

RESQML Technical Usage Guide

For RESQML v2.1

RESQML Overview	The RESQML standard facilitates data exchange among the many software applications used along the E&P subsurface workflow, which helps promote interoperability and data integrity among these applications and improve workflow efficiency and flexibility.
Version of Standard	2.1
Abstract	Detailed explanations of key concepts and how RESQML works for software/IT professionals.
Prepared by	Energistics and the RESQML SIG
Date published	11 November 2016
Document type	Usage guide
Keywords:	standards, energy, data, information, process, reservoir model, shared earth model



Document Information

DOCUMENT VERSION	1.0
DATE	11 November 2016
Language	US English

Usage, Intellectual Property Rights, and Copyright

This document was developed using the Energistics Standards Procedures. These procedures help implement Energistics' requirements for consensus building and openness. Questions concerning the meaning of the contents of this document or comments about the standards procedures may be sent to Energistics at info@energistics.org.

The material described in this document was developed by and is the intellectual property of Energistics. Energistics develops material for open, public use so that the material is accessible and can be of maximum value to everyone.

Use of the material in this document is governed by the Energistics Intellectual Property Policy document and the Product Licensing Agreement, both of which can be found on the Energistics website, <http://www.energistics.org/legal-policies>.

All Energistics published materials are freely available for public comment and use. Anyone may copy and share the materials but must always acknowledge Energistics as the source. No one may restrict use or dissemination of Energistics materials in any way.

Trademarks

Energistics®, WITSML™, PRODML™, RESQML™, Upstream Standards. Bottom Line Results.®, The Energy Standards Resource Centre™ and their logos are trademarks or registered trademarks of Energistics in the United States. Access, receipt, and/or use of these documents and all Energistics materials are generally available to the public and are specifically governed by the Energistics Product Licensing Agreement (<http://www.energistics.org/product-license-agreement>).

Other company, product, or service names may be trademarks or service marks of others.

Amendment History			
Std Version/ Doc Version	Date	Comment	By
2.1 / 1.0	11 Nov 2016	V2.1 of RESQML with v2.1 of Energistics common. For the list of release notes, see Appendix A (page 245).	Energistics and RESQML SIG

Table of Contents

Table of Contents.....	4
1 Introduction.....	9
1.1 What is RESQML?.....	9
1.1.1 What Business Challenges Does RESQML Address?	9
1.1.2 What's in a RESQML Data Exchange?.....	10
1.1.3 How does RESQML Work? A Simple Example	11
1.2 Audience, Purpose, and Scope	11
1.3 Resource Set: What's Included When You Download RESQML?	12
1.3.1 RESQML Resources	12
1.3.2 Energistics Common Technical Architecture Resources	13
1.3.3 Documentation Updates	14
1.4 What's New in RESQML v2.1?	16
1.4.1 Features of v2.0+	16
1.4.2 Moving from RESQML v2.0.1 to v2.1	17
1.5 Further Development, Refinement of Use, and Experimentation.....	17
2 Technical Architecture Overview.....	18
2.1 CTA: Main Components and What They Do	18
2.2 RESQML Data Model	20
2.2.1 ResqmlCommon Data Objects.....	22
3 RESQML: Key Concepts and Organization.....	24
3.1 Basic Definitions and Key Concepts	24
3.1.1 RESQML Data Objects	24
3.1.2 Using UUIDs and EIP Metadata for Traceability to Help Manage Different Versions of a Data Object	25
3.1.3 "Knowledge Hierarchy": Features, Interpretations, Representations, and Properties	29
3.1.4 Topology and Geometry	30
3.1.5 Specifying Relationships in RESQML	30
3.1.6 Jagged Array Object.....	31
3.1.7 Activity Model	32
3.1.8 Energistics Package	32
3.2 High-Level RESQML Model Organization	35
3.2.1 Feature and Interpretations	36
3.2.2 Representations	36
3.2.3 Geometry	36
3.2.4 Properties	37
3.2.5 Earth Model	37
3.2.6 Grids	37
3.2.7 Streamlines.....	38
3.2.8 Wells.....	38
3.2.9 Seismic	38
4 Knowledge Hierarchy: Features, Interpretations, Representations, and Properties.....	39
4.1 Features, Interpretations, Representations, and Properties	40
4.2 Topology and Geometry	42
4.3 How These Concepts are Used in the Knowledge Hierarchy.....	42
4.3.1 Feature Level.....	42
4.3.2 Interpretation Level.....	43

4.3.3	Representation Level	44
4.3.4	Property Level	44
4.4	Specifying Relationships in RESQML.....	45
4.4.1	Relationship Mechanisms: Data Object Reference.....	45
4.4.2	Example: "FIRP" Relationships	46
4.5	References.....	47
5	Representations (Shared Concepts)	48
5.1	Representations Overview.....	48
5.2	Indexing	49
5.2.1	Indexable Elements	49
5.2.2	Multi-Dimensional Arrays and HDF5 Data Storage.....	50
5.3	Patches	51
5.3.1	When to Use Patches.....	51
5.3.2	Characteristics of Patches.....	51
5.4	Subrepresentations.....	52
5.5	Representation Identities	53
5.6	Redefined Geometry Representations	53
5.7	Representation Set Representation.....	54
6	Geometry	56
6.1	Overview	56
6.2	Points, Lines, and Planes	57
6.3	Parametric Points and Lines.....	59
6.3.1	Cubic Splines.....	60
6.3.2	Tangential Cubic Splines.....	62
6.3.3	Natural Cubic Splines	62
6.3.4	Z Linear Cubic Splines	63
6.3.5	Minimum-Curvature Splines	64
6.3.6	Example: Tangential and Natural Cubic Splines.....	65
6.4	References.....	66
7	Properties	67
7.1	Overview of How it Works.....	67
7.2	Property	68
7.2.1	Property Kinds	69
7.2.2	Extending the Property Kinds List	70
7.2.3	Facets	70
7.2.4	Mapping Simulator Key Words to Property Aliases	71
7.2.5	Units of Measure	71
7.3	Geometrical Information Stored as a Property	71
7.4	Property Set	71
7.5	Time Information: Time Step and Time Series	72
7.5.1	Time Index, Time Series and Geologic Time	72
7.6	Property Series	75
7.6.1	Use Cases	75
7.6.2	Description.....	76
7.6.3	Indexing	77
8	Earth Model	78
8.1	What is a RESQML Earth Model and How Can it Be Used?.....	78
8.1.1	Model Feature: Definition and Use.....	78
8.1.2	Earth Model Interpretation: Definition and Use	79
8.2	Features and Components of an Earth Model Interpretation	80
8.2.1	Features	82

8.2.2 Structural and Stratigraphic Organization Interpretations	84
8.3 Examples: How to Use RESQML in the Life Cycle of an Earth Model.....	103
8.3.1 Example 1: Creating an Earth Model	103
8.3.2 Example 2: Possible Next Steps: How to Reuse/Update a Complete Earth Model for Static Property Modeling and Flow Simulation	113
8.3.3 Example 3: Strategy to Re-Engineer an “Ancient” 3D Grid.....	115
8.4 References.....	116
9 Structural & Stratigraphic Representations.....	117
9.1 Introduction	117
9.2 Individual Representations	118
9.2.1 Patch Management	119
9.2.2 Points.....	120
9.2.3 Polylines	120
9.2.4 Surfaces	120
9.2.5 Plane Set Representation	125
9.3 Organization Representations (Representation Sets and Frameworks).....	127
9.3.1 Representation Set Representation	128
9.3.2 Framework Representations	128
10 Grids	136
10.1 Grids: Quick Start.....	136
10.1.1 Topology.....	136
10.1.2 Geometry.....	136
10.1.3 Feature Interpretations	137
10.1.4 Properties	137
10.1.5 Advice on Grid Export	137
10.1.6 Advice on Grid Import.....	137
10.2 Example of Grids that can be Transferred with RESQML.....	138
10.2.1 Local Grids	139
10.3 Grid Topology	141
10.3.1 Fundamental Grid Classes	141
10.3.2 Truncated Cell Grids.....	142
10.3.3 General Purpose Grid	142
10.4 Grid Cell Geometry	143
10.4.1 Block-Centered Cells.....	144
10.4.2 Column-Layer Cells	144
10.4.3 Unstructured Cells	144
10.5 Grid Element Indexing	144
10.5.1 Indexes that Cannot be Uniquely Determined	146
10.5.2 Multi-Dimensional Arrays	146
10.5.3 Elements per Object	147
10.6 Faulted Grids	147
10.7 Additional Grid Geometry and Topology.....	148
10.7.1 Higher Order Grid Geometry and Properties	149
10.7.2 Finite Element Subnodes	149
10.8 Local, Child and Parent Grids.....	151
10.8.1 Local Grids	151
10.8.2 Parentage and Re-Gridding	151
10.9 Grid Feature-Interpretations	154
10.10 Unstructured Grids.....	155
10.10.1 Unstructured Geometry	155
10.10.2 Unstructured Grid Representation	158
10.10.3 Unstructured Grid Indexable Elements.....	159
10.11 Column-Layer Grid Geometry.....	161

10.11.1 Pillars and Columns.....	161
10.11.2 Coordinate Lines.....	161
10.11.3 Additional Information Included in this Construction.....	162
10.11.4 Supported Extensions.....	163
10.12 IJK Grids	164
10.12.1 IJK Grid Geometry	164
10.12.2 IJK Grid Representation	165
10.12.3 IJK Grid Indexable Elements	167
10.12.4 IJK Grid Origin	169
10.12.5 IJK Cartesian and Radial Cell Interpolation.....	169
10.13 Unstructured Column-Layer Grids	170
10.13.1 Unstructured Column-Layer Grid Geometry.....	170
10.13.2 Unstructured Column-Layer Grid Representation	171
10.13.3 Unstructured Column-Layer Grid Indexable Elements.....	172
10.14 Truncated Column-Layer Grid Representation.....	175
10.15 General Purpose (GP) Grid Representation.....	176
10.16 Grid Connection Set Representation	176
10.17 Grid Examples	179
10.17.1 Eclipse GRDECL File as an IJK Grid	179
10.17.2 4x3x2 Faulted IJK Grid	184
10.17.3 Fault Representations on the 4x3x2 Faulted IJK Grid.....	186
10.17.4 Association between IJK Grid and Horizons	189
10.17.5 Association between IJK Grid and Stratigraphic Column.....	192
10.17.6 Small IJK Grid with Nested Local Grid Refinement.....	194
10.17.7 IJK Grid with Proportional Layering	200
10.17.8 Higher Order Finite Element Prism24 Unstructured Column-layer Grid.....	202
10.17.9 Block-Centered General Purpose Grid.....	205
10.17.10 Unstructured Grid with Discontinuous Properties.....	206
10.18 References.....	212
11 Streamlines	213
11.1 Streamlines and the RESQML Knowledge Hierarchy	214
11.2 Streamlines: Feature.....	215
11.2.1 Flux	215
11.3 Streamlines: Interpretation.....	215
11.4 Streamlines: Representation	216
11.4.1 Lines	216
11.4.2 Streamline Sources and Sinks	217
11.4.3 Streamline Geometry.....	217
11.4.4 Polyline Interval Grid Cells	217
11.5 References.....	218
12 Wells	219
12.1 Special Requirements for Using WITSML 1.4.1 Data Objects with RESQML (in an EPC file).....	219
12.2 Wellbore Organization Overview	219
12.2.1 Wellbore Feature	220
12.2.2 Wellbore Interpretation	220
12.2.3 Wellbore Location.....	221
12.3 Well Logs	222
12.4 Markers and Links to Geological Boundaries	223
12.5 Interpretations and Links to Organizations	224
12.6 Blocked Wells	226
12.7 Example: Two Vertical Wells	227
12.7.1 Class Diagram	227

12.7.2 Instance Diagram	232
13 Seismic	235
13.1 Seismic Lattice	236
13.2 Seismic Lattice Representation	237
13.3 Seismic Line	238
13.4 Seismic Survey as a Part of the Geometry of the Business Object	239
13.4.1 Example.....	241
13.5 Seismic Survey as Extra Information on the Geometry of a Representation	243
13.6 Seismic Survey Representations on a Grid	243
Appendix A. Release Notes.....	245
A.1 Version 2.1	245
Appendix B. HDF5 Basics	247
B.1 Overview of RESQML HDF5	247
B.2 Adding HDF5 to your Code.....	247
B.3 HDF5 Handles	247
B.4 Reading from a RESQML HDF5 File	247
B.5 Writing Data to a RESQML HDF5 File.....	248

1 Introduction

1.1 What is RESQML?

RESQML is an XML- and HDF5-based data-exchange standard that facilitates reliable, automated exchange of data among software packages used in subsurface workflows. RESQML consists of a set of XML schemas (XSD files) and other standards-based technology, which developers implement into software packages. Software that has implemented RESQML can read and write the standard format.

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

1.1.1 What Business Challenges Does RESQML Address?

1.1.1.1 Subsurface Workflow Challenges

The exploration and production (E&P) subsurface workflow is lengthy, iterative, and complex. It involves many people from different disciplines, sometimes different companies, and use of many different software packages for complex analysis, interpretation, modeling, and simulation.

This multi-discipline, multi-company, multi-software environment is iterative and requires users to move data back and forth between different software packages. Many of these packages use different data formats—often proprietary and incompatible.

This inherently complex process and inability to easily exchange data means E&P companies and their people face challenges that include: knowledge loss, rigid workflows, difficulty characterizing and sharing uncertainty, data loss, and productivity loss.

1.1.1.2 How RESQML Helps Address These Challenges

RESQML-compliant software can read and write this standard, common format, eliminating data incompatibility and the need for reformatting. **Figure 1-1** below is a high-level overview of how RESQML works and the workflows it supports. The version 2.0+ capabilities (see Section 1.5 (page 13)) help RESQML deliver these benefits:

- Delivers a "knowledge hierarchy" to organize data and transform it into knowledge.
- Increases workflow flexibility, for example, with partial model transfers that allow you to update/transfer only data that has changed.
- Supports traceability, with universally unique identifiers for each top-level data object and key metadata for data sources, updates, dates of change, etc.
- Supports uncertainty management through increased ability to run more scenarios and realizations and reliably update models.
- Defines a rich set of subsurface data objects and enables transfer of detailed models and a variety of model types.
- Improves efficiency for both petro-technical and IT professionals.

For more information on the challenges of subsurface workflows, how RESQML helps address them, and some key supported workflow use cases, see the *RESQML Business Overview and Use Case Guide*. (For a link to this document, see Section 1.4 (page 12)).

RESQML Supported Workflows

How RESQML works: Commercial and in-house software packages that implement the RESQML standard can read and write the common format.

