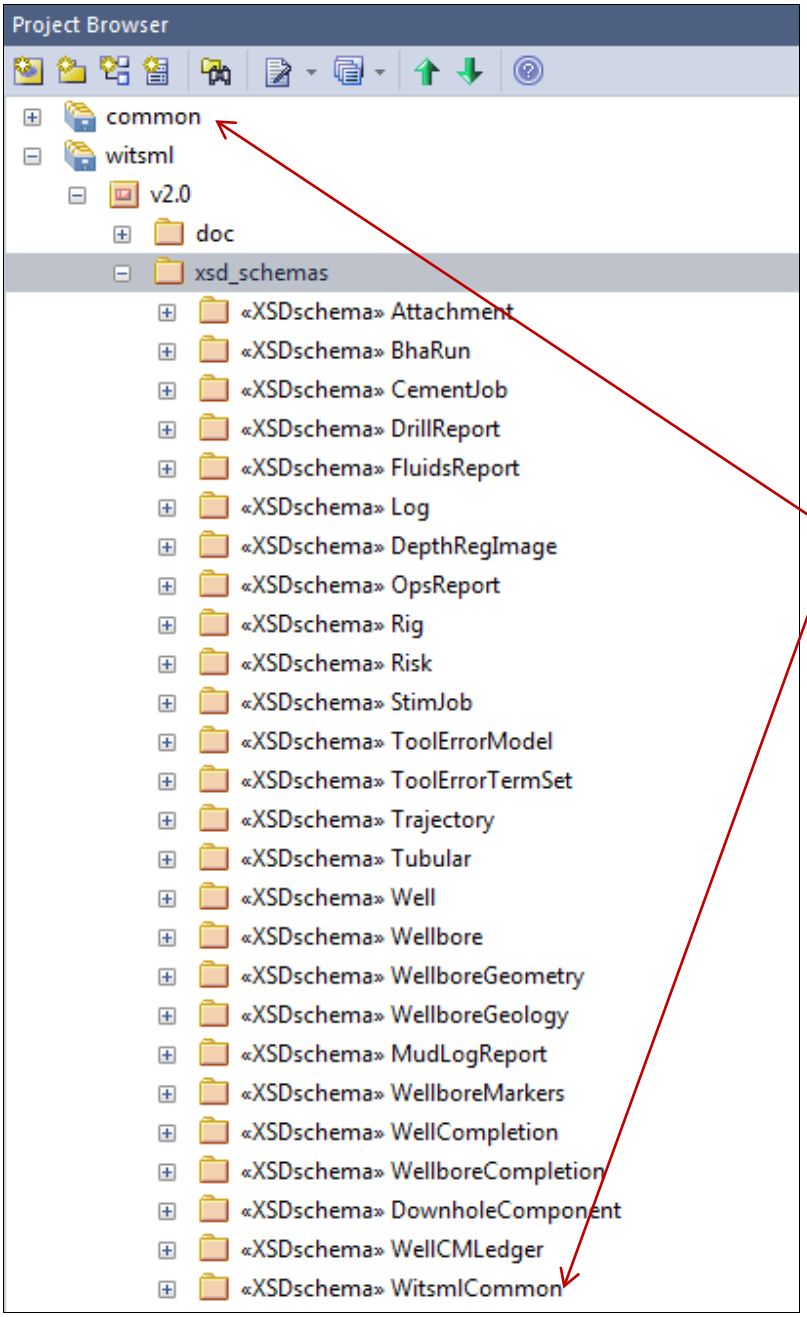

Figure 2–2. The WITSML data model includes the Energistics common, WITSML common, and a package/high-level schema for each of the main data objects.

Main packages of the WITSML UML model are listed and described here.
common contains the classes that are used by all Energistics standards, which includes classes to consistently define base data objects and references, coordinate reference systems, and units of measure. For more information, see the *CTA Overview Gui*de and *CTA Technical Reference Guide*.

witsml contains a version package (for example, ***v2.0)*** which contains these main packages:

- The **doc** class (folder), which contains a package for instance diagrams.
- The **xsd_schemas** class (folder) contains the packages within the EA project that are used to generate the XSD schemas (which are used to implement WITSML). It contains these packages:
  - **WitsmlCommon.** Shared data objects and related objects that are shared across all packages in a RESQML project. Many objects are extensions of data objects that appear in the shared Energistics-wide *common* package.
  - Package for each WITSML data object (Figure 2-2**)**, which are each described in other chapters in this manual. See the table of contents.

2.2.1 WITSML Common

Figure 2-3 shows the main data objects in WitsmlCommon.

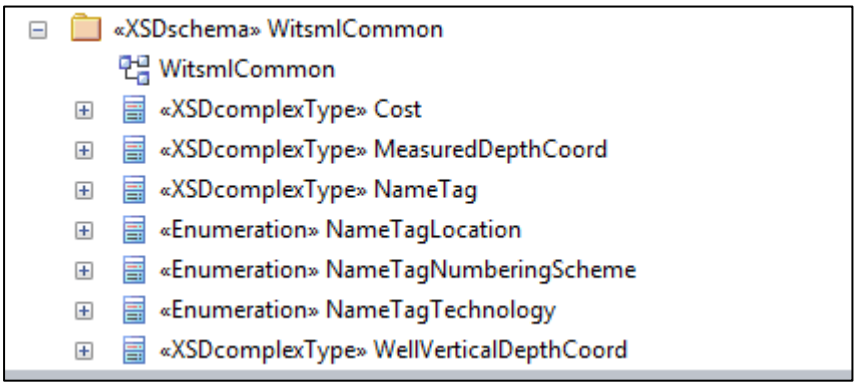

Figure 2–3. Data objects in the WITSML Common package.

2.2.1.1 Cost

The price of an item, with the currency specified.

2.2.1.2 MeasuredDepthCoord

A measured depth coordinate in a wellbore. Positive moving from the reference datum toward the bottomhole. All coordinates with the same datum (and same UOM) can be considered to be in the same coordinate reference system (CRS) and are thus directly comparable.

2.2.1.3 NameTag and Related Enumerations

In the field, various schemes can used to specify unique names and numbers for individual pieces of equipment. The NameTag object and related elements and attributes allow you to capture this information including information such as the tag technology and numbering scheme.

2.2.1.4 WellVerticalDepthCoord

A vertical (gravity-based) depth coordinate within the context of a well. Positive moving downward from the reference datum. All coordinates with the same datum (and same UOM) can be considered to be in the same coordinate reference system (CRS) and are thus directly comparable.

3 Key Concepts

This chapter describes key concepts that are important to understanding and successful implementation of WITSML. Most of these concepts are part of the Energistics Common Technical Architecture (CTA). The concepts are introduced here, with reference to more details in related documents.

3.1 WITSML Now Data Object Definitions Only

WITSML v1.4.1 and earlier included both data object definitions and an application programming interface (API). WITSML v2.0 (and future versions) defines ONLY the data model.

The Energistics Transfer Protocol (ETP) is the now the API for all Energistics domain standards.

For more information, see the *ETP Specification*.

3.2 Top-Level Data Objects and Object Identification

A top-level data object refers to those that:

- Inherit from the AbstractObject in Energistics *common*, which means they:
  - Are identified with a universally unique identifier (UUID).
  - Have a citation object that contains attributes such as (human-readable) name, creation and update information, etc.

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

3.3 Data Object Reference

In previous versions of WITSML, relationships between objects were a key component of object identify and were defined through an XML hierarchy or tree structure. For example: a log was identified uniquely in the context of a wellbore and its parent well.

Now that top-level objects are identified by UUID, relationships between objects are specified using a data object reference (DOR), a mechanism that allows any top-level object to reference another by its UUID. A log object now has a UUID and a DOR to its parent wellbore; the wellbore has a DOR to its parent well.

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

3.4 Data Object Organization

The nature of drilling and well operations drives the organization of the data model. As such, the data model (hierarchy of well, wellbore, and possible child objects of trajectory, logs, and mud logs) remains unchanged in WITSML v2.0. However, as describe in Sections 3.2 and 3.3 above, the mechanisms for specifying these relationships has changed.
Figure 3-1 shows the basic data model organization as implemented with data object reference. Each of these data objects is now a top-level object, identified by a UUID, and the relationships between the objects are specified using a data object reference (DOR).

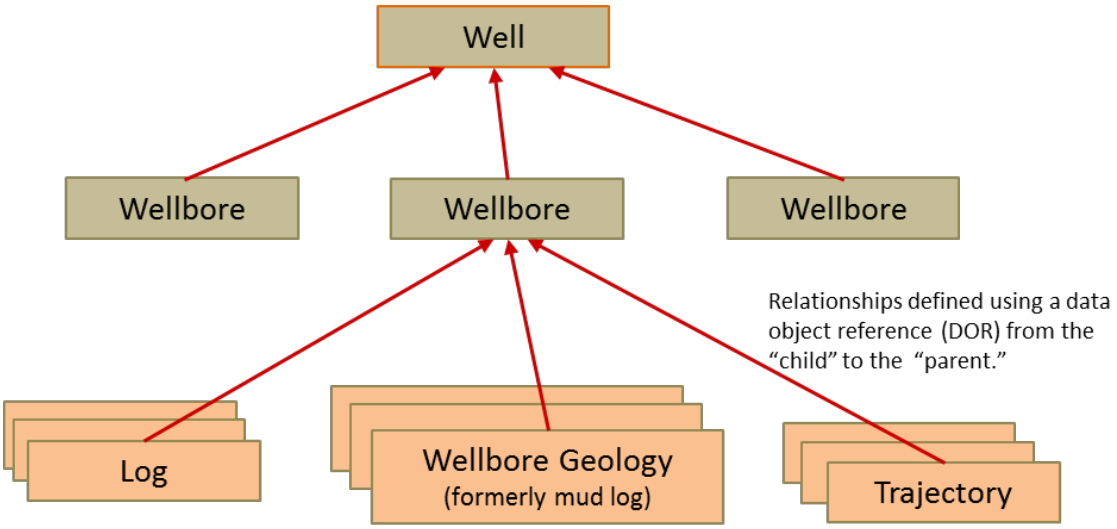

Figure 3–1. In WITSML v2.0, the basic organization of these top-level objects remains the same, but now the relationship is specified by a data object reference from the "child" its "parent."

For context, many of the other data objects reference a wellbore. For example, the report objects (e.g., drill, operations or fluid reports) must reference a wellbore. The individual chapters in this guide give necessary details.

3.5 "Growing" Data Objects and the New "Part" Object

Drilling and logging result in data objects that grow over time and as such WITSML refers to these as growing data objects, which are characterized by:

- Relatively large number of occurrences of recurring data that is indexed to time or depth.
- Additional occurrences of recurring data that are inserted (or updated) over time (which is what causes the data object to grow).

Growing data objects include: trajectories, logs, and wellbore geology (mud logs in v1.4.1). WITSML v1.4.1 included special instructions for how to manipulate and manage these objects with a SOAP API.

The Energistics Transfer Protocol (ETP) was designed to address many of the issues with the former API, so that data can be streamed in real time. To accommodate this capability required a change in the WITSML data model, namely the creation of a "part" sub-object for any object that you would want to stream data for, which includes all growing objects.

Figure 3-2 shows a portion of the UML model for the interpreted geology interval object (which is related to the wellbore geology object) and ***part_interpretedGeology*** interval which is used to stream data. Note that parent objects of part objects have a growing status field to indicate whether the object is active, closed, or inactive.

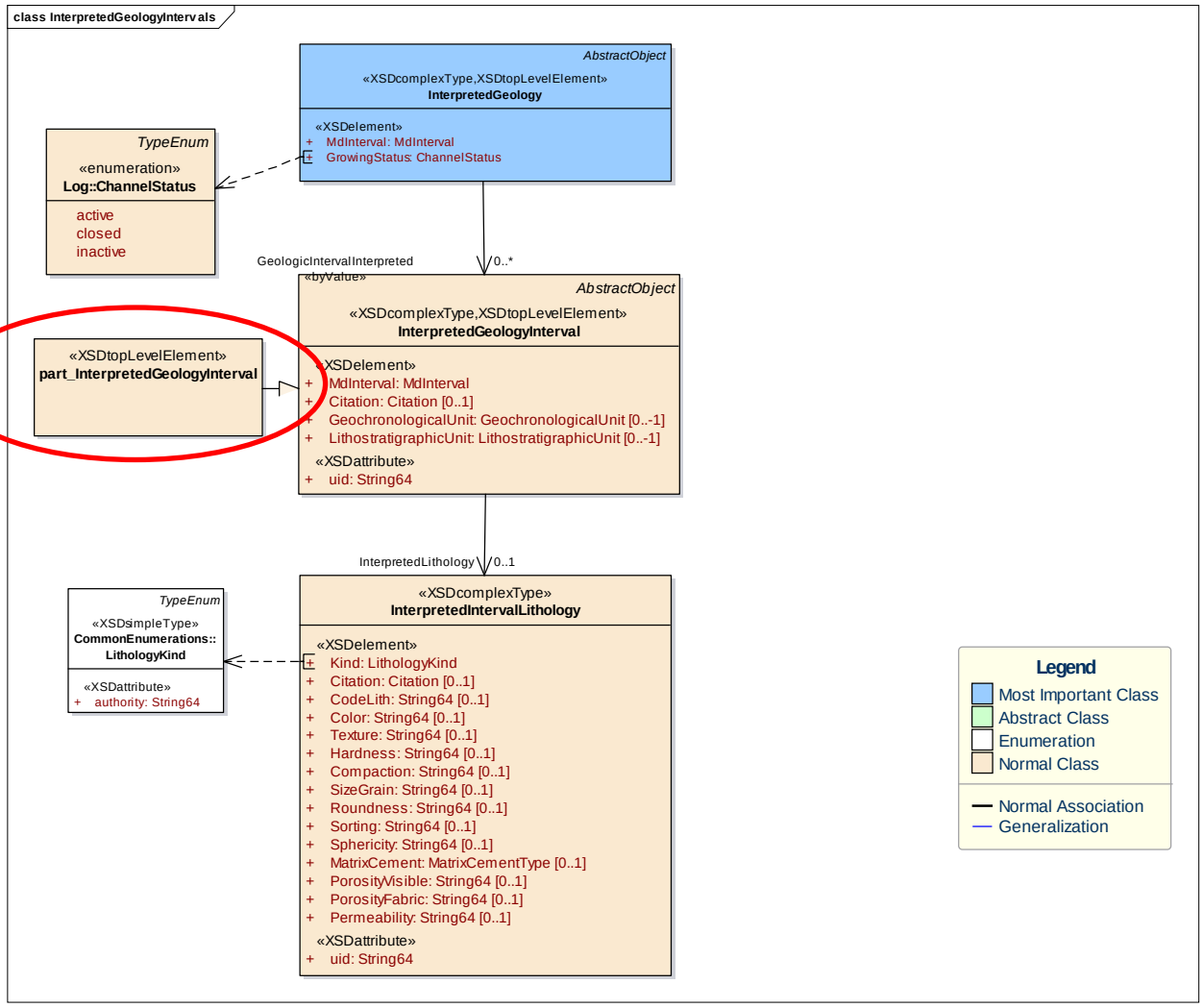

Figure 3–2. Objects named “part_” (circled above) are required for streaming of data using ETP.

4 Well, Wellbore, CRS, Trajectory, Tool Error Model, and Survey Program

This chapter provides brief definitions of a well, wellbore, coordinate reference system (CRS), trajectory and tool error model. It then explains the related WITSML data objects and how they are organized.

4.1 Basic Definitions

This section provides a brief overview of these real-world objects and their purpose in drilling and well-related operations. These real-world objects are represented as a data objects in WITSML. Each data object is discussed in more detail later in this chapter.

A **well** in the oil and gas industry is a boring in the Earth that has two main purposes:

- Exploration of the subsurface to determine viability of economic development and production of hydrocarbons.
- Production of hydrocarbons to the surface and conducting of a variety of operations and services for the safe and efficient operation of the wells.

Each well has a unique surface location on the Earth**.** Knowledge of this location and reference to it are vital for accurate and safe well-related operations and field development. The well is located on the Earth's surface (or in space) using a **coordinate reference system** (CRS).

To maximize operational efficiency and increase subsurface access, a well may contain multiple **wellbores**, which are the actual boreholes that comprise the well. A wellbore represents the path from surface to a unique bottomhole. While drilling a wellbore, measurements called **surveys** are taken at various locations as drilling progresses. These measurements determine the shape and location (curve) of the wellbore, which is referred to as the wellbore's **trajectory**. The individual locations where the measurements are taken are called **trajectory stations** or survey stations.

Like any tools, the tools use to conduct surveys are subject to inaccuracies. Errors in a survey can have significant impacts on the HSE, operations, and financial success of well. As such, **tool error models** are used to quantify the inaccuracies of a tool, so that associated risks can be defined.

4.1.1 Business Purpose: Well Data Object

The well provides the key context for activities that occur in a well and its wellbores, and the resulting data generated, such as drilling and logging. Generally we refer to business, operations and analysis activities occurring in or to a specific well (and typically wellbores in the well). Types of activities include:

- Monitor and measure well performance in hydrocarbon exploration and production. Better access to wellhead data helps well operators make better decisions that ultimately contribute to increased well productivity.
- Visualize operations and assets (**Figure 4-1**). With the help of reference points, well datum, locations, and a CRS, software can show all the wells on a map and use various graphics (e.g., green, yellow, red) to indicate performance problems.
- Accelerate delivery of well data to critical business applications and operational analytics. Data measured, gathered, and produced in wells is used in a variety of software applications from historical data capture, to analysis and performance of current operations, to predictive performance for future operations.
- Optimize the digital oil field. Companies use data generated from wells to improve efficiency and gain improved visibility to wells and assets that bring significant business benefits.

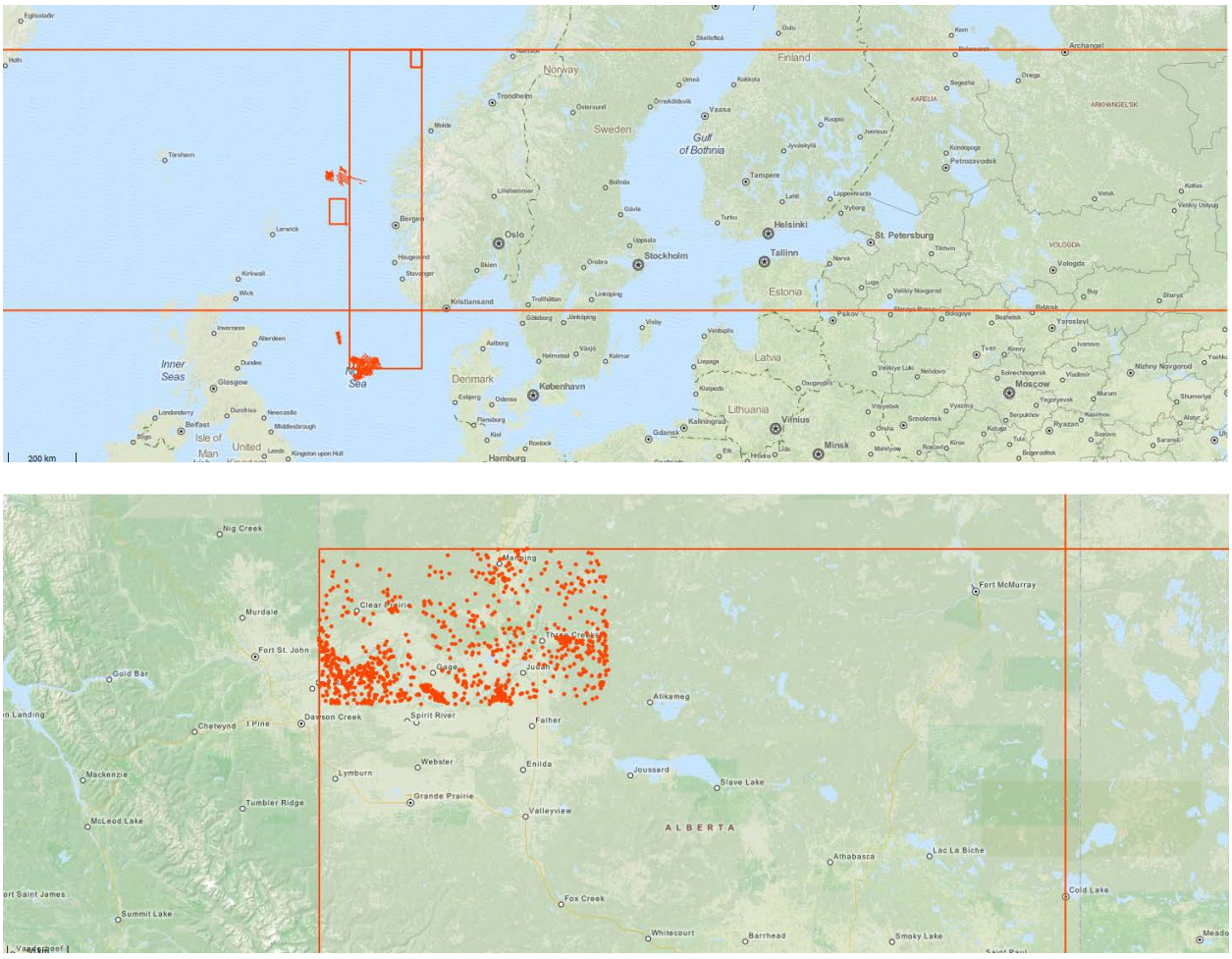

Figure 4–1. Wells visualized on map software provide improved visibility into operations and opportunities to improve well and asset performance. (*Image courtesy of Schlumberger.)*

4.1.2 Business Purpose: Wellbore Object

Data related to the wellbore is used to:

- Monitor all the drilling activity, measurement, status and all other information happening in the wellbore, information that is helpful in making some operational and business decisions.
- Compare current plan with the plan which was been made before drilling of wellbore, which could be helpful in making decision of whether to move forward with activity or not.
- Accelerate delivery of wellbore data to critical business applications and operational analytics.
- Track drilling activity, for example, where drilling operations have progressed in terms of measured depth (MD), true vertical depth (TVD), etc.

4.2 Data Model

The basic relationship between wells, wellbores and trajectories was described in Section 3.4 (page 18). As a reminder, while the nature of the relationship is hierarchical, these objects are all top-level objects so the relationships are specified with data object references from a child to its parent.

4.2.1 Well Data Object

The well data object (Well) (Figure 4-2) is used to capture the general information about a well, which is sometimes called a well header. This general information includes identifying and legal information (legal

name, license number, geographic location (country, state, county, region, and block), purpose (see enums), status (see enums) and more.

Additionally, the well has important related data objects related to its location including a well datum, well location and well reference point.

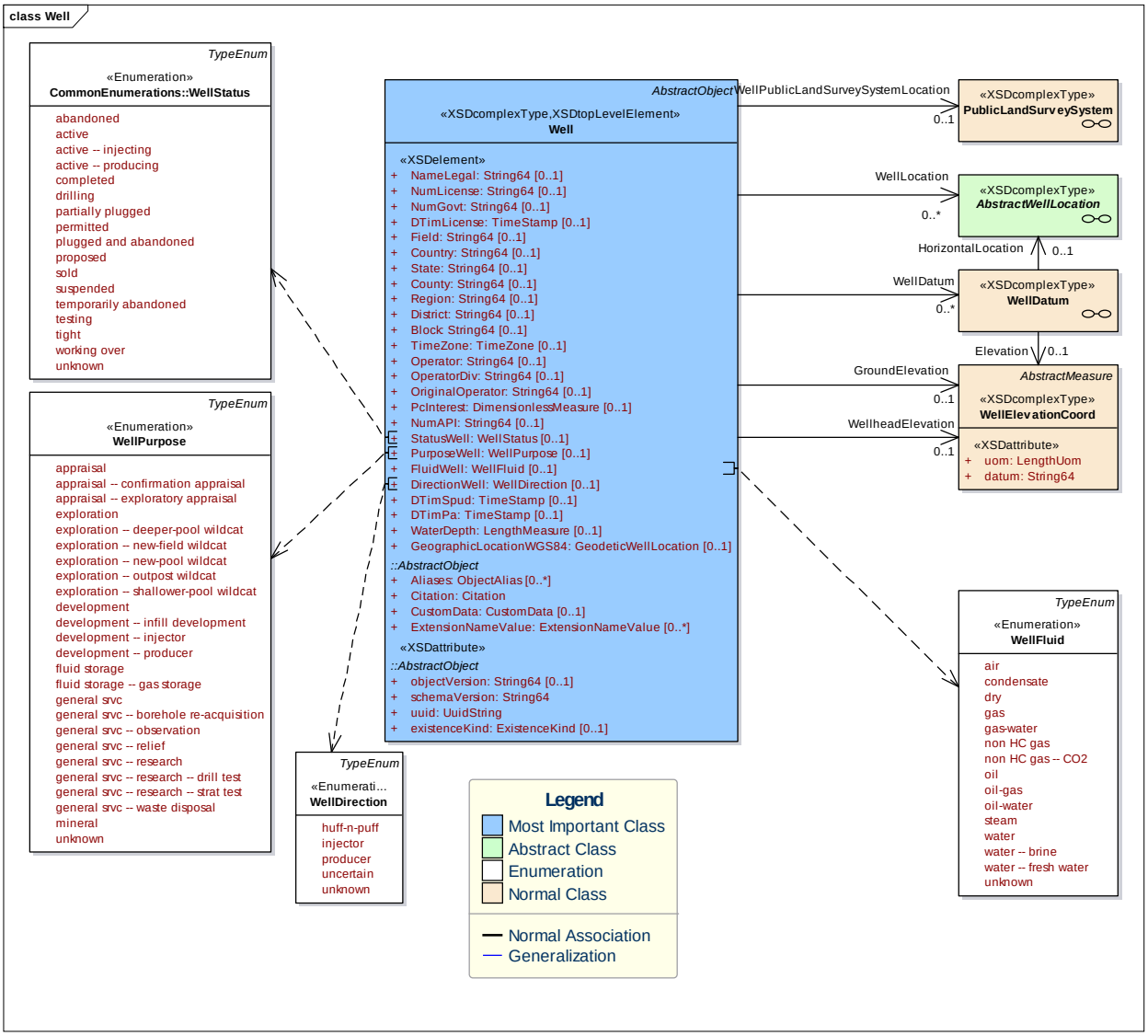

Figure 4–2. UML diagram of the well data object.

4.2.2 Wellbore Data Object

The wellbore (Wellbore) data object (**Figure 4-3**) is used to capture the general information about a wellbore, which is sometimes called a wellbore header. A wellbore represents the path from surface to a unique bottom hole. A well has one or more wellbores.

Like the well object, the wellbore object provides context. The wellbore references it parent well object using a data object reference. The wellbore serves as the parent of trajectory, log and wellbore geology data objects (which reference their parent wellbore using data object reference).

The wellbore object is used to capture data such as:

- Purpose: appraisal, injection, mineral, observation, producer, research, unknown, waste disposal, etc.

- Shape: deviated, horizontal, vertical, build and hold, unknown etc.
- Status of current activity such as: drilling, active, sold, proposed, testing, tight, suspended, etc.

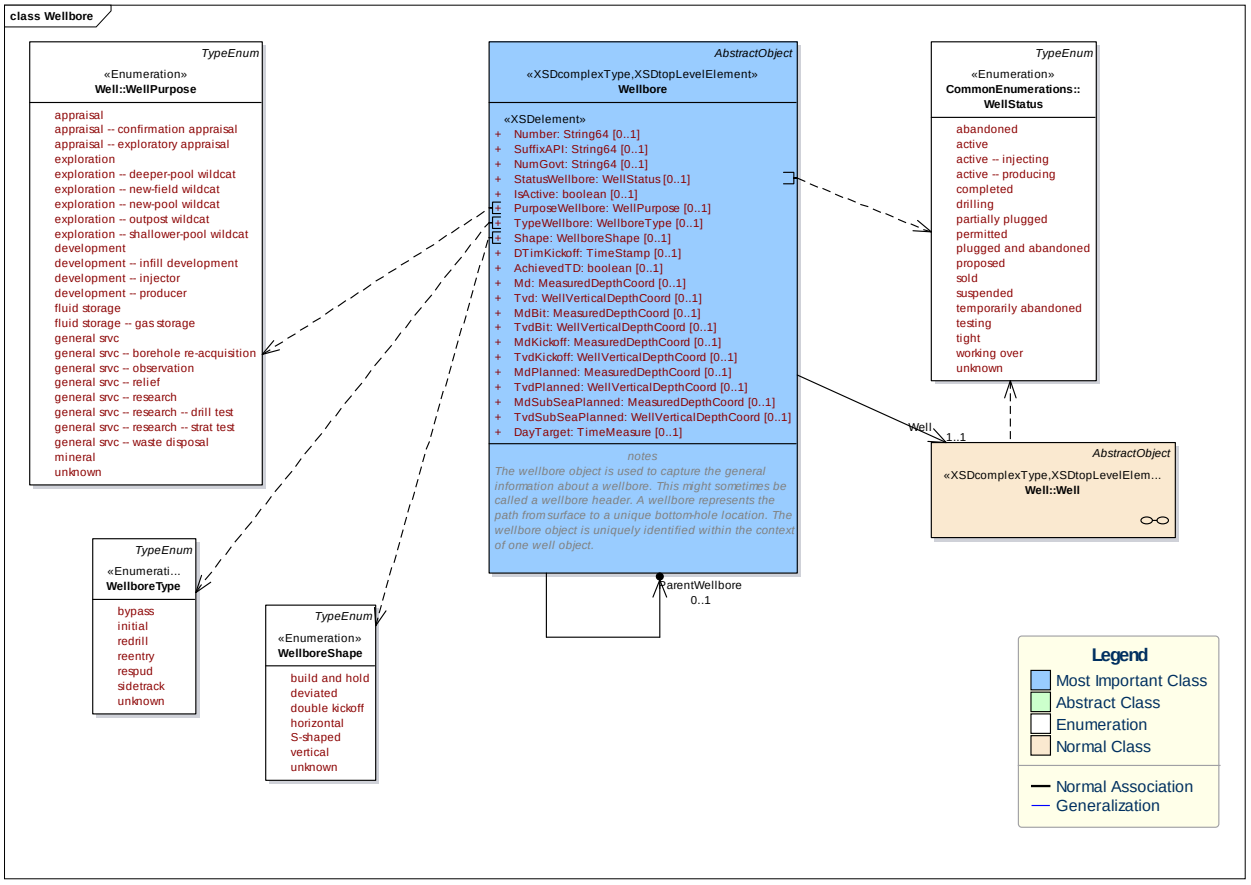

Figure 4–3. UML diagram of the wellbore data object.

4.2.3 CRS Data Object

Knowledge of a well's location (or points along its trajectory) and reference to it are vital for accurate and safe well-related operations and field development.

A coordinate reference system (CRS) uses a series of numbers to locate entities or objects (such as wellheads) in space. A CRS is a set of axes (a coordinate system (CS)) that is given a location through being related in position and orientation to a specific object. The relationship between the CS and CRS is defined through a datum.

For more information about the CRS, see the *Energistics CTA Overview Guide*.

4.2.4 Trajectory Data Object

The trajectory data object (Trajectory) (**Figure 4-4**) is used to capture information about a directional survey in a wellbore. The data from the survey actually defines the wellbore trajectory. A trajectory is a top-level data object and must reference a parent wellbore using a data object reference.
While drilling a well, contractors conduct measurement-while-drilling (MWD) surveys. MWD typically takes measurements of wellbore inclination from vertical and magnetic direction from north. Using basic trigonometry, a 3D path of the wellbore can be produced. The location in the wellbore where a measurement is taken is referred to as a survey station or trajectory station (TrajectoryStation). The set of data collected at all stations defines the wellbore trajectory.

4.2.4.1 Business Purpose

Data in the trajectory data object is used to:

- Identify the exact wellbore position, drill bit information, and related directional data.
- Drill in a planned direction into the formation which contains hydrocarbons (the target zone) and to keep the well within authorized (lease) areas (when drilling close to lease boundaries).
- Transmit all the survey stations in real time to a directional driller to help “steer” the well towards the target zone.
- Record deviations from the planned trajectory (and correct as necessary and possible); the information collected is referred to as a deviation survey.

4.2.4.2 Data Model

The trajectory data object:

- Must reference its parent wellbore.
- Is a growing data object because as measurements are taken (at increased depth) that data is added to the trajectory.
  - Includes a part_TrajectoryStation object to support streaming data with ETP.
  - Has a GrowingStatus, which describes the growing status of the Trajectory (active, inactive or closed)
- Contains this data:
  - Trajectory stations and measurement data recorded at each.
  - The type of measurement tool used and related information.
  - Trajectory station status, type, and related data for specific types.
  - Directional drilling information and measurements.
  - Calculated values and related uncertainty ranges.
- References a tool error model, which identifies inaccuracies of instruments or measurement systems related to the trajectory.

4.2.4.3 Important Elements and Best Practice

It is important to include certain elements in the trajectory and trajectory station data objects for them to be both valid and of practical use. Most required elements relate to object identification (UUIDs), etc. and high-level concepts, which are implemented as requirements in the schemas. Other elements represent a “best practice” that is helpful for the Trajectory data object to be properly interpreted.

Trajectory Object, Best Practice:

- **Definitive**: Identifies a final, definitive trajectory, as distinguished from other temporary or partial ones.
- **MdMn** and **MdMx**: minimum and maximum measured depths represented by the contained trajectory stations.
- **ServiceCompany**: name of the contractor who provided the service.

Trajectory Station:

- Required:
  - **uid**: unique identifier within the trajectory scope, for a trajectory station instance.
  - **TypeTrajectoryStation**: specifies the type of a directional survey station.
  - **Md**: measured depth, along the trajectory path, identifying where the survey measurement was taken.
- Best Practice:

Note: If inclination (Incl) and azimuth (Azi) are not provided, then Tvd, DispNs, DispEw should be; otherwise, it isn't possible to properly interpret the trajectory path.

- **Incl**: hole inclination angle, from vertical line descending from the well head.
- **Azi**: hole azimuth, corrected to a well's azimuth reference.
- **Tvd**: true vertical depth of the measurement from vertical.
  - DispNs: north-south offset from vertical, positive to the north.
  - DispEw: east-west offset from vertical, positive to the east.
- **DTimStation**: date and time the station was measured or created.
- **StatusTrajStation**: status of the station: open, rejected or position.

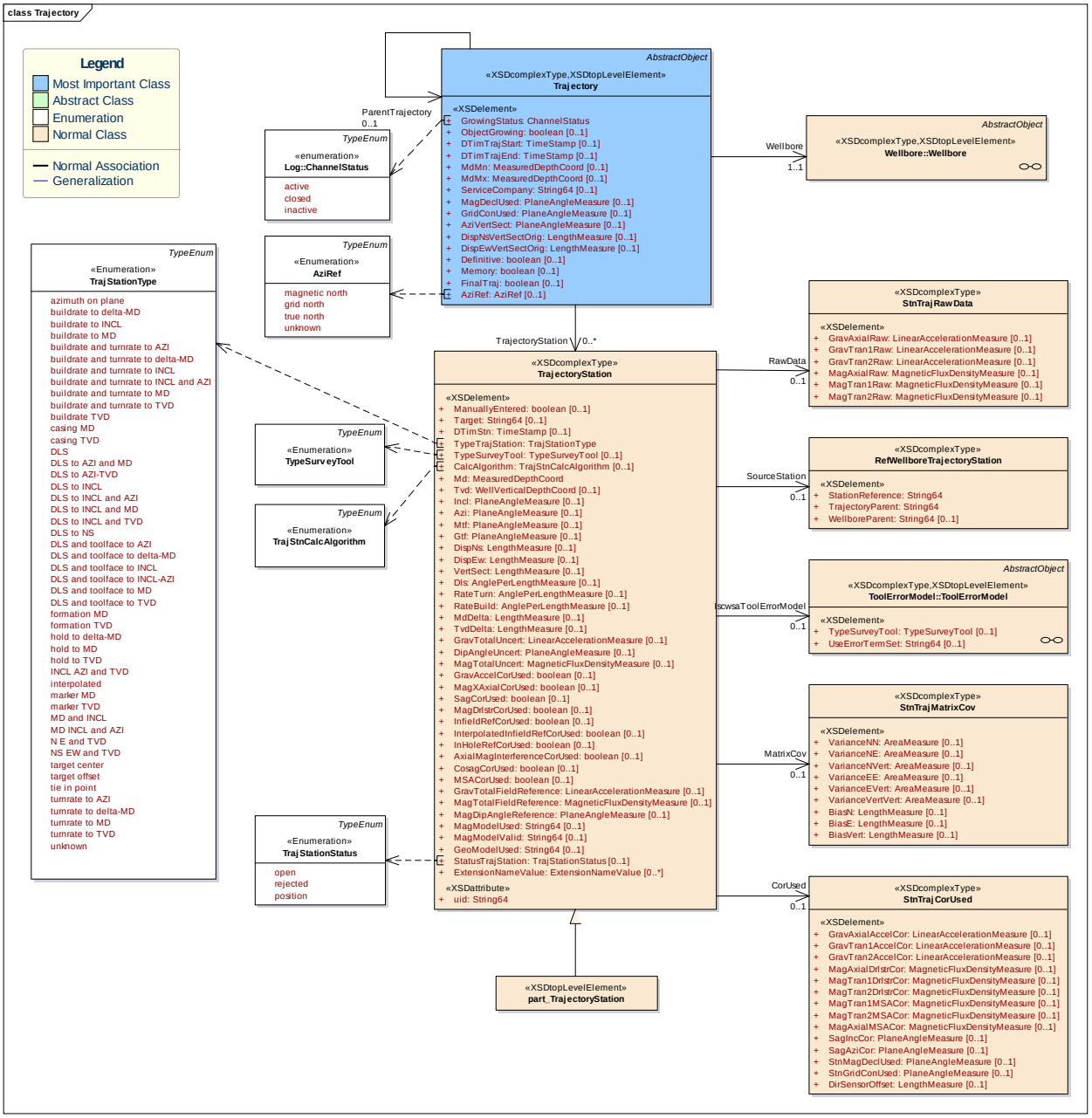

Figure 4–4. UML diagram of the trajectory data object.

4.2.5 Tool Error Model and Term Set Data Objects

Tool error models are used to specify the inaccuracies of instruments or measurement systems. The instrument inaccuracies are often referred to as “modellable errors” whereas the practical or usage mistakes are referred to as “unmodellable errors”. These error models define how various error sources affect the observations in the well and thus the positional uncertainty along the wellbore. Industry-standard error models are used to ensure that best practice has been adopted before assuming that the calculated uncertainties are representative. A tool error model term set provides the error terms that are used in the tool error model.

Work in the industry related to error models and the design of these data objects is based on work by the Industry Steering Committee on Wellbore Survey Accuracy (ISCWSA), which is affiliated with the Society of Petroleum Engineers Wellbore Positioning Technical Section (SPE-WPTS).

4.2.5.1 *Data Model*

The tool error model (ToolErrorModel) (**Figure 4-5**) data object is used to define a surveying tool error model. The tool error term set (ToolErrorTermSet) (**Figure 4-6)** data object is used to define a set of surveying tool error terms that may be used in a tool error model.

Important points:

- The tool error model and term set are each top-level objects.
- A trajectory references a related tool error model.
- The tool error term set references a tool error model.

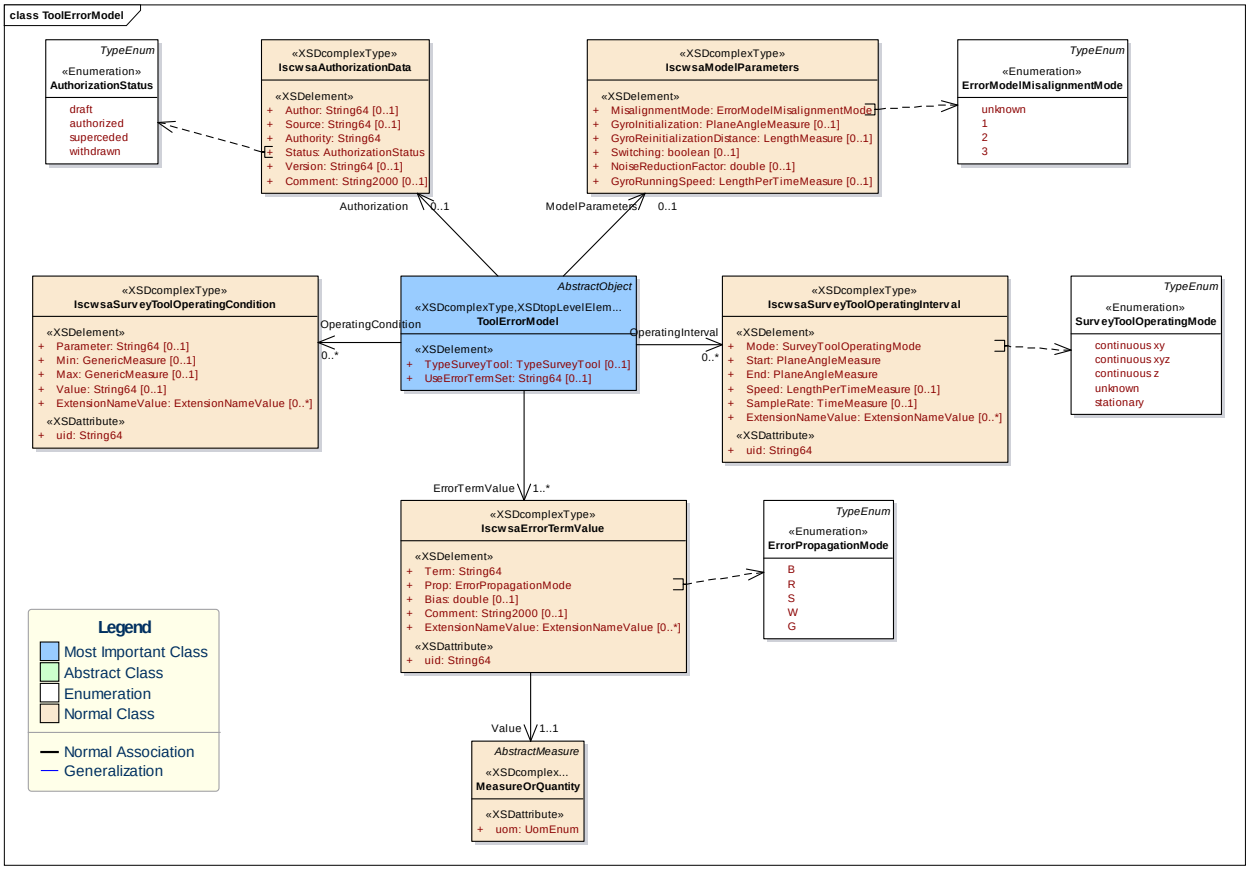

Figure 4–5. UML diagram of the tool error model data object.

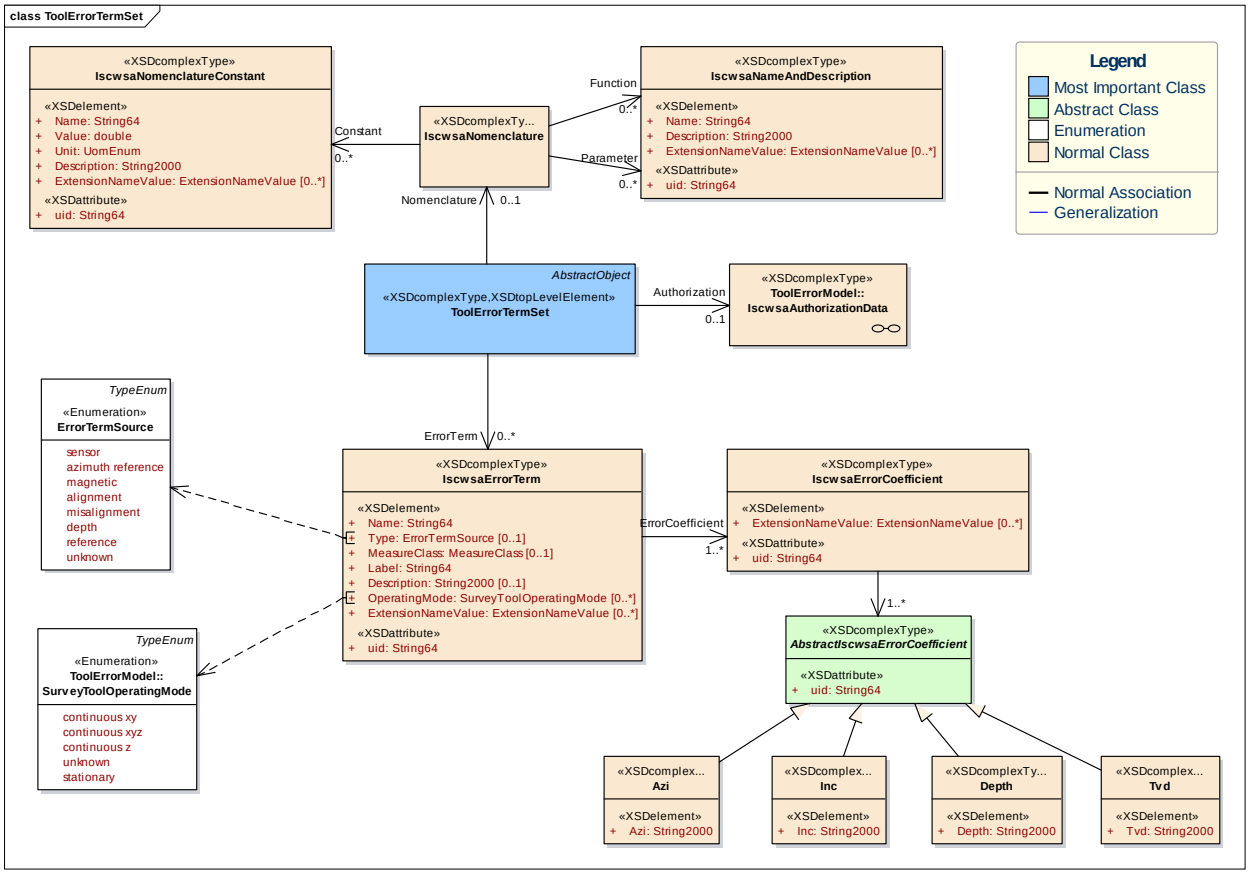

Figure 4–6. UML diagram of the tool error model term set.

4.2.6 Survey Program Data Object

The survey program (SurveyProgram) data object (**Figure 4-7**) captures information about the nature, range, and sequence of directional surveying tools run in a wellbore for the management of positional uncertainty. This object is uniquely identified within the context of one wellbore object.

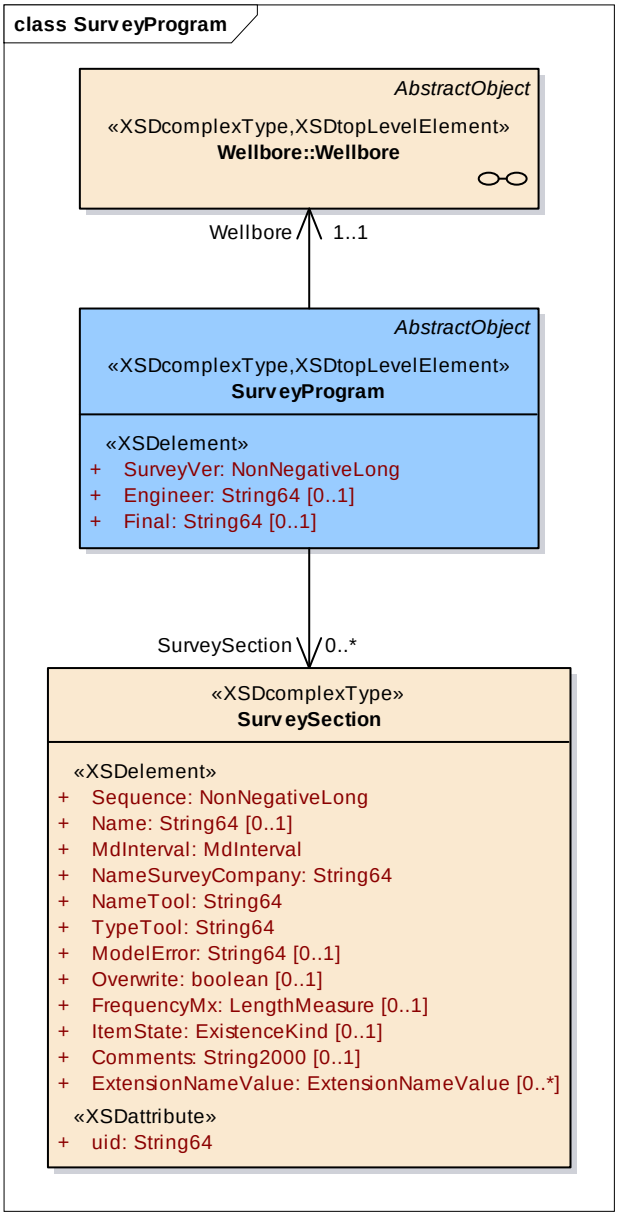

Figure 4–7. UML diagram of the survey program data object.

5 Log Data Object

This chapter provides brief definitions and an overview of logs and their use in the oil and gas exploration, production, and drilling operations.

It then explains the WITSML Log data object, key concepts, and how it has been designed to work. The Log data object underwent major redesign in v2.0. The main driver for this design change was so that realtime data could be streamed using the Energistics Transfer Protocol (ETP).

5.1 Overview of Logs

In the oil and gas industry, the term “log” can be used as either an action (verb) or a thing (noun). In very general terms, the action refers to measuring something related to drilling operations. A wide variety of surface and downhole sensors, hardware, and software technology are used to take these measurements.

The measurements occur over a measured time period or at different depths in the well (or sometimes both). The “things” that are measured include operational parameters, drilling progress and parameters, and formation properties. Sometime the measured properties or data are direct measures used in operations (e.g., rate of penetration (ROP) or hole size) and sometimes they are indicators (e.g. electrical resistivity as an indicator of hydrocarbon presence) or as inputs to additional analysis.

The output of the logging process—traditionally a very long piece of paper that records values at each time or depth index—is also referred to as a log. Today, this information is captured digitally, but still consists of a time or depth index and an associated measurement at that index. Logs are often informally referred to as “curves” (because when the data is plotted—an index and its associated value—it produces a curve). A Log may consist of 1 curve or a collection of many curves.

The volume of log data is huge (both historical and current) and has many vital uses. So log data is transferred frequently—as it is collected and used for analysis in realtime drilling operations, for overall field evaluation, and for future use when evaluating similar prospects and/or operational conditions.

5.2 Log Data Object

The WITSML Log data object is used to capture and transmit the data collected from a variety of logs. It is use to capture and transfer data for:

- MWD parameters such as pressure, temperature, and survey stations
- Surface or drilling parameters like ROP, RPM, WOB, etc.
- LWD parameters such as gamma rays, sonic, resistivity, etc.

The Log object is considered a growing object because as measurements are taken (over time or increased depth) that data is added to the Log.

However, in WITSML, a log object is constrained to capture curves from a single pass. This object is uniquely identified within the context of one wellbore object.

The Log WITSML object containing multiple logging parameters as channels can be transmitted in real-time and can be indexed to either Time or Depth. This object is declared as *growing object.* This will grow slowly with time as the wellbore is drilled.

5.2.1 Business Purpose

Data in the Log data object are used to:

- Support quick decision making in realtime drilling operations. WITSML is used to transfer surface and downhole parameters in real time to drilling operations systems.
- To improve geologic interpretation by transferring surface and well log parameters indexed to hole depth to.

5.3 Log Organization

The Log data object underwent major redesign in v2.0. The main driver for this design change was so that realtime data could be streamed using ETP. Figure 5-1 is a UML diagram of the v2.0 Log. At a high level, a Log is now organized as follows:

- **Channel:** corresponds roughly to the LogCurveInfo structure in WITSML v1.4.1.1 and directly corresponds to the ChannelMetadataRecord structure in ETP. In historian terminology, a Channel corresponds directly to a tag. Channels are the fundamental unit of organization for WITSML logs.
- **ChannelSet**: grouping of Channels with a compatible index for some purpose. As discussed earlier, each Channel effectively has its own index. By compatible index, we simply mean that all of the Channels are either in time or in depth using a common datum
- **Log:** primarily a container for one or more ChannelSets. Most of the information is now at the ChannelSet level. The concept of multiple ChannelSet in a single Log is significant change from WITSML v1.4.1.1 where each Log represented exactly one group of curves and their data

This chapter further explains these concepts and guidelines for working with this new design.

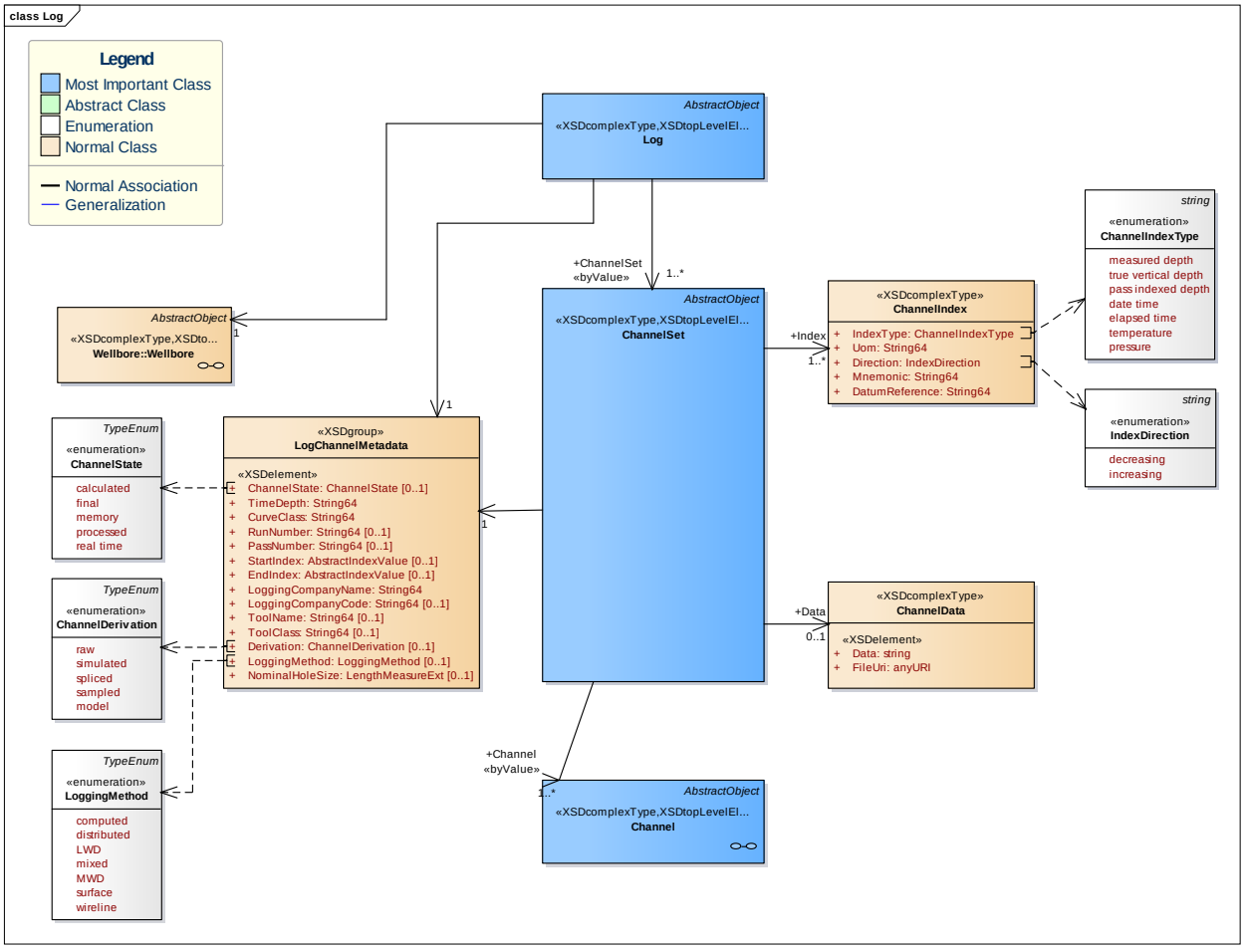

Figure 5–1. UML diagram of log data object.

5.3.1 Channel

The single biggest change occurring in WITSML 2 is the introduction of Channel as a top-level object in the store (Figure 5-2). A Channel is the measurement of one physical property versus a range of some other dimension, usually time, depth or both. Within a Channel, individual measurements are called data points. Each data point is a 2-tuple of an index value(s) and a data value.

In WITSML 1.x versions, the concept now called Channel existed only as child element (called a LogCurve) of a Log object. The uid of curves was not unique except within the context of the Log. With WITSML 2, a Channel has its own schema and exists in the store independent of any Logs that reference it. This means that a Channel can exist but not be in any Logs, and that a given Channel can appear in multiple Logs.

Like all top-level objects, Channel inherits from AbstractObject and thus shares the common Citation metadata elements of all Energistics data objects (e.g., guid, title, creation date, etc.).

Channels in WITSML also have a required relationship to a Wellbore.

As a consequence of being top-level, there is no longer the concept of a shared index between multiple channels, at the acquisition and storage level. Each channel is responsible for storing its own data as a set of indexed values. The notion of data alignment does not exist at the channel level (though it can show up in the Log object as discussed below). Of course, individual tools may or may not collect data on an index-aligned basis, and any given store may or may not chose to optimize its channel data storage based on common index values, but this is entirely implementation dependent and not part of this spec.

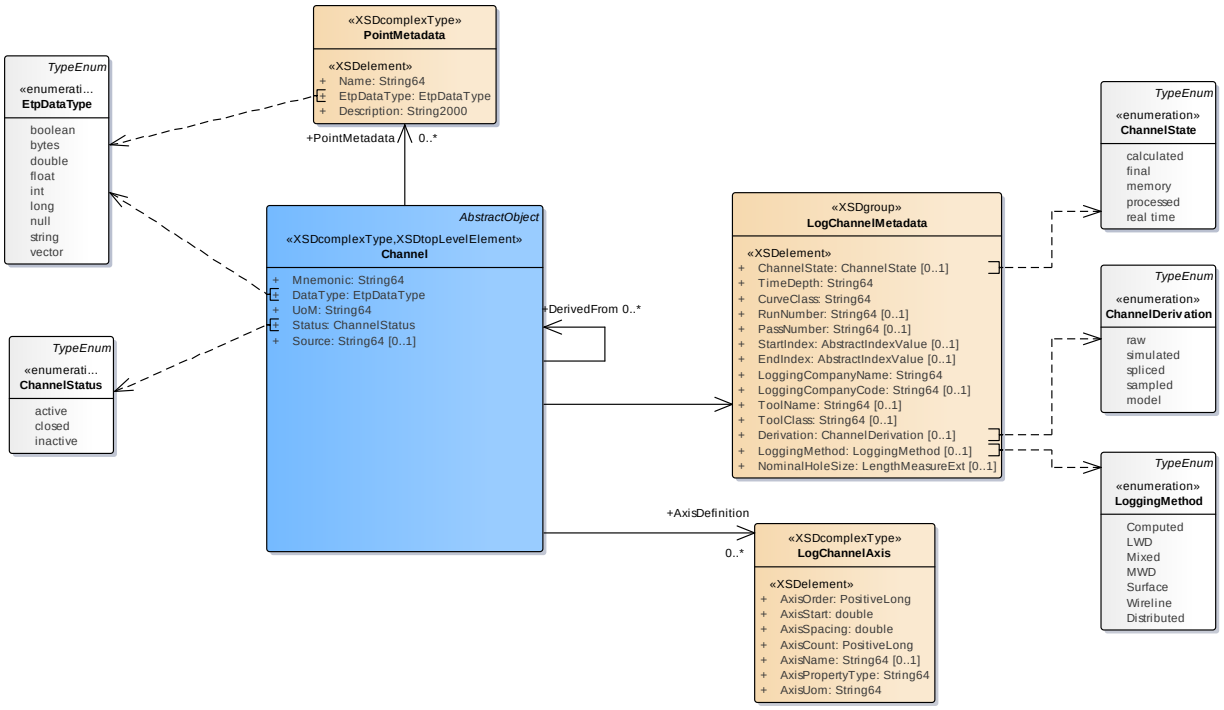

Figure 5–2. Channel model.

5.3.1.1 LogChannelMetadata

The WITSML SIG formed a working group to study the business requirements for organizing Log data. Stemming from the work of that group, WITSML 2 now has the concept of a metadata record that describes logging information in a way that will make it easier to find logs in a consistent repeatable way, across multiple vendors and service companies. This metadata appears on all Channel objects. The full

description of all of the elements in the metadata record can be found in the *WITSML Technical Reference Guide*.

The metadata record also appears on higher aggregations of Channels (such as ChannelSet and Log). In these cases, to aggregate various metadata tags, follow these rules:

1. In general, if all the Channels in the aggregate have the same value for a given metadata tag, then that value is promoted to the aggregate object. For example, if a ChannelSet is created where all Channels have the same LoggingCompanyName and Code, then those values appear in the metadata for the ChannelSet.
2. For StartIndex and EndIndex, the aggregate object values are a single range consisting of the minimum and maximum values from all of the input Channels.
3. All other fields can be replaced with either a '*' character (indicating it contains multiple values) or a comma-separated list of the distinct values found among all of the Channels in the set.

5.3.1.2 ChannelDerivation

WITSML 2 introduces the notion of Channel Derivation. All channels are tagged with metadata that indicate the derivation of the data values on the Channel. If a Channel's derivation is 'raw', then it is an original copy of all points as recorded by the channel source. The possible values for ChannelDerivation are:

raw	The channel contains raw measured data, directly from sensors.
simulated	Channel values are simulated.
spliced	Channel values are the result of 'splicing' or concatenating one or more channels of the same measure class. This can be used, for instance, to create a top to bottom composite channel from the values in multiple depth logs. (See Figure 5-3 below.)
sampled	The Channel values are derived from sampling or decimating another channel. This can be used, for instance, to create derived channels that can send sampled data to visualization screens using less bandwidth.
model	Channel values are based on some computed or modeled results of values in one or more channels. Could be a moving average from a channel, addition of two channels, etc.

In addition to the ChannelDerivation tag, the Channel object has an optional list of pointers to the Channels from which it was derived. Other than this metadata, the WITSML spec and the Channel object provide no specific information on how the splicing or sampling is actually done, which points came from which of the source channels, or what calculations are performed, etc. It is simply metadata for tracking purposes. In particular, a given store implementation may or may not optimize the storage of spliced channels such that the points are only stored once and re-assembled when the data for the spliced channel is retrieved.

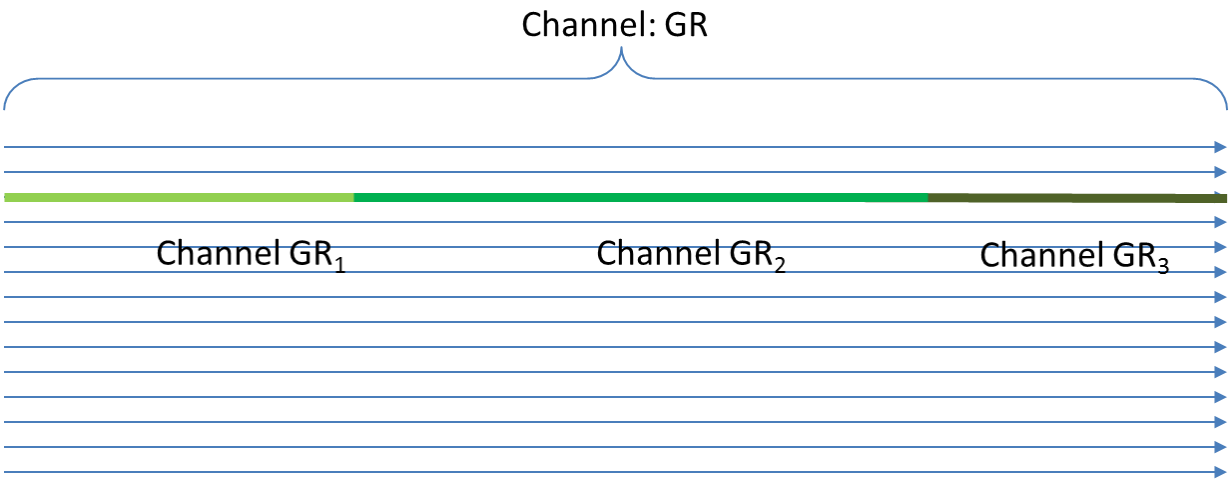

Figure 5–3. Spliced channels.

5.3.1.3 Data Point Metadata

WITSML 2.0 introduces that concept of data point metadata. These are additional data on a single data point, normally used to describe concepts such as quality, confidence, editor (if a point has been changed from the recorded value), etc. The Channel object contains a list of metadata data values that may appear on any given point in the Channel.

At this time, there is no standardization of the names or datatypes of point metadata. This may be addressed in a future release of WITSML.

5.3.1.4 Array Data

The measured values in a log channel are often a single number or string for a given index value, but it could also be an array. Common log channels which are arrays are array resistivity types like FMS and wave data from a sonic tool (where the sonde listens for a period of time at a given depth). These are simple 1D arrays but an array type sent as a log channel could be multi-dimensional. To support multidimensional arrays sent as log channels, the Channel object has member class, which defines the meaning of the axes of that array and how it will be serialized into XML (or on a wire if ETP is being used).

This LogChannelAxis class contains metadata which defines the array structure of the compound value. A single instance of LogChannelAxis defines one axis of an array type used in a Log Channel. An axis can have a name, and must have a property (time, measured depth, etc.) and a unit of measure (meters, seconds, etc.).

Note that the index of the log channel itself is not one of these axes. The multidimensional array is given at an occurrence of the primary index of the log channel. So to continue the previous example, the measured depth for the sonic wave data is not part of an array; the array is a vector (1D array) of evenly-sampled sonic intensity values.

Each axis must be a regularly sampled floating point number, so there are attributes for the starting axis index value, the increment along the axis, and the number of elements along this axis (dimension) of the array (AxisCount).