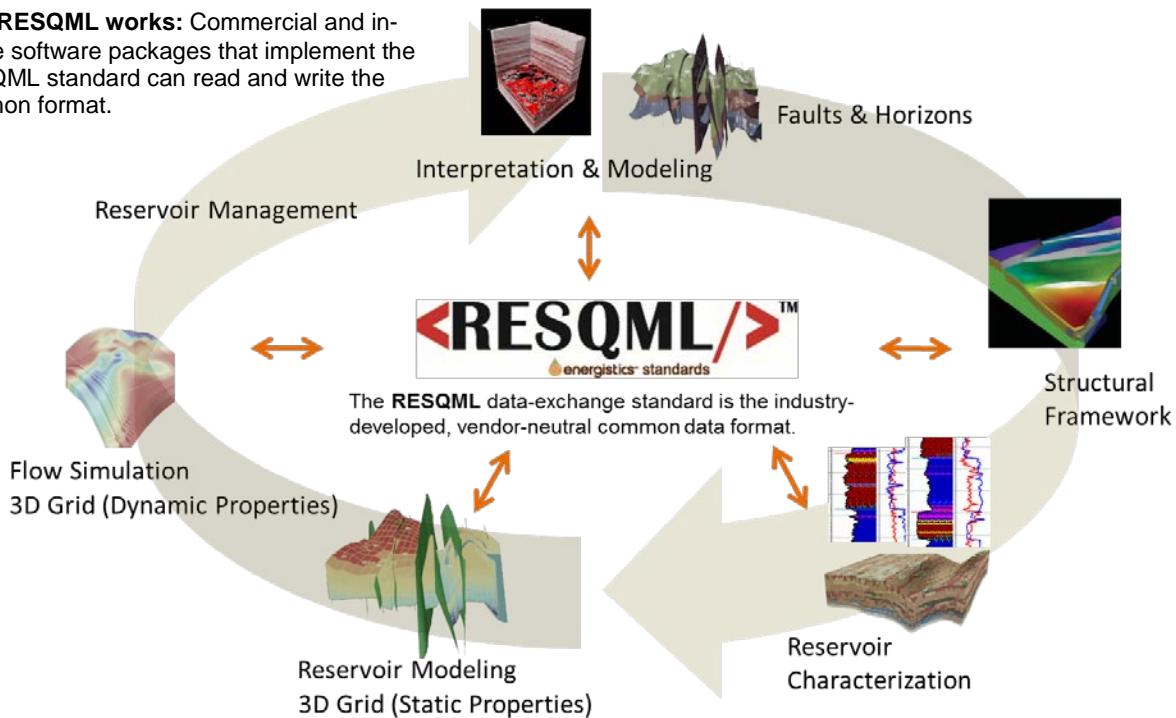


Figure 1-1. Implementing RESQML in software used in the E&P subsurface workflows streamlines data flow among the many different software packages used. The latest version supports more workflows and more flexible workflows. New capabilities provide a rich set of data objects, a well-defined knowledge hierarchy throughout the model, methods for specifying and transferring relationships among data objects, and the ability to group all of the information into a single, structured package with associated files for floating point array data.

1.1.2 What's in a RESQML Data Exchange?

When software applications exchange data using RESQML, that data exchange is composed of:

- One EPC file (a container file technology) that groups together in a single file all the XML data objects for the exchange. The Energistics Packaging Conventions (EPC) file is essentially a zip file.
- One (or possibly more) related HDF5 files that contain the floating point data for the XML data objects. HDF version 5 is a standard designed for the flexible and efficient I/O for high volume and complex data—particularly when compared to XML.

1.1.3 How does RESQML Work? A Simple Example

Figure 1-2 is an example of a very simple subsurface workflow using RESQML-enabled software. When users need to move data to the next software application in their workflow, they choose to write (export) data to the RESQML format. In this example, User A using Software A writes the data to the RESQML format, which is transported in an EPC file (sometimes called an Energistics package) and 1 (possibly more) related HDF5 file, which contains the floating point data.

That next software application may be a tool used by another discipline in the workflow or by a partner company in a joint venture. If that software application is RESQML-enabled, it can read (import) the EPC and HDF5 files containing the RESQML data and process the data in its native environment.

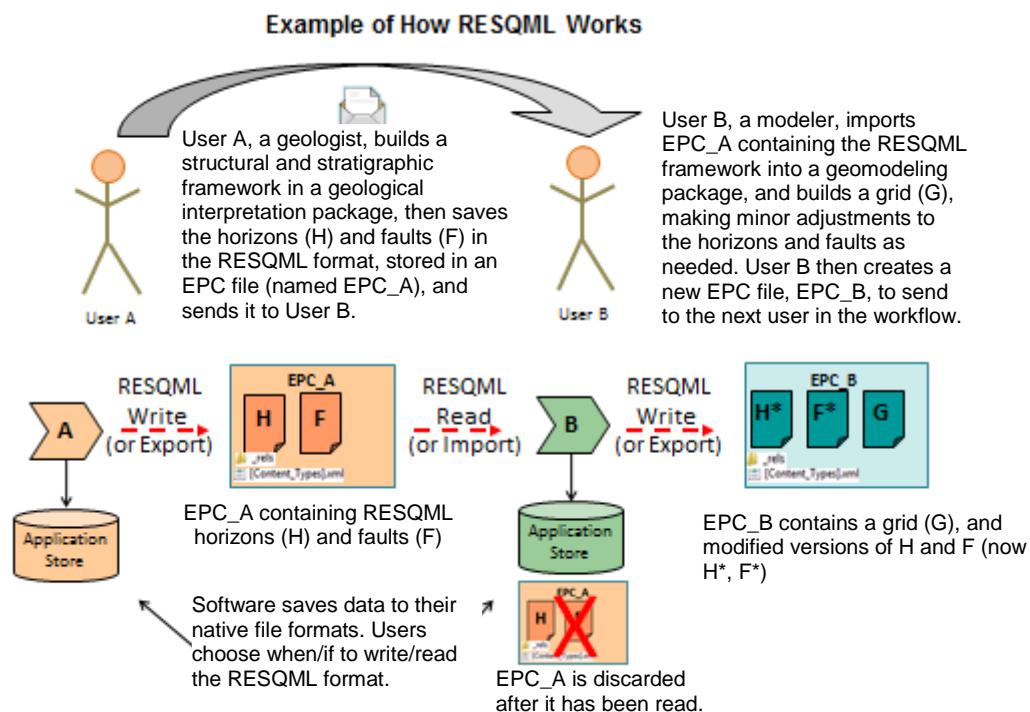


Figure 1-2. A user writes (exports) a file to the RESQML format, which is stored in an EPC file and may be read (imported) by other RESQML-enabled software.

1.2 Audience, Purpose, and Scope

This guide is intended for Information technology (IT) professionals—programmers, developers, architects and others—who are implementing RESQML into a software package.

This guide introduces and explains key RESQML capabilities and concepts—what it can do and how it works.

ASSUMPTION: This document assumes that the audience has a good understanding of XML and a general understanding of E&P subsurface workflows.

1.3 Resource Set: What's Included When You Download RESQML?

RESQML is a set of XML schemas (XSD files) and other technologies freely available to download and use from the Energistics website. When you download RESQML, you get a zip file structured as shown in **Figure 1-3** with the content listed in the tables below in these 2 main groups:

- **RESQML-specific** (Section 1.4.1, below). The schemas and documentation specific to RESQML. Note in Figure 1-3 (left) (red square):
 - ResqmlAllObjects.xsd is schema that includes all other schemas in shown.
 - The ResqmlCommon schema includes objects shared by other RESQML data objects.
- **Energistics Common Technical Architecture** (Section 1.3.2 (page 13)). (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.

To download the latest version of RESQML, go to the Energistics website at:
<http://www.energistics.org/reservoir/resqml-standards/current-standards>.

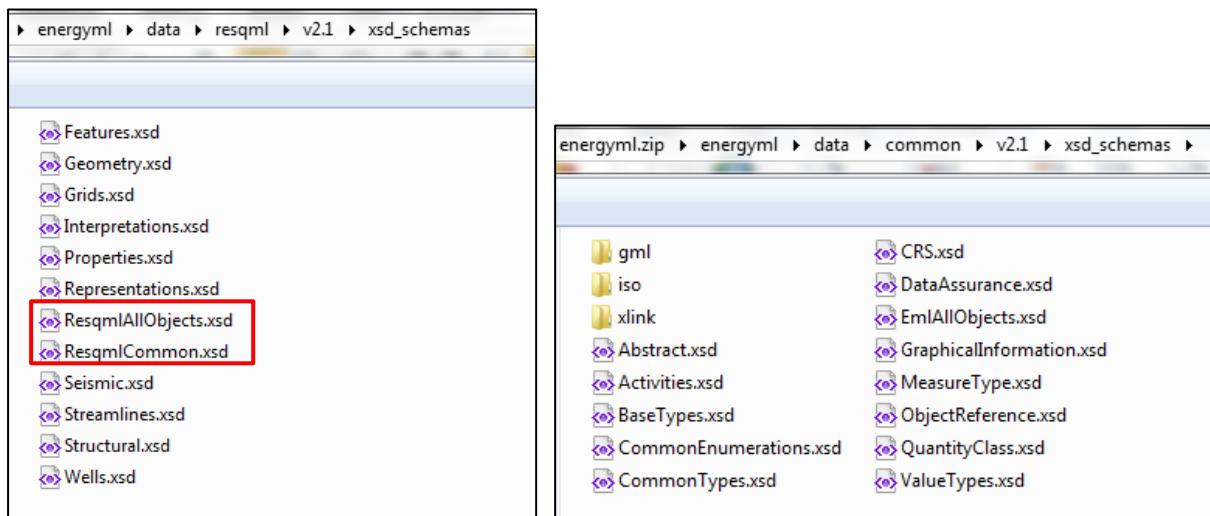


Figure 1-3. You download the RESQML standard as zip file from the Energistics website. It contains the artefacts show in the image, which are described in the tables below.

1.3.1 RESQML Resources

	Resource/Document	Description
1.	RESQML:XSD files	The RESQML package includes a readme file that details the contents of the download package.
2.	RESQML UML Data Model (XMI file)	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.	<i>RESQML Business Overview and Use Case Guide</i>	An introduction to RESQML for domain/petro-technical professionals. Provides: <ul style="list-style-type: none"> • an overview of the business value and domain challenge that RESQML helps to solve. • a representative list of supported use cases.

	Resource/Document	Description
4.	<i>RESQML Technical Usage Guide</i> (this document) START HERE	<p>Detailed explanation of RESQML key concepts and design intended for software/IT professionals.</p> <p>See also in download: Separate spreadsheet with spline equations related to Chapter6 (20131223 RESQML Cubic Splines.xlsx)</p>
5.	<i>RESQML Technical Reference Guide</i>	<p>Lists and defines all packages, data objects, elements, including related business rules. This document also identifies relationships between/among data objects and highlights significant information about relationships (if any). Generated from the RESQML UML model.</p>
6.	Examples	<p>The download includes an example Energistics package containing a basic RESQML model.</p> <p>Additionally, new examples will be added. Check the link above for updates.</p>

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	<p>Schemas (XSD files) for Energistics CTA data objects, which are contained in the folder named <i>common</i> and are further described in this document.</p> <p>NOTE: The <i>common</i> folder is included as part of the package when you download any of the Energistics domain standards.</p>
2.	UML Data Model (XMI file)	<p>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.</p> <p>NOTE: Objects in the CTA are in the folder named <i>common</i> and included in the UML for each Energistics domain standard.</p> <p>Energistics saves the UML model as an XMI file, a format that can be imported by any UML data modeling tool.</p>
3.	<i>Energistics Common Technical Architecture Overview Guide</i> (For CTA, start here.)	Provides an overview of the components that comprise the Energistics CTA.
4.	<i>Energistics common Technical Reference Guide</i>	<p>Lists and defines packages, data objects, elements, and relationships for the objects in the CTA <i>common</i> folder. Produced from the common UML package from which Energistics <i>common</i> XSDs are produced.</p>
5.	<i>Energistics Packaging Conventions (EPC) Specification</i>	Specifies the Energistics Packaging Conventions (EPC), which is the set of practices to store multiple files as a single entity for data transfer; this single entity is referred to as an "EPC file" (or sometimes, an "Energistics package"). EPC is an implementation of the Open Packaging Conventions (OPC), a container-file technology standard.
6.	<i>Energy Industry Profile of ISO 19115-1 (EIP)</i>	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

	Resource/Document	Description
		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.	<i>Energistics Identifier Spec</i>	Describes the syntax and semantics of data object identifiers used in Energistics data exchange standards, which include UUIDs and URIs, and object reference.
8.	<i>Energistics Unit of Measure Standard</i> (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 <i>common</i> .
9.	<i>Energistics Transfer Protocol (ETP)</i> (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.

1.3.3.1 Documentation Conventions

Documentation for RESQML observes the conventions listed in the following table.

	Document/Resource	Description
	Mandatory Behavior	Mandatory behaviors are specified as business rules as shown in the example below. For more information, see Section 1.3.3.2 below. BUSINESS RULE: Array length is the number of cells in the grid or the blocked well.
	Document Hyperlinks: Internal	Though no special text-formatting convention is used: All section, page and figure numbers in this and all RESQML and Energistics documents are hyperlinks. The table of contents is also hyperlinked.

1.3.3.2 Business Rules

Business rules are currently designated and documented in the UML model (in the EA software), the *RESQML Technical Reference Guide*, and, as appropriate, in other documentation.

Examples of business rules in the documentation include requirements for matching counts, for example, for array lengths to match cell counts for grids and blocked wells; or for other requirements, such as use

of the *Energistics Unit of Measure Standard* (see Section 2.2.4 (page **Error! Bookmark not defined.**)). The following is an example (for a specific class in the UML model) of a business rule:

BUSINESS RULE: Array length is the number of cells in the grid or the blocked well.

No method for business rule validation exists for the current version of RESQML, but validation approaches are being explored and considered for future versions.

1.3.3.3 *Naming Conventions*

RESQML adheres to the *Energistics Naming Conventions Specification* (currently in draft form). Of specific interest to RESQML is the list of reserved data object names below.

Reserved data object names typically appear as part of the name of an object of this kind, when it improves the clarity of the schema, but never otherwise.

- Abstract
- Feature
- Interpretation
- Representation
- Geometry
- Property
- Patch
- Array
- Set
- Reference
- Choice

1.3.3.4 *RESQML Jargon*

The notion of "transfer" is referring to the exchange of data between any two RESQML-enabled software packages. Frequently we also refer to "exporting" or "writing" data to the RESQML format and "importing" or "reading" data.

In this context, the terms "software," "software package," "application" and others are used interchangeably—no special distinction is made between these terms. Software that writes or exports RESQML data is informally referred to as a "writer", and a package that is reading or importing RESQML data is referred to as the "reader."

1.4 What's New in RESQML v2.1?

RESQML v2.1 includes:

- Design improvements including:
 - Improved data storage for time series and multiple realizations
 - Simplified treatment of patch indexing
- Incorporation of previous “add-on” data objects (added to the v2.0 data model in v2.0.1) into the main design packages:
 - Property series and streamlines have now been integrated into RESQML.
 - Activity model is now integrated into Energistics *common*.

1.4.1 Features of v2.0+

RESQML v2.0 (published in September 2014) added significant capabilities to the previous commercial version (RESQML, v1.1). For new RESQML users, the v2.0 changes are summarized here. The overarching design objectives were to:

- Improve the richness of existing domain objects.
- Add new domain objects to the data model to support the more sophisticated capabilities of the software packages used in E&P workflows.
- Add relationships among data objects.
- Support flexible workflows, for example, by supporting partial model updates.

In support of these design objectives, RESQML includes these capabilities:

- A subsurface knowledge hierarchy based on “features, interpretation, representations and properties” that describes subsurface features, multiple interpretations of those features, multiple representations of those interpretations, and the properties that may be “attached to” the representations.
- A standard methodology for the representation of subsurface scenarios and uncertainty.
- Support for structural, stratigraphic, and reservoir models:
 - At the interpretation level: definition of the involved geologic features with their relationships, i.e., contacts and their characteristics, within the context of the model.
 - At the representation level: topology and geometry of the model, either surface or volume. Ability to define contacts using nodes of associated geologic features.
- Associations between structural and stratigraphic features with the 3D reservoir grid.
 - Achieved using subrepresentations of the grid, which are defined by selecting particular elements of the grid such as the layers, pillars, cells, cell faces, etc.
- Structured and unstructured grids or a mixture of both. New grid capabilities include:
 - Block-centered grids defined by cell-centered geometry and properties, and their associated connectivity, to support both simple grids and current unstructured simulators.
 - Polyhedral and perpendicular-bisector (PEBI) grids to provide the visualization and geometry required by current fluid flow and geomechanical reservoir simulators.
 - Corner-point grids as used by reservoir simulators.
 - Tetrahedral grids as used by geomechanical simulators.
 - Ability to use parametric lines for pillar grid definitions.
 - Simplified grid descriptions for simple grid types, e.g., rectilinear or tartan grids.
 - Higher order extensions for grid cell geometry and for grid properties.

- Use of wellbore information from WITSML, the Energistics exchange standard for well and log information, which includes:
 - Support for wellbores, trajectories, logs, markers, and blocked wellbores.
 - Use of HDF5 storage.
 - Reference to existing WITSML data.
- Support for multiple topologies, geometries, and properties for multiple time steps and multiple model realizations.
 - These capabilities are shared across the schema and are not restricted to specific data objects.
- Ability to transfer only those parts of a model that have changed (partial model transfer), which enables the workflow flexibility to test alternative scenarios and characterize the range of uncertainty in structural and stratigraphic frameworks, as well as reservoir rock and fluid properties.
- Ability to group a variety of data objects and files from multiple sources into a single "package" for transfer using the new Energistics Packaging Convention (EPC), which is an implementation of the Open Packaging Conventions (OPC) standard.
- Ability to combine data referenced to different coordinate reference systems (CRS), for example, a RESQML reservoir model and WITSML wellbores, trajectories, and markers. The RESQML package supports a single global CRS and one or more local CRSs.

1.4.2 Moving from RESQML v2.0.1 to v2.1

RESQML v2.1 is considered a "breaking change" from the previous version (v2.0.1), so no transforms for conversion are available.

1.5 Further Development, Refinement of Use, and Experimentation

Data-exchange standards can be a "chicken-and-egg" situation. For example, only one of the six grid classes now supported in RESQML has seen wide-spread historical use in the industry; that is corner-point grids.

Lack of shared data standards for 2.5D unstructured column-layer grids and for fully 3D unstructured grids have limited their use and accessibility in the industry, and have impeded cooperation between different technical domains, e.g., flow simulation and geomechanical calculations. Now that these newer grid types are defined in RESQML and their data can be exchanged, it should enable and foster experimentation and learning about how the data might be shared and used.

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 RESQML key concepts, see Chapter 3. Chapters 5 – 13 describe RESQML 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., ResqmlCommon) 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 (See **Figure 2-2** below) :

- 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 used with RESQML, 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 WITSMML 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 RESQML Data Model

The RESQML UML model is grouped into packages as shown in **Figure 2-2** and explained below. Each of the schema packages contains classes that represent the key data objects (see **Table 2** (page 35)). The UML model is also the source for the XSD files (schemas), which are generated by an automated process.

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

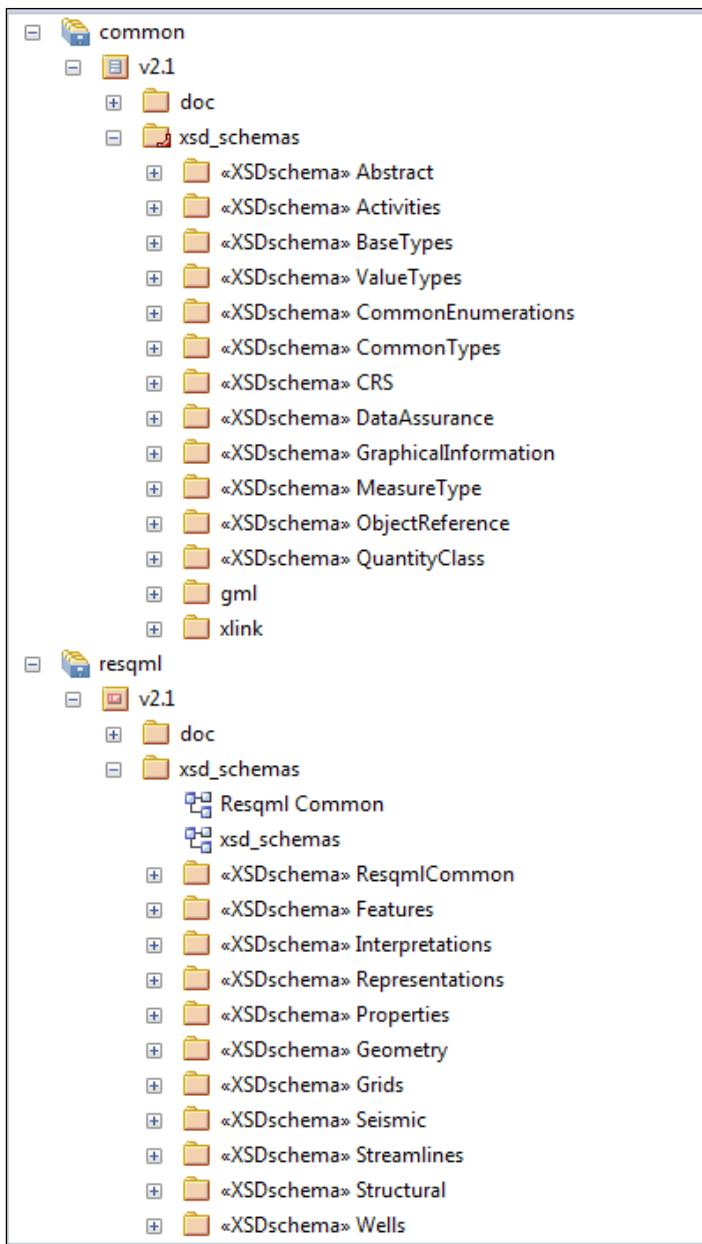


Figure 2-2. The RESQML UML model contains the groupings shown here and explained in the text below.

Main packages of the RESQML UML model are listed and described here.

common contains the classes that are used by all Energistics standards. For more information see references in 1.3.2 (page 13).

resqml contains a version package (for example, **v2.1**) which contains these main packages:

- The **doc** class (folder), which contains packages for:
 - **Package Diagrams** contains several diagram that show the main packages in RESQML and the data objects within each of those packages.
 - Use Case Diagrams. For future use.
 - **Instance Diagrams** includes instance diagrams and supporting class diagrams, which illustrate the attributes, inheritances and other relationships through examples.
- The **xsd_schemas** class (folder) contains the packages that are used to generate the XSD schemas. It contains these packages:
 - **ResqmlCommon**. 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. For more information, see Section 2.2.1 below.
 - **Features**. Feature data objects used throughout RESQML. (Feature data objects that serve only a single purpose are located in their appropriate packages, for example, Wells, Grids, and Seismic.)
 - **Interpretations**. Data objects that allow an interpreter to formalize his/her opinion of a feature throughout RESQML. Single-purpose interpretations—for example, wellbore interpretation—can be found in the appropriate package, in this example, the Wells package.
 - **Representations**. Representation data objects used throughout RESQML. (Single-purpose data objects are located in appropriate packages, for example, wells, grids, structural and seismic.)
 - **Properties**. Properties are attached to the indexable elements of a representation. Properties are designed to be transferred independently of a representation and are top-level data objects.
 - **Geometry**. Geometry is attached to the indexable elements of a representation. Geometries are not RESQML top-level data objects so cannot be transferred independently.
 - **Grids**. The features and representations used exclusively to define grids.
 - **Seismic**. Features and geometries used exclusively to define seismic data. Seismic data does not have interpretations; representations of seismic data are defined using data objects in the Representation package.
 - **Streamlines**. Feature and representation data objects used exclusively to define streamlines.
 - **Structural**. Representation data objects used exclusively for structural modeling. Features and interpretations for structural modeling are found in the Features and Interpretation packages.
 - **Wells**. The feature, interpretation, and representation data objects for defining wells.

2.2.1 ResqmlCommon Data Objects

Figure 2-3 shows the data objects in the ResqmlCommon folder. This section covers highlights of how these data objects are intended to work when RESQML is implemented in a software package. Classes and elements are defined in the UML model and the *RESQML Technical Reference Guide*.

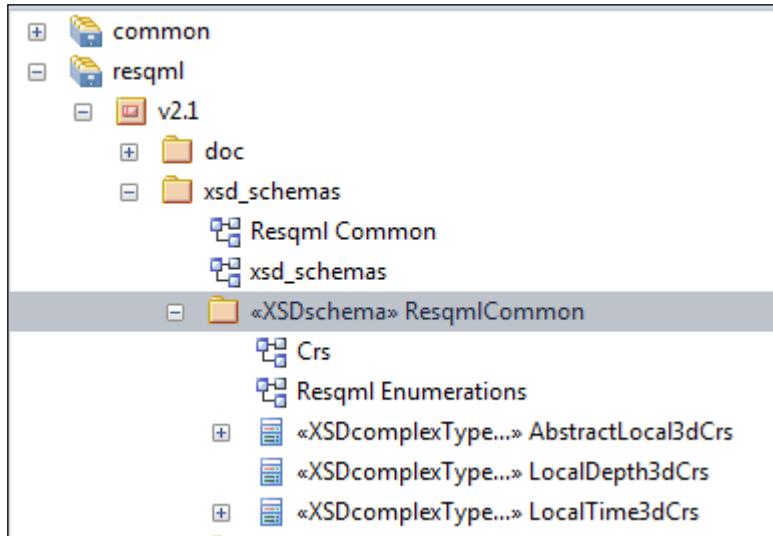


Figure 2-3. ResqmlCommon data objects.

2.2.1.1 Local Coordinate Reference Systems

To avoid numerical issues when computing relative distance between elements of a model, it is common practice in reservoir modeling to work within a local coordinate system. In RESQML, this local coordinate system is obtained by translation of the origin and areal rotation from projected and vertical global coordinate systems. It is also common to see elements in both the depth and the time domain.

RESQML contains one abstract class, `AbstractLocal3dCrs`, which references the projected and vertical CRS it is based on.

BUSINESS RULE: When the projected or vertical CRS is well known, it must have the same UOM as the UOM defined by the well-known CRS (projected or vertical).

Explanation: A well-known CRS already defines the UOM. When you indicate that you use a CRS EPSG code, e.g., 7500, if you go to the EPSG database, you find the constrained UOM. Some RESQML members did not want to depend on an EPSG database (or other external database), so RESQML copies the UOM of the well-known CRS into the RESQML CRS.

Because some projected and vertical CRSs can be anonymous, it also contains information such as units of measure, axis order, and orientation. Most of the time, this information is redundant with the actual projected and vertical CRS definition, but is required for an anonymous CRS and is also targeted toward applications that are not CRS aware. In any case, such information in the local CRS must match its projected and vertical CRS.

The two concrete specializations of the `AbstractLocal3dCrs` allow separation between the depth and time domains.

- `LocalDepth3dCrs` is referred by representations in the depth domain.
- `LocalTime3dCrs` is used by the representations in the time domain. `LocalTime3dCrs` also provides the unit of measure for time coordinates.

NOTE: When a CRS description is based on GML, it must include `gml:identifier` and the identifier must follow the UUID convention, so that a reader can quickly identify which representations are in the same CRS.

3 RESQML: Key Concepts and Organization

This chapter explains the "big picture"; it provides an introduction to and overview of RESQML basic definitions, key concepts, and organization.

For information related to the Energistics Common Technical Architecture, see the *CTA Overview Guide*.

3.1 Basic Definitions and Key Concepts

This section provides basic definitions and key concepts.

3.1.1 RESQML Data Objects

A RESQML data object is the atomic part of a RESQML transfer. It contains the semantics associated with the data model (or modeling data) in the context of RESQML. Key naming and organizing concepts of these semantics are explained in Section 3.1.3 (page 29) and Section 3.1.4 (page 30).

Data objects are exchanged mostly as XML files; associated explicit array data is transferred as one or more HDF5 files.

3.1.1.1 Data Object Identification and Traceability

- For identification, each RESQML data object must have a UUID. NOTE: For UUIDs, RESQML is case-insensitive.
- For traceability, RESQML uses EIP metadata, which includes information such as when a data object was created and updated and by what software.
- These stand-alone data objects are also informally referred to as *top-level* data objects or elements.

For information on how the UUID and EIP metadata can be used together to manage data object identity during extended data exchange during an interactive session, see Section 3.1.2 (page 25).

3.1.1.2 Relationships and References

A RESQML data object can be associated with many other data objects. For example, a CRS can be associated with many different geometries.

The use of multiple, independent data objects requires a way to specify relationship or associations between and among them, which is explained in Section 3.1.5 (page 30).

3.1.1.3 Packaging Together Data Objects

The use of multiple data objects also requires a way to group them together as a single package, which makes it easier for software and the people using it to understand the "whole" and all of its parts. For information on how RESQML packages data objects (and related parts), see Section 3.1.8 (page 32).

3.1.1.4 "Content" of a Data Object and Partial Model Transfer

A partial model transfer refers to the ability to transfer only the data in a model that has changed, instead of the entire model. However, to do this exchange correctly, you still need the context of the data objects that have not changed, so they must be included in the package (data transfer).

In support of partial model transfers, a data object, with only its mandatory identification data, may be transferred. Any other data contained in the data object, may or may not be included, depending on the requirements of the two applications exchanging data (for example, all the data for data object may have been transferred previously or may be transferred in the future).

For more information on partial model transfer, see the *RESQML Business Overview and Use Case Guide*.

3.1.1.5 Abstraction from Super Classes in the UML Data Model

To ensure consistency of design, all RESQML data objects inherit from the same abstract super class in the UML model. AbstractObject in the Energistics common package (in UML under v2.1 in the package named *Abstract*) contains the UUID, Schema Version, and mandatory EIP metadata citations.

For more information on the UML model organization, see Chapter 2.

For more information on common data objects and how they work, see the *Energistics CTA Overview Guide*.

3.1.2 Using UUIDs and EIP Metadata for Traceability to Help Manage Different Versions of a Data Object

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

3.1.2.1 Overview

In **Figure 3-1**, Software A and B represent applications used in consecutive stages of a subsurface workflow. When new information becomes available for a given model in Software A, users of Software B will typically have to reload a new RESQML document containing the entire updated model. If users of Software B already modified the model, they will have to either:

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

3.1.2.2 How it Works

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

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

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

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

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

3.1.2.3 Conflict Resolution: Frequent Exchange in One Session

Figure 3-1 shows an example workflow for resolving data conflicts resulting from frequent update and exchange of data models during one session.

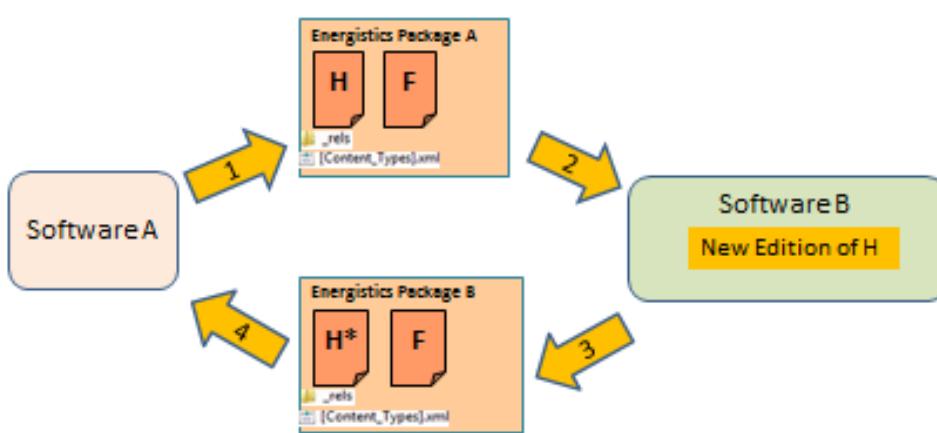


Figure 3-1. Conflict resolution: frequent exchange in a single session.