Here is a sample array:

		Time (ms)				
		1.0	2.0	3.0	4.0	5.0
Angle (deg)	0.0	.90	.82	.83	.81	.88
	90.0	.88	.85	.82	.87	.80
	180.0	.79	.84	.88	.87	.83
	270.0	.78	.76	.83	.77	.90

The identifier of any of these points is the coordinate stated as an x,y pair, or in this case a time/angle pair like (1.0, 10.0) [the value .9] or (2.0, 30.0) [the value .77]. If we generalize the example, an address in an array is like a spreadsheet with x being a column and y being a row. So instead of calling the value .9 as (1.0, 10.0) in a spreadsheet we would call it A1 or the .77 value would be B3. For the benefit of computer programs the column and row indexes are referred to as integers, rather than letters for columns and integers for rows. The start and increment are floating point numbers, and the count is an integer.

5.3.2 ChannelSet

A ChannelSet is grouping of Channels with a compatible index for some purpose. As discussed earlier, each Channel effectively has its own index. By compatible index, we simply mean that all of the Channels are either in time or in depth using a common datum.

Figure 5–4. Channelset is a grouping of channels with a compatible index, for a specific purpose.

There are two main purposes for ChannelSets.

- The ChannelSet can be used as a URI (see the Energistics Identifier Spec) for the purposes for the ETP real-time Channel description and streaming. By passing a ChannelSet URI in the ChannelDescribe message, the consumer can receive metadata on all of the Channels in the set and then begin streaming the channels of interest.
- The ChannelSet can be included in a Log (as described below) in which case it functions as the metadata description of the log data that will be written in ASCII form in the Log file.

5.3.2.1 ChannelData Element

The bulk data for an individual ChannelSet within a Log is sent as a JSON-compatible set of arrays within the ChannelData XML element. By JSON-compatible, we simply mean the data block could be processed by a JSON.parse() function. The data block (at this time) does not use the full features of JSON.

The size, dimensionality and expected datatype of the arrays can be derived from the descriptions of the Channels in the ChannelSet. To serialize data, follow these rules:

1. Whitespace is not significant, but it is recommended that individual rows are on a single line, for readability. This rule becomes irrelevant for large array channels.
2. Arrays are delimited by square brackets; array elements are delimited by commas.

```
[123, 456, 7.89]
```

3. Strings are in double quotes with backslash escaping.

```
[123, “hello”]
```

4. As in JSON, null values are represented by the word null itself (case sensitive). This is a significant change from WITSML 1.x, where consecutive commas designated no data.
5. Dates and times (in particular, time index values) are strings in ISO 8601 format. As in ETP, all time index values must be UTC.
6. In general, where the expected size of an array is known from the metadata and the actual size of the array is smaller, then the remaining elements of the array are assumed to be null. For instance, if a set has 12 channels and a given data has 8 elements, the final four channel values are null.
7. The entire data block is enclosed in a top-level array which represents the entire data set.

```
[
]
```

8. Each element of that array is itself an array representing one ‘row’ of log data. That is, an index (or indexes) and all of the channel values for that index. These correspond to one <data/> tag in WITSML 1.4.1.1 logs.

```
[
      [First row of data],
      [Second row of data],
      [Third row of data]
]
```

9. Within a data row, the index value(s) are distinctly separated from the channel values as an array. This is also in contrast to the earlier formats where the index was simply another channel that was identified as being the index and had to be identified by array position.

```
[
      [[Index Values], [Channel Values]],
      [[Index Values], [Channel Values]],
      [[Index Values], [Channel Values]]
]
```

Putting together these requirements and rules, a section of a simple gamma log, with a primary depth index, a secondary time index, and two data channels GR1AX, GR2AX might look like this:

```
[
[[2496.840,”2009-06-22T05:21:03.0000000Z”], [53.9, 49.3]],
[[2497.500,”2009-06-22T05:34:30.0000000Z”], [55.3, 50.6]],
[[2498.053,”2009-06-22T06:59:31.0000000Z”], [54.9, 50.2]],
[[2500.099,”2009-06-22T06:39:56.0000000Z”], [56.8, 51.9]],
[[2500.621,”2009-06-22T07:06:02.0000000Z”], [50.5, 46.2]],
[[2501.001,”2009-06-22T06:46:28.0000000Z”], [52.9, 48.4]],
[[2501.141,”2009-06-22T06:26:54.0000000Z”], [60.6, 55.5]],
[[2503.449,”2009-06-22T06:13:51.0000000Z”], [null, 49.3]],
[[2503.774,”2009-06-22T11:15:19.0000000Z”], [50.5, 46.2]],
[[2504.228,”2009-06-22T06:20:24.0000000Z”], [59.2, 54.1]],
```

```
[[2504.237,”2009-06-22T07:12:31.0000000Z”], [52.5, 48]],
[[2504.272,”2009-06-22T06:33:26.0000000Z”], [52.5, 48]]
]
```

10. When the Channel value itself is an array, it is simply written that way. If the Channel array is multi-dimensional, the dimensionality is not reflected in the data block. It is simply an array whose size is the product of the size of all dimensions described in the AxisDefintions of the Channel.
11. If a Channel contains value-level metadata (such as quality, confidence, etc.) as described in the PointMetadata element of the Channel, then the values for that channel are themselves an array in which the first element is the Channel value and the remaining elements are the point metadata. Using the example above, if the GR1AX channel described point metadata of ‘confidence factor’ as floating point number, it might look like this:

```
[
    [[2496.840,”2009-06-22T05:21:03.0000000Z”], [[53.9, 0.9], 49.3]],
    [[2497.500,”2009-06-22T05:34:30.0000000Z”], [[55.3, 0.9], 50.6]],
    [[2498.053,”2009-06-22T06:59:31.0000000Z”], [[54.9     ], 50.2]],
    [[2500.099,”2009-06-22T06:39:56.0000000Z”], [[56.8, 0.9], 51.9]],
    [[2500.621,”2009-06-22T07:06:02.0000000Z”], [[50.5, 0.9], 46.2]],
    [[2501.001,”2009-06-22T06:46:28.0000000Z”], [[52.9, 0.9], 48.4]],
    [[2501.141,”2009-06-22T06:26:54.0000000Z”], [[60.6, 0.9], 55.5]],
    [[2503.449,”2009-06-22T06:13:51.0000000Z”], [null       , 49.3]],
    [[2503.774,”2009-06-22T11:15:19.0000000Z”], [[50.5, 0.9], 46.2]],
    [[2504.228,”2009-06-22T06:20:24.0000000Z”], [[59.2, 0.9], 54.1]],
    [[2504.237,”2009-06-22T07:12:31.0000000Z”], [[52.5, 0.9], 48]],
    [[2504.272,”2009-06-22T06:33:26.0000000Z”], [[52.5, 0.9], 48]]
]
```

Some things to note.

- If PointMetadata exists on the ChannelDefinition, then all values MUST appear as an array in the data set, whether the metadata exists for that specific point or not.
- If the entire point does not exist, you can still simply use null.
- If a given point has no metadata (as in the 3rd row), then it can simply be left off.

12. It is a recommended practice to enclose the entire data block in a CDATA section, particularly if there are Channels with human-editable contents, such as comments or messages. For this purpose it is import to leave whitespace between the square brackets of the CDATA section and those of the array.

```
<![CDATA[

[
     [[2496.840,”2009-06-22T05:21:03.0000000Z”], [[53.9, 0.9], 49.3]],
     [[2497.500,”2009-06-22T05:34:30.0000000Z”], [[55.3, 0.9], 50.6]],
     [[2498.053,”2009-06-22T06:59:31.0000000Z”], [[54.9     ], 50.2]],
     [[2500.099,”2009-06-22T06:39:56.0000000Z”], [[56.8, 0.9], 51.9]],
     [[2500.621,”2009-06-22T07:06:02.0000000Z”], [[50.5, 0.9], 46.2]],
     [[2501.001,”2009-06-22T06:46:28.0000000Z”], [[52.9, 0.9], 48.4]],
     [[2501.141,”2009-06-22T06:26:54.0000000Z”], [[60.6, 0.9], 55.5]],
     [[2503.449,”2009-06-22T06:13:51.0000000Z”], [null       , 49.3]],
     [[2503.774,”2009-06-22T11:15:19.0000000Z”], [[50.5, 0.9], 46.2]],
     [[2504.228,”2009-06-22T06:20:24.0000000Z”], [[59.2, 0.9], 54.1]],
```

```
     [[2504.237,”2009-06-22T07:12:31.0000000Z”], [[52.5, 0.9], 48]],
     [[2504.272,”2009-06-22T06:33:26.0000000Z”], [[52.5, 0.9], 48]]
]

]]>
```

5.3.3 Log

The WITSML 2 Log is very simple object, in that it is primarily just a container for one or more ChannelSets as show in **Figure 5-5**. Most of the information is at the ChannelSet level. The concept of multiple ChannelSet in a single Log is significant change from WITSML where each Log represented exactly one group of curves and their data (NOTE: Technically, the WITSML 1.4.1.1 Log allowed for multiple blocks of data, but this was just to optimize transmission of sparse data for real time. With ETP, this requirement no longer exists.). In WITSML 2, each ChannelSet represents a disjoint set of channel data. There are many possible use cases that would dictate what ChannelSets would go together in a Log, and how they would relate to each other.

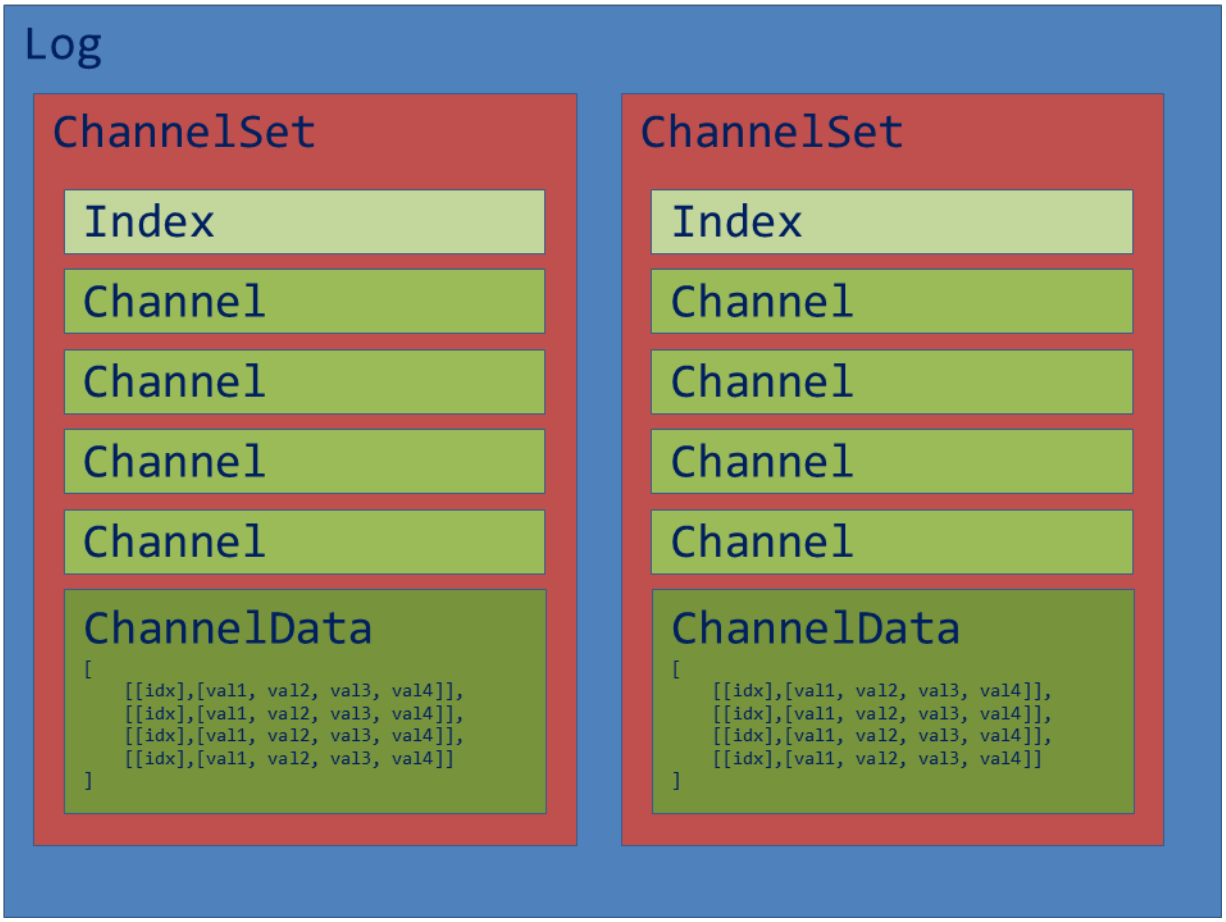

Figure 5–5. Organization of channel data within a log

Some of the possibilities:

- A Log could contain a depth run in one ChannelSet with the data for a re-log of one section of the hole in another ChannelSet.
- A single Log could contain all of the data for a run of a Production Logging Tool (PLT).
- Data from multiple acquisition systems can be represented as multiple ChannelSets in a Log.

The Log object does not include any specific information about how the ChannelSets relate to each other, except as can be gleaned from a comparison of the metadata for each object in the log.

6 Wellbore Geology and Mud Log Report Data Objects (formerly Mud Log object)

The current version of WITSML now has these main data objects related to wellbore geology (WITSML v1.4.1 had only a single mud log object):

- Wellbore geology
- Mud log report

In this latest version, the original v1.4.1 mud log object was divided into these two objects to better support mud logging workflows—that is, sometimes a mud log report is prepared at the same time the wellsite geologist is performing the assessment; sometimes a report is prepared for multiple assessments, after the wellsite geologist has completed several evaluations. This chapter describes the basic organization and usage of each of these objects.

6.1 Wellbore Geology Data Object

The wellbore geology data object (WellboreGeology) (**Figure 6-1**):

- Provides the unambiguous modeling of data for different levels of geological description and interpretation; it covers the areas of geology and hydrocarbon show evaluation that are performed at the wellsite.
- References a parent wellbore.
- Is a growing data object because as measurements are taken (at increased depth) that data is added to the wellbore geology object.
  - Includes several “part_ objects” to support streaming data with ETP. These part objects exist for geological intervals for cuttings, interpreted lithology, and show evaluation. The sections below provide more information.

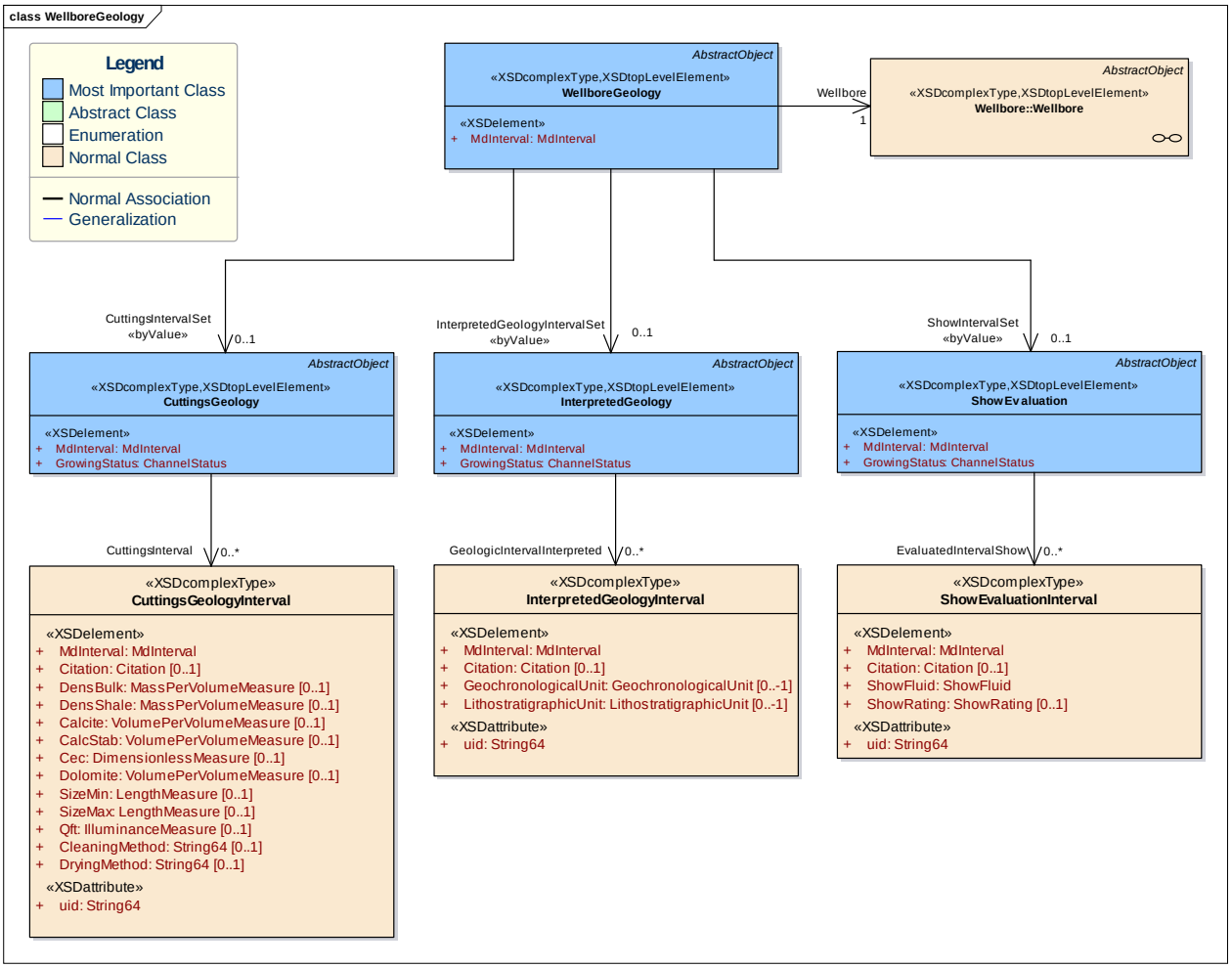

Figure 6–1. UML diagram of the wellbore geology data object—high-level view. The main sub-objects include: cuttings geology interval, interpreted geology interval, and show evaluation interval.

The wellbore geology object addresses the following main tasks in wellbore geology, which is reflected in the object design:

- Cuttings descriptions (CuttingsGeologicInterval)
- Geological interpretation (InterpretedGeologyInterval)
- Show evaluation (ShowEvaluationInterval)

Figure 6-2 shows the use of separate geology intervals for cuttings, interpreted lithology, and show evaluation as they might be presented on a graphical mud log. The following sections describe the related data objects and their usage.

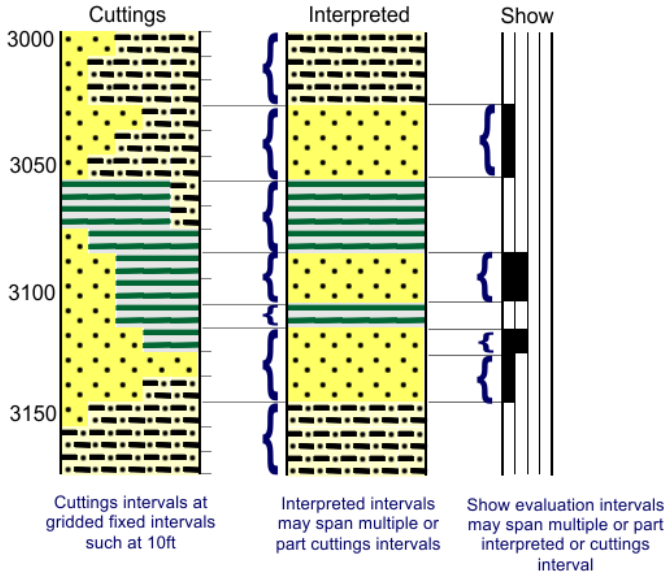

Figure 6–2. Mud log showing cuttings, interpreted lithology, and show evaluations.

The following pseudo-code example shows the hierarchy of the main wellbore geology objects, which are explained below.

```
WellboreGeology
      ShowEvaluation[0..1]
            ShowEvaluationInterval[0..*]
      InterpretedGeology[0..1]
            InterpretedGeologyInterval[0..*]
                  GeochronologicalUnit[0..*]
                  LithostratigraphicUnit[0..*]
                  InterpretedIntervalLithology[0..1]
      CuttingsGeology[0..1]
            CuttingsGeologyInterval[0..*]
                  CuttingsIntervalLithology[0..*]
                        CuttingsIntervalShow[0..*]
                  LithologyQualifier[0..*]
```

6.1.1 Cuttings Geology Interval (CuttingsGeologyInterval)

A mud logging company takes regular samples of drilled cuttings while the well is being drilled and examines the cuttings to determine the rock types (lithologies) present and the presence of any hydrocarbon shows. The results of these observations and determinations are recorded and associated with the depth interval of the cuttings sample.

The cuttings geology interval object (**Figure 6-3**) is used to capture and transfer this information. The cuttings samples typically contain a mix of different lithologies in each sample because there may have been multiple rock types that were drilled within the sample depth interval and there can also be mixing of cuttings as they travel up the wellbore and are collected on the shale shakers. Therefore, the cuttings geology interval needs to have multiple cuttings interval lithology elements so that the percentages of each lithology in the sample, along with the more detailed geological description of each lithology, can be recorded.

Section 6.3 (page 48) (which is an appendix for this chapter) describes the procedures for performing an analysis of hydrocarbon show on the cuttings samples. These show tests are performed on individual cuttings of selected rock types and record the staining and fluorescence of the cuttings as observed under

natural and ultraviolet light. To record the results of these tests, use the cuttings interval show sub-object (within the cuttings interval lithology element).

For realtime streaming of cuttings description data, transfer the cuttings lithology using the part cuttings interval object. For each cuttings interval described, create a new part cuttings interval object.

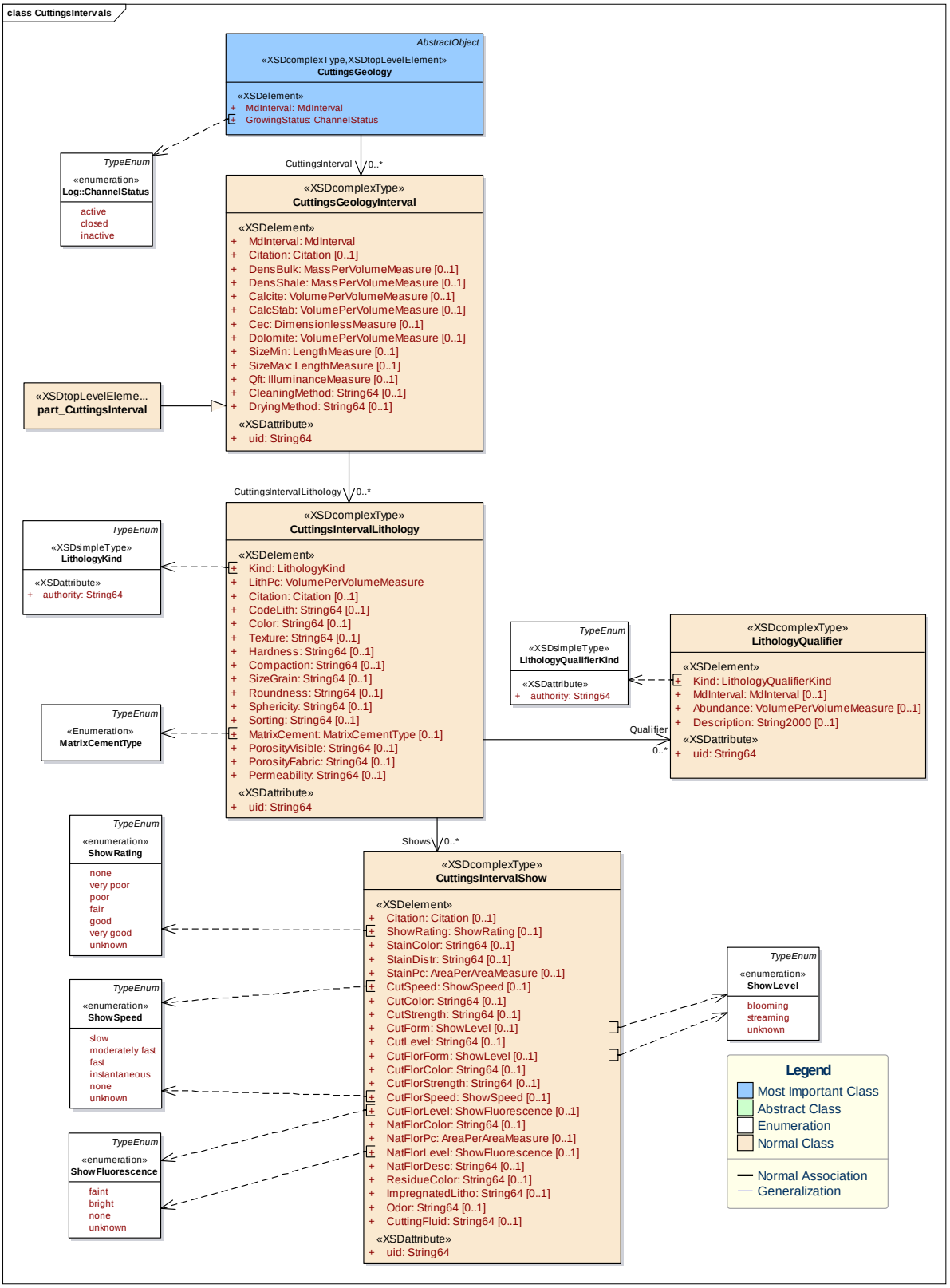

Figure 6–3. MUL diagram of the cuttings geology interval data object, which is used to capture data about lithologies, shows, and show evaluations.

6.1.2 Geological Interpretation (InterpretedGeologyInterval)

To estimate the location of the boundaries between the different lithology types, the mud logging company and/or a wellsite geologist may then make an interpretation of the actual lithology sequence along the length of the wellbore by correlating the percentage lithologies observed in the cuttings samples with other data, typically the drill rate and gamma ray log curves. This analysis creates a sequence of individual lithologies along the wellbore, which can be represented in WITSML as interpreted geology interval elements (**Figure 6-4**), each having a single interpreted interval lithology sub-object that captures the detailed geological description of the lithology.

For realtime streaming of the interpreted lithology, use the part interpreted geology interval object. Create a new object to stream each identified lithological interval.

To present a geological prognosis for the well in a non-realtime environment, use the geochronological unit, lithostratigraphic unit, and interpreted interval lithology objects in multiple instances of interpreted geology interval with either congruent or overlapping depth ranges.

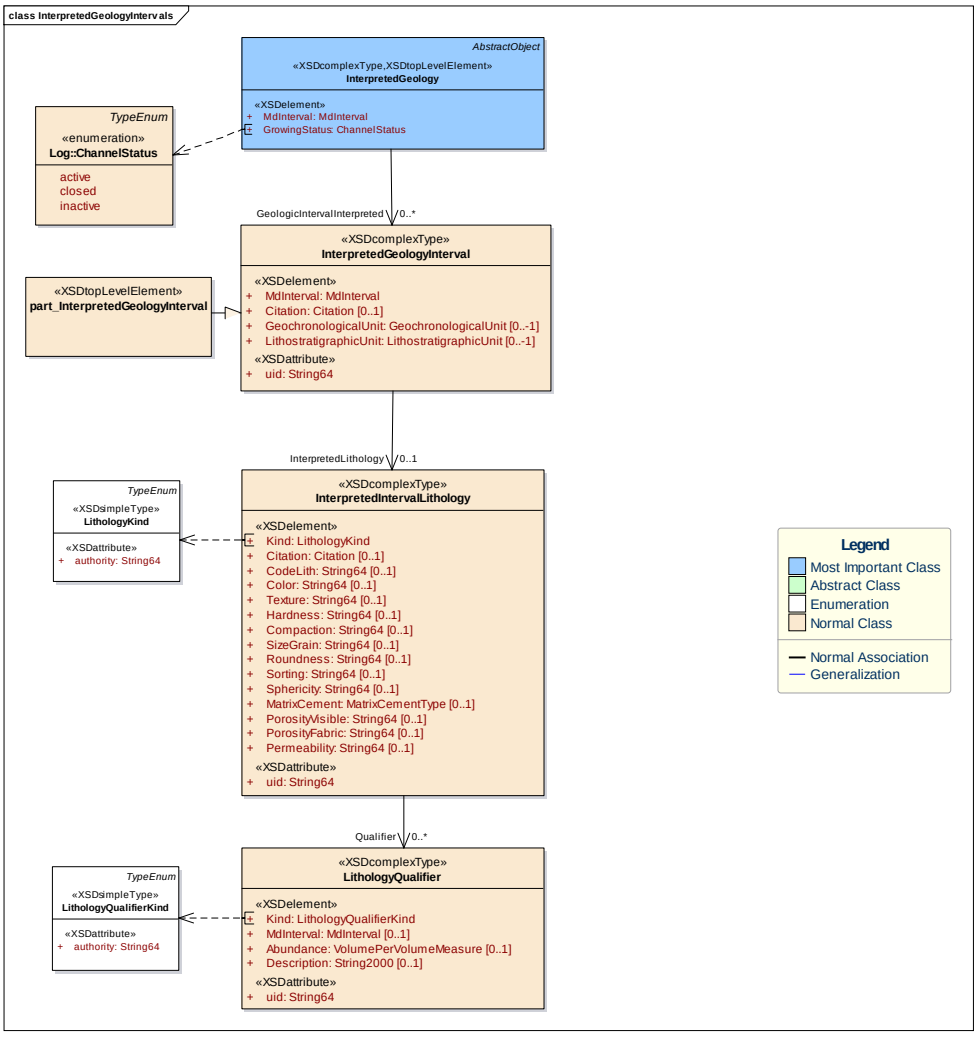

Figure 6–4. UML diagram of the interpreted geology interval data object, which is used to capture interpreted data used to estimate the location of the boundaries between the different lithology types.

6.1.3 Show Evaluation (ShowEvaluationInterval)

In a similar way to the interpreted lithology, there may also be an interpretation of the quality of hydrocarbon shows along the wellbore. Use the show evaluation object to contain the set of show

evaluation interval objects. Each instance of show evaluation interval contains a depth interval and a show rating.

For realtime streaming of the show evaluation data, use the part evaluated interval show object. Create a new object for each new evaluation that is available.

6.2 Mud Log Report Data Object

The mud log report (MudLogReport) data object can be used to create a report of geological, hydrocarbon evaluation, and drilling parameters observed while drilling a wellbore. The report represents the activities typically performed by mudlogging or wellsite geological personnel at a drilling location. The data from this object can be used to generate the graphical mud log or wellsite geology log that is the service product from these wellsite operations.

The mud log report object uses the objects from wellbore geology object (described above) to carry the detail of specific intervals of cuttings geology, interpreted geology, and hydrocarbon show but, the mud log report object also adds the capability to describe other associated data including:

- Information about the service company and its personnel
- Gas readings (mud gas, gas peaks)
- Chromatographic analysis of gas content
- Drilling parameters
- Links to other wellbore information, such as drilling reports and log data

The mud log report is a growing object. As drilling progresses along the wellbore, new instances of the mud log report interval are generated and added to the object. These intervals may be transferred in real-time via ETP using the part_MudLogReportInterval object.

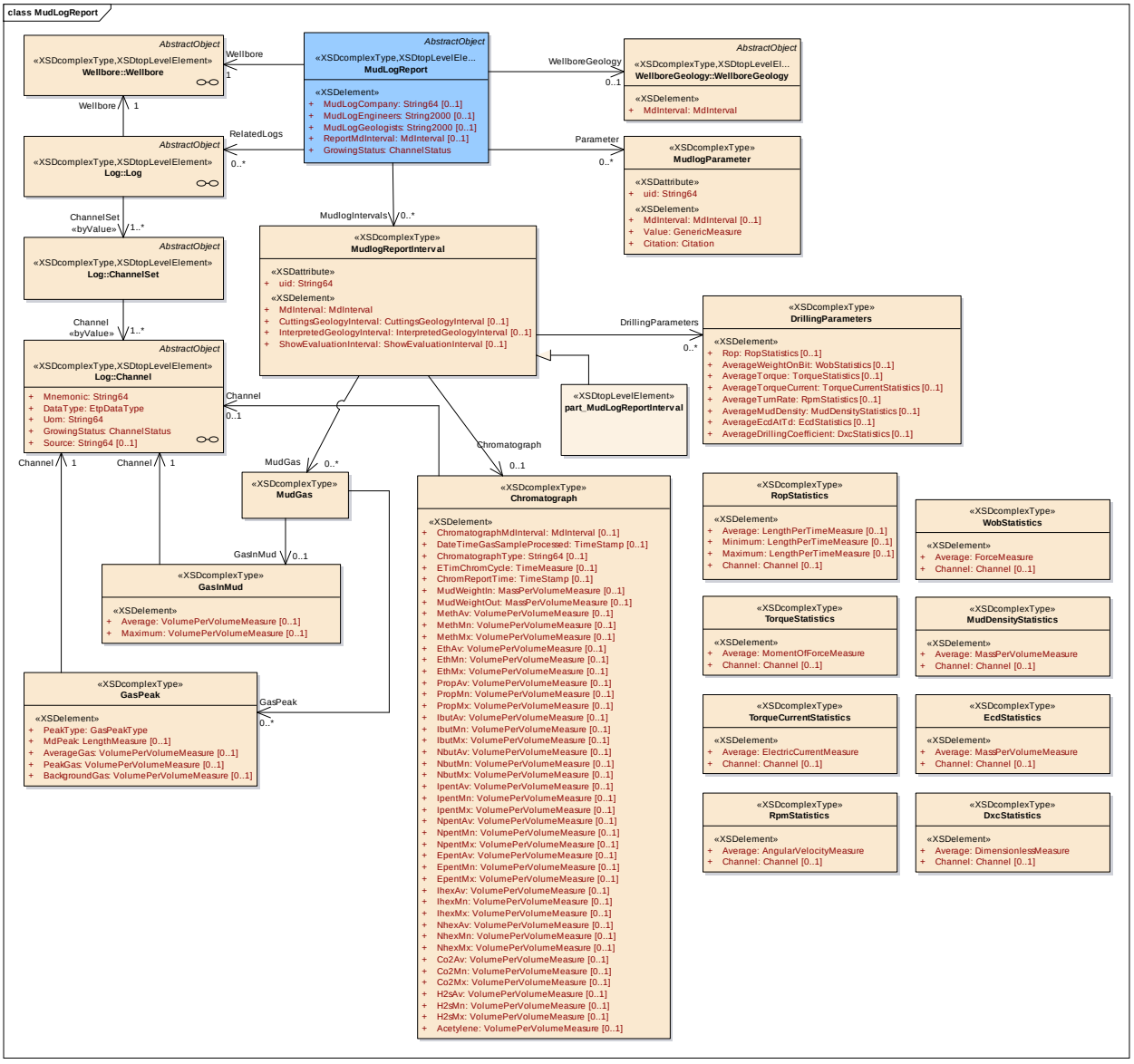

Figure 6–5. UML diagram of the mud log report data object.