The scenario is:

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

3.1.2.4 Conflict Resolution: Exchange in Multiple Sessions with Persistent Data Store

Figure 3-2 shows an example workflow for resolving data conflicts resulting from updates and exchange during multiple sessions with persistent data store.

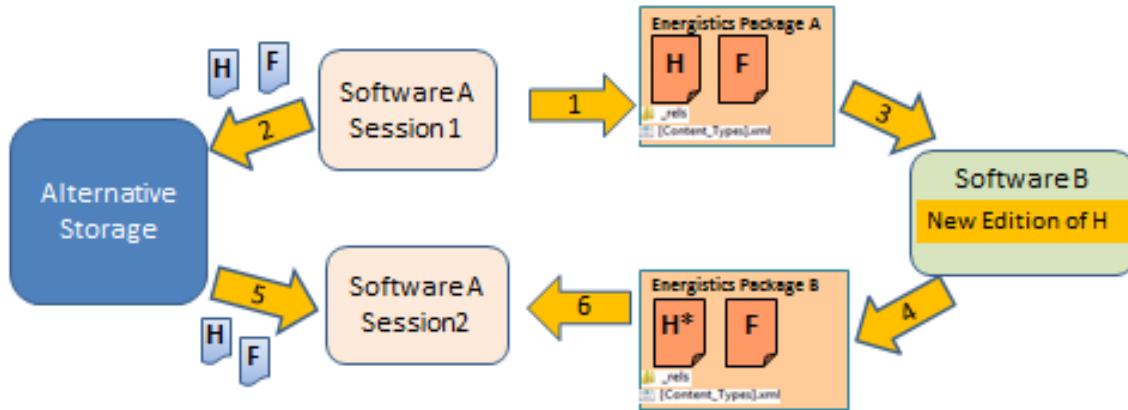


Figure 3-2. Conflict resolution: exchange with alternative persistence.

The scenario is similar to the single-session scenario:

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

3.1.2.5 Conflict Resolution Example

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

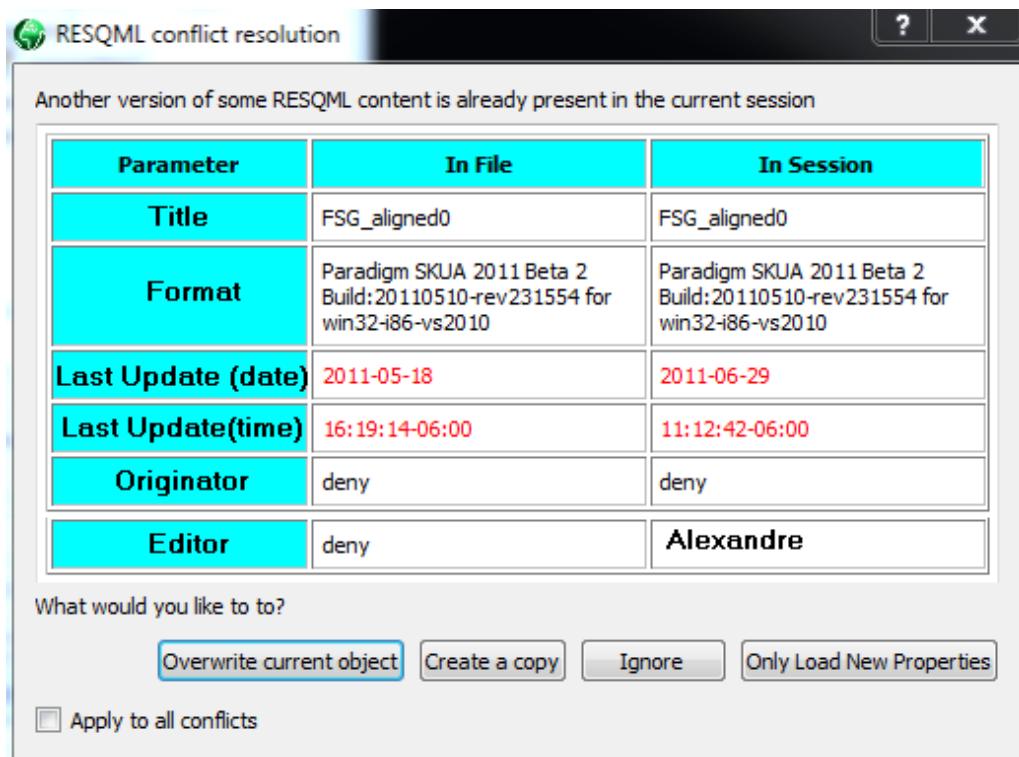


Figure 3-3. Conflict resolution example of how RESQML and EIP metadata can be used in software to resolve data conflicts.

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

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

3.1.3 "Knowledge Hierarchy": Features, Interpretations, Representations, and Properties

RESQML has now expanded the concept of "data" to include the relationships between data objects, which allows a more precise classification. The current design now supports the transfer of abstract subsurface features, human interpretations of those features, the data representations of those interpretations, and the properties indexed onto those representations, which results in a well-defined knowledge hierarchy of feature/interpretation/representation/properties (informally referred to as "FIRP")—a key organizing concept in RESQML. Geometry is included in the RESQML representation.

The table below defines these terms; each of these is a type of RESQML data object.

- For a high-level list of RESQML data objects and their organization, see Section 3.2 (page 35).
- For more information about these classifications, the knowledge hierarchy, and how it works, see Chapter 4 (page 39).

Construct/Term	Definition
Feature	<p>Something that has physical existence at some point during the exploration, development, production, or abandonment of a reservoir. For example: It can be a boundary, a rock volume, a basin area, but also extends to a drilled well, a drilling rig, an injected or produced fluid, or a 2D, 3D, or 4D seismic survey.</p> <p>RESQML has two categories of features:</p> <ul style="list-style-type: none"> Geological, for objects that exist a priori, in the natural world, for example, rock formations and how they are positioned. NOTE: "Geological feature" is not explicitly defined in the RESQML data model, but feature definitions identify those that are categorized as geological. Technical, for objects that exist by the actions of humans, for example, wells or seismic surveys. Because the decision to deploy such equipment is the result of studies or decisions by humans, technical features are usually not subject to the same kind of large changes in interpretation as geologic features. However, because technical features are subject to measurement error and other sources of uncertainty, they can be considered as subject to "interpretation". In the RESQML knowledge hierarchy, technical features do not require an interpretation.
(Feature) Interpretation	<p>RESQML uses the definition of David Gawith, which explains an interpretation as a single consistent description of a feature. An interpretation is subjective and very strongly tied to the intellectual activity of the project team members. The initial curiosity and reasoning of the people on the project team initiates the early pre-screening campaign (remote sensing, surveys). They make hypotheses that consist of as many interpretations as necessary to describe the features. (Gawith and Gutteridge 2007; for citation, see Section 4.5 (page 47)).</p> <p>A typical example of interpretation is based on the opinion of the geometry of a feature. Other specific interpretation information may include the description of the throw of a fault, the horizon classification in a sequence stratigraphic approach, or the contacts in a structural model.</p> <p>NOTE: The RESQML formal name is actually "feature-interpretation" and some of the class names use this full term. For conciseness of documentation, we use simply "Interpretation" where this usage is not confusing.</p>
(Feature) Interpretation Representation	<p>A digital description of a feature or an interpretation. For example, currently in RESQML, a horizon interpretation may be represented by a point set, a set of triangulated surfaces, or a set of orthogonal grids.</p> <p>A representation contains the topology and the geometry of the data objects. These terms are defined in Section 3.1.4 (below).</p> <p>NOTE: Like "feature-interpretation" in the previous table row, this term is formally "feature-interpretation-representation", but we use "representation" for</p>

Construct/Term	Definition
	conciseness.
Properties	A property can be attached to any indexable element of any representation. Properties refer to semantic variables (for example, porosity, permeability, etc.) and the corresponding data values, which are recorded in arrays, which may be stored in HDF5 datasets. For more information about properties, see Chapter 7 (page 67).

3.1.4 Topology and Geometry

For an optimized design and to support workflow flexibility, topology, geometry, and properties are handled differently than in the hierarchical approach of RESQML v1.1.

To support partial model updates, the topological description of a data object has been separated from its properties. This construction is consistent with the knowledge hierarchy (explained in Section 3.1.3 (page 29)) in which a description of a feature without properties is an interpretation, while a representation is required for property data. This construction is also similar to how seismic data is stored in SEGY, with the seismic trace headers and header information separated from the seismic trace records and data.

Term	Definition
Topology	Each representation contains a topological description, which defines how to associate nodes and other "indexable elements" to represent points, lines, surfaces or volumes (like structured and unstructured grids). For complex objects like simulation grids, much of the topological description can be implicit. For more information about indexing, see Section 5.2 (page 49) and the <i>RESQML Technical Reference Guide</i> .
Geometry	Each representation contains its geometry, which is the spatial location of each selected indexable element, mainly nodes. This information may be provided as numerical arrays stored in HDF5 datasets, or specified implicitly. Geometry may be contained within a representation (as just described) or, if not used to define the geometry of a representation, implemented as a "point property". For more information about geometry, see Chapter 6 (page 56).

3.1.5 Specifying Relationships in RESQML

RESQML uses the standard XML hierarchical relationship, for example, where one data object is contained inside another. However, the nature of subsurface description means there are many possible hierarchies in a reservoir model with more complex relationships between RESQML data objects, which cannot be modeled in a single hierarchical fashion. For example:

- Relationships between feature, interpretations, and representation that create the knowledge hierarchy defined in Section 3.1.3 (page 29).
- Relationship between data objects of the "same level" of the knowledge hierarchy, such as two horizon features that intersect to form a contact feature, and which may be combined to make-up part of a structural framework.

To address these more complex relationships, RESQML uses a data object reference, which is explained in Section 4.4 (page 45).

Additionally, relationships are specified in the context of an Energistics Package, which is explained in Section 3.1.6 (page 31).

3.1.6 Jagged Array Object

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

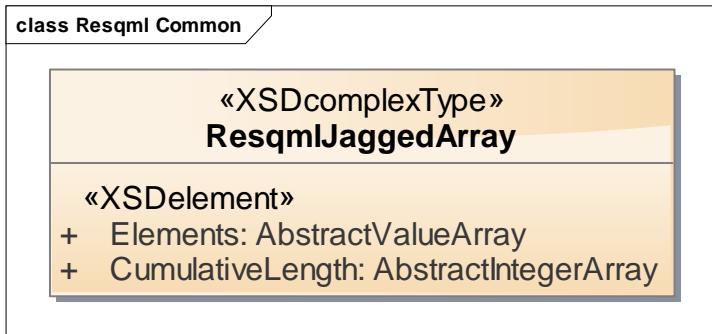


Figure 3-4. Jagged array object.

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

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

3.1.7 Activity Model

The purpose of the activity model is to capture:

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

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

RESQML through the Energistics CTA has basic mechanisms for defining activities and referencing affected data object(s). For more information, see the *Energistics CTA Overview Guide*.

3.1.8 Energistics Package

Use of an object-relationship data model and individual data objects requires a way to package together all related “parts” as a single, coherent RESQML package for data exchange. This packaging is done using the Energistics Packaging Conventions (EPC). The collective object is referred to as an EPC File (or sometimes an Energistics package). For more information on EPC, see the *Energistics Packaging Convention Specification*.

Table 1 lists the types of files that can be contained in an Energistics package. **Figure 3-5** (page 33) shows a high-level overview of the relationship among these files.

Table 1—Parts of an Energistics Package	
EPC Part	Definition, Purpose, and Requirements
XML data objects	<p>XML data objects defined in RESQML v2.0 or data objects from other Energistics data-exchange standards, such as WITSML or older versions of RESQML, may be included.</p> <p>These files use the file extension: <i>.xml</i></p> <p>NOTE: An EPC file may have xml files from multiple versions of the Energistics standards (RESQML, WITSML, and PRODML). For example, an EPC file can contain xml files saved in RESQML v2.0.1 and v2.1.</p>
EpcExternalPartReference	<p>Any file that is part of an Energistics package but stored externally to the package must have an external part reference that points to the external part.</p> <p>For example, because an HDF5 file is designed for random access (not streaming) and can already compress its data sets, RESQML requires that an HDF5 file(s) be stored outside of an Energistics Package. However, to accurately maintain all relationships, the package requires use of an external reference to the HDF5 file.</p> <p>For more information about HDF5 files, see Section 3.1.8.1 (page 33).</p>
Other types of files	<p>Optionally, other files that contain additional, informal information relevant to the contents of the package, such as these listed below. As a guideline these files are stored in a folder named “media”.</p> <ul style="list-style-type: none"> • PDF • Video • SEGY • Images • Microsoft Word documents
_rels	Describes the relationship(s) between parts in the Energistics package.

Table 1—Parts of an Energistics Package	
EPC Part	Definition, Purpose, and Requirements
[ContentType].xml	Special file required for EPC to associate file name extensions used in the package with specific mime content types.

Energistics Packaging Conventions (EPC)

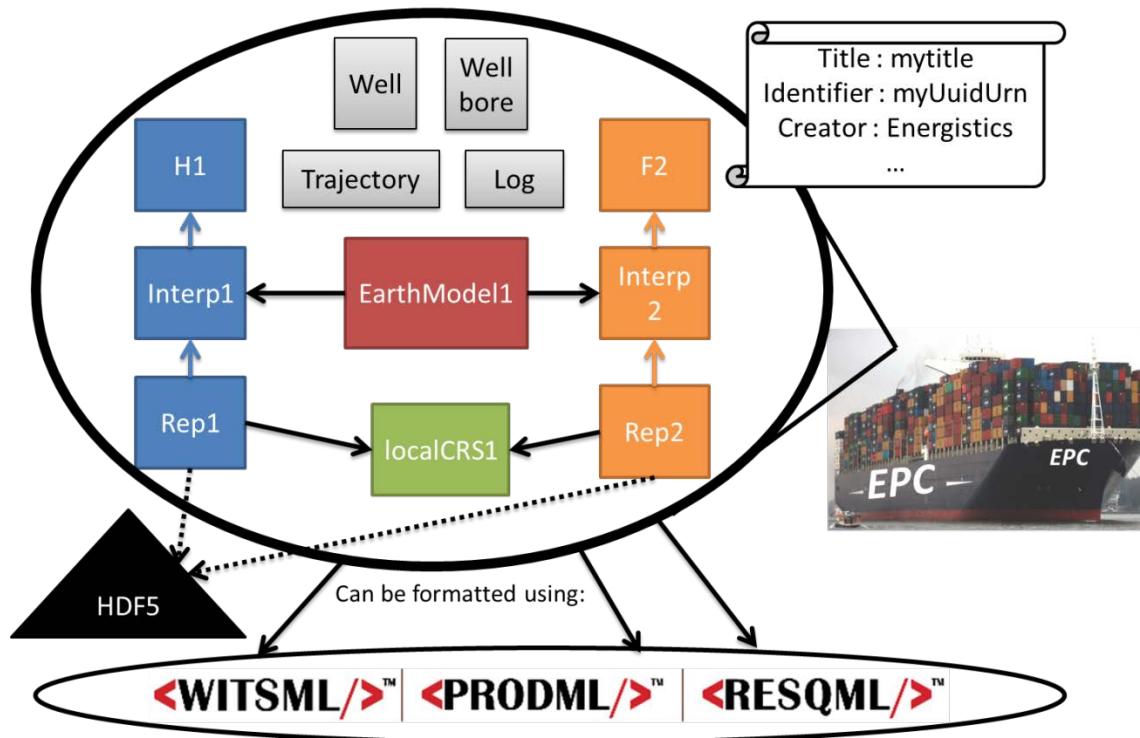


Figure 3-5. Diagram showing how EPC provides the technology to group together related files and exchange them as a single package. (Container ship photo from Wikipedia:

http://fr.wikipedia.org/wiki/Fichier:CMA_CGM_Marco_Polo_arriving_Port_of_Hamburg - 16. 01. 2014.jpg. Licence : Creative Commons paternité – partage à l'identique 3.0 (non transposée)

3.1.8.1 EPC External Part Reference and HDF5 Files: Use in RESQML

In some cases, it may be desirable to store parts of an Energistics package externally to the EPC file; the most common example is HDF5 files, which store large explicit arrays and numerical data in RESQML.

Hierarchical Data Format (HDF) is a data model, library, and file format for storing and managing data. It supports an unlimited variety of data types, and is designed for flexible and efficient I/O and for both high volume and complex data—particularly when compared to XML. HDF version 5 is part of the Energistics Common Technical Architecture and is used in RESQMLv2+ and in other Energistics standards.

The Energistics package is designed for data streaming, and in some implementations, has limitations in the amount of data which may be included. In contrast, an HDF5 file is designed for large data files, random access (not streaming), and can already compress its data sets. As a consequence, **RESQML requires that HDF5 files be stored outside the Energistics package**. To accurately maintain all relationships, the package requires use of an external reference to the HDF5 file.

The following items describe how to store and reference HDF files in the context of an Energistics package in RESQML:

- HDF5 files must be stored outside the Energistics package.
- When HDF5 files are used with Energistics standards they shall use the file extension: *.h5*
- It is recommended that the HDF5 files are stored in the same location as the Energistics package.
- Each HDF5 file must have an EPC external part reference, which is a proxy that points to the HDF5 file. This reference must be stored inside the Energistics package.
 - The HDF5 external part reference must have a relationship file that defines the actual location of the physical HDF5 file (which is recommended to be the same location as the Energistics package).
 - The corresponding entry in the relationship file must have a type attribute set to:
<http://schemas.energistics.org/package/2012/relationships/externalResource>
 - As a top-level XML data object, an external part reference must have a UUID, which must be included as an attribute of the physical HDF5 file called "uuid" to allow cross validation. This attribute should be stored at the root level of the HDF5 file.
 - The data type of the UUID in the HDF5 file must be a single string of 36 characters. In the C version of the HDF5 library this data type is *H5T_C_S1*. The UUID must be stored in its canonical format (lower case letters only) including the 4 dashes.
 - The *EpcExternalPartReference* attribute "Filename" and "MimeType" should be null. These attributes are used for non-EPC transfers. A reader must use the *Filename* and *MimeType* from the relationship file.
- You may have an array that is so large that it will not fit on a physical disk; therefore, a single array may reference multiple HDF5 files. A single *ExternalDataset* could therefore contain multiple *ExternalDatasetParts*, each of which references a single *EpcExternalPartReference*. However, each *EpcExternalPartReference* can be associated with only one HDF5 file.

3.2 High-Level RESQML Model Organization

Table 2 presents a high-level overview of the current data model and schema package organization. This table shows the main classes of data objects and an example listing of data objects within these classes. The following sections provide an overview of key RESQML domain concepts, how they map to this organization, and links to detailed chapters.

Table 2—RESQML High-Level Organization (2.1)	
Classes/Data Objects	Example Top-Level Data Objects (Schemas are the "official" source for available data objects.)
EA Packages Containing Features	Features
• Feature	Boundary Feature, Rock Volume Feature, Model
• Seismic	CMP Line Feature, Seismic Line Set Feature, Seismic Lattice Feature, Seismic Lattice Set Feature, Shot Point Line Feature
• Streamlines	Streamlines Feature
• Wells	Wellbore Feature
EA Packages Containing Interpretations	Interpretations
• Interpretations	Boundary Feature Interpretation, Boundary Feature Interpretation Plus its Rank, Feature Interpretation Set, Generic Feature Interpretation, Geologic Unit Interpretation, Stratigraphic Column Rank Interpretation, Stratigraphic Occurrence Interpretation, Structural Organization Interpretation, Earth Model Interpretation, Fault Interpretation, Fluid Boundary Interpretation, Geobody Boundary Interpretation, Geobody Interpretation, Rock Fluid Unit Interpretation, Horizon Interpretation, Stratigraphic Unit Interpretation, Rock Fluid Organization Interpretation, Stratigraphic Unit Interpretation
• Wells	Wellbore Interpretation
EA Packages Containing Representations	Representations
• Representations	Polyline Set Representation, Redefined Geometry Representation, Representation Set Representation, SubRepresentation
• Structural	Grid2d Representations, Grid2d Set Representations, Non-Sealed Surface Framework Representations, Plane Set Representations, Point Set Representation, Polyline Representation, Sealed Surface Framework Representation, Sealed Volume Framework Representation, Triangulated Set Representation
• Grids	IJK Grid Representations, Unstructured Column Layer Grid Representations, Unstructured Grid Representations, Truncated IJK Grid Representations, Truncated Unstructured Column Layer Grid Representations, (General Purpose) GPGrid Representations, Grid Connections Representation, Local Grid Set
• Wells	Wellbore Trajectory Representations, Wellbore Frame Representations, Wellbore Frame Marker Representations, Blocked Wellbore Representations, Deviation Survey, MD Datum
• Seismic	Seismic Survey representations take advantage of previously defined representations, with the addition of seismic coordinates to their geometry.
• Streamlines	Streamlines Representation

Table 2—RESQML High-Level Organization (2.1)	
Classes/Data Objects	Example Top-Level Data Objects (Schemas are the "official" source for available data objects.)
Properties Package	Properties
• Abstract Values and Points	Continuous Property, Discrete Property, Categorical Property, Comment Property, Points Property, Boolean Property, Boolean Array From Discrete Property Array,
• Property Lookup	Double Lookup, Double Table Lookup, String Lookup, String Table Lookup,
• Related Objects	Property Types, Time Series, Property Sets, Units of Measure
Geometry Package	
	Contains the classes used to define and exchanged geometry data. In general, geometry is attached to a representation with either a finite extent or planes, which are infinite. However, when spatial locations need to be stored for specific elements in a representation, then this information is stored similarly to properties using property points.

3.2.1 Feature and Interpretations

Features and interpretations are discussed further in the context of the RESQML knowledge hierarchy (see Chapter 4 (page 39)).

Specific types of features and interpretations are discussed in various chapters; for specific information on:

- Developing an earth model, see Chapter 8 (page 78).
- Streamlines, see Chapter 11 (page 213)
- Wells, see Chapter 12 (page 219)
- Seismic, see Chapter 13 (page 235)

3.2.2 Representations

Representations for various business-objects may have unique sets of requirements, for example, the representation of a horizon compared to the representation of a well. However, all representations share a set of capabilities for various indexing schemes (indexable elements, patches and multi-dimensional arrays) and representation identities.

- These shared capabilities are explained in Chapter 5 (page 48).
- For specific information on:
 - Structural and stratigraphic representations, see Chapter 9 (page 117).
 - Grids, see Chapter 10 (page 136).
 - Streamlines, see Chapter 11 (page 213)
 - Wells, see Chapter 12 (page 219)
 - Seismic, see Chapter 13 (page 235)

3.2.3 Geometry

Geometry is not an independent, top-level RESQML business object, which means that the points, lines, surfaces or volumes that may be described by RESQML cannot be transferred independently of one of the representations described in the previous section. With RESQML, you can define geometry in 2 main ways:

- Explicitly, i.e., the explicit (X,Y,Z) coordinates of a point (which was the only option available in RESQML v1.1.).
- Implicitly, which allows much more compact data transfer and provides more information about the underlying geometry of a grid, or wellbore trajectory, or fault.

For more information on geometry, see Chapter 6 (page 56).

3.2.4 Properties

Properties are independent data objects—not contained within other data objects. Property array data must be stored in HDF5 (see Section 3.1.8.1 (page 33)).

Properties may be Boolean, continuous, discrete, categorical, comments, or points. Property lookup tables are supported so that tables of production data or relative permeability tables or any other tabulated information can be included in a RESQML transfer.

Properties may be grouped, for example, to simplify the transfer of reservoir simulation data, which may involve grouping multiple properties at a single time or a single property across multiple times. The grouping mechanism supports multiple realizations and complex time relations. For example, time relations allow field surveillance data to be grouped with “near-by” times from a simulation model for comparison.

Property types, units of measure, and their management are not specific to RESQML; they are part of the Energistics Common Technical Architecture. RESQML v2+ supports a rich set of units of measure based on the Energistics Unit of Measure Standard (compared with RESQML v1, which only supported SI units).

For more information on properties, see Chapter 7 (page 67).

3.2.5 Earth Model

The RESQML "FIRP" knowledge hierarchy was defined in Section 3.1.3 (page 29). This hierarchy is useful for several reasons, but provides the most information within the context of the earth model.

In Table 2, an earth model can consist of all of the objects categorized as geologic features, interpretations, and representations. Many of the geologic representations listed in the table have clear relationships with the corresponding interpretations and features. RESQML has the functionality to let you specify relationships to create organizations building up to a coherent earth model. In addition, RESQML provides generic digital representations corresponding to points and surfaces, or their collections, e.g., for 2D grids, triangulated sets, or polyline sets.

For more information on how RESQML helps with developing and updating earth models, see Chapter 8 (page 78).

3.2.6 Grids

The second large group of representations in the table is the grid group, which represents predominantly reservoirs. A gap analysis of current industry usage led to the design of six different grid representations:

- Three grid representations are fundamental, and although there has been a lack of data-transfer standards, they are of well-known types and include: IJK grid, unstructured column-layer grid, and unstructured cell grid.
- The other three are mixtures between the first three and consist of two truncated grid representations and a general purpose grid representation. The general purpose grid is notable because it is more of a grid description development toolkit than a specific grid type, and recognizes that unstructured grid usage is still evolving. Included are geometry-free grid representations, and both lower order and higher order finite element grids.

For more information about grids, see Chapter 10 (page 136).

3.2.7 Streamlines

In a reservoir engineering context, streamlines are a way to visualize and represent fluid flow. They are especially useful for understanding the volumetric sweep relationships between wells or within a reservoir. They have many applications and, for example, have been used as a basis for fluid flow simulation, sweep management, well rate optimization, and infill well placement.

For more information about streamlines, see Chapter 11 (page 213).

3.2.8 Wells

The design of the wellbore representation draws heavily on the WITSML expertise within Energistics. The wellbore trajectory representation describes the geometry of the trajectory, and the wellbore frame representation provides the topological support for properties or relationships. Specific wellbore frames exist to represent geologic markers, and another wellbore frame is used to represent discretized wellbores along a grid.

For more information about wells, see Chapter 12 (page 219).

3.2.9 Seismic

Seismic representations take advantage of previously defined representations, with the addition of seismic coordinates to their geometry. A seismic feature describes a seismic lattice, a seismic line, or their collections. Any of the representations already described may be associated with a seismic feature, and then seismic coordinates, e.g., inline and crossline coordinates, may be associated with the geometric points of the representation. (In general, these coordinates are not integers because the points of a representation need not align with the geometry of a survey.)

The oil and gas industry already has a robust standard for the exchange of seismic data using the SEGY format. Therefore, RESQML does not attempt to redefine a new seismic data standard but can reference SEGY files in an Energistics package.

For more information on seismic, see Chapter 13 (page 235).

4 Knowledge Hierarchy: Features, Interpretations, Representations, and Properties

As introduced earlier in Chapter 3 (page 24), RESQML v1.1 had the concept of data "versions." RESQML has now expanded the concept of "data" to include the relationships between data objects, which allows a more precise classification.

The current design now supports the transfer of abstract subsurface features, human interpretations of those features, the data representations of those interpretations, and the properties indexed onto those representations, which results in a well-defined knowledge hierarchy of feature/interpretation/representation/properties (informally referred to as "FIRP")—a key organizing concept in RESQML (**Figure 4-1**). Additionally geometry is integral to RESQML organization.

The knowledge hierarchy is set up to associate and retrieve the different levels of information on data objects exchanged between software applications, in context, which essentially allows users to monitor the progress of the related business processes.

This chapter:

- Provides definitions for features, interpretations, representations and properties concepts, and introduces topology and geometry.
- Explains how these different levels of data objects fit together into a knowledge hierarchy—a crucial new RESQML concept for more precise data organization and more efficient data exchange—and refers to specific detailed chapters, as necessary.
- Explains how to specify relationships between data objects in RESQML.

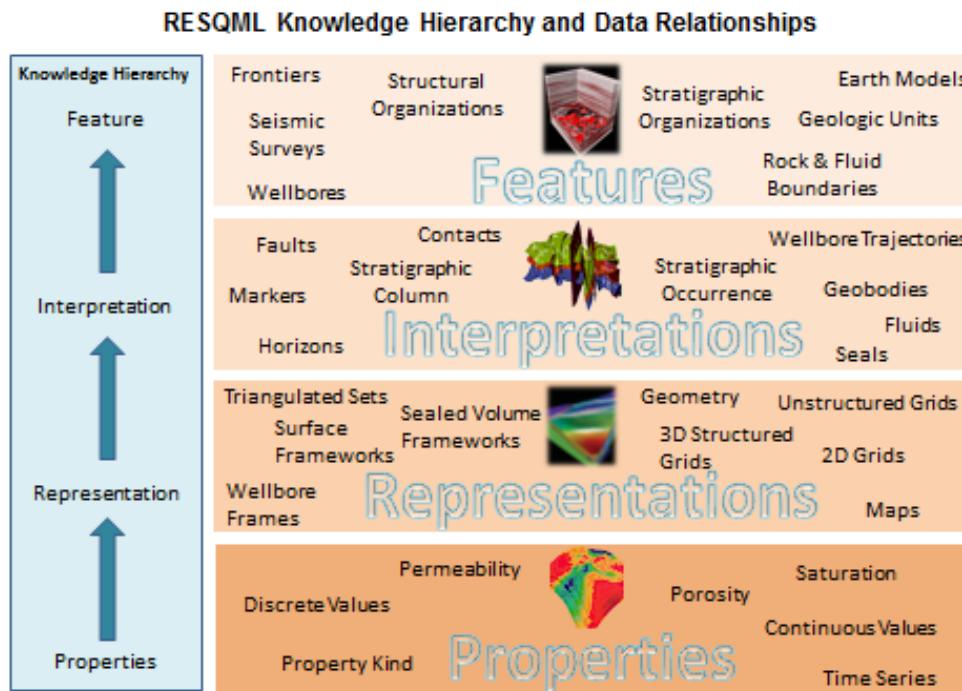


Figure 4-1. The feature/interpretation/representation/properties knowledge hierarchy (referred to informally as "FIRP") is a new concept in RESQML; it makes data organization more precise and data exchange more efficient.

4.1 Features, Interpretations, Representations, and Properties

The table below (repeated from Chapter 3 for convenience) defines these terms; each of these is a type of RESQML data object. Sections below describe additional details about each and the role of each (how it is used) in the knowledge hierarchy.

For a high-level list of RESQML data objects and their organization, see Section 3.2 (page 35).