6.3 Appendix A for Chapter 6: Hydrocarbon Evaluation

This section gives information on how a mud logger performs hydrocarbon evaluations. It is provided to help people implementing WITSML understand the type of data gathered, used, and transferred using the Wellbore Geology object.

6.3.1 Introduction

Although petrophysical analyses, coupled with well testing, may give a conclusive determination of the presence of commercial quantities of oil, it is still the mud logger's responsibility to report and log all hydrocarbon shows.

Unfortunately, there are no specific criteria to establish if a show represents a potentially productive interval. The color and intensity of the stain, fluorescence, cut, cut fluorescence, and residual cut fluorescence vary with the specific chemical, physical, and biological properties of each hydrocarbon accumulation. The aging of the shows (highly volatile fractions dissipate quickly) and flushing by drilling

fluids or in the course of sample washing tend to mask or eliminate evidence of hydrocarbons. The presence or absence of obvious shows cannot always be taken as conclusive. In many cases, the only suggestion of the presence of hydrocarbon may be a positive cut fluorescence. In other cases, only one or two of the other tests may be positive.

6.3.2 Sample Examination Procedure for Hydrocarbon Shows

It is a good practice, to examine both unwashed and the sample for lithological evaluation under the UV box. This will allow you to take note of the background fluorescence. In practice, this is rarely done due to time constraints.

One of the first signs of an entry into a possible hydrocarbon-bearing zone would be a drilling break. This is an increase in ROP due to the porous nature of the sediment drilled. On most wells, the drilling program would call for a flow check and a CBU when two to five meters into the drilling break. CBU is circulating until all the samples are recovered and evaluated.
If there is a significant gas peak, a spot sample should be taken and examined under the UV light. The spot sample should include an unwashed sample, lithological sample and a mud sample.

The mud logger must get all the relevant chemicals and tools ready before drilling a potential hydrocarbon zone.

Below is a typical flow of sampling procedures that are done when hydrocarbon shows are encountered.

1. Whenever there are significant gas shows, the mud logger must get a mud sample from either the flowline or the possum belly, aside from the regular sample or bottoms up sample. If the significant gas peak arrives in between sampling intervals, a spot sample is caught along with a mud sample.

2. The mud sample is poured into a shallow dish and placed under UV light to observe the signs of fluorescence in the mud or droplets of immiscible oil popping to the surface. If nothing is seen, water is added to the mud and the mixture is stirred. Again the sample is observed under UV light.

3. The unwashed sample is also observed under UV light. The distribution, color, and intensity of the fluorescence, if any, is noted.

4. For the lithological samples, smell the sample first before observing it under the microscope. If an oil odor is detected record it (see section on Odor). Under the microscope, an estimate is made of the percentage of oil-stained cuttings. The appearance (oily, waxy, dry residue), distribution (even, patchy, spotty, streaks) and color of the oil stains will be described. If oil-stained cuttings are observed these can be separated and placed into the depressions of the spot plate.

5. The sample tray is then observed under UV light. The proportion of the cuttings fluorescing and its distribution (even, spotty, pinpoint, patchy, and streaky), color and intensity of the fluorescence, if any, is noted. Some samples, if none were selected in Step 4, are placed into the depressions of the spot plate.

6. The samples in the spot plate are observed under the microscope for any oil stains. This is also to verify the lithology of the sample that will be subjected to the solvent cut test.

7. If no stains were observed, the spot plate is put inside the UV box and the samples are subjected to a solvent cut test.

6.3.2.1 Odor

Odor may range from heavy, characteristic of low gravity oil, to light and penetrating, for condensate. Some dry gases have no odor. Describe the odor, as oil odor or condensate odor. Report its strength as good, fair or faint. Faint odors may be detected more easily on a freshly broken of odor (in cores) or after confining the sample in bottle for 10 - 15 minutes.

6.3.2.2 Staining and Bleeding

The amount by which cuttings and cores will be flushed on their way to the surface is largely a function of their permeability. In very permeable rocks only very small amounts of oil are retained in the cuttings.

Often bleeding oil and gas may be observed in cores, and sometimes in drill cuttings, from relatively tight formations. The amount of oil staining on cuttings and cores is primarily a function of the distribution of the porosity and the oil distribution within the pores. The color of the stain is related to oil gravity. Heavy oil stains tend to be a dark brown, while light oil stains tend to be colorless. The color of the stain or bleeding oil should be reported. Ferruginous or other mineral stains may be recognized by their lack of odor, fluorescence or cut.

6.3.2.3 Fluorescence

There are numerous fluorescent compounds in crude petroleum. A number of them will fluoresce in the visible spectrum. Generally, dense, low gravity oils will fluoresce at the longer, infrared, red and orange visible wavelengths, while the lighter, high gravity oils will fluoresce at shorter, yellow or blue, visible wavelengths. Unfortunately, crude oil is not the only material found in cuttings that will fluoresce under ultraviolet light. Some minerals and sample lithologies will also fluoresce. The fluorescence colors emitted are dark and the intensity is low. Below is a table of fluorescence of some common minerals and lithologies.

Table 2. Common Minerals and Lithologies and Their Fluorescence Color

Mineral	Fluorescence color
Dolomite and magnesian limestone	Yellow, yellowish-brown to dark brown
Aragonite and calcareous mudstones	Yellow-white to pale brown
Chalky limestones	Purple
Foliated paper shales	Tan to grayish brown
Anhydrite	Blue to mid gray
Pyrite	Mustard yellow to greenish brown

Some mud additives may also exhibit traces of fluorescence. Pipe dope, the heavy metalized grease used to lubricate and seal threaded tool joints of drill pipe and drill collars, is a source of fluorescence contamination. Pipe dope has a very bright gold, white or bluish-white fluorescence normally indicative of a light, high gravity oil or condensate. In natural light, it has a heavy, viscous appearance and a blue-black or brown metallic color, while high gravity oil or condensate is transparent and gold in natural light.

A reservoir containing heavy, low gravity oil might be passed over because bright white fluorescence of pipe dope might mask the darker color of the oil. In addition, low gravity crude has is dense and dark brown in natural light. It is important to check all samples under ultraviolet light, regardless of whether oil is suspected.

Examination of mud, drill cuttings and cores for hydrocarbon fluorescence under ultraviolet light often indicates oil in small amounts, or oil of light color which might not be detected by other means. All samples should be examined. Color of fluorescence of crude ranges from brown through green, gold, blue, yellow, to white; in most instances, the heavier oils have darker fluorescence. Distribution may be even, spotted, or mottled, as for stain. The intensity range is bright, dull, pale, and faint. Pinpoint fluorescence is associated with individual sand grains and may indicate condensate or gas. Mineral fluorescence, especially from shell fragments, may be mistaken for oil fluorescence, and is distinguished by adding a few drops of a solvent.

Hydrocarbon fluorescence will appear to flow and diffuse in the solvent as the oil dissolves, whereas mineral fluorescence will remain undisturbed.

6.3.2.4 Solvent Cut Test

Oil-stained samples which are old may not fluoresce; thus failure to fluoresce should not be taken as decisive evidence of lack of hydrocarbons. All samples suspected of containing hydrocarbons should be treated with a reagent. The most common reagents used by the geologist are chlorothene, petroleum ether, and acetone. The next topic provides a list of reagents that can be used for the solvent cut test.

6.3.2.4.1 Chemicals Used for Solvent Cut Test

The solvent cut test is useful in determining the quality of the show. You can use different solvents for this test:

- chloroform (trichloromethane)
- carbon tetrachloride (tetrachloromethane, perchloromethane)
- ethylene dichloride (sym-dichloroethane, 1,2-dichloroethane, ethylene chloride, dutch oil)
- methylene chloride (methylene dichloride, dichloromethane)
- 1,1,1-trichloroethane (methyl chloroform, chlorothene)
- 1-1-2-trichloroethane (vinyl trichloride, beta-trichloroethane)
- trichloroethylene (ethylene trichloride, triclene, tri, trike or TCE)
- acetone
- petroleum ether

The most common reagents used by the geologist are chlorothene, petroleum ether, and acetone. The next topic provides a list of reagents that can be used for the solvent cut test. The use of ether gives a more delicate test for soluble hydrocarbons than chlorothene or acetone. However, the ether being used should be tested constantly, for the least presence of: any hydrocarbon product will contaminate the solvent and render it useless. Chlorothene is recommended for general use although
it too may become contaminated after a long period. Acetone is a good solvent for heavy hydrocarbons but is not recommended for routine oil detection.

CAUTION: CARBON TETRACHLORIDE is a cumulative poison and SHOULD NOT BE USED for any type of hydrocarbon detection.

CAUTION: Proper ventilation is important when using petroleum ether as it may have a toxic effect in a confined space. In addition, PETROLEUM ETHER AND ACETONE are very INFLAMMABLE and must be KEPT AWAY FROM OPEN FLAMES.

6.3.2.4.2 Solvent Cut Test Procedure

The most reliable test for hydrocarbons is the solvent cut test. Another name for it is the cut fluorescence or “wet cut" test. If hydrocarbons are present, fluorescent "streamers" will be emitted from the sample and the intensity and color of these streamers are used to evaluate the test. Some shows will not give a noticeable streaming effect but will leave a fluorescent ring or residue in the dish after the reagent has evaporated. This is termed a "residual cut." It is recommended that the "cut fluorescence" test be made on all intervals in which there is even the slightest suspicion of the presence of hydrocarbons. Samples that may not give a positive cut or will not fluoresce may give a positive cut fluorescence. This is commonly true of the high gravity hydrocarbons that may give a bright yellow cut fluorescence. Distillates show little or no fluorescence or cut but commonly give positive cut fluorescence, although numerous extractions may be required before it is apparent. Generally low gravity oils will not fluoresce but will cut a very dark brown and their cut fluorescence may range from milky white to dark orange. An alternate method involves picking out a number of fragments and dropping them into a clear one-or two-ounce bottle. Chlorothene or acetone is poured in until the bottle is half full. It is then stoppered and shaken. Any oil present in the sample is thus extracted and will color the solvent. When the color of the cut is very light, it may be necessary to hold the bottle against a white background to detect it. If there is only a slight cut, it may come to rest as a colored cap or meniscus on the top surface of the solvent. Before a solvent cut test is to be performed, ensure the reliability of your solvent by placing a few drops of the solvent on a depression in the spot plate and observing it under UV light. If the solvent is “good”, no fluorescence should be observed.
To do a solvent cut test:
1. Place a few drops of solvent, enough to immerse the sample, on the sample in the depression in the spot plate. Be careful not to get the cutting agent into the rubber of the dropper as it might “contaminate” the solvent by giving it a pale yellowish fluorescence.
2. Observe the following:

- Cut speed: This is an indication of both the solubility of the oil and the permeability of the sample. The speed can vary from instantaneous to very slow.
- Cut nature: coloration of the solvent with dissolved oil may occur in a uniform manner, in streaming manner or in a blooming manner. A streaming cut also indicates low oil mobility.
- Cut color fluorescence and intensity: Observe the color and the intensity of the oil in the solvent under both UV and natural lights. The cut color observed under UV light could be called a cut color fluorescence (example: bright blue white cut fluorescence).
- Cut color and intensity: After observing the sample under UV light observe the sample under natural light. The cut color observed in natural light is just called cut color (example: very light brown cut color or no cut color).

  The shade of the cut depends upon the gravity of the crude; the lightest crudes giving the palest cuts, therefore, the relative darkness should not be taken as an indication of the amount of hydrocarbon present. A complete range of cut colors varies from colorless, pale straw, straw, dark straw, light amber, amber, very dark brown to dark brown opaque.
- Residue color and intensity: The solvent dissolves rapidly under the heat of the UV light, sometimes leaving a residue of oil around the cutting on the spot plate. The true color of the oil can then be observed. The intensity and opacity of color, especially of the residue, is an indicator of the oil density and the quantity of oil originally in the cutting.

  A faint "residual cut" is sometimes discernible only as an amber-colored ring left on the dish after complete evaporation of the reagent. The hydrocarbon extracted by the reagent is called a "cut."

3. If the sample shows all the possible signs of being oil bearing, but has no solvent cut, the sample is crushed using the metal probe and it is observed for a solvent cut. The cut is called a crushed cut.

7 Other Data Objects

This chapter describes these data objects:

- Rig and rig utilization
- Tubular (Section 7.2)
- BHA run (Section 7.3)
- Wellbore geometry (Section 7.4)
- Wellbore markers (Section 7.5)
- Risk (Section7.6)
- Attachment (Section 7.7)
- Depth-registered image (Section 7.8)

7.1 Rig and Rig Utilization Data Objects

Beginning with v2.0, the rig data object has been split into these two top-level data objects, which are shown in **Figure 7-1**.

- **Rig** contains the typically static information associated with a rig, such as: owner and related information; characteristics and identifying information; permanent installed equipment, such as derrick and cranes, characteristics of those; and ratings and service information. This is a globally unique object with no required references.
- **Rig Utilization** contains the dynamic information associated with use of a particular rig, on a particular wellbore, for a specific purpose. The rig utilization object contains dynamic data specific to a particular use, for example: information associated with the operation (start/end times, hole depths, air gap, etc.); details of installed equipment at the time operation, for example: mud pumps, centrifuge, pit, shale shaker, blowout preventer (BOP), etc.

  The rig utilization object must reference:

  - One rig object
  - One wellbore

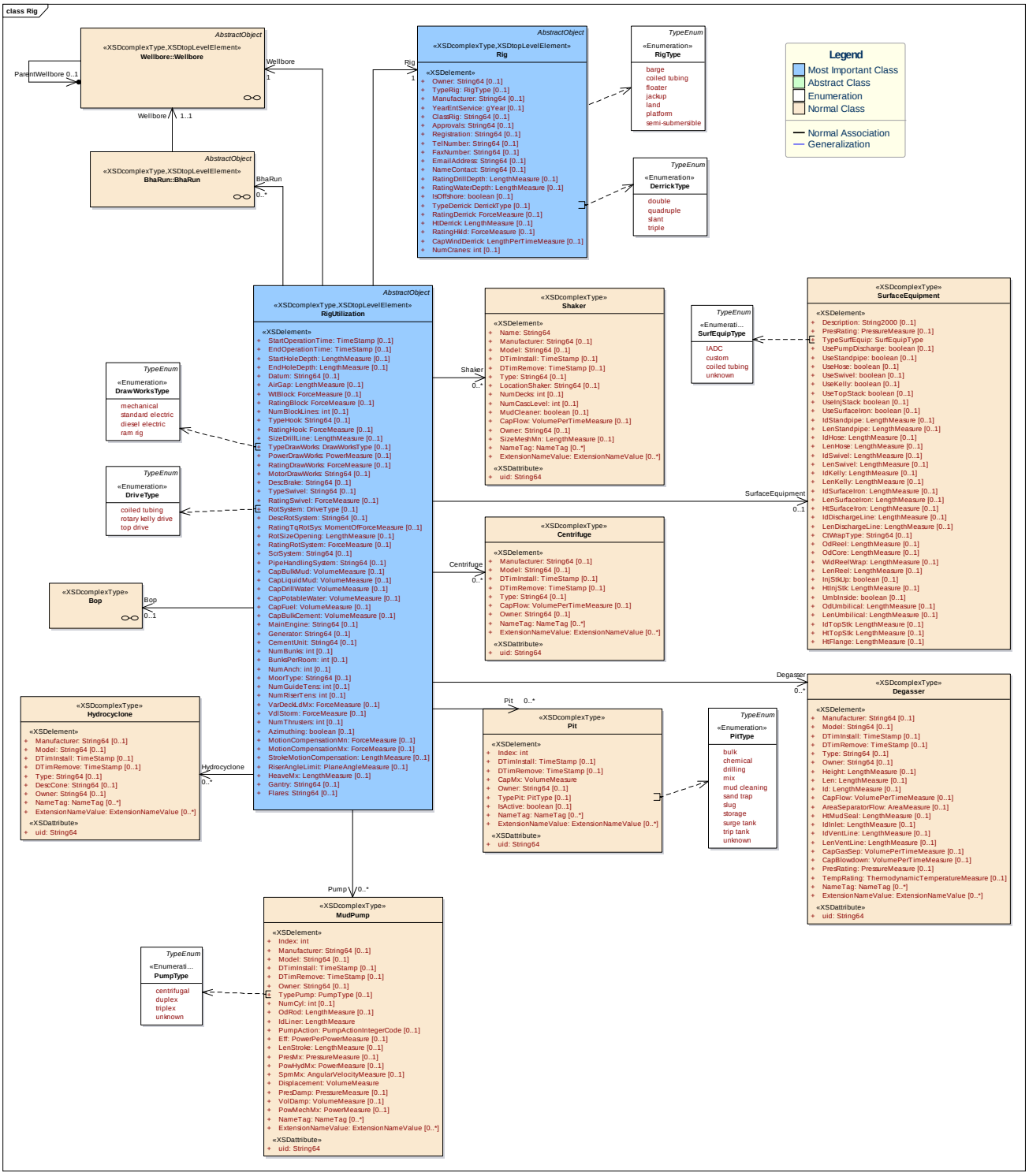

Figure 7–1. UML diagram of the rig and rig utilization data objects.

7.2 Tubular Data Object

The tubular object is used to capture information about the configuration of a string of pipe, including drill strings, coiled tubing, and casing. Some example workflows using the tubular object include: BHA

component tracking, hydraulic calculations, displacement calculations, bit properties and statistics, and many others.

The tubular object is uniquely identified within the context of one wellbore object. Typically, a tubular document (an instance of a tubular object) is created when a pipe string is being designed or planned, and, as the pipe string is being constructed, the components and lengths in the document can be edited and finalized. For a drill string, when it is deconstructed, the tubular document is also updated to populate the bit record components.

The tubular object is a static (not growing) object. As such, the document does not change over time; a new tubular document should be created for each change to the configuration of the string being planned or used. Reference to how a particular string was or will be used is referenced in the BHA run object.

7.2.1 Data Model

Figure 7-2 shows a UML diagram of the main tubular data object. Remember, all Energistics data object schemas (XSD files) are produced from the UML model; so the UML model reflects the schema organization. The basic organization of this set of objects is:

- **Tubular**: an ordered list of components that comprise a string and particular attributes (size, weight, order, etc.) and properties of those components.
  - **Tubular component** (optional) (**Figure 7-3**): Specify additional details about each of the components in the string
    - **Tubular tool** (optional) (**Figure 7-4**): In the tubular component object, if you specify a component as a tool, use this object to provide additional details about that tool. For example, details can be provided for sensors, MWD, or rotary steerable tools.
    - **Tubular bit record** (optional) (**Figure 7-5**): In the tubular component object, if you specify a component as a bit record, use this object to provide additional details about that bit record. For example, statistical data about the bit can be used in bit performance evaluation workflows.

The tubulars object must reference a “parent” wellbore.

The duration that a tubular was used in a string can be specified in a referenced BHA run object.

7.2.1.1 Tubular Object Overview

Drill sites normally collect information regarding any tubular that the well site works with. This information is then put together in a tally as the tubular is constructed, as it is going into the wellbore. The tubular object (**Figure 7-2)** is a way to transmit this collected tubular information. Typically, a separate tubular document is created for each configuration of the drill sting. This is not dependent on the standard of a run, but dependency resides on the design of the components of the string. An example of a non-run dependency tubular string is when casing is run. The first tubular document created contains the casing, inner string, and landing string components. However, when the casing tubular components are detached from the tubular string, a second tubular object is created and that document only contains the inner string and landing string tubular information.

Tubular object references to other objects to enable workflows are common. Reference to the wellbore geometry object is common because it allows consumers of data to know what tubulars were in a particular configuration of the wellbore design. The BHA run object can also reference the tubular object for consumers to link BHA run statistics to particular tubular configurations.

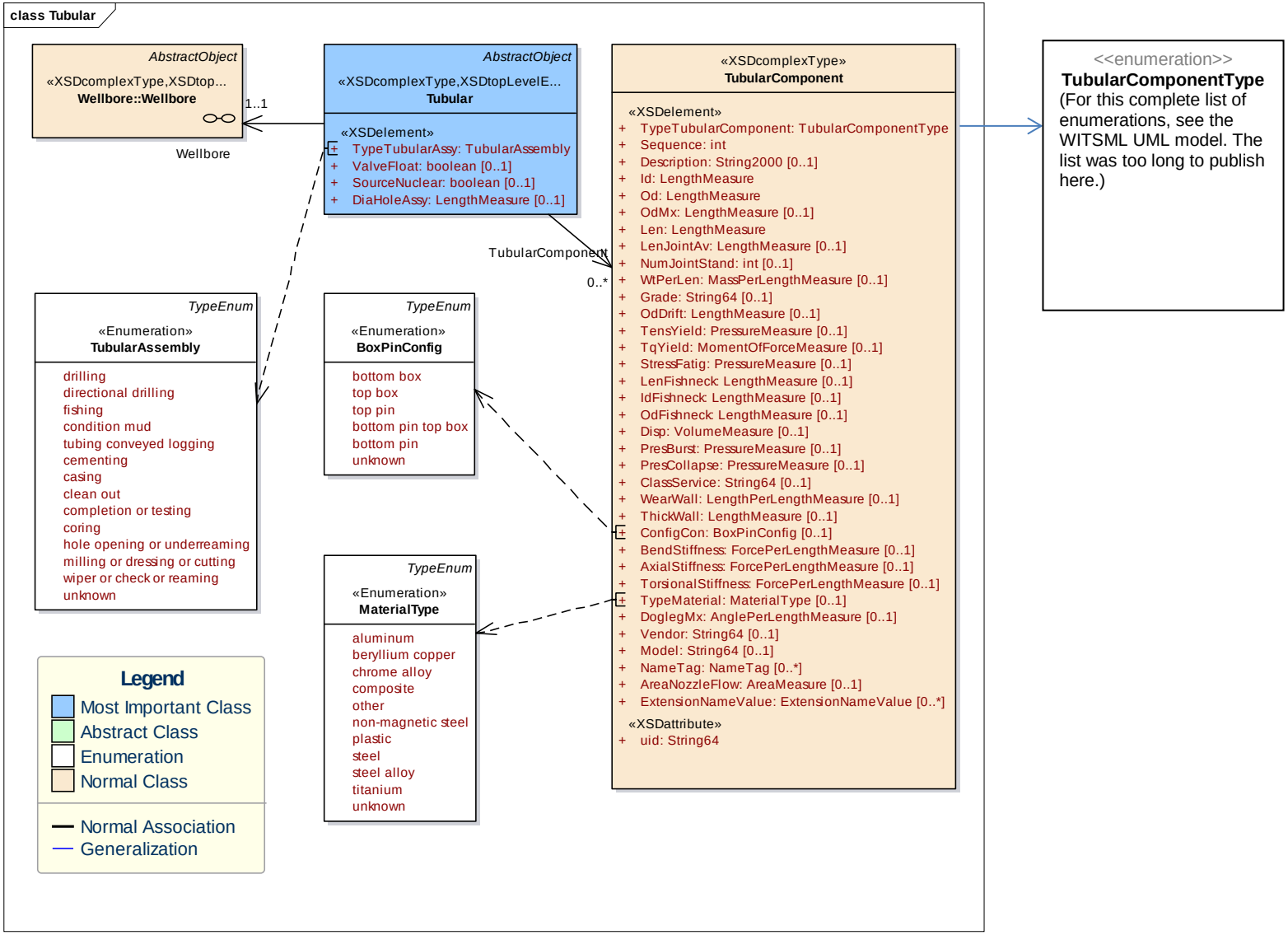

Figure 7–2. UML diagram of the tubular data object. NOTE: The diagram would not fit on this page, so the image has been slightly adapted (see the blue arrow). For the complete diagram, see the XMI file in the WITSML download..

7.2.1.2 Tubular Component Overview

To include more details about the components that make up a tubular string, use the tubular component object (**Figure 7-3).** Note that this object is a "child" of the tubular object (that is, tubular component is NOT an Energistics top-level element).

The figure shows the different lists of particular components that can be enumerated. Examples of the information found in this object include:

- Connection joints, including diameters and connection details, such as thread type and yield point.
- Motor type and detail of the motor such as bend settings and flow rate settings
- Stabilizer details of blade length and type
- Bit nozzle information for type and orientation
- Hole opener size and type
- Jar information such as type and action needed to activate
- The tubular component object can also reference these child objects, which are further explained below:
  - Tubulars tool object
  - Bit record object

Components

Data for component details should be populated as they are assembled in the string. The service provider or the tubular and relate data can recommend appropriate frequency of data updates.

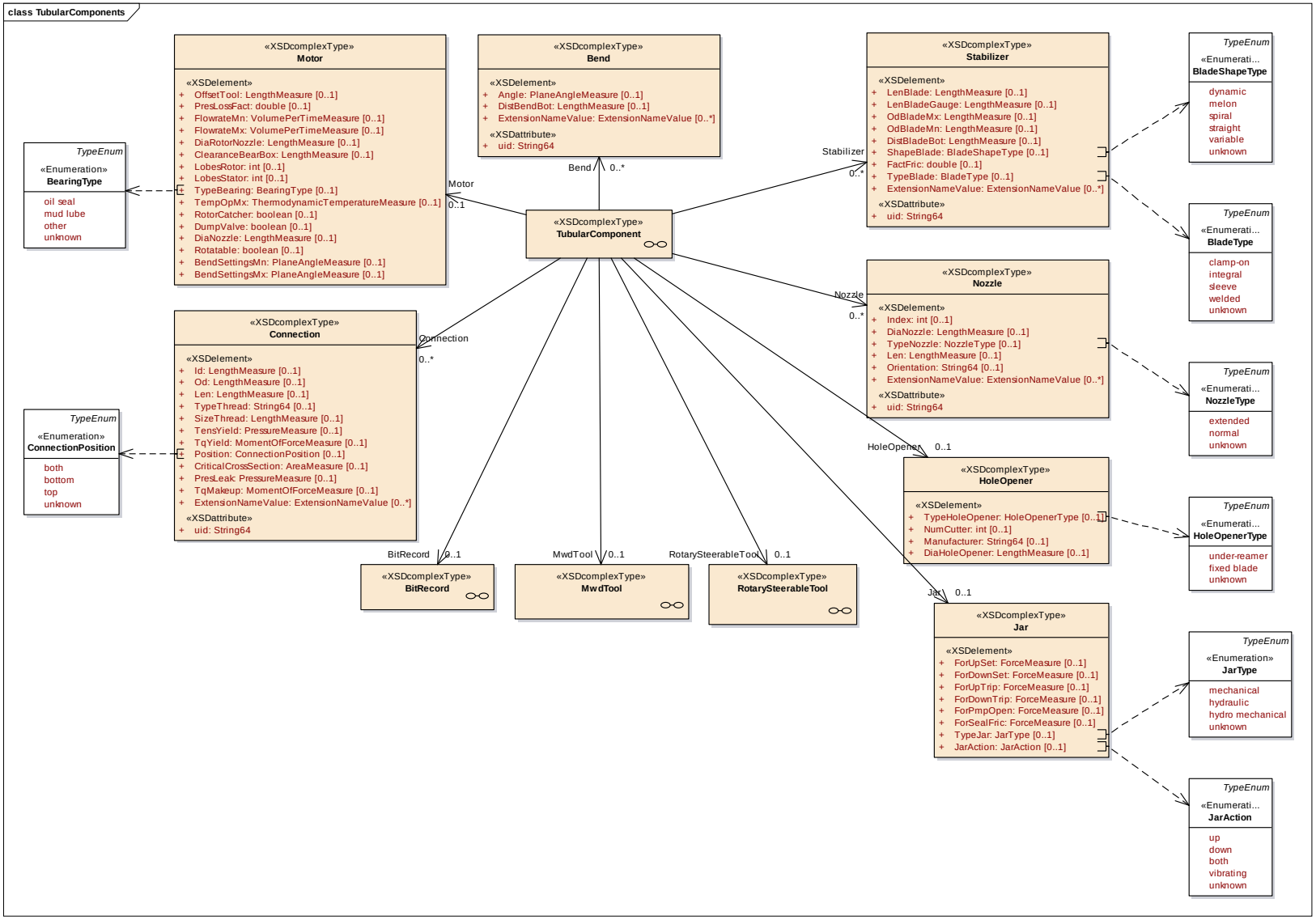

Figure 7–3. UML diagram of the tubular components.

7.2.1.3 Tubular Tool Overview

Tubular tool components are the details for MWD and directional tool sensors (**Figure 7-4).** This object stores data for:

- MWD tool parameters
- Sensor type and class
- Rotary steerable tool details for type and measurement

This data supports workflows such as data analytics and tool performance tracking. This information should be populated as the tool component is added to the object.

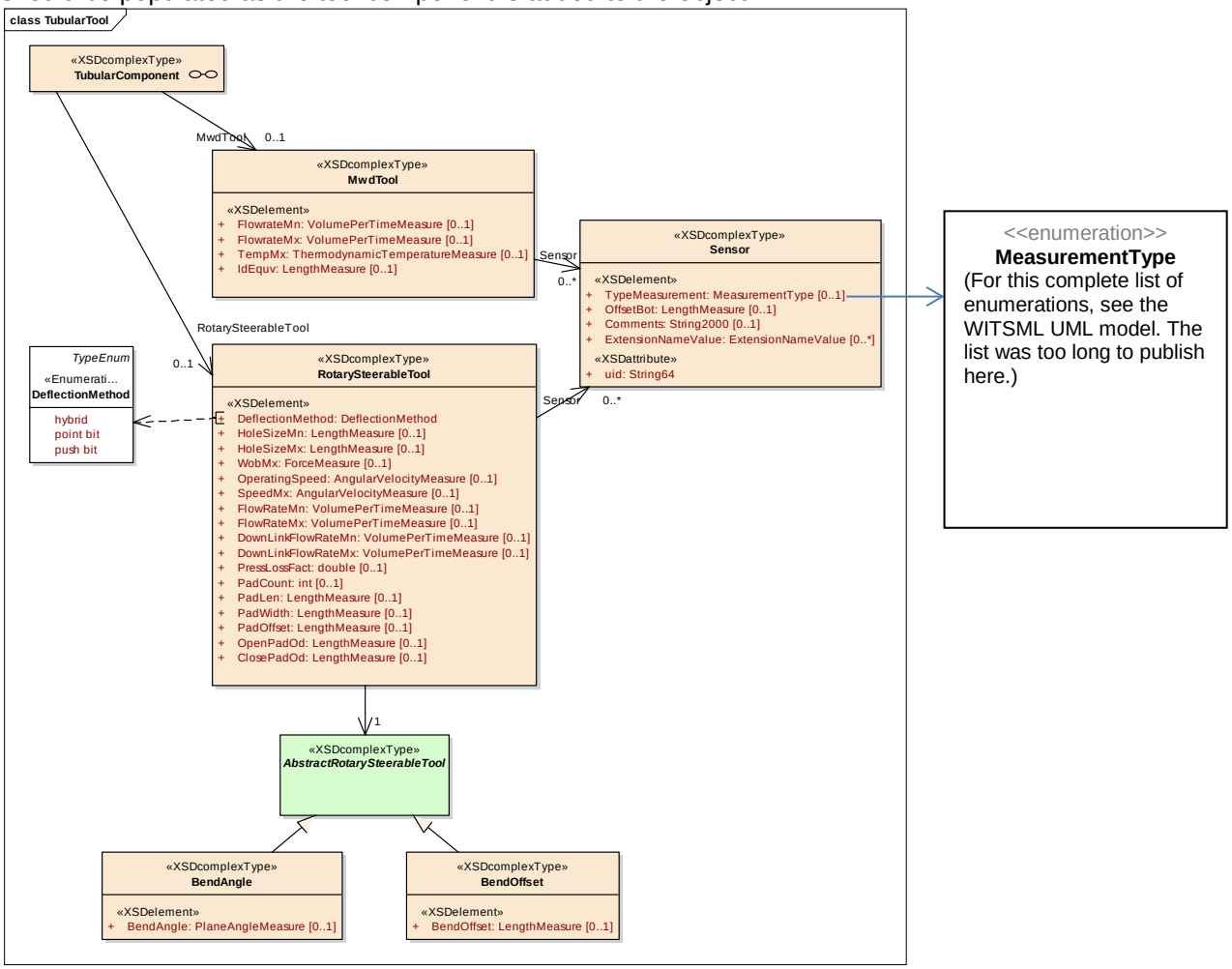

Figure 7–4. UML diagram of the tubular bit record.

7.2.1.4 Tubular Bit Record Overview

If the tubular has an associated bit and you want to capture related information about its performance, use the tubular bit record object (**Figure 7-5)**. This object is also a child of tubular component and contains data for the bit related to:

- Identification (number, type, manufacturer)
- Reason for and condition parameters when the bit was pulled from the hole.

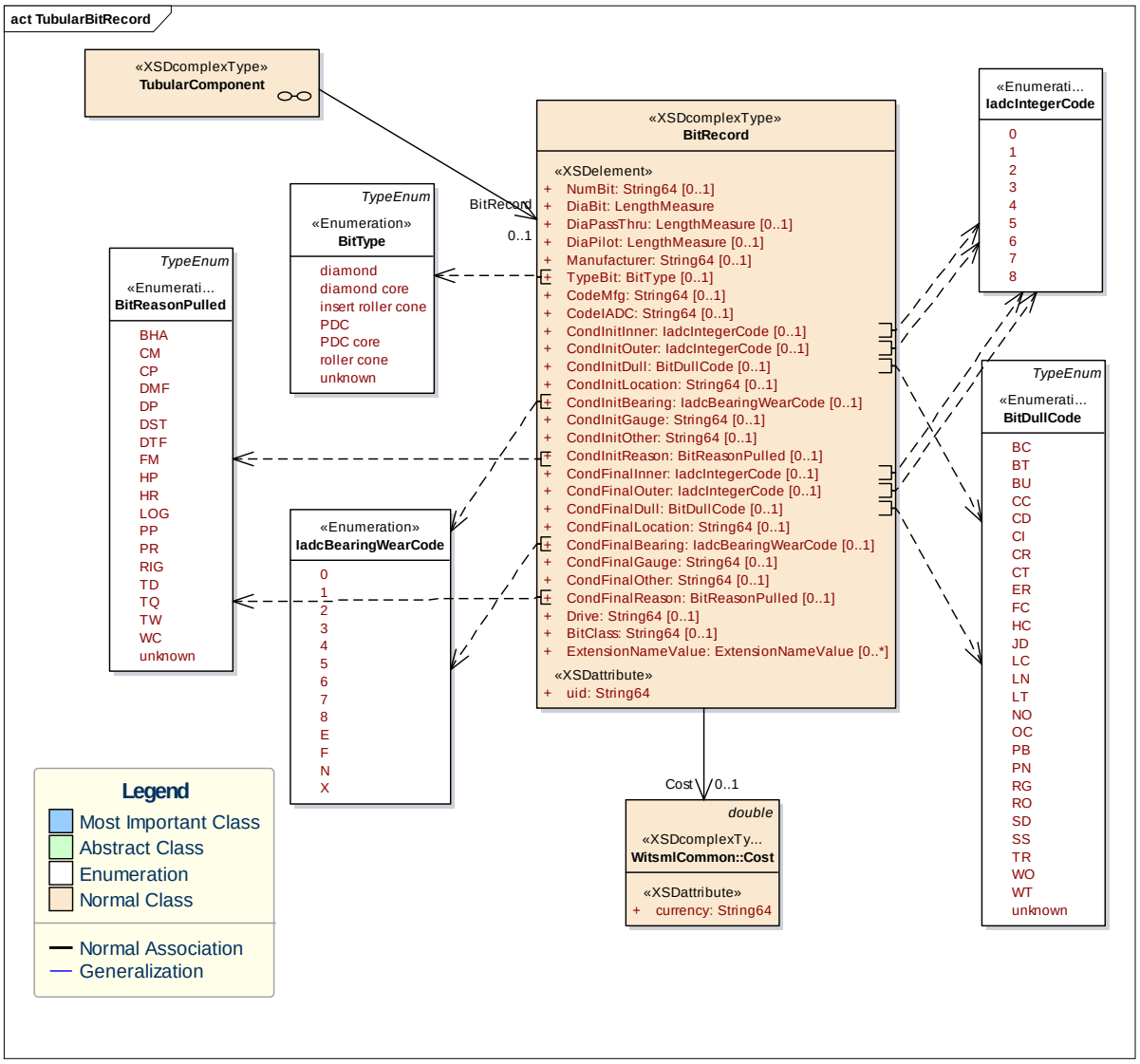

Figure 7–5. UML diagram of the tubular bit record.

7.3 BHA Run Data Object

The BHA run (BhaRun) data object is used to capture information about one run of a tubular string into and out of the hole. A “run” begins when the first component of the string—typically the bottomhole assembly (BHA)—is lowered below the rotary table on the trip into the hole, and ends when the final component of the string (typically the BHA) is above the rotary table on the trip out.

The type of data captured includes identifying information about the run itself and key statistics and calculated values for the operation performed (drilling, logging, etc.) during the run. Information about the tubular string used on the run is stored in the Tubular data object.

7.3.1 Data Model

Figure 7-6 shows the UML diagram of the BHA run data object.

The top-level BHA run object:

- Captures information about the run itself, which includes:

  - Identifying information about the run (e.g., run number, start/stop times, etc.)
  - Purpose and status (final, progress, plan) of the run
  - Information about estimated and actual dogleg severity

- References:
  - The wellbore in which the run occurs
  - The configuration of the string (Tubular)
  - Drilling parameters (DrillingParams) object, which contains statistical and calculated operations data for the run, related to depths, activities, temperature, pressure, flow rates, torque, etc.

    These statistical and computed values are often based on raw data that are captured in a channel data object (see Section 5.3.1 (page 33)), for example, hook load and weight on bit. Other values, for example mud class, come from other sources and are captured here.

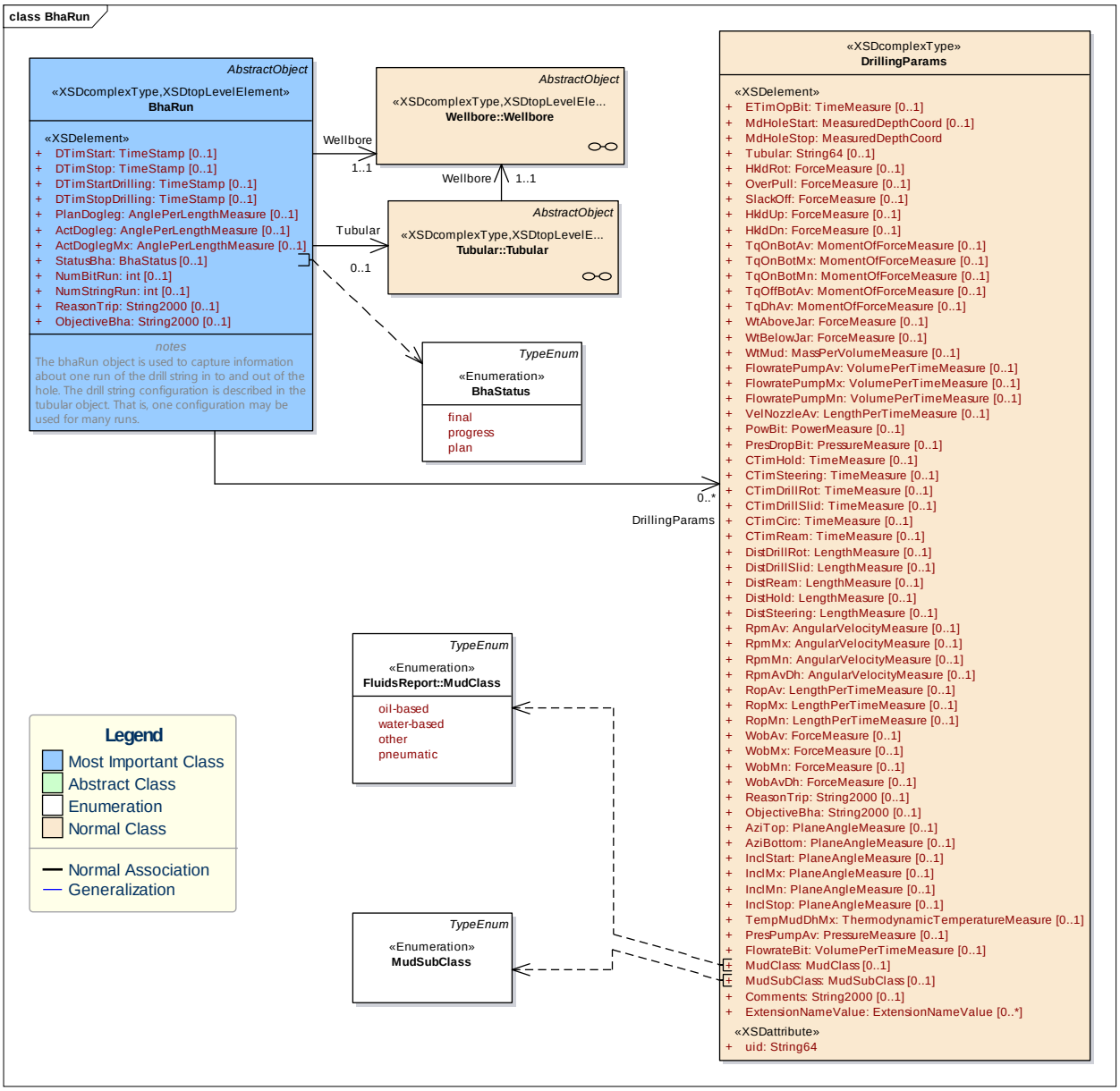

Figure 7–6. UML diagram of the BHA run data object.

7.4 Wellbore Geometry Data Object

The wellbore geometry (WellboreGeometry) data object captures information about the configuration of the permanently installed components in a wellbore. It does not define the transient drilling strings (see the Tubular object, Section 7.2) or the hanging production components (see the Completion object, Chapter 8).

Data in this object supports workflows that need a well profile.

7.4.1 Data Model

Figure 7-7 shows the UML model for the wellbore geometry data object. Remember, all Energistics data object schemas (XSD files) are produced from the UML model; so the UML model reflects the schema organization. The basic organization of the object is:

- **Wellbore Geometry** (WellboreGeometry). Contains the main data for wellbore location such as measured depth at bottom (at the time this data was captured), water depth, and air gap. It also includes a growing status, which is used when streaming data using ETP. This object:
  - Must reference one parent wellbore object.
  - May optionally reference BHA run object.
- **Wellbore Geometry Section** (WellboreGeometrySection). Defines the "fixed" components in a section of a wellbore, which includes properties and dimensions for open hole, casing, or risers.
- **part_WellboreGeometrySection,** Object used to stream data in real time using ETP.

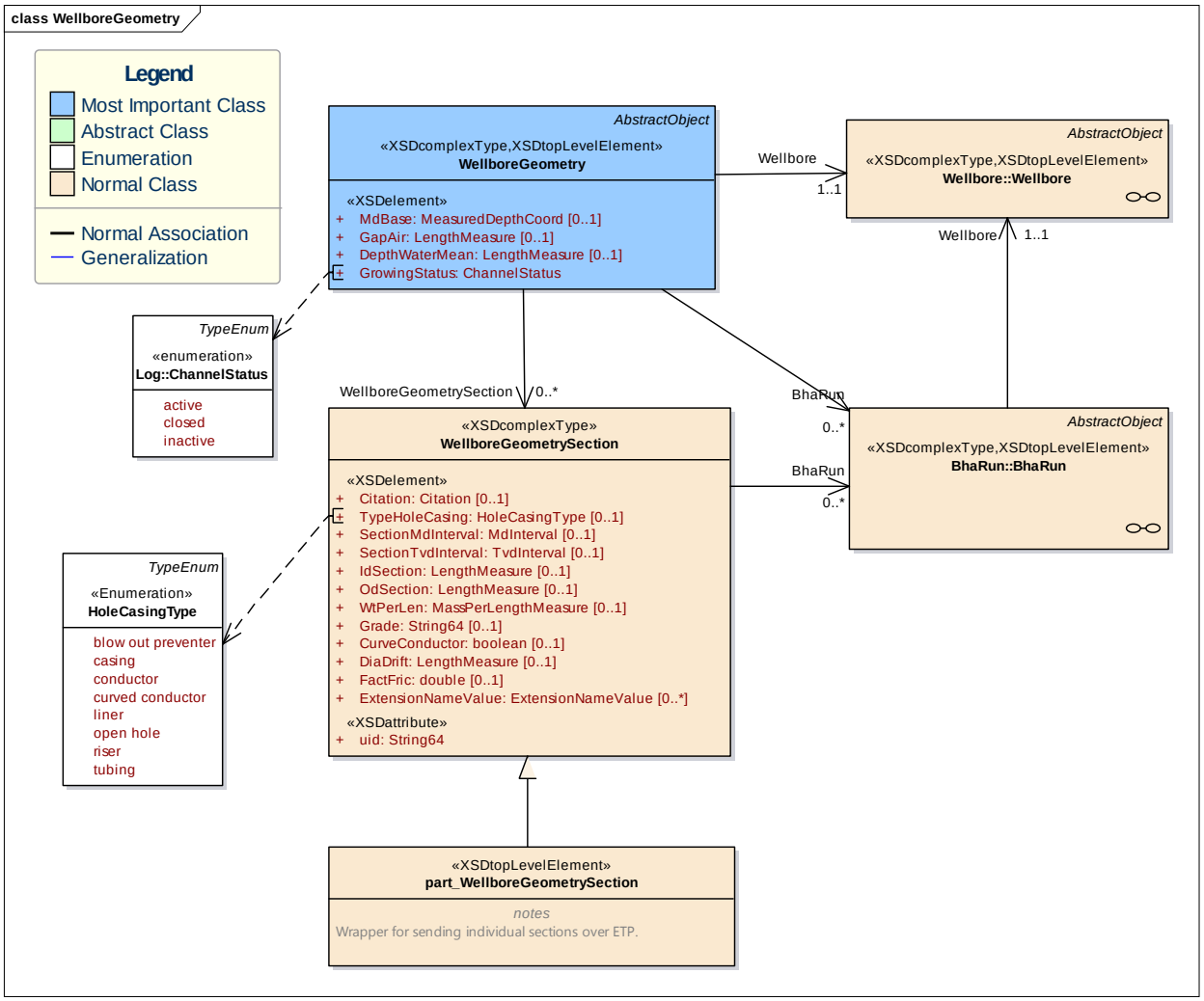

Figure 7–7. UML diagram of the wellbore geometry object.

7.5 Wellbore Markers Data Object (formerly Formation Marker)

Knowledge of rock formations is vital input to assessing the total and recoverable volume of hydrocarbons, and for effective drilling operations. Some of this data is gathered while drilling wells. For example, a break in continuity of a formation when observed passing through different wells indicates structural discontinuities in the subsurface and might indicate potential faults/folds, which may be structural traps. Similarly the lithology in different types of subsurface formations provides information about potential stratigraphic traps.

The wellbore markers object (WellboreMarkers) (which was Formation Marker in v1.4.1) is used to capture information about geologic formations that were encountered in a wellbore. This data is typically captured during drilling using logging while drilling (LWD) tools.

7.5.1 Business Purpose

Data captured and transferred using the wellbore markers object has these uses:

- Helps with drilling planning and operations; for example:
  - Guide geosteering (the act of adjusting the borehole position (inclination and azimuth angles) “on the fly” while drilling a borehole) to reach one or more geological targets, when encountering obstacles that require deviation from the drilling plan.
  - Optimize drilling operations based on the formation angle, for example, a horizontal reservoir could be better exploited with a horizontal well.
- Helps with assessing hydrocarbon potential and reservoir quality, by contributing information to determine:
  - Chronostratigraphic age. The maturity of the formation is useful information in estimating the potential of source rock (where hydrocarbons were initially produced).
  - Formation angles (dip information) and continuity of the reservoir based on the depth and angles of the formation in different wells.
  - Sequence stratigraphy, which is important in identification of a petroleum system and its potential for production of hydrocarbons.

7.5.2 Data Model

The wellbore marker (**Figure 7-8**) is a top-level data object that can optionally be assembled into a set of wellbore markers. It has these key characteristics:

- Contains this basic information:
  - Chronostratigraphic top, which is a reference to the geochronological unit of the wellbore geology object (see Section 6.1.2 (page 46)).
  - Lithostratigraphic top, which is a reference to the lithostratigraphic unit of the wellbore geology object (see Section 6.1.2 (page 46)).
  - TVD, dip angle and direction.
- Optionally references a wellbore and/or trajectory data object.

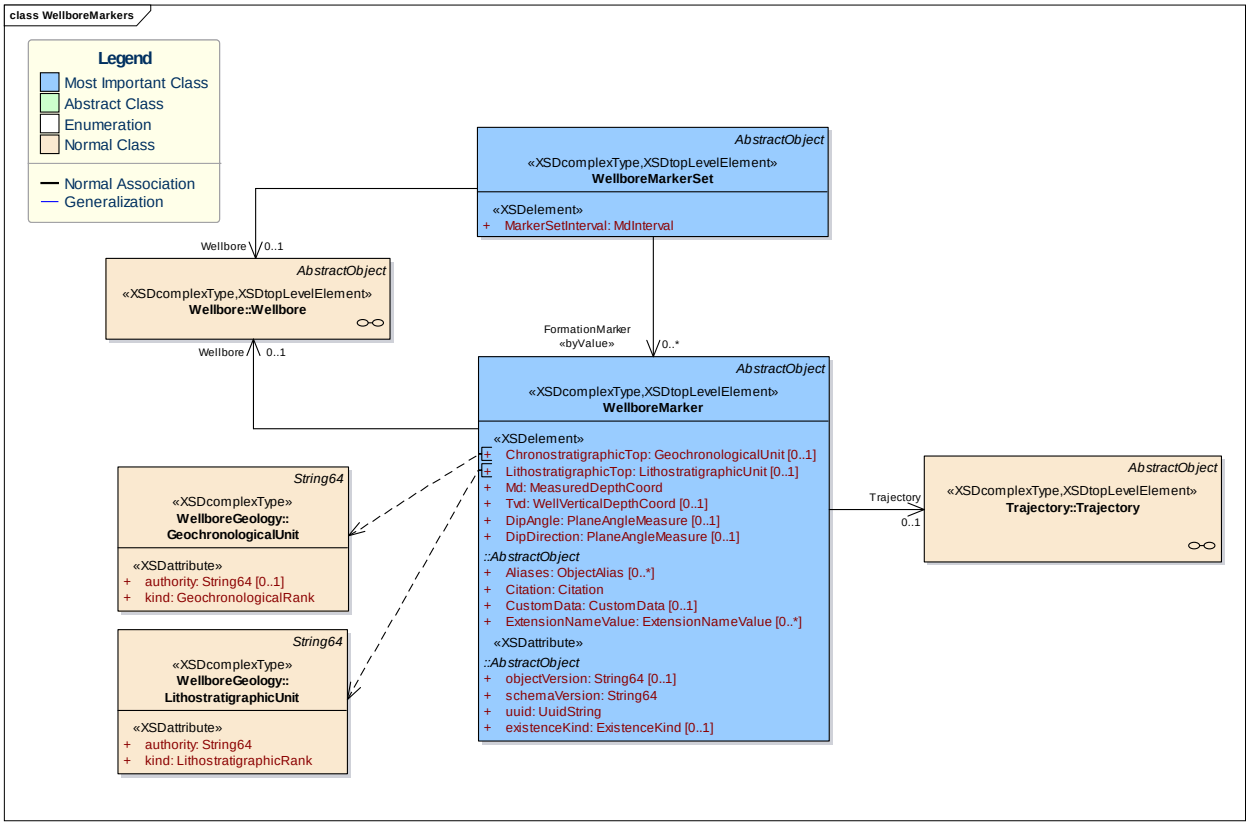

Figure 7–8. UML diagram of the wellbore marker and wellbore marker set objects.

7.6 Risk Data Object

Drilling risks and events are defined throughout the planning and execution of a well. It is crucial to record and share information about risks and events affecting rig operations. However, the ability to effectively share data on a larger scale between wells, fields, and blocks is a challenge. This data is particularly important during the planning phase of a well, when access to all available offset risk information is crucial for a successful and safe design of the well.

The risk data object (Risk) enables efficient and complete data transfer of risk and event data between different users and software applications. This information allows the integration of relevant risks into the proposed well design. As the well is drilled, the risk object can be used to alert the rig and remote support teams of potential issues ahead of the bit.

7.6.1 Data Model

Figure 7-9 is the UML diagram of the risk data object. Certain elements are required for the risk object to be both valid and of practical use. Other elements represent a “best practice” that is helpful for the risk to be properly interpreted. The risk data object:

- Must reference its parent wellbore.
- Must include these elements:
  - **Type:** the type of risk, e.g. gas kick.
  - **Category**: the category of risk, e.g. hydraulics or equipment failure”

Best Practice:

- **DTimStart** and **DTimEnd**: if applicable, date and time that the risk started and ended.
- **MdHoleStart** and **MdHoleEnd**: if applicable, the measured depth at the start and end of the risk.

  - **TvdHoleStart** and **TvdHoldEnd**: if the risk is associated with a depth range and MdHoleStart/MdHoleEnd have not been provided, then these should be provided.
- **SeverityLevel**: severity level of the risk, values 1 through 5, with 1 being the lowest.
- **ProbabilityLevel**: probability level of the risk, values 1 through 5, with 1 being the lowest.
- **Summary**: a summary description of the risk. If needed, provide more information in **Details.**
- **Contingency**: the plan of action if the risk materializes.
- **Mitigation: p**lan of action to ensure the risk does not materialize.

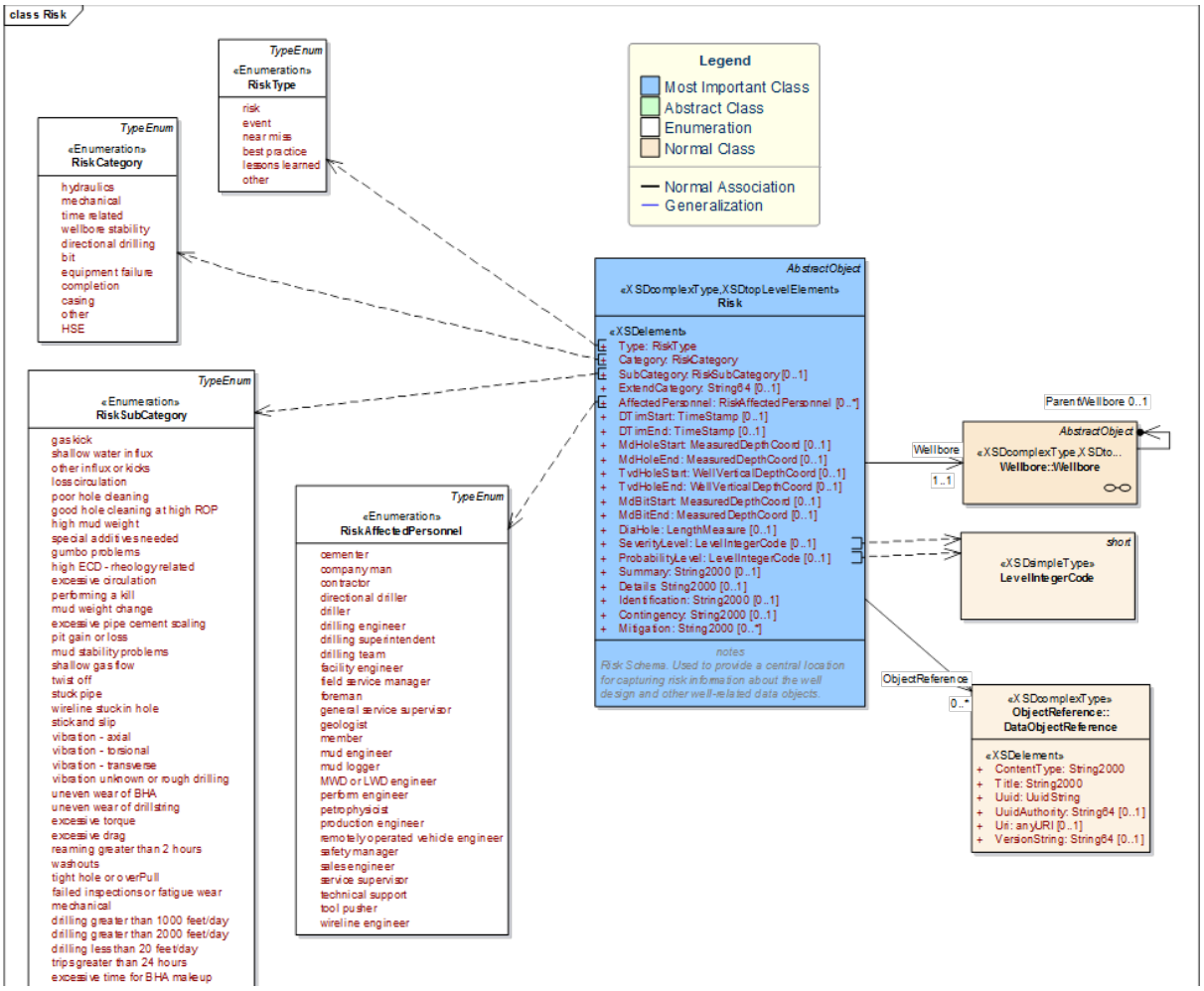

Figure 7–9. UML diagram of the risk data object. Note: RiskSubCategory has been truncated to fit the page. For the complete list, see the WITSML UML model (XMI file).

7.7 Attachment Data Object

Use the attachment data object (**Figure 7-10**) to "attach" any binary file to a WITSML object or sub-object. The attachment object is a top-level WITSML data object and references the object it is attached to using the CTA data object reference. If used, the attachment object must also reference its parent wellbore object. Attachments are typically images (pictures, schematics, etc.), which may be attached to:

- Top-level objects (e.g., attach a picture of the rig to a rig data object).
- Child objects such as a trajectory station (e.g., a schematic of the wellbore trajectory).
- Specific depth of an interval (e.g., pictures of cuttings from a particular depth).

The “content” tag contains the binary file. It is base64 encoded, so it can be contained in an XML document. Additionally, it contains some metadata such as file type (i.e., jpg, bmp, and pdf), description, etc.

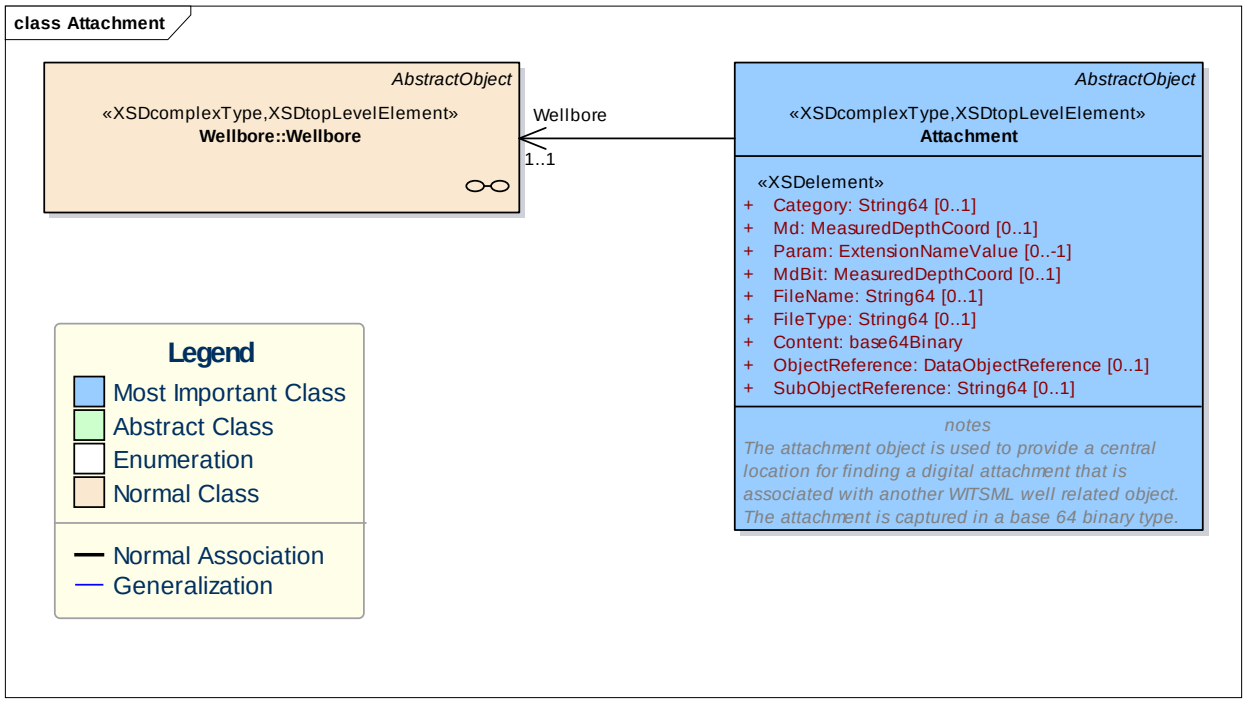

Figure 7–10. UML diagram of the attachment data object.

7.8 Depth-Registered Image Data Object

The set of data objects for raster well log depth registration (DepthRegImage) provides a common, industry-standard depth calibration (registration) format that improves on and replaces existing proprietary standards. It allows service companies, data vendors, and customers to more readily associate depth registration information with the correct log and move well logs and registration information between software systems.

The work to design these data objects began with an assessment of current popular, proprietary formats contributed by team members and work done previously by the WITSML SIG.

7.8.1 Business Purpose

Well logs have been used in oil and gas operations for more than 100 years. Today, log data is captured digitally. But until about 1980, logs were recorded on paper. Because the life cycle of an oil field is long, “old logs” provide vital data for modern-day operations.

Paper logs are scanned and saved as a raster format (for example, a .TIF file format), which are converted to digital data—or digitized. However, accurate interpretation of the raster format requires additional processing and the resulting data.

A depth registration is performed that provides digital depth references on the raster log image. The results of the calibration are stored in another file, separate from the image, which is commonly known as the depth registration file. The primary role of the depth registration file is to match pixels from the well log image with depth pixels represented in the wellbore. This file acts as a vehicle for the storage, maintenance, and continuity of the raster image and its calibration for future use.

Understanding the data in these logs requires both the image file and the depth registration file. But in the course of business, problems arise and the files get separated (often resulting in someone calibrating the

file again), then multiple versions of the file may exist and there is uncertainty on which is the correct version.

Additionally, various vendors offer depth calibration services—but each vendor's depth calibration file had proprietary and incompatible formats.

For details of the supported use case, see Section 7.9.4 (page 71)

7.8.2 Example Log

Figure 7-11 is the top portion of a raster well log, which includes the header and the top part of a log section. This figure is referenced when defining data objects below.

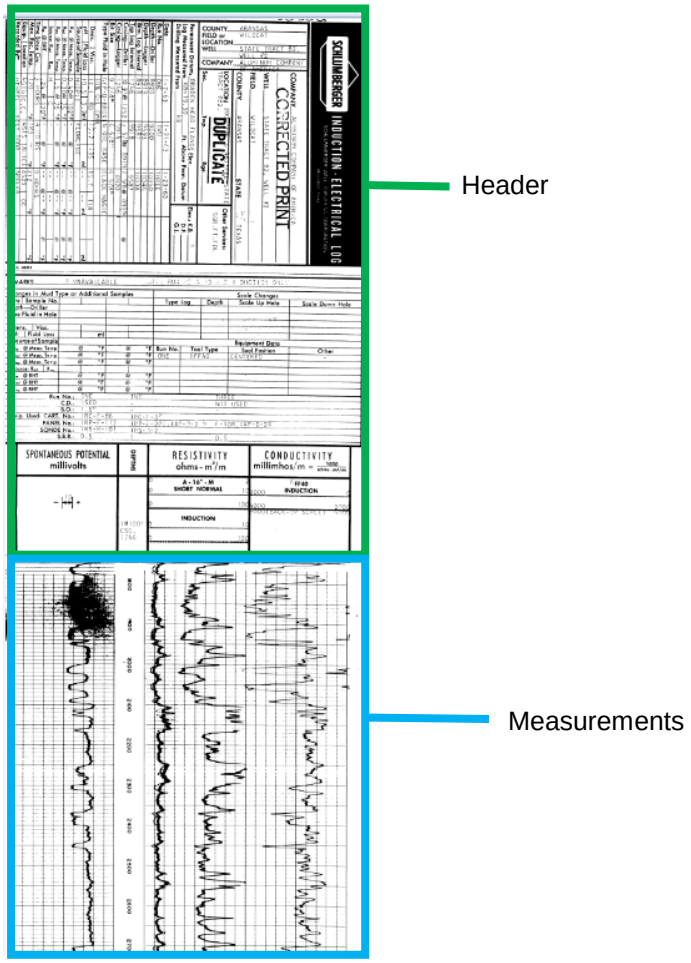

Figure 7–11. Example raster well log, with main areas highlighted.