For each of these levels, each instance of the corresponding data objects is uniquely identified with a UUID and metadata (a citation data object). The ability to uniquely identify each instance allows a user to start a business process (for example, define horizons and faults or build an earth model) by creating an instance of one feature. This begins the modeling process.

As work progresses in the business process, the user can create interpretations of the feature, and representations of the interpretations. Properties can also be added to representations and geometry defined. This process can continue (according to the requirements of the specific business process), with as many interpretations and representations as the business process requires.

Construct/Term	Definition
Feature	Something that has physical existence at some point during the exploration, development, production, or abandonment of a reservoir. For example: It can be a boundary, a rock volume, a basin area, but also extends to a drilled well, a drilling rig, an injected or produced fluid, or a 2D, 3D, or 4D seismic survey. RESQML has two categories of features: geological and technical.
	Geological Feature. Objects that exist <i>a priori</i> , in the natural world, for example: the rock formations and how they are positioned with regard to each other; the fluids that are present before production; or the position of the geological

Construct/Term	Definition
	<p>intervals with respect to each other. Some of these objects are static—such as geologic intervals—while others are dynamic—such as fluids; their properties, geometries, and quantities may change over time during the course of field production.</p> <p>Technical Feature. Objects that exist by the action of humans. Examples include: wells and all they may contain, seismic data (surface, permanent water bottom), or injected fluid volumes. Because the decision to deploy such equipment is the result of studies or decisions by humans, technical features are usually not subject to the same kind of large changes in interpretation as geologic features. However, because technical features are subject to measurement error and other sources of uncertainty, they can be considered as subject to “interpretation”. In the RESQML knowledge hierarchy, technical features do not require an interpretation.</p>
(Feature) Interpretation	<p>RESQML uses the definition of David Gawith, which explains an interpretation as a single consistent description of a feature. An interpretation is subjective and very strongly tied to the intellectual activity of the project team members. The initial curiosity and reasoning of the people on the project team initiates the early pre-screening campaign (remote sensing, surveys). They make hypotheses that consist of as many interpretations as necessary to describe the features. (Gawith and Gutteridge 2007; for citation, see Section 4.5 (page 47).</p> <p>A typical example of interpretation is based on the opinion of the geometry of a feature. Other specific interpretation information may include the description of the throw of a fault, the horizon classification in a sequence stratigraphic approach, or the contacts in a structural model.</p> <p>NOTE: The RESQML formal name is actually "feature-interpretation" and some of the class names use this full term. For conciseness of documentation, we use simply "Interpretation" where this usage is not confusing.</p> <p>v2 uses the term "interpretation" instead of alternative terms that were used in v1.1, such as "version" or "opinion".</p> <p>Most of the information contained as attributes or enumerations in individual feature interpretations or organization interpretations will help users understand how the representations of the geologic objects should be built or have been built, if the representation is already associated to the given interpretation.</p>
(Feature) Representation	<p>A digital description of a feature or an interpretation. For example, currently in RESQML, a horizon interpretation may be represented by a point set, a set of triangulated surfaces, or a set of orthogonal grids.</p> <p>A representation contains the topology and geometry of a structural feature.</p> <p>BUSINESS RULES: Representations in RESQML must be in a single projected 2D CRS. All representations associated with the same interpretation may be in either time or in depth/elevation, but a mixture of time, depth and elevation vertical reference systems is NOT allowed.</p> <p>Representations may not always be associated with interpretations, although this pattern is recommended for subsurface representations for which interpretations exist.</p> <p>A representation contains the topology and the geometry of the data objects. These terms are defined in Section 4.2 below.</p> <p>NOTE: Like "feature-interpretation" in the previous table row, this term is formally "feature-interpretation-representation", but we use "representation" for conciseness.</p>
Properties	<p>A property can be attached to any indexable element of any representation.</p> <p>Properties refer to semantic variables (for example, porosity, permeability, etc.) and the corresponding data values, which are recorded in arrays, which may be</p>

Construct/Term	Definition
	<p>stored in HDF5 datasets.</p> <p>For more information about properties, see Chapter 7 (page 67).</p>

4.2 Topology and Geometry

To support partial model updates, the topological description of a data object is separate from its properties. This construction is consistent with the knowledge hierarchy explained above, in which a description of a feature without properties is an interpretation, while a representation is required for property data. This construction is also similar to how seismic data is stored in SEGY, with the seismic trace headers and header information separated from the seismic trace records and data.

The table below (also repeated here from Chapter 3, for convenience) defines these terms and points to more detailed chapters.

Term	Definition
Topology	<p>Each representation contains a topological description, which defines how to associate nodes and other “indexable elements” to represent points, lines, surfaces or volumes (like structured and unstructured grids). For complex objects like simulation grids, much of the topological description can be implicit.</p> <p>For more information about indexable elements, see Section 5.2 (page 49) and the <i>RESQML Technical Reference Guide</i>.</p>
Geometry	<p>Each representation contains its geometry, which is the spatial location of each selected indexable element, mainly nodes. This information may be provided as numerical arrays stored in HDF5 datasets, or specified implicitly. Geometry may be contained within a representation (as just described) or, if not used to define the geometry of a representation, implemented as a “point property”.</p> <p>For more information about geometry, see Chapter 6 (page 56)</p>

4.3 How These Concepts are Used in the Knowledge Hierarchy

This knowledge hierarchy is set up to associate and retrieve different levels of information for data objects exchanged between applications, for each of the levels of the hierarchy defined in Section 4.1 (page 40) and shown in **Figure 4-1**. For more information how to specify the relationships among these different data objects, see Section 4.4 (page 45).

We have four levels of the knowledge hierarchy (informally referred to as "FIRP"):

Feature

Interpretation

Representation

Properties

4.3.1 Feature Level

A feature can be any top-level business object required by a business process, such as the individual data objects or organization data objects described above or listed in Table 2 (page 35). The creation of a feature is a declaration that this is a business object that will be further studied and developed over a span of time. As such, it must be uniquely identified, and we must be able to access all data and information related to it, including future interpretations of it, future representations of those interpretations, and the properties attached to the representation. A feature may be:

- An individual data object, such as a rock volume or boundary feature.

- Or a complex business object of interest on which users will study, collaborate and further develop using various software applications, which RESQML refers to as a *model*.

When beginning a series of data exchanges between software packages (or applications), the user must determine a reference feature data object that will be the entry point and the "pivot location" for identifying related interpretations and representations.

The role of the feature is to ensure a unique reference, which is why each feature has a UUID and metadata. The attached metadata (a citation element) reports the circumstances of the creation of this feature.

At the beginning of an exchange between software packages, to avoid redundancy and to ensure reliable management of the data object, the "reader" software must verify whether or not it has already imported this specific feature data object. If it has already imported the feature, it must not import it again. Instead, it should retrieve the data it has stored on that feature and determine if the new data being exchanged contains any updates or new information.

IMPORTANT! This specific feature data object must not be modified; rather, all changes to the data object are made at the interpretation, representation or property level. As data objects are exchanged back and forth among software applications, each package updates metadata (appropriately, based on the changes it made) for interpretations, representations, and properties.

For more information on features and interpretations in the context of an earth model, see Chapter 8 (page 78).

4.3.2 Interpretation Level

The interpretation level corresponds to the results of a step done by one user or software agent in a business process.

Most of the information contained in an individual interpretation or organization interpretation is as attributes or enumerations, which helps users understand how the topology and the geometry of the geological objects and organization representations should be built or have been built, if the representation is already associated with the given interpretation.

Like a feature, an interpretation is uniquely identified by a UUID and metadata. Interpretations have two additional types of information:

- **Geologic knowledge**, which describes (usually by enumerations) the hypothesis used to obtain the associated representation (embedded directly as an attribute of the interpretation):
 - Structural example: a horizon interpretation is unconformable below
 - Grid example: a reservoir organization interpretation is based on a specific stratigraphic organization interpretation
- **Link to a representation** (which could be of several types) which is obtained by applying the hypothesis declared in the interpretation using one software package (or application). We can have several representations based on the same hypothesis, for example:
 - **Structural example:** On one hypothesis: a horizon interpretation is unconformable below:
 - Surface Representation 1: a point set
 - Surface Representation 2: a triangulated surface
 - Surface Representation 3: an interval edge in a grid (which in this case is a subrepresentation of a grid; for more information, see Section 5.4 (page 52)).

In this example, these 3 representations may come from 3 different software packages; that is, we have 3 different representation of the same interpretation.

- **Reservoir example:** On one hypothesis, a reservoir: “rock” fluid organization interpretation is contained in an earth model interpretation based on a specific stratigraphic column we can have 3 representations of the rock volume:
 - Rock Volume Representation 1: a structured explicit grid oriented north-> south with 50X50 cells and 45,000 layers
 - Rock Volume Representation2: an unstructured grid oriented northwest -> southeast with 20,000 cells
 - Rock Volume Representation 3: a structured “parametric lines” grid oriented north-> south with 50X50 cells and 45,000 layers (For more information on parametric lines, see Section 6.3 (page 59)).

Note that these three grids, which can have different orientations due to the direction of the flux (a simulation with different Injector wells), are based on the same stratigraphic organization interpretation. For more information on grids, see Chapter 10 (page 136).

4.3.3 Representation Level

The representation level corresponds to the 3D modeling expression of the feature initialized at the beginning (as described in Section 4.3.1 above).

Often, a representation contains a topological description and very often the 3D position of the “indexable elements” of this topology. Additionally, it has more concepts for organizing its data; for more information about representation concepts (including indexable elements) see Chapter 5 (page 48).

NOTE: In RESCUE (predecessor to RESQML), software packages exchanged data only at the representation level. For example, in RESCUE, software only exchanged a grid representation and we did not know how and when it was modeled, nor the horizons and faults used to construct the grid. In RESQML v1.1, horizons and faults had features and interpretations.

With the RESQML knowledge hierarchy, a representation "knows" the UUID of the interpretation on which it was based (see Section 4.1 (page 40)). With this information, a user can return to the last geomodeling process and build a new representation, based on the same principles as the previous one, but with different parameters. For example:

- **Structural example:** A user can use a triangulation to represent a horizon previously represented as a 2D grid. After more operations, he/she can come back to the original 2D grid information data for precise fitting purposes.
- **Reservoir example:** A user can create a new representation by executing a local grid refinement after having reduced the number of cells of an existing representation. The stratigraphic column attached remains the same.

4.3.4 Property Level

The property level was set up to easily transfer newly updated properties for a well-known existing representation. (For more information on properties, see Chapter 7 (page 67)).

The property level corresponds to valuating the indexed elements of a representation. The properties (which can also be location properties, i.e., geometry) are in this case attached to a specific topological description. For one specific topological description—if you know the UUID of the representation to which they are attached—you can attach several properties with different time stamps and exchange these properties between different software without exchanging the representations itself—you only need to specify the representation.

For example, if a software package makes a change to a data object at the interpretation level, the software must update the data object's metadata including the "Last Change" element (date and time the data object was last changed) and the "Editor" element (the person or organization that made the change). This type of change management tracking can be used for re-evaluation of models and field appraisal management.

4.4 Specifying Relationships in RESQML

RESQML uses the standard XML hierarchical relationship, for example, where one data object is contained inside another. However, the nature of subsurface description means there are many possible hierarchies in a reservoir model with more complex relationships between RESQML data objects, which cannot be modeled in a single hierarchical fashion. For example:

- Relationships between feature, interpretations, representations, and properties that create the knowledge hierarchy defined above (which is informally referred to as "FIRP").
- Relationship between data objects of the "same level" of the knowledge hierarchy, such as two horizon features that intersect to form a contact feature, and which may be combined to form a part of a structural framework.

To address these more complex relationships, RESQML uses a data object reference, which is explained in Section 4.4.1 below. Section 4.4.2 (page 46) shows an example of how it works.

Additionally, relationships are specified in the context of an Energistics Package, which is explained in Section 3.1.6 (page 31).

4.4.1 Relationship Mechanisms: Data Object Reference

The data object reference is a special mechanism in the schema (`DataObjectReference` in the common package of the schema/EA model) that is used to specify relationships between and among RESQML data objects. The reference mechanism is specified in the data object schema and includes:

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

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

4.4.1.1 *"Direction" of Data object References*

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

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

- Parent-child:
 - In a one-to-one parent-child relationship, the data object reference is from the parent to the child.
 - In a one-to-many parent-child relationship—and always for the "FIRP" relationship (see the example in Section 4.4.2 (below))—the data object reference is from the child to the parent.
- *isbasedOn*, where one data object is based on another data object defined at the same level. The data object reference goes from one class to another to express that the source object which is BasedOn a target object needed to collect all the information gathered by the target object.

4.4.2 Example: "FIRP" Relationships

The relationships between features, interpretations, representations and properties are parent-child relationships held by the child(ren). **Figure 4-2** shows an example, which is further explained in the text below. As explained in the previous section, the child data objects specify the relationship.

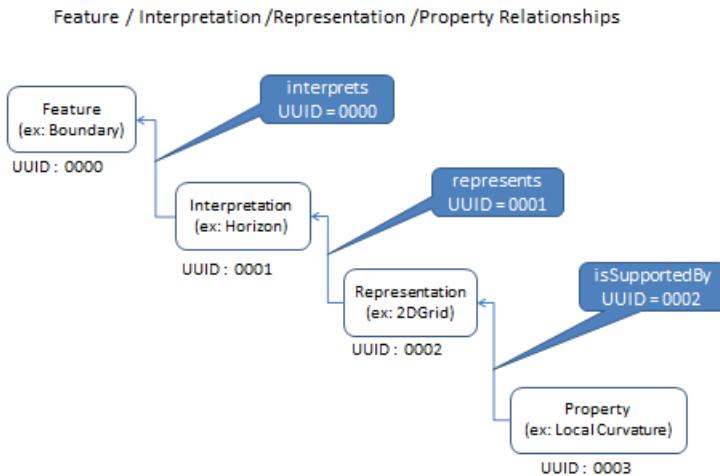


Figure 4-2. Example of relationships in RESQML for FIRP. Property "is supported by" (or provides values for) a representation; a representation "represents" an interpretation; an interpretation "interprets" a feature.

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

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

Figure 4-3 shows the addition of several children, which include a new interpretation, representation and property.

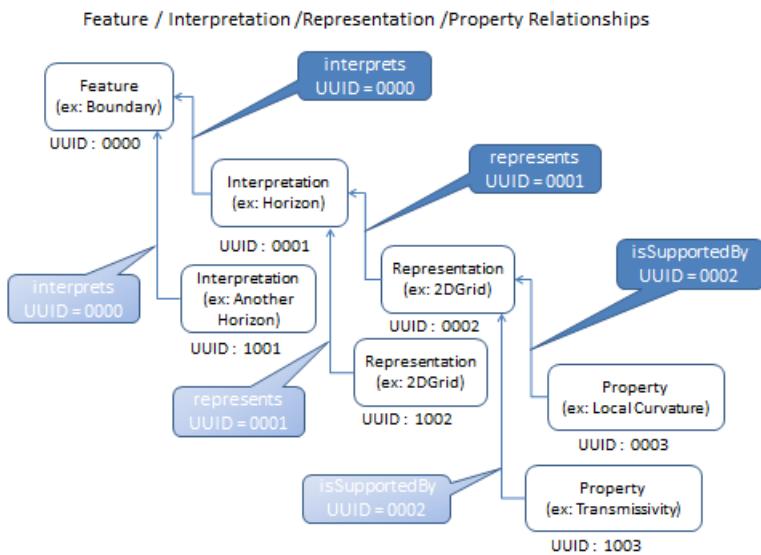


Figure 4-3. The previous example is extended to show multiple children with the addition of: another interpretation of the boundary feature (UUID 1001), another representation of the horizon interpretation (UUID 2002), and another property on the first representation (UUID 1003).