7.8.3 Data Model

Figure 7-12 is the UML model of the depth reg image data objects. This set of objects defines the raster log image as a series of rectangles, the contents of each rectangle (DepthRegTrack (**Figure 7-13**)), and how the rectangles are related. These objects bring together information about the image boundary, the header/alternate sections, metadata on log sections, calibration points, and depth curves. The schemas describe the dimensions of the raster image, rather than a dump of the data points of the raster curves, in a format that all other providers can use. Specific objects on the diagram include:

- **DepthRegImage.** The top-level object that contains all the information about the composition, layout, and depth registration of the raster well log file.

- **DepthRegLogRect**. Defines a region of an image containing a header or alternate sections as defined on Figure 7-11.
- **DepthRegRectangular**. Uses four corner points (ul, ur, ll, lr) to define the position (pixel) of a rectangular area of an image, using x-y coordinates. Most objects point to this object because most are rectangles, and use this schema to define each rectangle.
- **DepthRegLogSection**. Defines the description and coordinates of a well log section, the curves on the log. An important element to note is **log:refNameString**; it is a reference to the actual log/data (in a WITSML server) that this raster image represents; this object does not contain the log data.

  The three arrows at the lower left of this object—upperCurveScaleRect, lowerCurveScaleRect, and whitespace—are used to identify the corner points of rectangles bounding those elements of the raster log.
- **DepthRegCalibrationPoint**. Defines a mapping of pixel positions on the log image to significant locations on the log. Specifically, pixels along the depth track are tagged with the matching measured depth for that position.
- **DepthRegParameter** Defines parameters associated with the log section and includes top and bottom indexes, a description string, and mnemonic.

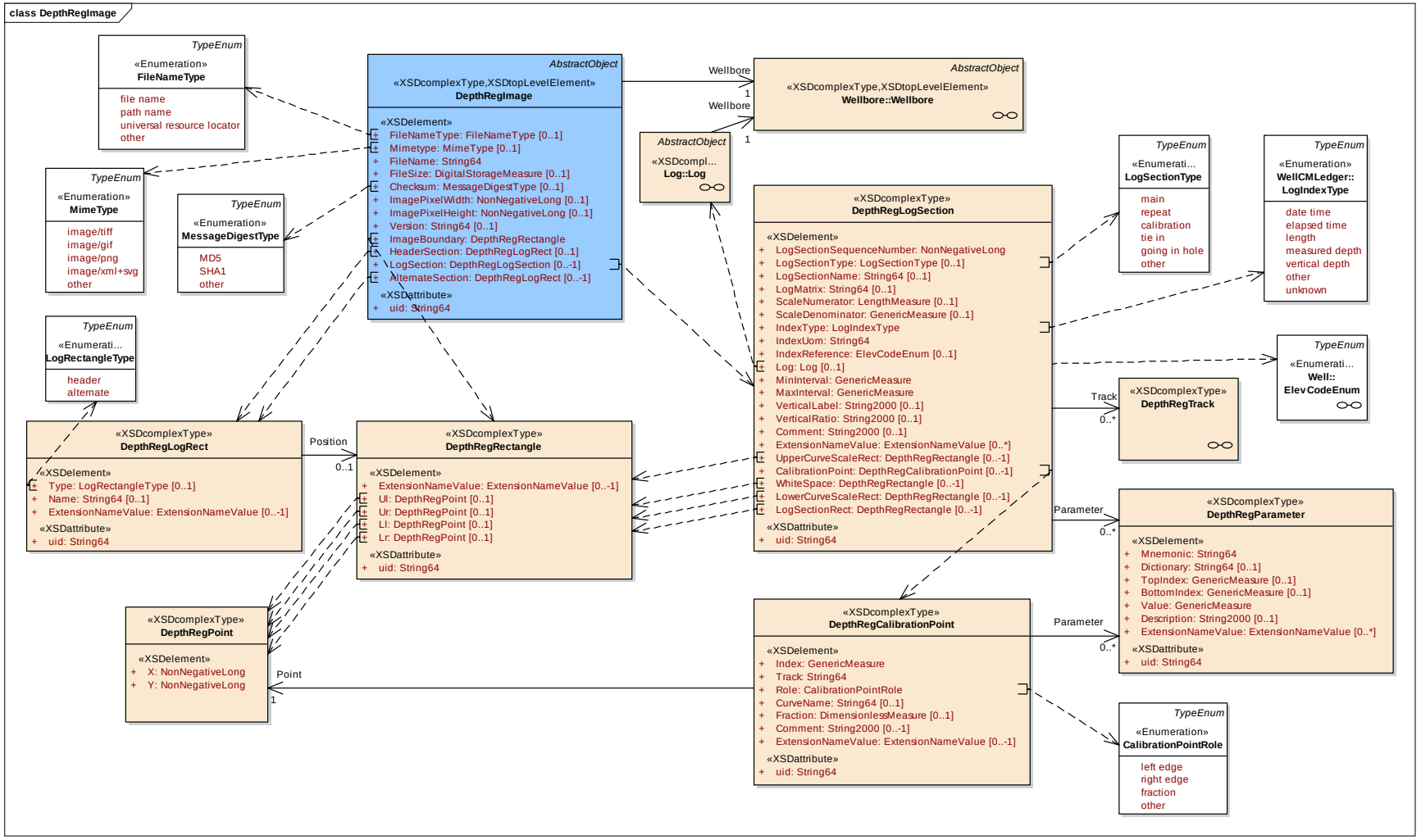

Figure 7–12. UML diagram of the depth register image data object.

Figure 7-13 is a UML diagram of the DepthRegTrack, which identifies the rectangle for a single log track. The DepthRegTrackCurve provides a description of the actual curve, including elements such as line weight, color, and style, within a log track.

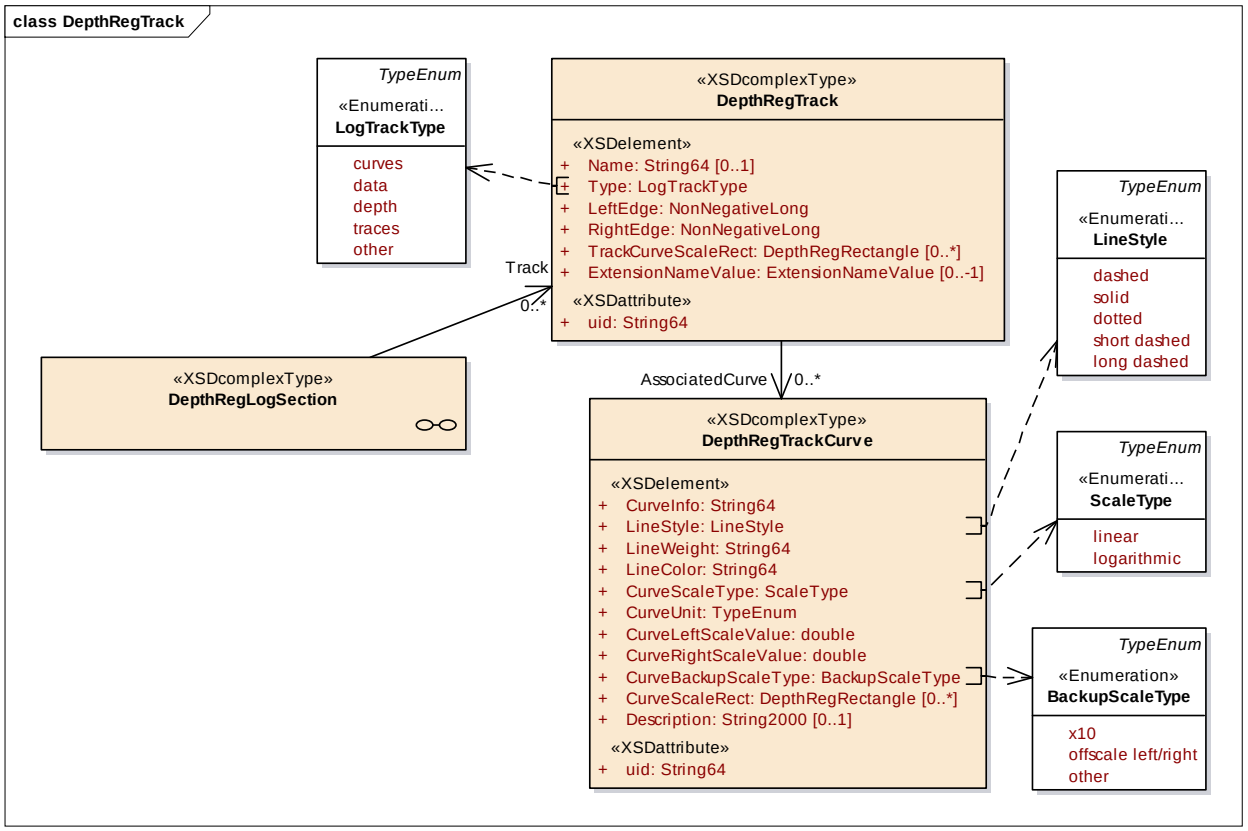

Figure 7–13. UML model of a depth registration track, a single "rectangle" in the depth reg image object.

7.8.4 Use Case: Digitizing Depth-Registered Raster Well Logs

Depth registration allows for the measured depth to be derived from a given location on a printed or raster image (**Figure 7-14**). Advanced depth registrations may allow for obtaining additional information such as the curves present and their corresponding values at a given depth.

Digitizing is the process of converting graphical features into vector values. In the case of well logs, and in particular scanned well logs, the process involved creating depth/value pairs of the well log curves on the log from an image of the log. Digitizing is often required when the only available well log existing is in hardcopy or scanned format. As a result of well logs being created since the 1930s for millions of wells in the US alone, digitizing a log is often the only way to obtain digital values of the well log curves. Obtaining the well log curve values may be required to perform log editing, splicing or other advanced petrophysical calculations.

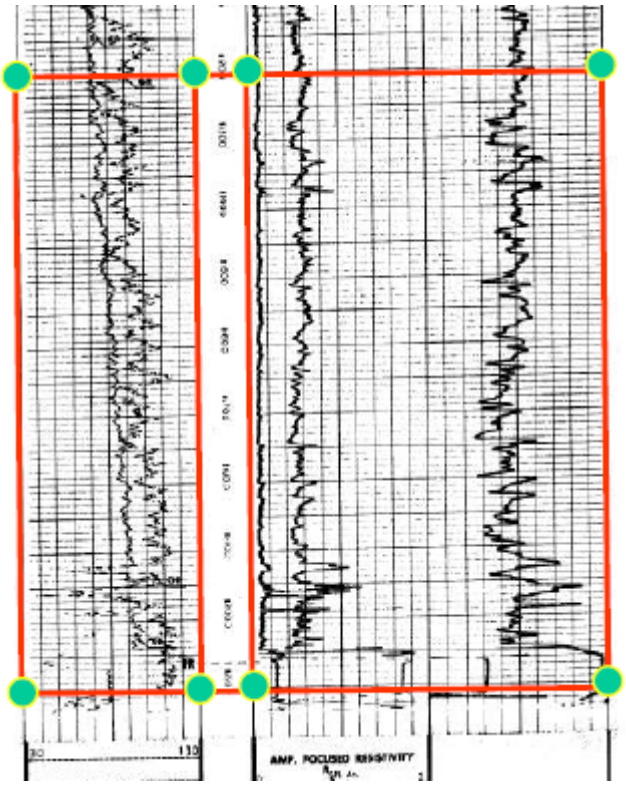

Figure 7–14. Registration data required to understand a digitized log image includes specifying coordinates of rectangular areas on the log.

7.8.4.1 Process for Digitizing a Log

To digitize a log, the following process must be followed:

1. Identify the associated well using header information.
2. Identify the well log tracks, for each track:
   - Identify track left and right edges
   - Identify depth markers in the track, tag depth
3. For each curve on a well log track, do the following:
   - Identify track left and right scale values
   - Identify curve backup regime
   - Capture curve points either manually or using an automated process
4. For all captured curve information, use mathematical transformations to convert the captured curve pixel locations to depth/value pairs for each curve.
5. Resample curve depth value points to industry standard depth step (i.e. 0.5ft, 0.1524m).
6. Generate flat file of the curves and metadata in industry standard format, such as LAS, ASCII or Excel.

7.8.4.2 Key Metadata to Support Digitization

- Well Information
  - UWI
  - Log Datum
  - Location

- Logging Information
  - Log Service
  - Logging Vendor
  - Logging Parameters (BHT, Mud Resistivity, date, etc.)
- Raster Image Depth Registration Information:
  - Well log header rectangle
  - Well log interval(s)—array of:
    - Log Scale (i.e., 1”/100’, 1:1000)
    - Description (i.e. Main Pass, Repeat Section)
    - Upper Scale Section Rectangle
    - Lower Scale Section Rectangle
    - Depth Registered Log Interval
      - Well Log Track(s)—array of:
        - Track depth segments - array of
          - Segment spanning left and right sides of track
          - Associated depth value
        - Associated curve(s)—array of:
          - Curve Mnem
          - Line Style (solid, dashed)
          - Line Color
          - Curve Scale Information:
            - Scale Type (Linear, Logarithmic)
            - Units (GAPI, MV, OHMM)
            - Left and Right Scale Values
            - Backup Scale Type (Offscale Left/Right, x10, etc.)

7.8.4.3 Example

Figure 7-15 shows how a properly depth-calibrated raster log can be digitized. Note that the depth track grid lines span the track and each depth grid segment overlays the corresponding depth timing mark. The digitized curves overlay the actual curves in the image. The pixel curve overlays can be converted to depth/value pairs using the registration information.

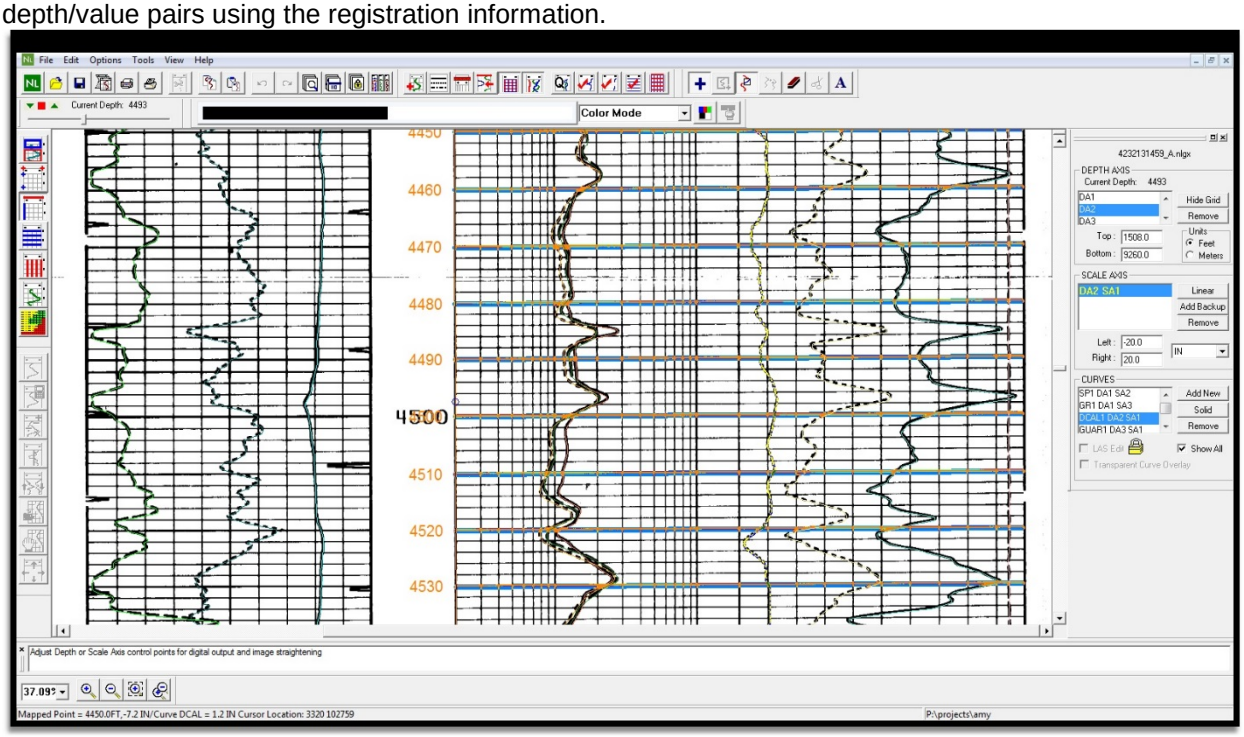

Figure 7–15. To accurately digitalize a raster log, you must first calibrate it. (Image courtesy of Neuralog, Inc.)

8 Completion Data Object

The Completion data object is actually a set of four top-level data objects and other children objects that describes a well completion throughout its whole history. With a standardized description, completion data can be accurately, reliably, and consistently exchanged between the many software applications used to plan, manage, and record work on a completion over the life of a well.

8.1 Scope, Use Cases, and Key Concepts

The set of completion data objects includes the completion equipment from the sandface connection to the reservoir, up to and including the tubing spool/tubing hanger, for use cases defined in this chapter. It can be used to transfer or exchange data between software applications or databases for these high-level workflows:

- Drilling and initial installation of the completion.
- Handover from drilling to production.
- Transfer of data between engineering applications.
- Well services through the life of an asset (which may involve links to other WITSML or similar data objects, for example, for the details of drilling, cementing, fracturing and other “job” types yet to be supported).

These use cases are supported:

- “Snapshot” of a completion at one instant in time
- “Change Log” of events, capturing the changes to a completion between two moments in time
- “Cumulative History” of the well over its whole life

The three main use cases relate to each other to enable systems to track progress and changes made to the completion, and to transfer the actual configuration at any instant in time. However, while the set of completion data objects supports these use cases, the ability of individual software applications to support them may vary.

These concepts are explained below with diagrams to relate them to real-world assets and activities.

Note that while the well completion hardware is modeled explicitly in detail and operations are handled with as much detail as existing data objects allow, flow paths of fluids through the whole completion are modeled in a limited sense. The connections to the reservoir are modeled, and these are associated with ports (or flows) at the wellhead. However, currently the detailed path in between is not explicitly described.

8.1.1 Snapshot Use Case

The snapshot use case refers to exchanging data that represents the well completion at a given instant in time. Everything in the data transfer represents equipment in the completion at the instant specified by the user. Conceptually, a snapshot corresponds to a completion diagram (**Figure 8-1**). However, a snapshot does NOT include any information about how components were installed or any service-related information.

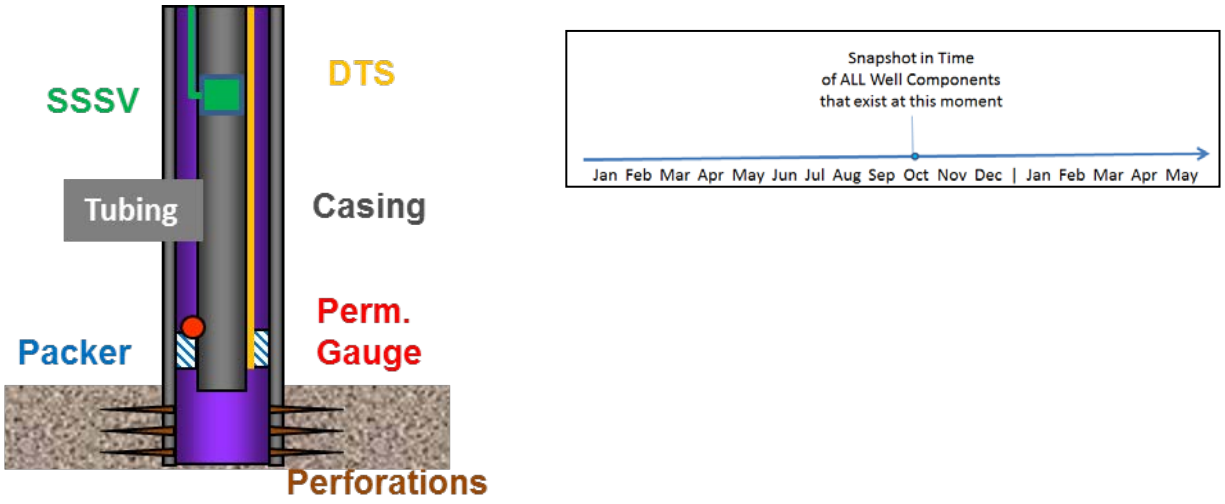

Figure 8–1. The snapshot use case is effectively the same as a completion diagram.

Note that the snapshot can include flow paths of fluids through the whole completion, modeled in a limited sense. The connections to the reservoir are modeled, and these are associated with ports (or flows) at the wellhead. However, the detailed path in between is not explicitly described.

8.1.2 Change Log Use Case

The change log use case is about exchanging ONLY the data that has changed over a specified time period. Components that did not change are NOT exchanged. This capability is useful to allow repeated synchronization of applications or databases.

Well and completion configurations change by performing or executing a "job", a single completion or well service (For more information, see Section 8.1.4, page 78). Information about the job and resulting changes are captured in an "event ledger" called the Well CM Ledger (CM stands for “construction and maintenance”), which captures necessary data about both the job that created the change and the result (e.g., equipment change) of the job. (For more information about the Well CM Ledger, see Section 8.2.2, page 84.)

Figure 8-2 shows an example perforation job (in a well that has already been cased, tubing has been run, and is ready to produce).

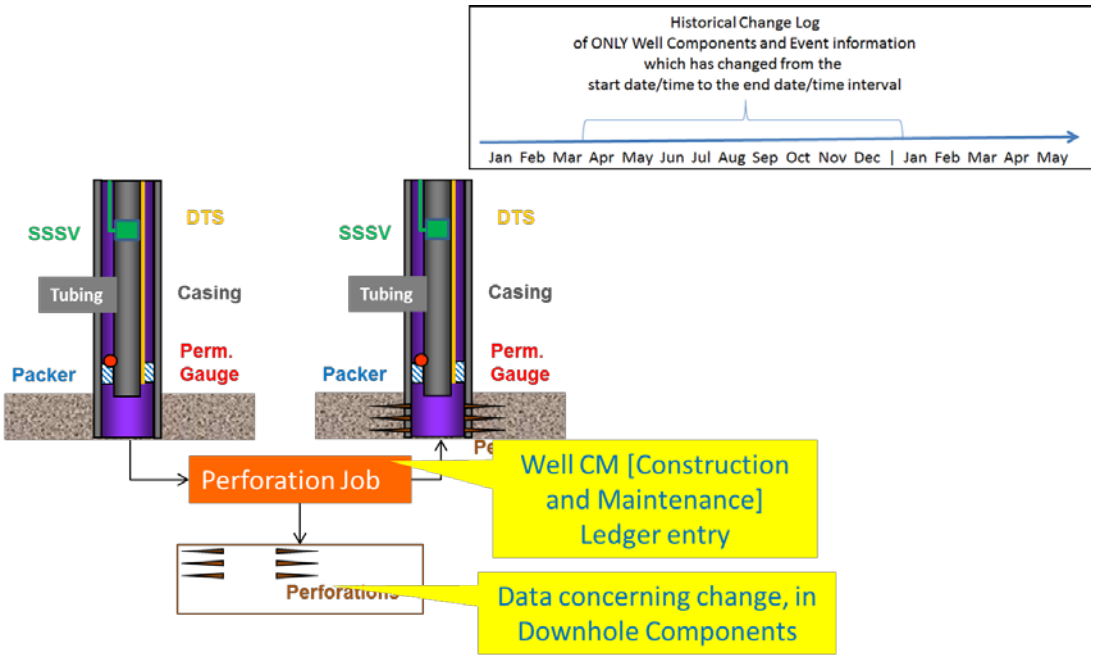

Figure 8–2. Example Well CM Ledger entry for a single "job".

The ledger in this case would have a single entry for this perforation job. The entry contains data about:

- **Job as a process.** In our perforation job example, job data would include which company performed the job, when it was run, equipment used, etc.
- **Result of the job.** The equipment or change associated with the well. In this example, it would be the addition of the perforations, but can also include installation, removal or replacement of equipment.

For this perforation example, the time at which the perforations become active is recorded in the equipment data object. Thus a snapshot for a time after the perforation job would show the perforations; whereas, a snapshot for a time before the job would not show the perforations.

8.1.3 Cumulative History

The cumulative history use case refers to the transfer of data corresponding to the whole history of the completion. The ledger described in the previous section is also used to determine all events and equipment changes (additions, replacements, and removals) as shown in **Figure 8-3.**

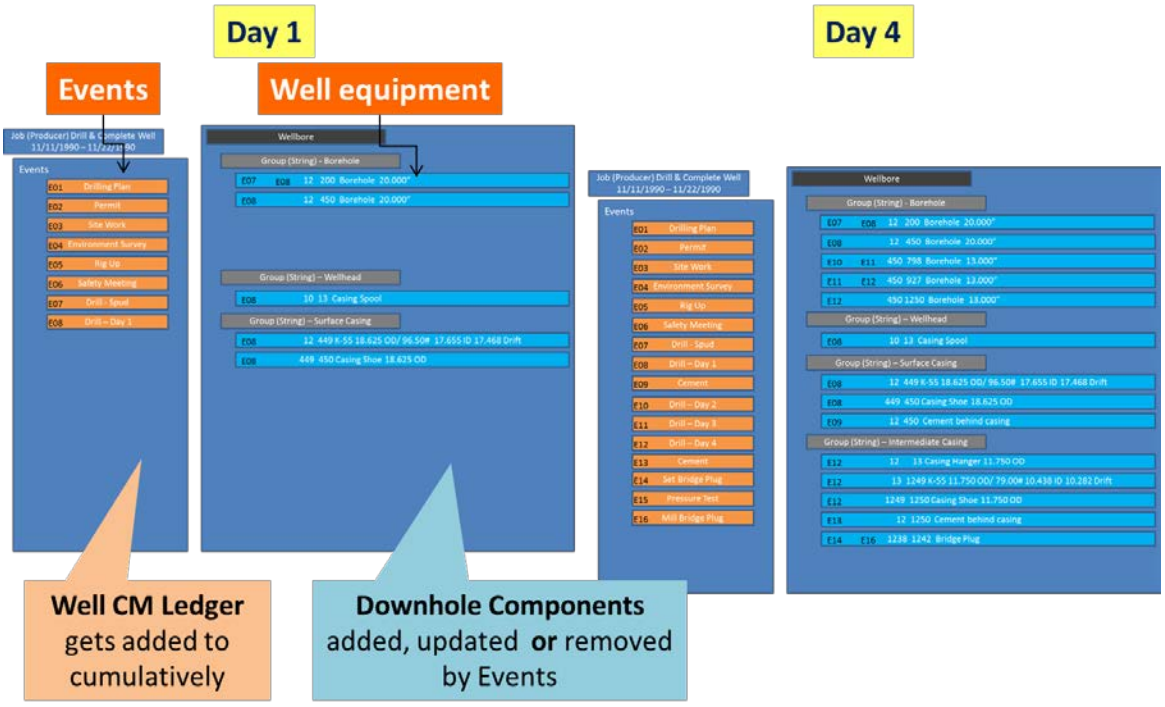

Figure 8–3. Example well cumulative history.

Figure 8-3 shows the data for the first few days of a well being drilled and completed. The events are shown as sequentially numbered and this ledger continues to grow through the life of the well. The "Wellbore Equipment" shows the physical installation of the completion. This installation includes the wellbore itself (i.e., the hole drilled in the earth) and the hardware (casing, tubing, etc.) installed in it. Note that each item of equipment contains the ID (the "Exx" in the figure) of the event that installed it, allowing the details (including date and time) of each piece's installation and removal to be discovered. When an element of equipment is removed, a second ID is added, which has the event that resulted in its removal.

For example, the last element of equipment is a bridge plug used to pressure test the casing; it was installed by event E14 and then later removed by event E16. The pressure test, E15, is between the installation and drilling out. The equipment entry shows both install and remove events at the time of the day 4 cumulative history.

Note also that the borehole equipment entries for each day get "removed" and replaced by a new deeper borehole when appropriate during the drilling phase, with the day's drilling report being the event that triggers this new equipment entry. This can be seen by looking at, e.g., the second borehole size running on from 450 depth units, which is drilled to 798 by event E10 ("drilling day 2"). Then this borehole data object is removed by event E11 ("drilling day 3") and replaced by a deeper borehole to 927, and finally is established by E12 as drilled to 1250.

The snapshot use case can be thought of as a query on a cumulative history at a given instant in time. The change log use case can be thought of as a query on a cumulative history between two points in time, giving the "delta" events between those times.

8.1.4 Jobs, Events, and Service Company Data

8.1.4.1 Job vs. Event

The notion of a *job* is an important one when considering a well completion; it is through a series of jobs that the completion is actually built, modified, or removed.

However, exactly what constitutes or defines a *job* can vary widely from company to company. **Figure 8-4** shows several examples of common jobs—from both the operator/producer and service company perspectives—related to wellbore completions.

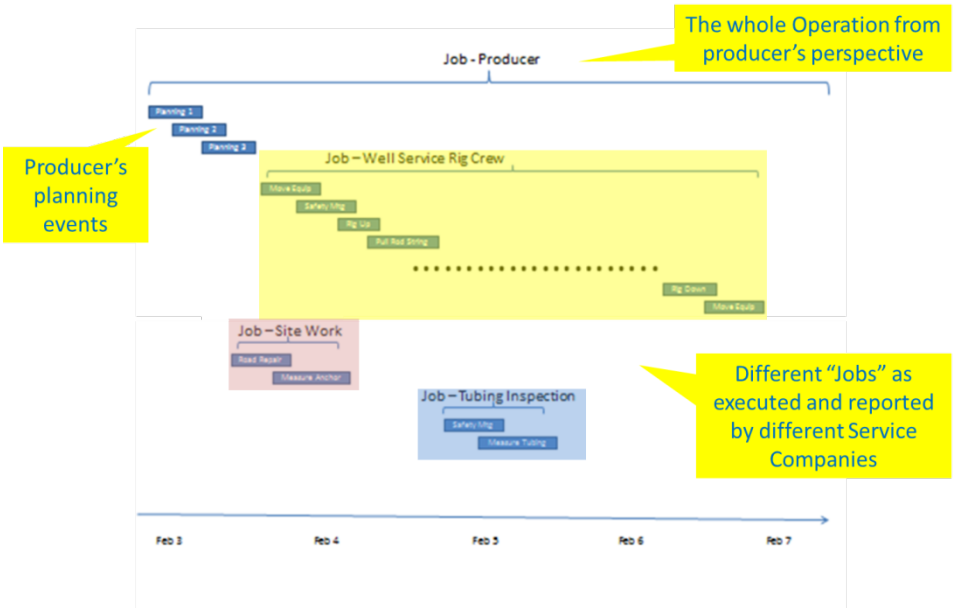

Figure 8–4. Example well services jobs from both the producer and service company perspective.

The term *job* can be used to refer to everything from the entire scope of work required to install the completion (from the operator's perspective), to one single task of that overall job, for example, perforating the well. There are typically jobs within jobs, which form a nested or hierarchical relationship.

To avoid the varying definitions of the common industry term *job*, the completion data object instead uses the concept of an *event*.

8.1.4.2 Event: Definition

An *event* used in the context of the completion data object is any action or scope of work associated with the work on completion. Events are transferred in the Well CM Ledger data object (see Section 8.2.2, page 84).

Note that events may include activities that do NOT result in a physical change to the wellbore equipment, for example, reports, inspections, and meetings. However, the events of primary interest for the completion data object are those that add to, modify, or remove components from a completion.

To accommodate the nested or hierarchical nature of the work, events may be nested within events, by referring to a parent event. Thus, an event such as "complete the well" may have no parent, but subsequent events that were part of the completion operation may show the "complete the well" event as their parent.

Any WITSML data object can be referenced in an event, in addition to the downhole component data objects, as explained below. In addition, specific "Event Extension" data objects can be referenced, where these are specific to the kind of event, e.g., the stim job data object for a stimulation event. These data objects supply the operational details of an event to supplement the changes to the completion itself transferred using the set of completion data objects.

8.1.4.3 Service Company Data

The set of completion data objects includes the ability for service companies to submit data records to their operator clients corresponding to individual jobs (events) that they perform. Figure 8-4 shows an example workflow for this process. Within the overall job/event or complete operation from the well producer's (operator's) perspective, there may be multiple service providers, each of whom carries out

some of the stages of work. These are shown as colored bands in the figure. The concept is that each of these providers may supply Change Log type data objects containing the events and wellbore equipment elements for which they were responsible.

8.2 Data Model

The table here lists and describes the top-level data objects that define the set of completion data objects. Details of each are explained further in this section. Each of these objects has been converted into its own package in the UML model. For more information, see the XMI file downloaded with WITSML.

Data object	Parent Object	Description
Downhole Component	Well	Contains all the wellbore equipment in a well. Components are identified as to which wellbore they are in. This data object covers the openhole boreholes, the strings of equipment, the components in each string and the details of these components, and the connections from these elements to the reservoir.
Well CM Ledger	Well and Wellbore	This name is short for "Well Construction and Maintenance Ledger". Contains details of events (which were defined in Section 8.1.4.1, page 78). Each event also has a pointer to elements of downhole components that the event changed. NOTE: Energistics *common* now provides an Activity Model data object, which may be implemented as an alternative to the Well CM Ledger.
Well Completion	Well	Represents a "flow" or "stream" from the well (e.g., from a wellhead port) that is associated with a set of wellbore completions. When there is more than one such wellbore completion, the flows from them commingle in the well (the wellbore completions may be located in multiple wellbores). The well completion represents this commingled flow. It can be associated with business concept such as field, rights, etc.
Wellbore Completion	Well, Wellbore and Well Completion	Each wellbore completion represents a flowing connection between wellbore and reservoir. It contains contact intervals, which reference the *physical* aspects of these connections detailed in downhole components.

8.2.1 Downhole Component

The downhole component data object contains all information about the physical completion, all of the well equipment data. The major elements of this information are shown in **Figure 8-5** and further explained in the sections below.

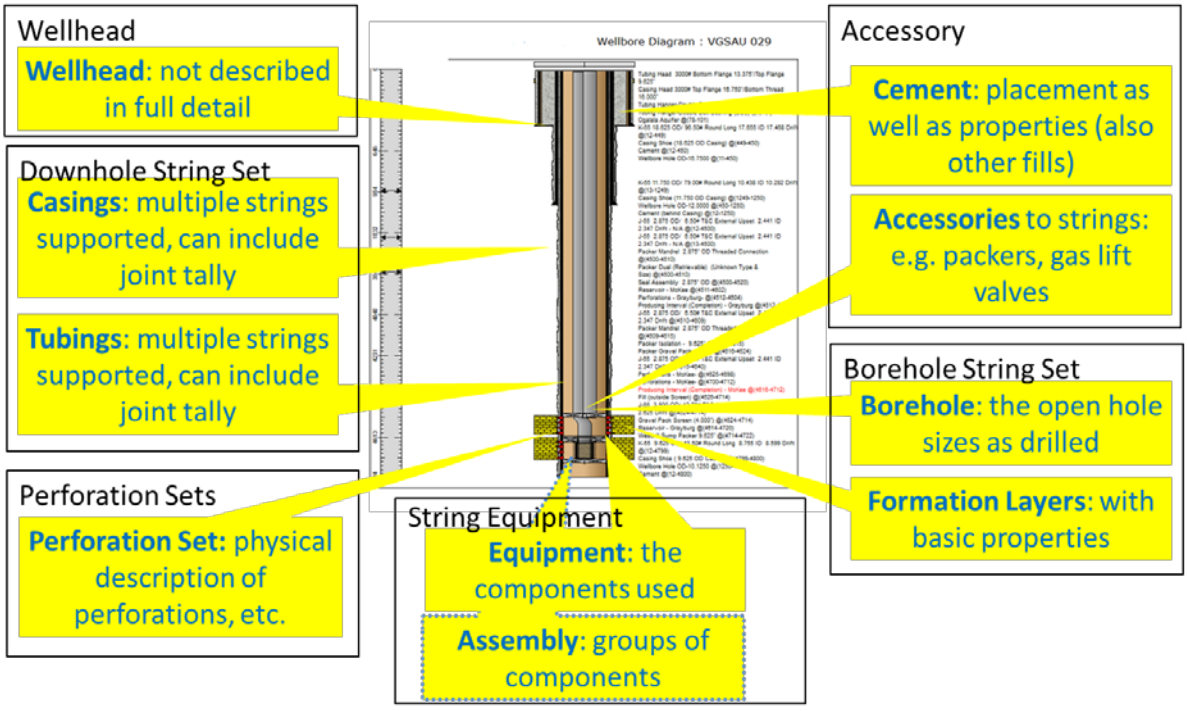

Figure 8–5. Scope of down hole component. These elements are described further in the text below.

Each of the constituent elements of the downhole component object (e.g., each item of equipment in a string) has a date element called event info. This event info element represents either the beginning or ending event for the item, and it contains a timestamp and a pointer reference to the well CM ledger entry corresponding to this event (thereby providing the link referred to above between event and equipment).

8.2.1.1 Wellhead

In this version of the data model, the wellhead is treated as a kind of *downhole string* (for details, see Section 8.2.1.3, page 82). It contains some data elements but is not intended to be comprehensive. A more detailed wellhead model could be developed in future releases.

8.2.1.2 Borehole String Set

This section of the schema is used to describe the borehole itself (the as-drilled hole through the earth) and some of the earth-related information, including the completed (flowing) intervals, the layers drilled through, etc.

Element	Description
Borehole String	Used to define the drilled hole that corresponds to the wellbore. A collection of contiguous and non-overlapping borehole sections is allowed. Each section has depth range, diameter, and kind.
Geology Feature	Aquifer or reservoir features are supported, and a range of parameters can be reported for each.
Accessory	Equipment or other data object associated with but not part of the drilled open hole being described. For example: cement, junk in hole.

8.2.1.3 Downhole String Set

This section of the schema is used to describe all strings in the completion, including casing, tubing, and rod strings.

A completion may have multiple sets of strings which may be nested each inside another, or run in parallel as in dual string completions; all strings are contained in a parent wellbore. Each string is composed of equipment, and may also contain accessories and/or assemblies.

Element	Description
Downhole String	A collection of strings is allowed. Strings: • Are considered to be contained in parent wellbores. • Each has a type (casing, tubing, or rod). • Contain a set of string equipment and can contain string accessories and assemblies (described below in this table). • Have install and removal dates.
String Equipment	Each string is created by mechanically coupling individual pieces of equipment, for example, tubing, gravel pack screens, etc. The string equipment element is used to define the set of equipment that makes up a particular string. Each item of equipment in the string equipment set includes this information: • Type defines the type of equipment from an enumerated list. • Depth defines the MD where the item of equipment is located in the string. • Connection next refers to the piece of equipment in the string to which the current equipment is connected. • Run history identifies if the equipment has been previously used. To avoid duplication of data, common equipment properties (such as diameter, model, materials, and drift) are stored in the equipment set. Each item of string equipment contains a reference (equipment ref UID) to the corresponding kind of component in the equipment set. Hence, string equipment covers “usage properties” and equipment set covers “common properties”. For more information, see Section 8.2.1.4, page 83.
Accessory	A string can contain one or more accessories, which refers to equipment or objects that are related to or part of the string but do NOT mechanically couple the string together, for example, a gas lift mandrel. “Fills” in the spaces between the strings are also accessories. For example: cement, or gravel or “accidental” objects, such as sand fill or dropped fish. Accessory components use the same data object as string equipment. They use data elements in the data object to define where the accessory is relative to the string and string equipment with which they are associated: • A Boolean “outside string” flag to indicate if the accessory is outside its parent string. • Two reference elements, “inside component” and “outside component”, point to the component either inside the accessory or outside it, to which it is attached. To describe a cemented casing string: • String type = casing o String equipment = casing pipe, casing shoe, etc. o ID = 1 then • Accessory = cement • Outside string = true

Element	Description
	• Outside component = [using *refContainer*, ID = 1]
Assembly	The concept of an assembly is used to refer to a collection of equipment as one "thing". It is expected that an assembly will be used where the level of detail required varies depending on users' needs and/or the software applications' capability to process and store those details. Its use is implementation-specific. For example, one piece of software (or user) may only care that there is an assembly representing "an ESP". In contrast, another piece of software or user may want to know about the detailed components of the ESP. In this case, the data that could be passed so the ESP components are all contained in the assembly called "ESP" and the first user would not need further information, while the second can look at the assembly element and get a list and structure of the components inside. Within an assembly element, the sub-components available (in string equipment) are the same as available in the string equipment element itself. Assembly can be used recursively.

8.2.1.4 Equipment Set

This section of the schema contains a collection of kinds of equipment that comprise the string. Each kind of equipment in the set has a type (what it is) and attributes common across all instances of that type of equipment. The string equipment then references these common attributes (as described above in Section 8.2.1.3, page 82).

Element	Description
Type	An enum list (equipment type) which is listed in the equipment schema.
Equipment Property Group	Common properties about the equipment, for example, the manufacturer, diameters, materials, etc. These properties apply to all equipment types .
Property	Extended (or "custom") properties for these equipment types. These properties are those which are unique to the particular equipment type.

The following simplified example shows how the string equipment and equipment set can be used. Suppose we have the following equipment at the specified depths:

1. Tubing 3.5 in. 0–2,000 m (meters)
2. Packer 2000–2005 m
3. Tubing 3.5 in. 2005–3,000 m

Conceptually this is represented in the data object as follows:

- Downhole Component
  - Downhole String name = "Tubing"
    - String Equipment Set
      1. Tubing 3.5" 0-2,000 meters, install date = abc, etc…EquipmentRefUID= A
      2. Packer 2000-2005 meters, install date = abc, etc…EquipmentRefUID= B
      3. Tubing 3.5" 2005-3,000 meters, install date = abc, etc…EquipmentRefUID= A
  - Equipment Set
    - ID= A, Type = tubing, OD = 3.5", ID = 3.0", manufacturer = abc, etc.
    - ID= B, Type = packer, OD = 4.9", seal kind = xyz, etc.

The efficiency of this approach is self-evident.

8.2.1.5 Perforation Sets

This section contains information about perforations in the downhole components. Each set has a reference to its related downhole string, and borehole string, and then gives the perforation details. For how this fits into the flow path description, see also Section 8.2.5.

8.2.2 Well Construction and Maintenance Ledger (Event Ledger)

NOTE: Energistics *common* now provides an activity model data object, which may be implemented as an alternative to the well CM ledger.

The "event ledger" concept is this: each time an activity associated with a well happens (an "event" as defined in Section 8.1.4.2, page 79) it is recorded in the ledger, which is named the well construction and maintenance ledger.

Recording this information tracks the history of all events and makes it possible to derive the complete history of the completion, including the use cases described above. A single event entry fills an instance of the data object well CM Ledger and can contain this information:

Information	Description
Identities	Well and wellbore name, event name, etc.
Event details	More detailed information about the operation of an event, e.g. depths, type, times, duration, etc.
Completion equipment reference	References to the downhole string (see Section 8.2.1.3, page 82), and the string equipment component being worked on within this string.
Operational and management information	Business associate, work order number, cost, rig ID, flags to indicate if the operation was planned or unplanned, etc.
Event Extension	Contains a reference to a WITSML data object containing the job data. These are listed as elements that reference AbstractEventExtension in the well CM ledger data object.
Participant	A reference to any WITSML data object associated with the event.

8.2.3 Well Completion

This is a top-level object that works with other objects to report flow paths from reservoir to surface. See Section 8.2.5.

8.2.4 Wellbore Completion

This is a top-level object that works with other objects to report flow paths from reservoir to surface. See Section 8.2.5.

8.2.5 Flow Paths from Reservoir to Wellhead

Two top-level objects are used in the flow path logic:

- Well completion
- Wellbore completion

These work together and with other sub-objects in a rather involved way to be able to report flow from defined places along the wellbore to defined surface streams. The following table summarizes how this is done. See also **Figure 8-6**, page 86.

Element	Parent	Description
Well Completion	Well	Occurs as a top-level data object. Represents a “flow” or “stream” in the well. The data object also contains business information about this flow, such as field ID, ownership rights, etc. Referenced by the wellbore completion. Multiple wellbore completions can

Element	Parent	Description
		be associated with a well completion. In this way, a commingled set of connections with the reservoir can be associated with a flow within the well. Generally, a well completion is associated with a single "wellhead stream", i.e., flow from a port in the wellhead. However, this is not always the case. For example, in cases where liquid and gas are separated downhole, the liquid may flow up the tubing and the gas up the casing annulus. In this case, there would be two "wellhead streams" from two ports (tubing and casing annulus), but both would be associated with the same well completion because they both come from the same set of commingled wellbore completions. The wellhead streams are reported in the PRODML Simple Product Volume data object, with a link to the Well Completion. For more information, see the *PRODML Technical Usage Guide* (Sections 5.2.3 and 6.2.2).
Wellbore Completion	Wellbore	Occurs as a top-level data object. Each individual wellbore completion data object represents a completion (i.e., open to flow) interval along a wellbore. Meaning "this section of wellbore is open to flow". Has data for depth, API numbering, type of fluid, etc. Contains a reference to a Well Completion data object (which is the flow stream from the wellhead), see above. Contains a Contact Interval Set, which represents the physical connection(s) to the reservoir.
Contact Interval Set	Wellbore Completion	Occurs in the wellbore completion data object. Contains one or more "xxx interval" objects, each representing the details of a single physical connection between well and reservoir, e.g., the perforation details, depth, reservoir connected. Meaning: this is the physical nature of a connection from reservoir to wellbore. The kinds of interval allowed are: • Gravel pack (refs downhole string) • Open hole (refs borehole string) • Perforated (refs perforation set) • Slotted (refs string equipment) Each interval has an ID, details about the interval itself, including history, and a reference to the data object giving full details (the kind of equipment is shown in parentheses in the list).
Perforation Set	Downhole Component	This section contains information about perforations in the downhole components. Each set has a reference to its related downhole string, and borehole string, and then gives the perforation details. It will be referred to from the perforated type of contact interval.
String Equipment	Downhole Component	This section contains the individual items of equipment deployed in the downhole component. A given item of equipment can have a reference to a perforation set, meaning "this item of equipment contains the perforations of *thi*s perforation set".

Note that flow is *not* modeled through the components making up the completion (for example, through sliding sleeves, specific tubing strings, etc.). Such a full "flow path" may be added as a later development.

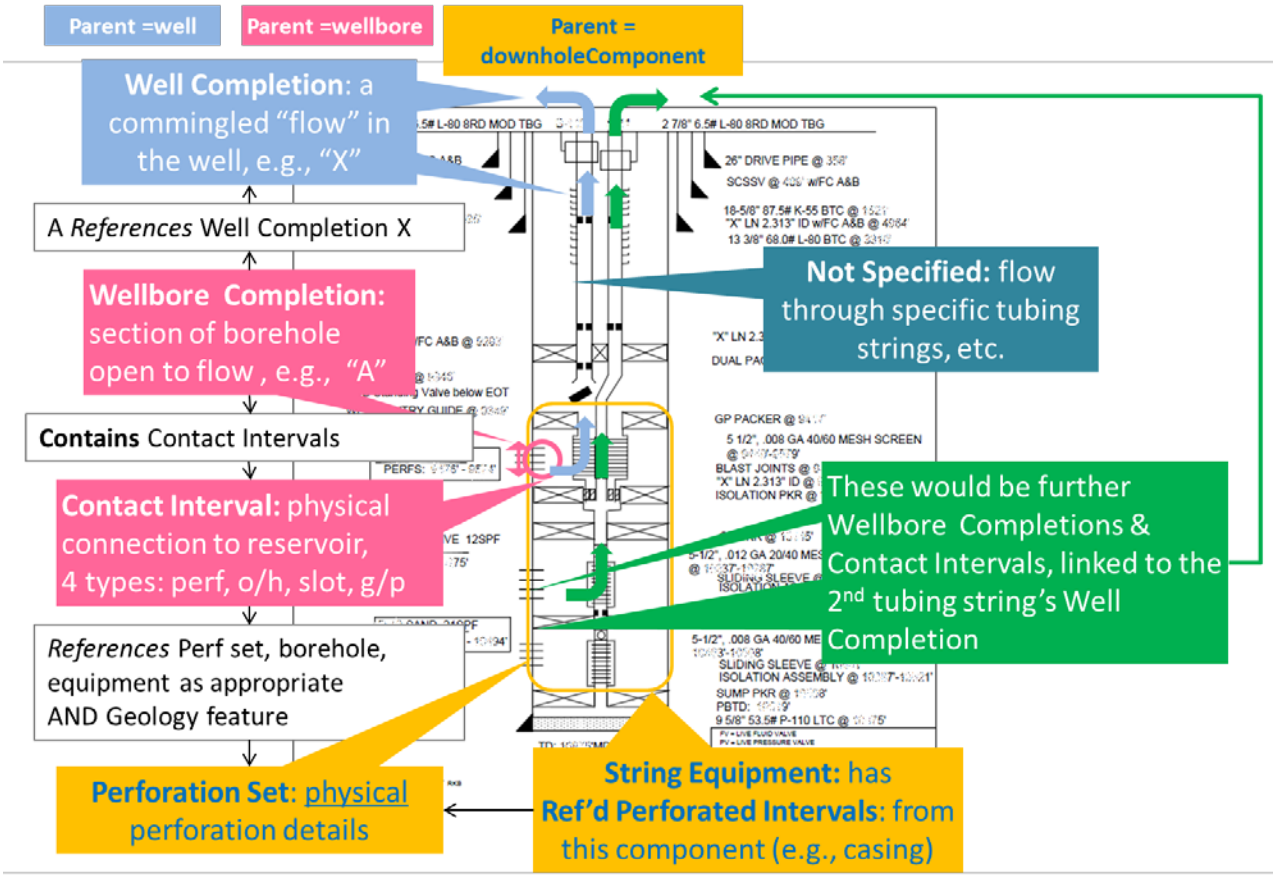

Figure 8–6. Scope of flow representation, showing elements and linkages used.

8.3 Role of "Well" and "Wellbore" Concepts as Applied in Completion

Most WITSML data objects are children of the wellbore data object. This organization is logical given that drilling operations are happening in only one wellbore within a well, at any given time. However, in a completed well with multiple wellbores, installed equipment spans all wellbores. For this reason, in the completion data object, the parent data object is the well, although many of the child data objects of completion fit under the wellbore concept.

Figure 8-7 shows the main associations with well (blue boxes) and with wellbore (pink boxes).

- The wellhead is associated with well.
- The whole set of completion equipment, downhole components, is also associated with well.
- Individual strings and the equipment in them, and boreholes are associated with wellbore.
- Concerning flow, wellbore completions and contact intervals are associated with wellbore.
- However the well completion data object, representing flow out of the wellhead, is associated with well because it may be a commingled flow from multiple wellbores.
- The well CM ledger can associate with multiple wells. Each entry in the ledger is associated with a well and, if appropriate, also with a specific *wellbore*.

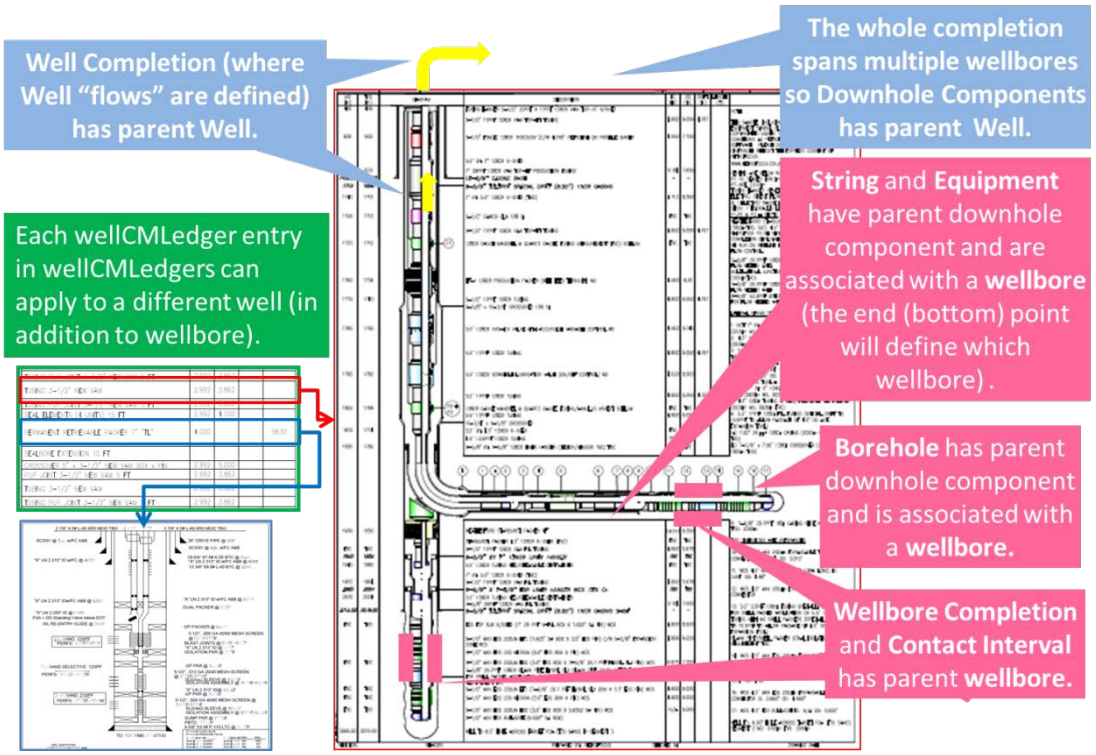

Figure 8–7. Completion data object in the context of well (blue boxes) and wellbore (pink boxes).

9 Stim Job Data Object

This chapter provides the business purpose of the stimulation job (StimJob) data object, describes changes between the previously published version (with WITSML v1.4.1 in 2012) and the latest version, and explains how the latest version works, with a focus on new key capabilities.

9.1 Introduction

The stim job data object was developed so that service companies can deliver stimulation job design and post-job reports to operators in an industry-defined standard electronic format. The current version leverages the stim job object last published with WITSML v1.4.1.1 (in 2012), with some significant redesign. Drivers for this redesign include the move to the Energistics v2.0 architecture and to:

- Accommodate industry changes (e.g., increasing use of horizontal wells and number of treatment stages in a well).
- Address issues with the previous stim job object and add design capabilities (in addition to reporting).

9.1.1 Business Purpose

Stimulation jobs, such as fracturing services, are performed on wells for the primary purpose of enhancing oil and gas production. Fracturing treatments are a very complex process and generate large volumes of data. This data is very valuable in determining the efficiency and optimization of oil and gas production from a well and at a broader level, the effect on the reservoirs within a field; therefore, it is essential that all data generated is properly recorded so it can be provided to the operator of the well.

The stim job object now allows the transfer of information describing both a stimulation design and the as-executed job, including a summary report of what was pumped, how it was pumped, observations made during the job, and summary interpretations of observed data.

The summary report content is already essentially standardized across fracturing service providers, with only slight differences in format and content. Historically, this data was provided to the well operator as a paper report. The previous version of the stim job object made it possible for fracturing service companies to consistently create and transmit an electronic version of this data to the operator.

9.1.2 Terminology

The following table provides brief definitions of terms as they are used in the context of the stim job data object.

Term	Definition
Job	A series of operations that stimulate a set of stages through high-pressure pumping of fluids within a single well over a period of time.
Stage	A section of a well that is being stimulated. NOTE: To better align with industry, the definition of the term ‘stage’ has been revised from the previous version of the stim job object.
Step	A pumping task performed during the stimulation of a stage. NOTE: In the previous version of the stim job object, this was known as a ‘stage’.
Flow path	The geometry of the path the fluid takes while a stage is being stimulated.

9.2 Overview of Major Changes and New Features

Because the previous version of the stim job object was widely used, this section provides a summary of changes from the previously published version to the current version.

Table 3. Summary of High-Level Changes v1.4.1. 1 Objects/Elements to v2.0

Change	Description/Explanation
Stage is now Step	The object previously named 'Stage' is now a 'Step' (StimJobStep).
Stim flow path is now optional	While very important, the flow path information is not always available. In the previous version of the stim job object, flow path was required, which hindered data transfer if flow path data was not available. Making it optional means it easier to transfer other stimulation job data even when flow path data is not available. Most elements on the StimFlowPath object have been moved to either StimJobStage or StimJobStep.
Treatment-related elements	Some elements were removed (because they were not being used) and the remaining elements moved to "Kind".
Custom data deleted	It was not being used.
Stim Material Catalog	New object added for each stimulation job. Use of this catalog eliminates the need to repeat material data on each stage; now it may be referenced from the catalog and only listed once per job.
Proppant	Has been moved to the new material library and additional ISO properties can be provided. For more information, see Section 9.3.3 (page 92).
Plural (a.k.a., "containers") no longer used.	Previous version of the StimJob object (and WITSML) required use of a plural object (to group together multiple occurrences of an object). This convention was eliminated by the Energistics CTA.
Wellbore geometry	Wellbore geometry step number in stage and has been moved to the flow path object.
NEW: Reference to WITSML Logs	The stim job object can now reference related logs in WITSML using the data object reference of the Energistics CTA.

9.3 Data Model

Figure 9-1 is a high-level UML diagram of the stim job object. Each stimulation job can have multiple stages (StimJobStage) (which itself is an Energistics top-level object), and each stage can have multiple steps (StimJobStep) that comprise the treatment or stimulation of that stage and are "children" of stage. **Figure 9-2** shows details of the stim job stage and other related objects for the stage, which are explained below.

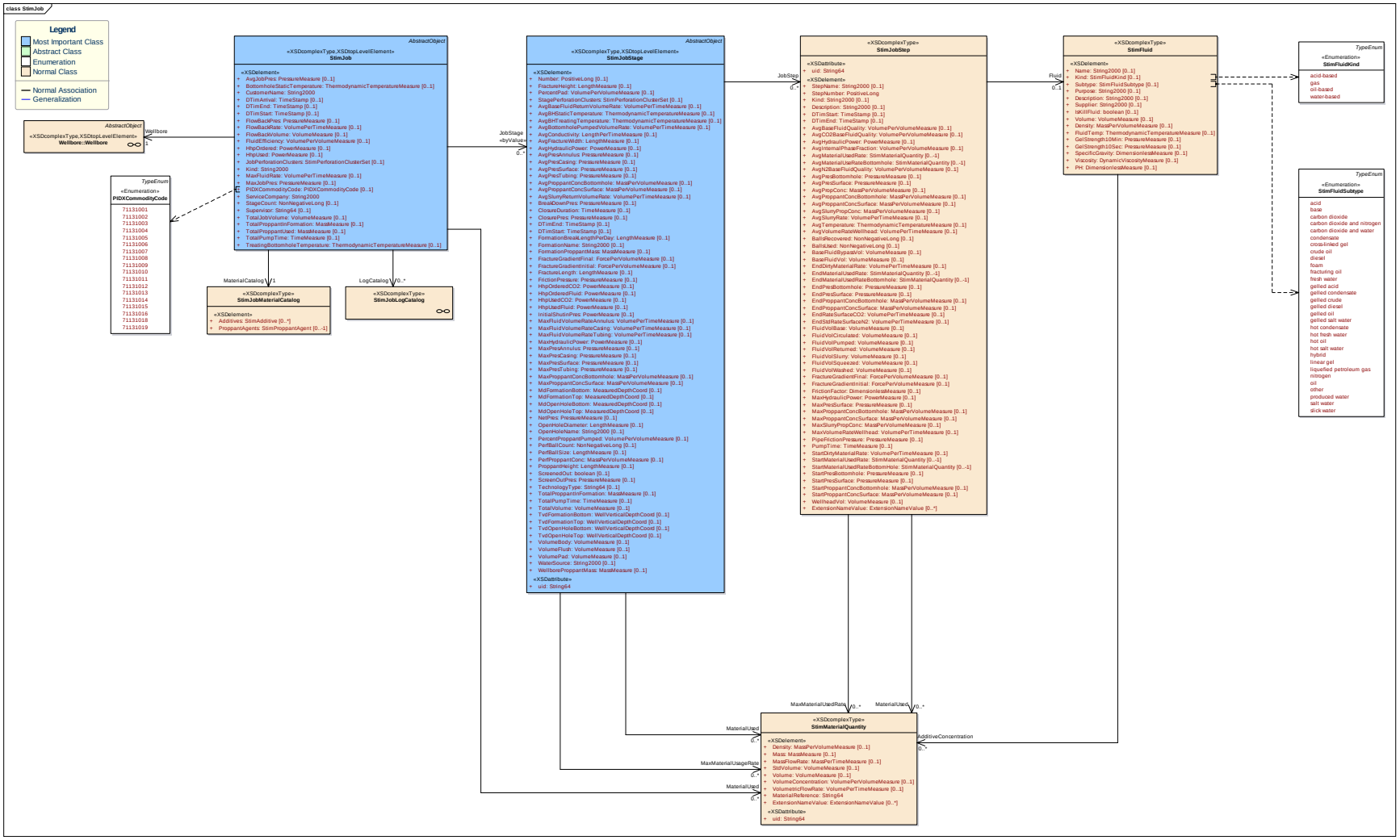

Figure 9–1. UML diagram of the stimulation job data object. A stim job may have multiple stages and a stage may have multiple steps.

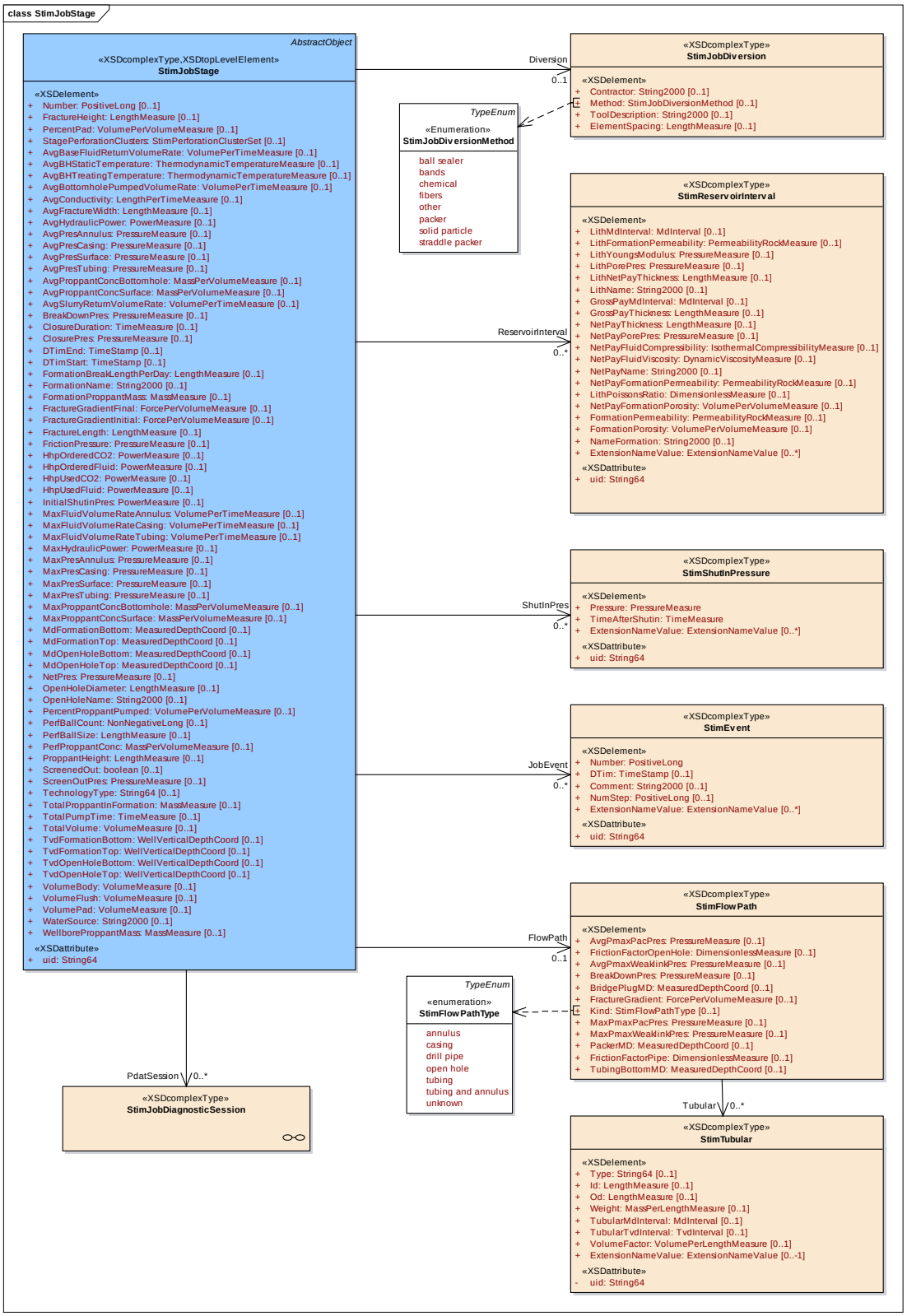

Figure 9–2. UML diagram of the stim job stage object and related objects.

9.3.1 Stim Job Stage

Figure 9-2 shows details of the stim job stage and other related objects for the stage. A stage represents a section of a wellbore that is being stimulated. The stim job stage object makes it possible to describe a summary of the operations for stimulating the stage and also records the details associated with the fluid flow path and the individual steps used during the stimulation.

For example, the stage object contains an optional reference to a flow path (StimFlowPath), which makes it possible to record the path of the fluid flow while stimulating the stage, and for a production enhancement job, can specify

9.3.1.1 Stim Flow Path

While stimulating a stage of the well, the fluid can move through a series of conducting tubulars that move the fluid from the surface pumps to the portion of the wellbore being stimulated. This data can be captured using the stim job flow path object (within the stage object). To allow stim job designs to be captured, this object is optional because this information usually is not available. The stim flow path can also reference the the tubular used (StimTubular).