4.5 References

Gawith, D and Gutteridge, P. (2007). Redefining what we mean by shared earth model. *First Break*, 25:10. EAGE <http://www.earthdoc.org/publication/publicationdetails/?publication=27669>

5 Representations (Shared Concepts)

A representation has two distinct and complementary roles in RESQML:

- It is an important component of the feature/interpretation/representation/property knowledge hierarchy where it corresponds to a 3D modeling expression of a feature that was initialized at the beginning of a business process (as explained in Section 4.1 (page 40)). For example, the same horizon feature-interpretation can have a 2D grid representation or a triangulated set representation.
- It supports the geometry and properties of data objects in RESQML. The geometry of a representation is contained within the representation, while properties may be attached to the representation. Each property is “attached” to the indexable elements of a representation, which may be as simple as the nodes on a single triangulated surface or as complex as the cell, nodes, faces, pillars, coordinate lines, columns, etc., for a 3D grid.

All representations share some key concepts, which are explained in this chapter. For more information on these related topics, see the indicated chapter:

- For geometry, see Chapter 6 (page 56).
- For properties, see Chapter 7 (page 67).
- For structural/stratigraphic representations, see Chapter 9 (page 117).
- For grids, see Chapter 10 (page 136).
- For streamlines, see Chapter 11 (page 213).
- For wells, see Chapter 12 (page 219).
- For seismic, see Chapter 13 (page 235).

5.1 Representations Overview

Representations of different business objects (e.g., a horizon versus a reservoir versus a seismic survey) have different requirements and elements. However, all RESQML representations share several common concepts. This chapter explains these shared concepts, which include:

- **Indexable elements** (see Section 5.2 (page 49)). Geometric or topological elements in a representation that can be enumerated by a contiguous set of integral numbers, which are called **indices**. The indices are used to uniquely specify how properties and geometry are associated with elements in a representation. A subset of indexable elements can be identified by an ordered subset of indices. Indices can be multi-dimensional.
- **Patches** (see Section 5.3 (page 51)). A Patch is a mechanism in RESQML that provides a clear way of ordering indices to avoid ambiguity in their definition. For example, the representation of a horizon may consist of 10 triangulated surfaces. To correctly order the geometry or properties on this representation, the software importing or reading that horizon must know the indices within each of the 10 triangulated surfaces AND how the 10 triangulated surfaces are sequenced.

All classes with "patch" in their name (e.g. NodePatch, PolylineSetPatch, Grid2DPatch, etc.) have a patch index, which is implicitly defined by the order in which that patch object appears in a RESQML file. For instance, the first NodePatch would have a patch index of 0, the second a patch index of 1, etc. This order is used to uniquely sequence the elements of the patches.

- **Subrepresentations** (see Section 5.4 (page 52)). A subrepresentation is a logical and ordered subset of one or several existing representations. It is itself a representation. Subrepresentations are used to define the topological elements of new representations that get their geometry from some previously defined existing representations.
- **Representation identities** (see Section 5.5 (page 53)). Relationships (or semantics) between nodes of representation or subrepresentations.

In RESQML v1.1, gridding information was exchanged at the representation level only. Now, a grid representation (and all the other representations) is associated with the UUID of an interpretation of a geologic feature, so that a user can revisit the last geomodeling process and build a new representation that preserves the interpretation. For example:

- For structural modeling, the user can use a triangulation to represent a horizon that was previously represented by a 2D grid. After more operations, the user can go back to the original 2D grid representation for precise fitting purposes. He can do this because these two representations are linked to the same interpretation.
- For reservoir grids, the user can create a new representation by specifying a local refinement of an existing grid after having reduced the number of cells of an existing representation. The stratigraphic column (which is a stratigraphic organization interpretation) attached remains the same. He can do this because the two grid representations are linked to the same interpretation.

5.2 Indexing

RESQML now makes extensive use of the concept of indexing to order and reference the indexable elements of a representation. RESQML uses both one-dimensional and multi-dimensional arrays of elements. So that all elements may be referenced in a consistent and uniform fashion, each multi-dimensional index must have a well-defined 1D index.

5.2.1 Indexable Elements

Table 4 shows the RESQML indexable elements and indicates for which representation or representations these elements are defined. This list of indexable elements is used to:

- Contain geometry within a representation
- Attach properties to a representation
- Identify portions of a representation when expressing a representation identity
- Construct a subrepresentation from an existing representation

Several specialized indexable elements, e.g., hinge node faces, have been included to add higher order geometry to a grid representation, and are not available for other purposes.

Table 3—RESQML Indexable Elements and Applicable Representations

Indexable Elements	Applicable Representation(s)
cells	grids, wellbore frames
column edges	column-layer grids
columns	column-layer grids
contacts	surface frameworks
coordinate lines	column-layer grids
edges	triangulated sets, grids
edges per column	column-layer grids
enumerated elements	subrepresentations
faces	sealed volume frameworks, grids
faces per cell	grids
hinge node faces	grids (geometry only)
intervals	wellbore frames, column-layer grids

Table 3—RESQML Indexable Elements and Applicable Representations	
interval edges	column-layer grids
I0	IJK grids
I0 edges	IJK grids
J0	IJK grids
J0 edges	IJK grids
layers	column-layer grids
lines	streamlines
nodes	all representations
nodes per cell	grids
nodes per edge	grids
nodes per face	grids
patches	all representations
pillars	column-layer grids
radial origin polyline	IJK grid (geometry only)
regions	all representations
representation	all representations
subnodes	grids
triangles	triangulated sets

5.2.2 Multi-Dimensional Arrays and HDF5 Data Storage

Each element of a multi-dimensional array within a representation must have a well-defined 1D index, to allow elements to be uniquely referenced for properties, geometry, data storage or any other purposes, e.g., subrepresentations. The data ordering is uniquely specified at the representation level and this ordering is inherited by the array constructions for points and geometry.

For example:

- For a 2D array ($N_1 \times N_2$) with indices $I_1=0,\dots,N_1-1$ and $I_2=0,\dots,N_2-1$, then the 1D index is $I_1+N_1*I_2$.
- For a 3D array ($N_1 \times N_2 \times N_3$) with indices $I_1=0,\dots,N_1-1$, $I_2=0,\dots,N_2-1$ and $I_3=0,\dots,N_3-1$, then the 1D index is $I_1+N_1*I_2+N_1*N_2*I_3$.

This ordering choice is sometimes called “fastest to slowest”, with the first index in the equation varying the fastest, and the last index varying the slowest. RESQML is not restricted to 3D arrays, for example, the “faces per cell” on an IJK grid follow a 4D ($6 \times N_1 \times N_2 \times N_3$) array indexing.

HDF5 Data Storage. When stored in HDF5, the data storage order is the RESQML index order. It is very important to understand this relationship between indexing and data storage in RESQML. Index order for the elements within a representation is specified by the schema documentation, and this data order is preserved in the HDF5 data storage. However, because of how the HDF5 array storage works, this means that an $N_1 \times N_2$ RESQML array is stored as a $N_2 \times N_1$ HDF5 array (N_1 fastest, N_2 slowest). To avoid confusion, use of the words “first” and “last” needs to clearly distinguish between the RESQML index calculation and the HDF5 data storage context. The important point: when viewed as an equivalent 1D array, the HDF5 data storage ordering and the RESQML index ordering are identical.

NOTE: For a brief introduction on implementing HDF5, see Appendix B (page 247).

Lattice Offsets. Geometry and properties use multi-dimensional lattice offset constructions for points and values, respectively. The ordering of the offsets follows the ordering of the indices in the multi-dimensional index calculation and hence is opposite to the ordering of the HDF5 data storage.

An example of the use of multi-dimensional arrays. The coordinate line nodes on a faulted grid, where N1=coordinateLineCount and N2=NKL. However, the dimensionality of an array may vary with context; for example, the coordinate lines themselves may be either a 1D or a 2D array. In the special case of an unfaulted grid, the coordinate lines are a 2D array indexed by NIL x NJL, and the coordinate line nodes are a 3D array indexed by NIL x NJL x NKL.

Lists, Time Series and Realizations. When points or multi-dimensional (count>1) property values are stored in HDF5, this introduces an additional dimension, which is always the fastest. For example:

- An N dimensional array of points3d is stored as an N*3 HDF5 array of coordinates.
- Alternatively, an N dimensional array of points2d, is stored as an N*2 HDF5 array of coordinates.
- An array of facies proportion curves, (count>1), is stored as an N*count HDF5 array of values.

BUSINESS RULE: To facilitate data validation and hyper-slabbing of the data, RESQML requires that data be stored with the maximum dimensionality possible. For the example of coordinate line nodes given earlier, this rule implies that instead of always using a 2D array format, which is possible, that a 3D array format is used for an unfaulted grid.

5.3 Patches

A Patch is a mechanism in RESQML that provides a clear way of ordering indices to avoid ambiguity. For example, the representation of a horizon consists of 10 triangulated surfaces, to correctly represent the same horizon, the software importing or reading that horizon must know the indices within each of the 10 triangulated surfaces AND how the 10 triangulated surfaces are sequenced.

5.3.1 When to Use Patches

Representations with unique indexing of their elements DO NOT require Patches. For example, a (lower order) corner-point grid has an indexing scheme that can be defined without using Patches. However, a RESQML general purpose (GP) grid (an unconstrained hybrid of any of the other RESQML grid types) is much more complex and variable, with no “natural” sequence. For a reader to correctly interpret a GP grid, the software that created the GP grid must:

- Explicitly define each Patch (specify the indices) that comprise the grid.
- Designate the correct order of the Patches.

If a representation includes indexable elements both specified within patches and external to patches, then Patch Index = 0 is defined to be the representation itself.

5.3.2 Characteristics of Patches

Patches have these characteristics, which must be followed by the software that creates the Patches (the “writer”):

- RESQML defines many types of Patches. Each Patch data object includes the word “patch” in its name, e.g., TrianglePatch.
- Each Patch must itself have a range of indexable elements.
- Each Patch has an (implicit) Patch index corresponding to the order on which the patches are stored by the writer into the XML instance.
- Each Patch has a unique zero-based Patch index defined by its order minus one in the RESQML XML instance, and which defines the order of the Patches. For example, Patch 0, Patch 1, Patch 2, etc.

5.4 Subrepresentations

A subrepresentation is a logical and ordered subset of one or several existing representations. It is itself a representation. Because RESQML has separated the representation concepts of topology, geometry, and property values, we can select a range of nodes, edges, faces, volumes, or any of the indexable elements from the topological support of representations. By extracting such a list of simple topological element indices, we define a subrepresentation.

A subrepresentation may describe a different feature-interpretation using the same geometry or property as the "parent" representation(s). In this case, the only information exchanged is a set of potentially non-consecutive indices of the topological support of the representation(s).

As described in **Figure 5-1**, a subrepresentation refers to a list of supporting representations and contains one to several subrepresentation patches. Each of these subrepresentation patches can define up to 2 element indices.

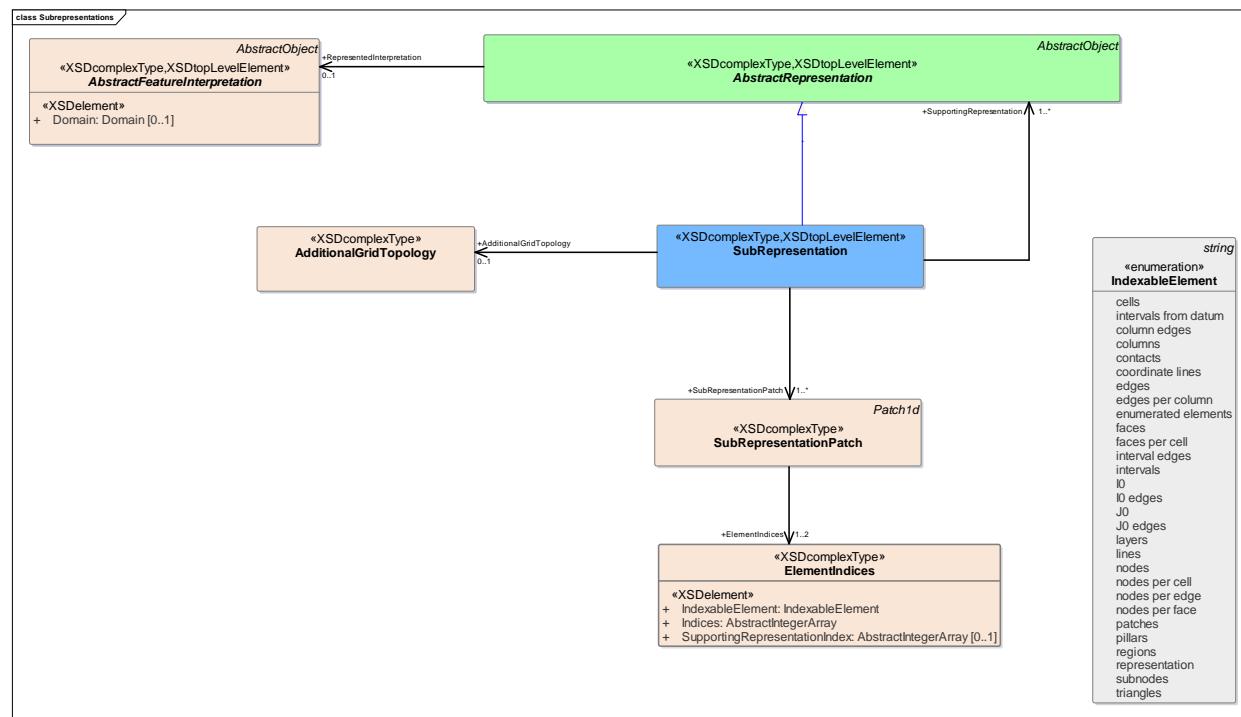


Figure 5-1. Subrepresentation.

In the common case, only one “element indices” is provided. An “element indices” identify the selection of nodes, edges, faces, volumes, or any of the indexable elements from one or several supporting representations. The selection of indexable elements is done using the “Indices” UML attribute. The selection of the corresponding supporting representations is done using the “SupportingRepresentationIndex” UML attributes (the values are the indices of the supporting representation in the list of supporting representations contained in the subrepresentation). If the Subrepresentation only refers to a single SupportingRepresentation, then the “SupportingRepresentationIndex” UML attribute is not needed.

In the case two “element indices” are provided, the first one has to be considered in the context of the second one. For example, it allows to select a face from a reservoir grid cell by means of its local index in the reservoir grid cell. It could be for instance the face 1 of my hexaedron cell 25 of my reservoir grid UUID_A.

5.5 Representation Identities

It is also possible to add semantics about the relationships between subrepresentations, even if they come from different representations. The table below describes different identity relationships. This type of relationship is commonly used to identify the contacts between representations in model description, but may also be used to relate the components of a grid (pillars) to those of a structural framework (faults).

Concept	Definition: A set of subrepresentations or representations):
Collocated (Sub)Representations	Is Collocated if there is bijection between the simple elements of all of the participating (sub)representations. This implies there is the same number of simple elements. The geometric location of each set of simple elements mapped through the bijection is intended to be identical <i>even if the numeric values of the associated geometries differ, i.e., due to loss of spatial resolution.</i>
Equivalent (Sub)Representations	Is Equivalent if there is a map giving an association between some of the simple topological elements of the participating (Sub)Representations.
Previously Collocated (Sub)Representations	Were collocated at some time in the geologic past , but not necessarily in the present day earth model.
Previously Equivalent (Sub)Representations	Were equivalent at some time in the geologic past , but not necessarily in the present day earth model.