9.3.1.2 Other Object Referenced from Stage

Additional related objects include:

- **Stim reservoir interval**. Information about the properties of the section of reservoir associated with this stage. You can also record the high-level overview of the reservoir intervals that are being stimulated using one or more reservoir interval objects.
- **Stim diversion**. Information on the method and tool for a diversion.
- **Stim shutin pressure**. A pressure measurement taken a certain time after the well has been shut in. This object is intended to be used with the performance enhancement schemas in PRODML.
- **Stim Events**. A log of events that occurred while simulating the stage.

9.3.2 Stim Job Step

During the stimulation of a stage, a series of steps are performed (Figure 9-1). These steps are synonymous with a pumping schedule, which provides a description of materials and fluids that are pumped at some rate and concentration over a period of time.

9.3.3 Specifying Stim Fluid, Materials, and Material Quantities

In the previous version of the stim job object, materials had to be repeated for each stage of a stimulation job. Because stimulation jobs in current industry practice now can contain 60 or more stages, a new, more efficient approach was needed.

A stim job object can now contain a catalog of materials for that particular job (see Figure 9-1 and **Figure 9-3**); any stage of the stimulation job can reference material(s) in the catalog, thereby eliminating the need to repeat the materials for each stage.

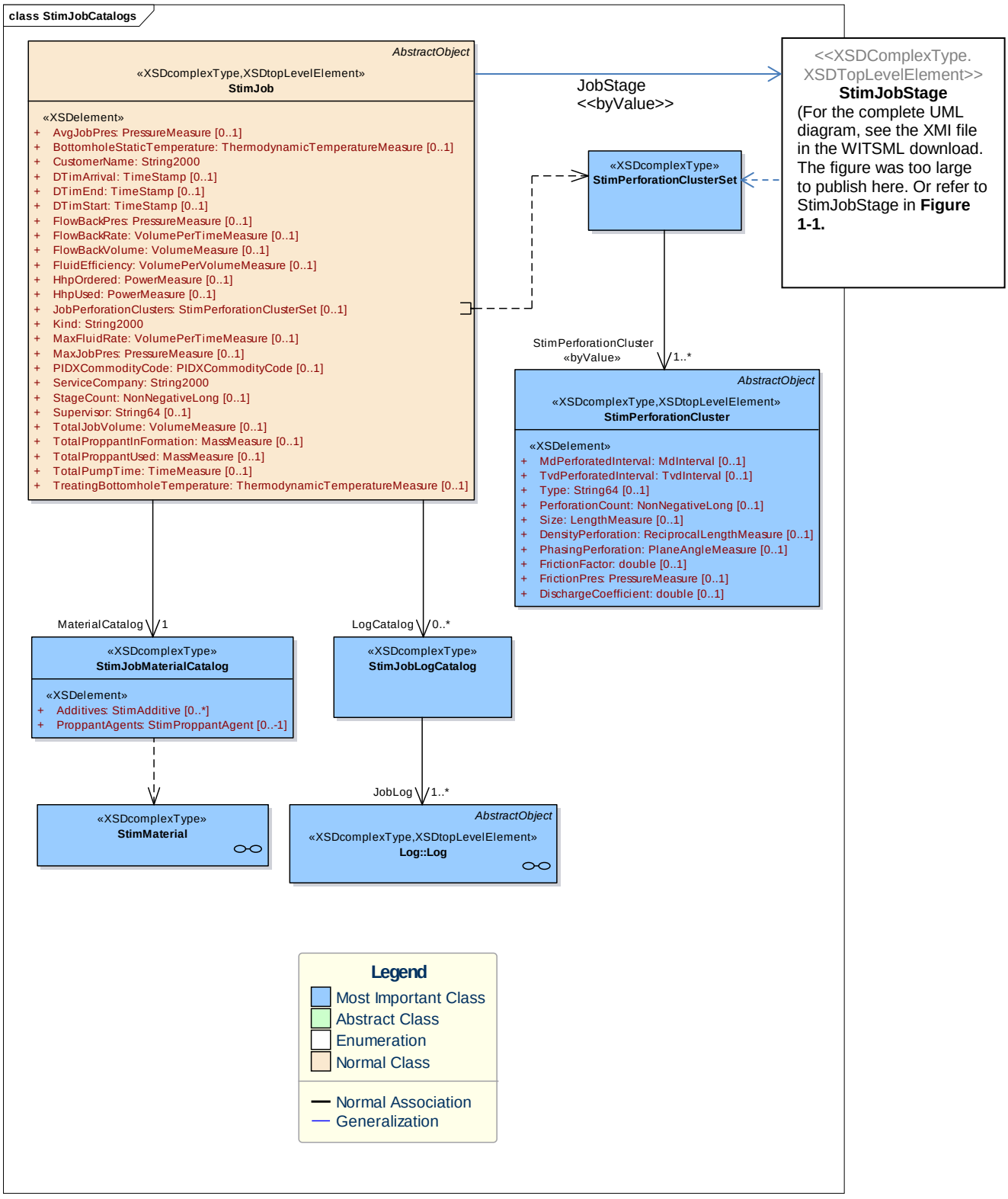

Figure 9–3. Each stim job object now has a library of materials that each stage can reference, instead of repeating the materials for each stage.

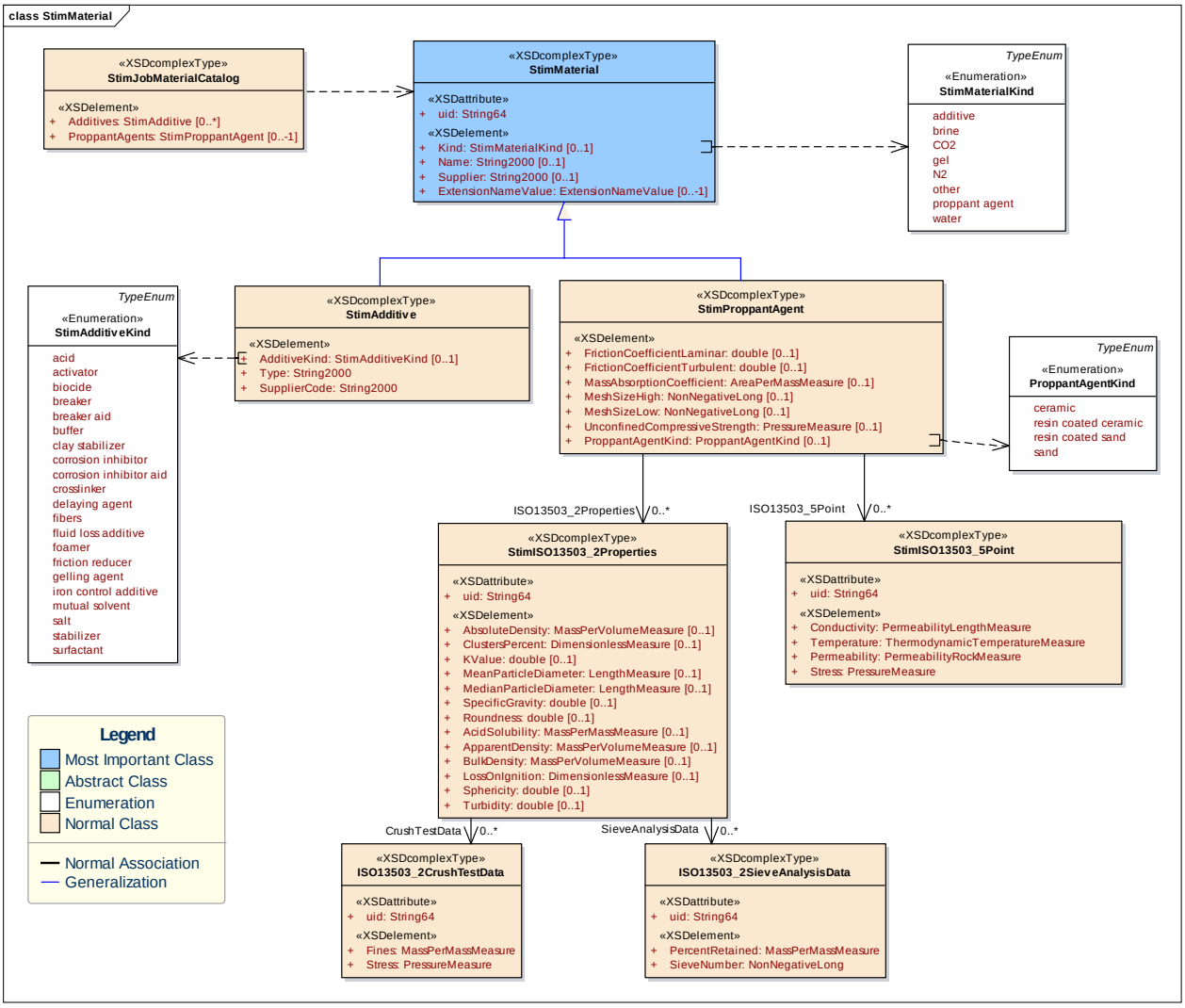

Figure 9–4. Stim material quantity (on Figure 9-1) references a stim material using a "material reference" string, which is the UUID.

The stim fluid object is the fluid (without the proppant) that is carrying the proppant downhole. Proppants as a concept refers to the materials left in the well or consumed in the process of making the stimulation.

The stim material catalog object contains a set of stim material objects representing additives and proppants that are used during a stimulation job. These materials can be used as a catalog by referencing the material from stim material quantity using a UUID instead of repeating the material properties.

Figure 9-4 shows a UML diagram of individual materials. For proppants, additional details can be provided (if available). For example, properties related to ISO 13503.2 properties (e.g., density, solubility, apparent density) and crush tests, and ISO 13503.6 point data (e.g., conductivity, permeability and stress) can describe useful details of the proppant material properties.

The stim fluid object uses the stim material quantity objects to get the "material reference" string, which is the UUID of stim material quantity. Each material type has a UUID as an attribute; and references are always that UUID.

9.3.4 Perforations

In the current version, a stimulation job contains a set of perforation clusters, which are weakly referenced to a perforation library (similar to the material library described above) using the UUID. In the previous version, perfs were on the interval to capture that data as part of the current stimulation job.

However, if perforations already exist (i.e., if you are refracking a previously stimulated well), then that perforation information is needed to accurately design the new stim job.

9.3.5 Referencing Logs

Logs taken during the stimulation job or used as reference while designing or conducting the stimulation job provide useful information. The stim job object can reference any relevant WITSML logs using the data object reference, a component of the Energistics CTA.

10 Cement Job Data Object

Accurate planned and actual cement data is crucial from a regulatory and performance perspective. Using WITSML to transmit this data to a client's system of record can greatly improve data quality, efficiency of cement job performance reviews, and regulatory reporting and compliance.

10.1 Data Model

(**Figure 10-1**) shows the UML model for the cement job and cement job evaluation objects and related objects in this set. Remember, all Energistics data object schemas (XSD files) are produced from the UML model; so the UML model reflects the schema organization. This object has two main top-level data objects and the following sub-objects organized like this:

Cement Job (CementJob) object. Contains identifying information about the cement job, and references related objects about the design and execution of the cement job, including a final cement job report. Child objects include:

- Cement Fluid (CementingFluid)
- Cement Additives (CementAdditive)
- Cement Job Design (CementJobDesign)
  - Cement Stage Design (CementStageDesign)
    - Cement Pump Schedule (CementPumpScheduleStep)
  - Cement Tops (FluidLocation)
- Cement Job Report (CementJobReport)
  - Cement Stage Report (CementStageReport)

Cement Job Evaluation (CementJobEvaluation) object. Contains data for the testing and evaluation of a previously performed cement job.

More information about the design and relationships of these objects are explained below the figure.

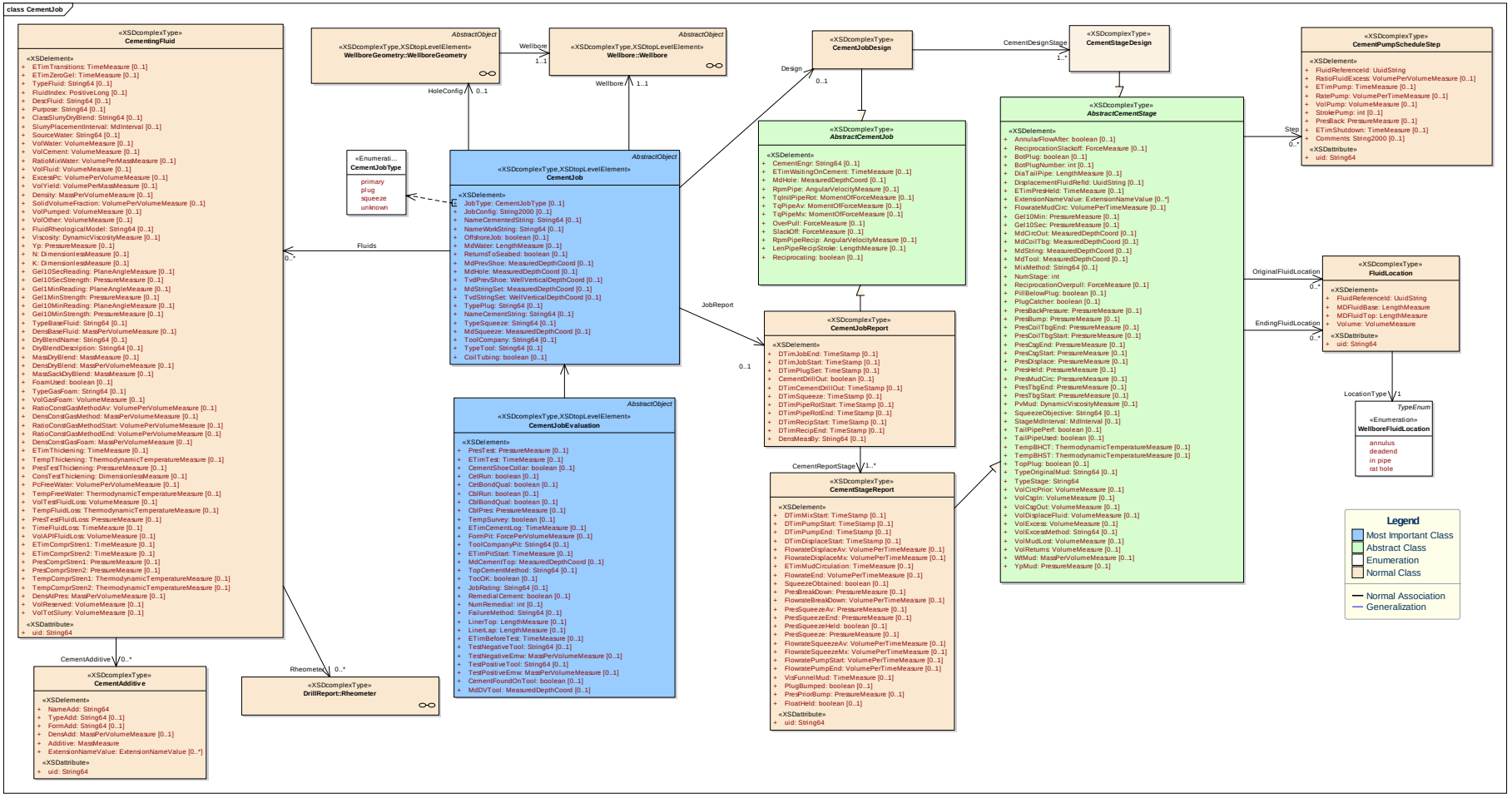

Figure 10–1. UML model of the cement job object

10.1.1 Cement Fluids

Capturing accurate cement fluid specifications and designs is crucial for tracking cement job performance. Quality data can assist in determining the best cement slurry to use for a particular application/operation.

The cement fluids object captures data such as: the type of cement fluids used along with the composition of fluids in a particular cement slurry, which can differ greatly from one well to another. Accurately capturing data pertaining to the cement fluid temperature, gel strength, water content viscosity, foaming agents can assist in optimizing cement fluid designs.

10.1.2 Cement Additives

Cement additive specifications and designs are also crucial for tracking cement job performance. The type of cement additives used along with the composition can affect the properties and characteristics of a cement slurry and can be specifically engineered or standardized based on cementing operations. It is imperative to capture the types and quantity of all additives, retardants, and accelerants that have been added to a cement slurry.

10.1.3 Cement Job Design

At the highest level of the cement job object, design data can be captured regarding the type of cementing operation, associated string cemented, associated wellbore, and start and end dates. After the data has been imported into a system of record, data can be aggregated and analyzed for general performance data with regard to types of cement operations, duration of operation, etc.

10.1.3.1 Cement Stage Design

Each cement job consists of one or more cement stages, which is a measured depth range in the wellbore over which a cementing operation is performed. Initial, final, and sustained pump rates and pressure are recorded for each stage. This data is compared with the actual data in the cement stage report.

10.1.3.1.1 Cement Pump Schedule

The cement pump schedule stores the planned information for cement fluid ratios, volumes, pump rates, and managed pressures of each cement stage and from a relational perspective is a “child” of the cement stage.

10.1.3.2 Cement Tops

The cement top or “fluid location” contains the estimated and interpolated top and bottom depth of pumped cement. This data is used to help generate regulatory reports and determine if certain formations and other water tables are properly isolated.

10.1.4 Cement Job Report

After a cementing operation is complete, a cement job report is generated, which is used to compare actual vs planned cementing information. Other data captured in this object includes the thickness of cement plugs and pipe movement (pipe reciprocation) while pumping.

10.1.4.1 Cement Stage Report

The cement stage report contains data regarding the actual volumes, rates, pressures and other related data during the cementing operations, for a particular stage. Using the cement stage report, planned vs. actual data is compared and clients are able to conclude the success of a cement operation (PA, squeeze, etc.).

10.2 Cement Job Evaluation

The cement job evaluation is a top-level object that captures the overall success of a cementing operation. This includes data pertaining to the cement bond quality measured from cement surveys, bond

logs, temperature logs, tools that were used for measurements, as well as data from pressure tests that measures the cements ability to isolate and contain formation fluids and pressures.

11 Other Report Data Objects

This chapter contains information for these report objects:

- Drill
- Ops (Section 11.2)
- Fluids (Section 11.3)

11.1 Drill Report Data Object

During drilling operations, operators are required to report information daily to their partners and government agencies. The drill report (DrillReport) data object has been designed to capture required and some supplemental information for daily drilling reports.

NOTE: For daily a daily drilling report format designed for service companies to report to operators, see the Section 11.2 (page 101) (OpsReport).

11.1.1 Business Purpose

The drill report object captures data for:

- All kinds of wells such as production, injection, appraisal, observation, etc.
- Main aspects of each activity performed on the rig daily, based on an extensive list of standard activity codes
  - Information related to activities, such as: instruments or tools used, their measurements, test reports, and incident reports.

The fast and efficient of data provided in the drill report has these benefits:

- Helps rig site users communicate more efficiently with onshore asset and management teams and partners to promote safer, more efficient operations.
- Facilitates compliance to daily government reporting requirements.
- Improves capability for faster analytics, improved decision making, and performance optimization or enhancement.
- Helps improve asset performance through better understanding of key areas such as key performance indicators (KPIs), rig downtime, and rig utilization.
- Promotes safety through tracking human factors such as crew performance in executing required safety programs.

11.1.2 Data Model

Figure 11-1 is a UML diagram of the drill report. Key design features include:

- The top-level object is small set of attributes that identifies the report, for example: report interval (start and end time covered by the report); the well datum and the bit record.
- Required reference to the wellbore where the activity is occurring.
- Optional reference to the fluids report (see Section 11.311.3 (page 102)).
- Optional references to multiple (0..*) drill activities, to capture each activity that occurred during the reporting period.
- Optional references to status info.
- Optional references to other WITSML data objects that were created during the reporting period, for example:
  - Trajectory stations
  - Logs

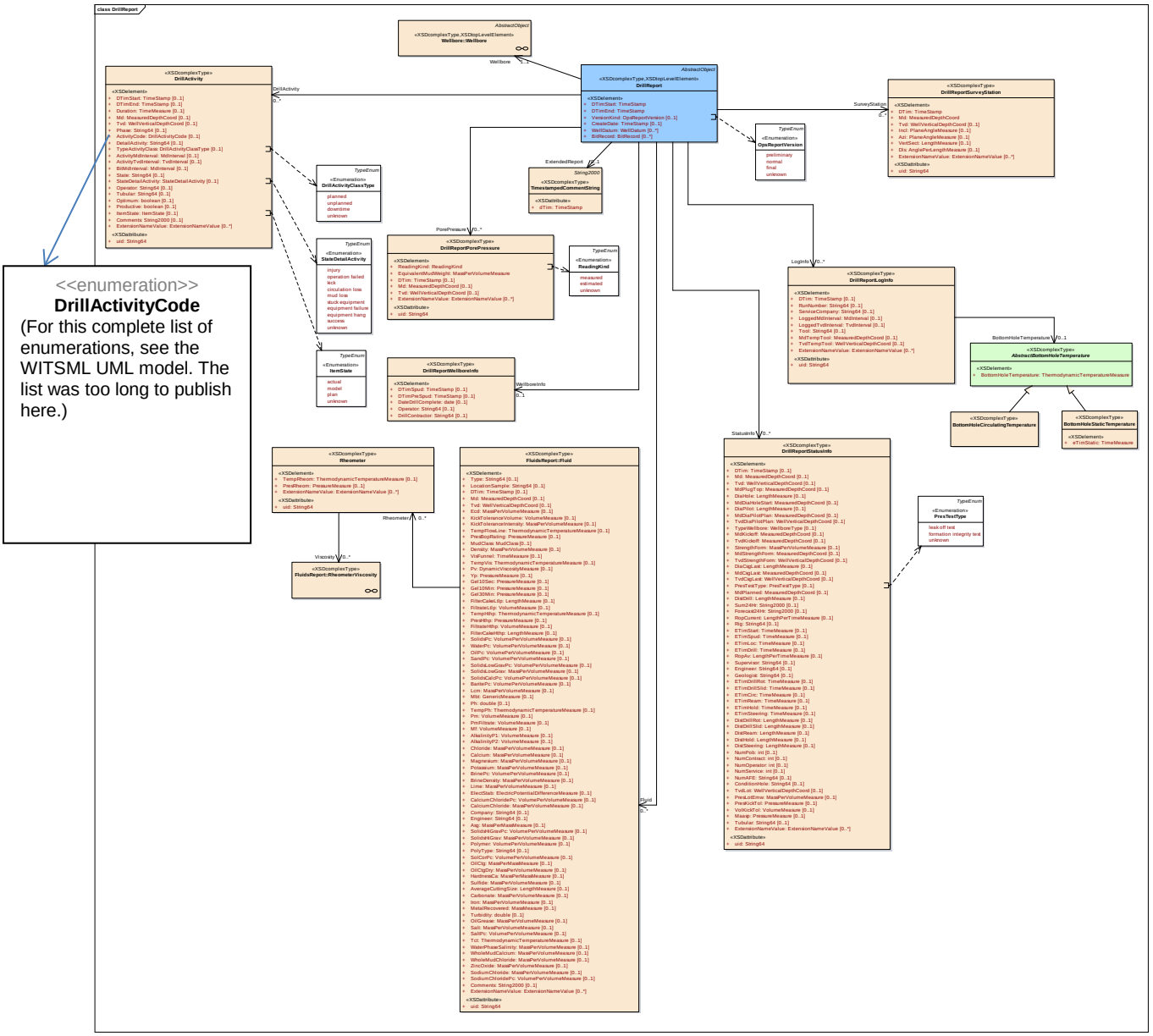

Figure 11–1. UML model of the drill report data object. NOTE: The diagram would not fit on this page, so the image has been slightly adapted (see the blue arrow). For the complete diagram, see the XMI file in the WITSML download.

11.2 Ops Report Data Object

The operations report (OpsReport) data object (**Figure 11–2**) captures daily drilling and related information for service companies to report to operators. The report includes information about the operations performed, major equipment status/performance, rig response, HSE events, and weather.

For daily drilling report designed for operator to share with partners and regulators, see Section 11.1 (page 100).

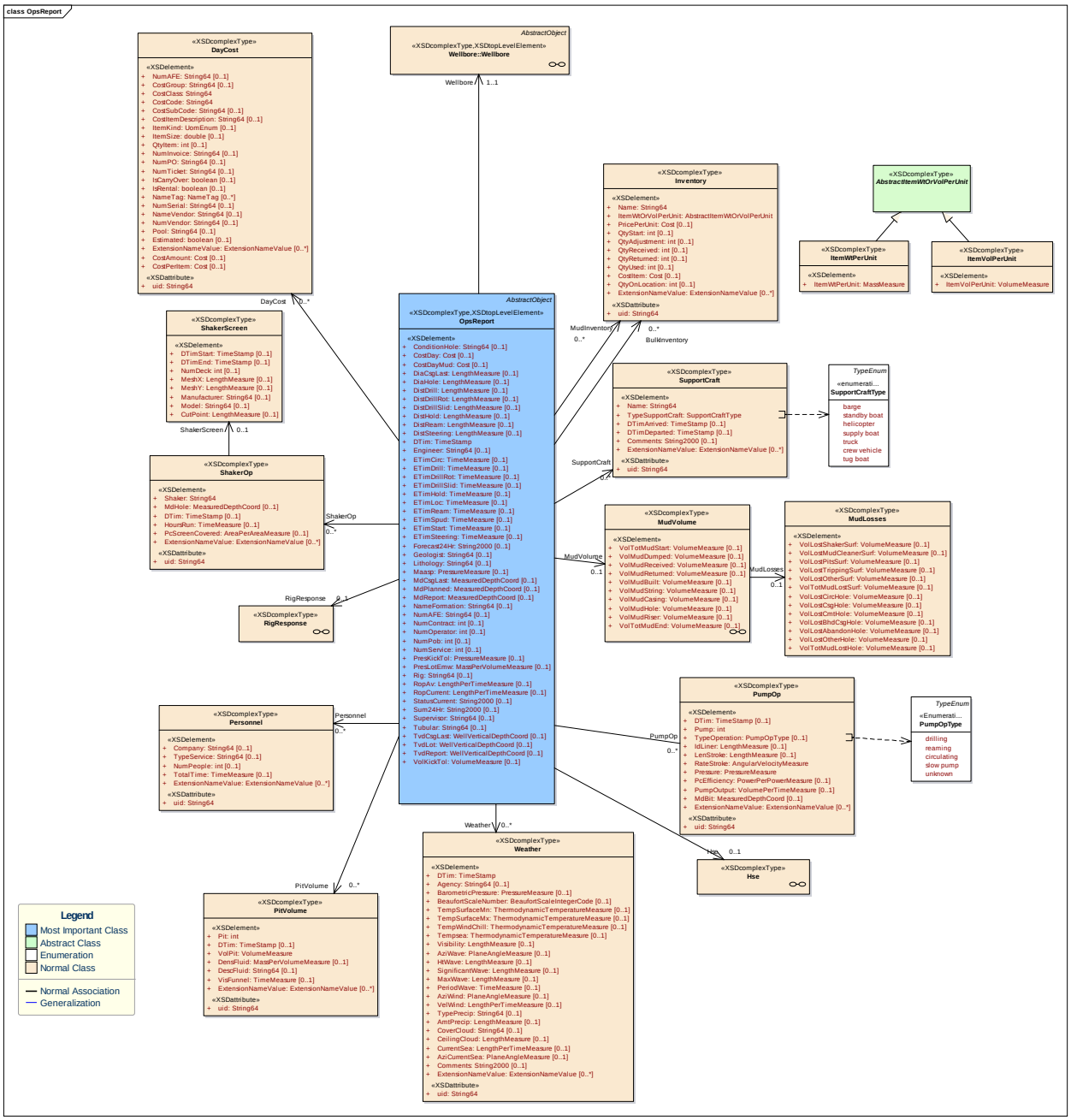

Figure 11–2. UML diagram of the ops report data object.

11.3 Fluids Report

The composition and properties of fluids used in well operations (e.g., drilling fluids, completion fluids, etc.) are crucial to safe and efficient operations. As such, fluids used are analyzed, and composition and property measurements are recorded throughout operations—often continuously for certain fluids—and the additives or dilution compounds added are also monitored, measured, and recorded. The monitoring and data capture is done by the mud engineering service company at the wellsite, and then data is transferred to an operator's systems, for record keeping and future analysis.

The fluid report (FluidReport) data object is designed to capture and transmit the measured properties of well fluids for a given operation.

11.3.1 Data Model

Figure 11–3 shows the UML model for the FluidReport data object. Remember, all Energistics data object schemas (XSD files) are produced from the UML model; so the UML model reflects the schema organization. The basic organization of the object is:

- Fluids Report: Contains basic report information such as the report number, time stamp, and associated wellbore depth data.
  - Fluid: Captures specific information about the fluid composition and properties; a report can have multiple fluids.
    - Rheometer measurements: Optionally, the fluid object can reference rheometer measurements, which are captured in the drill report object.

The fluids report object must reference a “parent” wellbore object.

11.3.1.1 Fluids

Well fluids are used to provide hydrostatic pressure, stabilize the hole (drilling mud), suspend and transport debris and cuttings to the surface, and cool the bit. Many factors and properties impact the quality and performance of fluids, so those attributes are captured in the fluids object. For example, water quality impacts the mixing of water-based drilling muds and the effectiveness of additives, such as potassium chloride to inhibit clay swelling. Well fluid temperatures can also greatly affect the properties of drilling fluids with solids suspension and the overall life and performance of additives. The properties and quality of completion fluids can impact the overall productivity of a well further so capturing those fluid properties can be crucial when troubleshooting future production problems.

The fluid object has been designed to be used with all types of well-related fluids (drilling, completions, etc.). As such, not all elements apply to all types of fluids. Most elements are optional; only provide data for elements applicable to a particular fluid type.

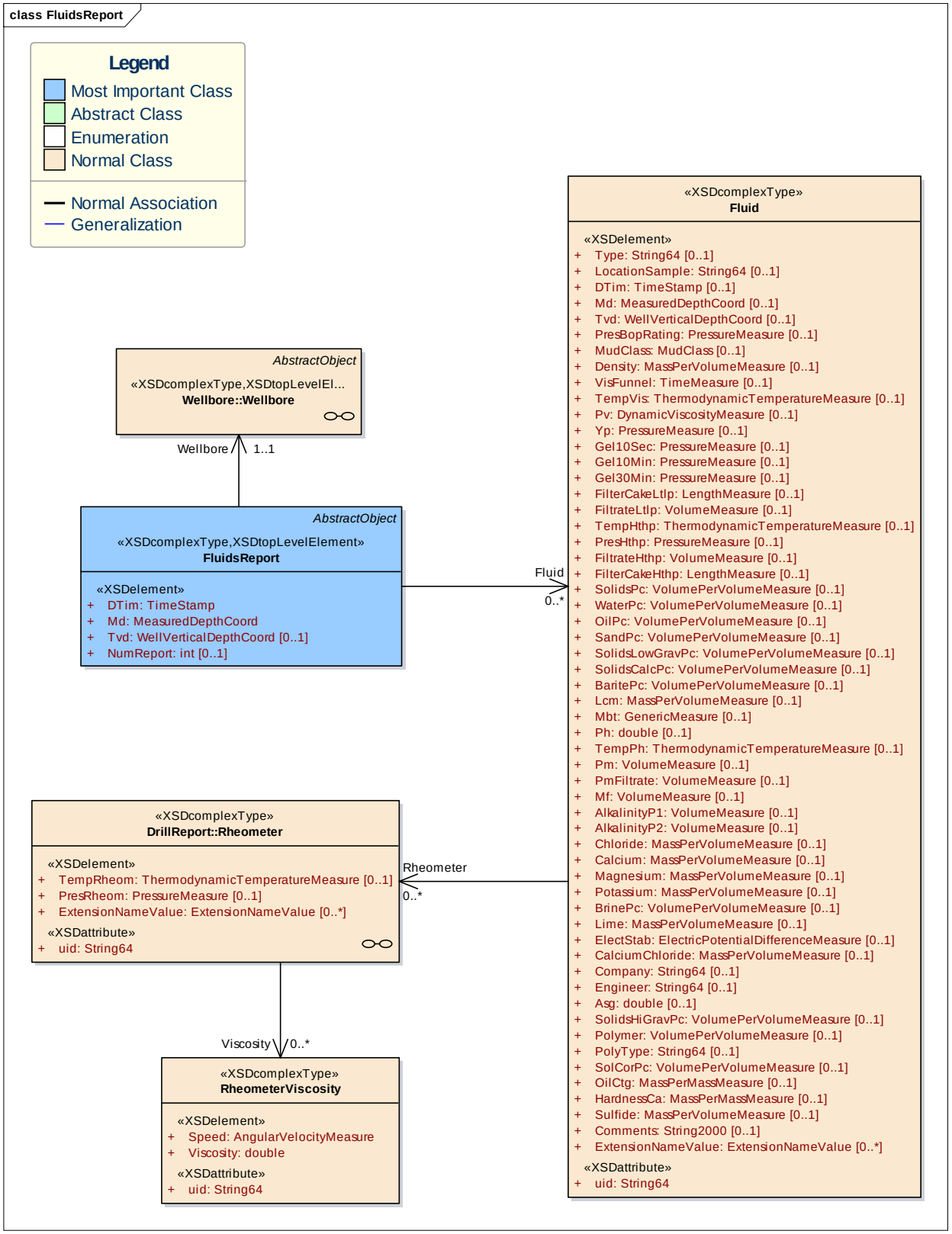

Figure 11–3. UML model of the fluids report object.