5.6 Redefined Geometry Representations

RESQML provides us with the ability to modify the geometry of a representation. However, because geometry is central to the definition of a representation, we can only do this by creating a new representation based upon the existing representation. This capability supports a number of very useful cases:

- Time-dependent geometry for structures that evolve in time (geomechanics, basin evolution)
- Related representations with one in depth and one in time
- Related representations with one in one coordinate reference system and one in another

Figure 5-2 shows the construction. Any representation with geometry may be selected as the supporting representation. The use of a “patch of geometry” construction allows grids with additional grid geometry to have all patches of their grid geometry redefined.

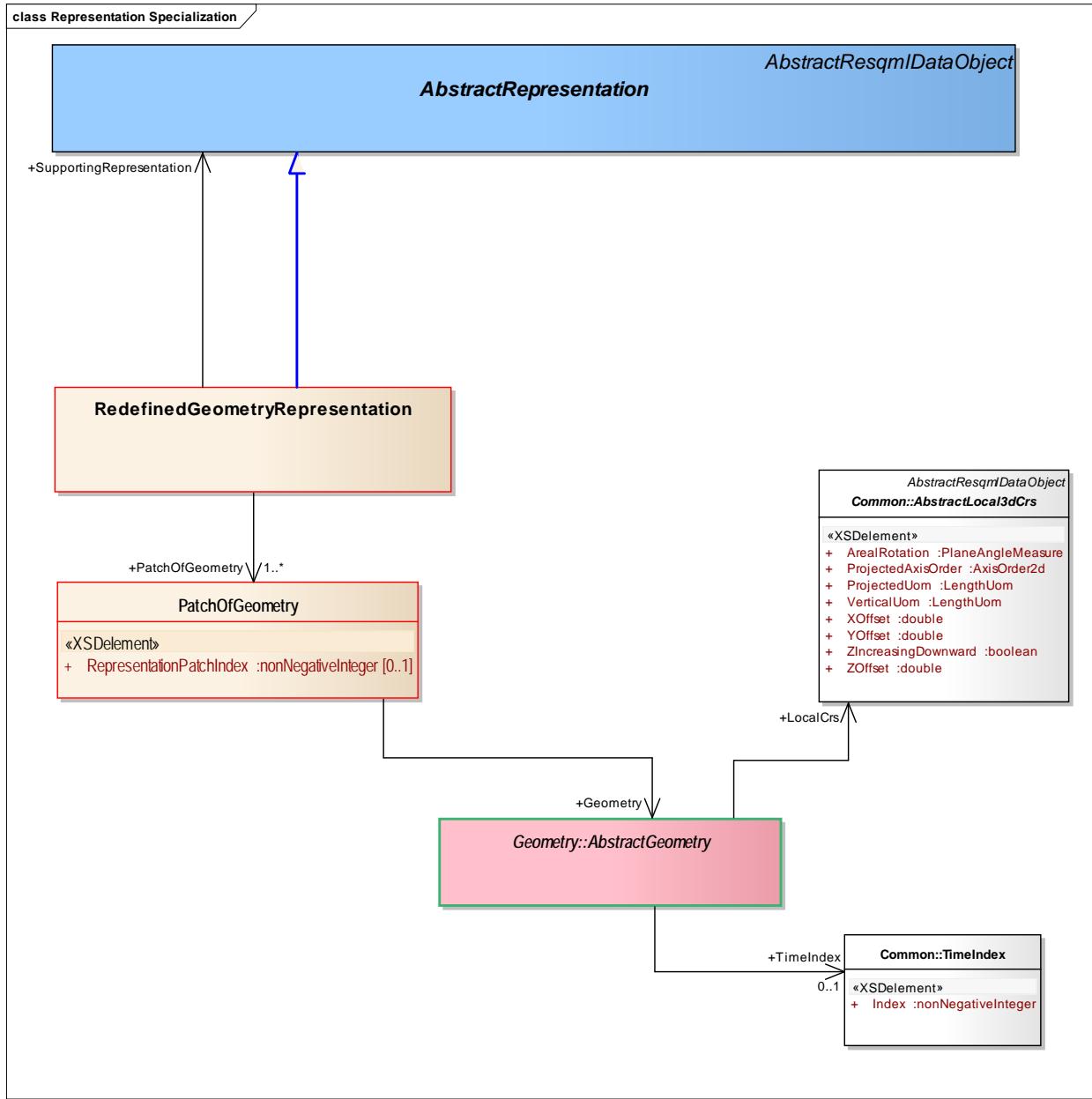


Figure 5-2. Redefined geometry representation.

5.7 Representation Set Representation

Representation set representations (**Figure 5-3**) are an indexed assemblage of individual representations. The representation index given is internal and defined in the context of a particular representation set interpretation instance. This representation index is implicit. It follows the order on which the representations are given in the XML instance by the software writer. This representation index is used in the non-sealed surface framework, sealed surface framework, and sealed volume framework.

Representation set representations are used to group together individual representations into a “bag” of representations. These “bags” do not imply any geologic consistency. For example, you can define a set of wellbore frames, a set of wellbore trajectories, and a set of blocked wellbores.

If the bag contains homogeneous individual representations, the "isHomogenous" Boolean may be set, and then the reader software need only read the first individual representation to understand the kinds of individual representations that are gathered in the set.

The representation set representation is also the parent class of the framework representations. Because the framework representations inherit from this class, they inherit the capability to gather individual representations into sealed and non-sealed surface framework representations, or sealed volume framework representations. The representation index is used to define the contact patches.

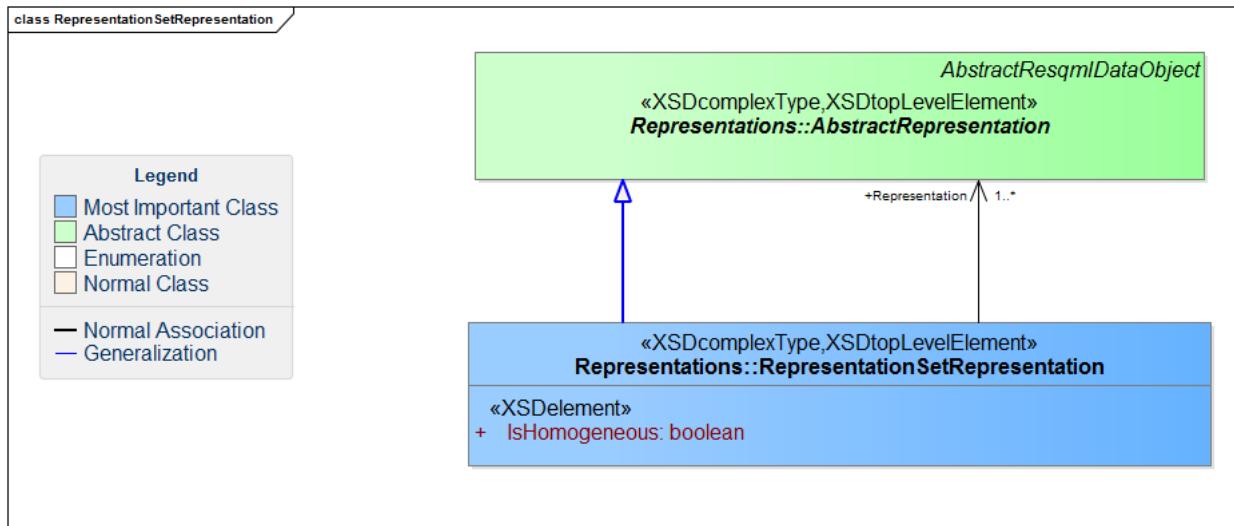


Figure 5-3. Representation set representation.

6 Geometry

RESQML has two main ways of implementing geometry:

- When geometry applies to the specification of a representation, then the methods described in this section are used.
- When geometry valued point properties are to be attached to an existing representation, then a particular type of property (points property) is used. For more information, see Section 7.3 (page 71).

This chapter describes general methods. For details on how geometry is applied to specific representations or data objects, refer to these chapters:

- For earth models, see Chapter 8 (page 78).
- For structural and stratigraphic representations, see Chapter 9 (page 117).
- For grids, see Chapter 10 (page 136).
- For streamlines, see Chapter 11 (page 213).
- For wells, see Chapter 12 (page 219).
- For seismic, see Chapter 13 (page 235).

6.1 Overview

RESQML provides descriptions for the geometries of points, lines, and planes (**Figure 6-1**). These descriptions are in the context of a local depth or time 3D coordinate reference system, and to support time varying geometries, a time index. In the following text references to a Z coordinate are appropriate for use in a depth 3D coordinate reference system. If working in time, then Z is replaced by T.

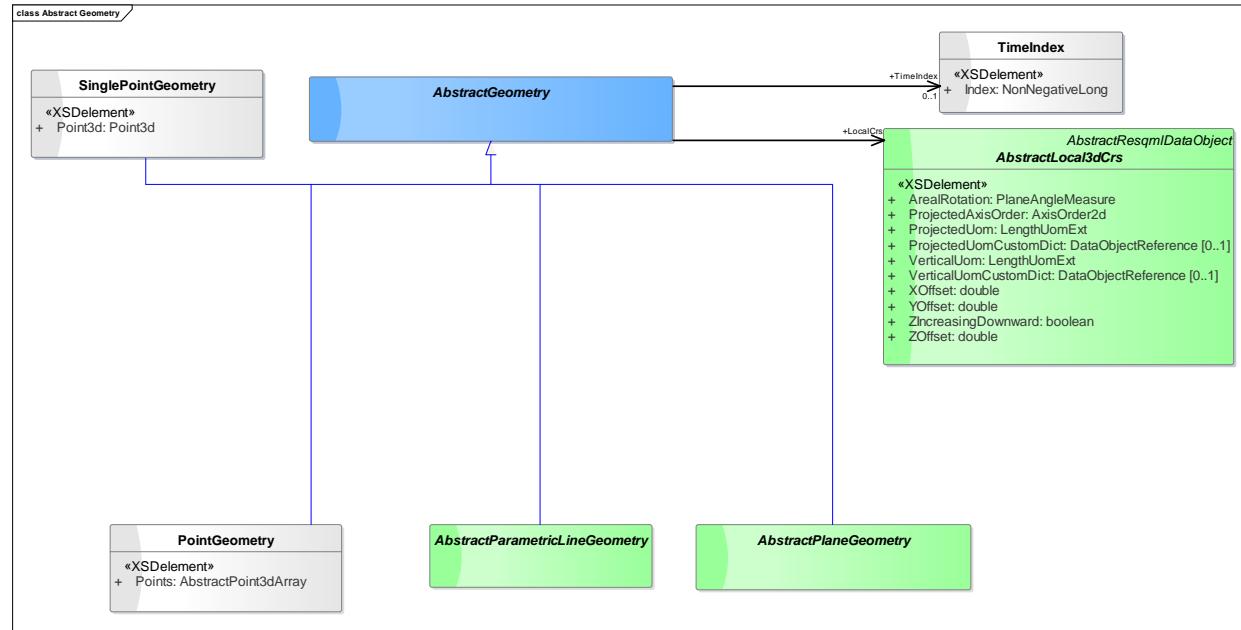


Figure 6-1. Class diagram showing geometries of points, lines, and planes.

Points are organized into arrays of 3D points, line geometry is parametric, and there are simple descriptions for horizontal or tilted plane geometry.

RESQML provides implicit geometric descriptions for the parametric points on a parametric line. A familiar example from the grid domain is the COORD/ZCORN description of an Eclipse grid coordinate line (**Figure 6-2**), in which a value for Z may be used to infer values for X and Y. RESQML includes vertical,

piecewise linear, cubic and minimum-curvature spline parametric lines. Unlike the COORD/ZCORN description, RESQML does not restrict the spline parameter to be depth. This allows support for over-turned geologic structures or for more general relationships.

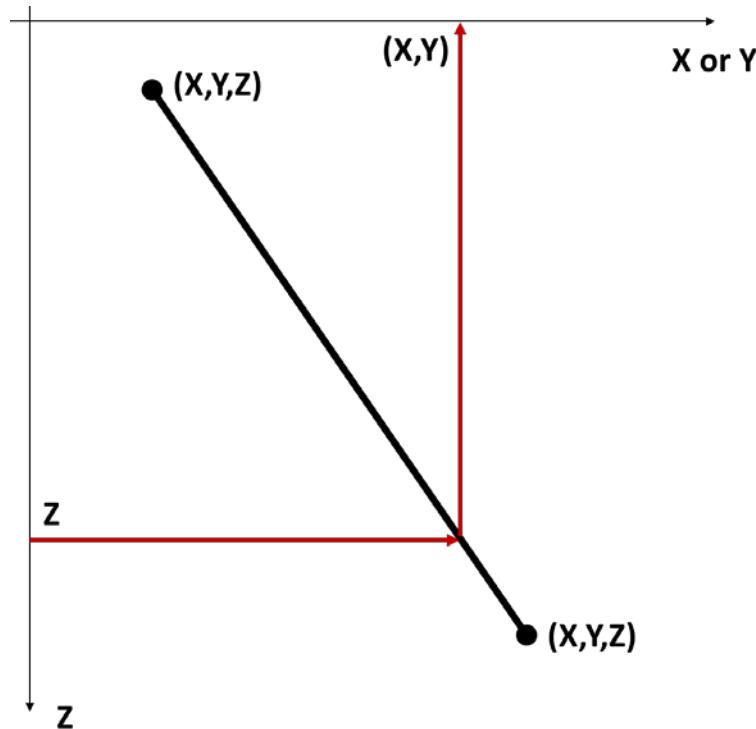


Figure 6-2. Eclipse COORD/ZCORN construction in which two (X,Y,Z) control points and a value for Z may be used to infer values for (X,Y).

The use of arrays for points and parametric lines is very similar to that of the RESQML property values. However, unlike properties, which are top-level data objects, geometry only occurs within a representation. As with the property arrays, the dimensions of these arrays are not specified within the schema geometry, but are inherited from their context within a representation.

6.2 Points, Lines, and Planes

Table 5 summarizes the different construction types available for the geometry of points, lines, and planes. Changes for this latest version of RESQML include a much richer list of construction types and a simplified (restricted) design that supports only HDF5 for explicit point arrays (previously both XML and HDF5 arrays were supported).

Table 4—Point, Line and Plane Geometry

Construction Type	Definition/Explanation
Point Geometry	
Point3d HDF5 Array	Explicit HDF5 array of N*3 coordinates, (X,Y,Z)
Point2d HDF5 Array	Explicit HDF5 array of N*2 coordinates (X,Y) for a 3D point (X,Y,0)
Point3d ZValue Array	Points extracted from another representation, and then modified to have new Z coordinate values
Point3d Lattice Array	Points in a lattice created from a start point and offset vectors
Point3d From Representation Lattice Array	Points extracted from a representation, using a lattice of indices to

Table 4—Point, Line and Plane Geometry	
Construction Type	Definition/Explanation
	space the sampling
Point3d Parametric Array	Points defined parametrically using parametric lines
Line Geometry	
Parametric Line Geometry	Parametric line geometries are described in more detail in the next table
Parametric Line From Representation Geometry	Parametric line extracted from a representation
Plane Geometry	
Horizontal Plane Geometry	Definition of the infinite geometry of a horizontal plane provided by giving its unique Z value.
Titled Plane Geometry	Used to describe the infinite geometry of a potentially tilted plane from three points.

Point, line, and plane geometry are described in the following figures (**Figure 6-3**, **Figure 6-4**, **Figure 6-5**). Each individual construction is also described in the *RESQML Technical Reference Guide*.

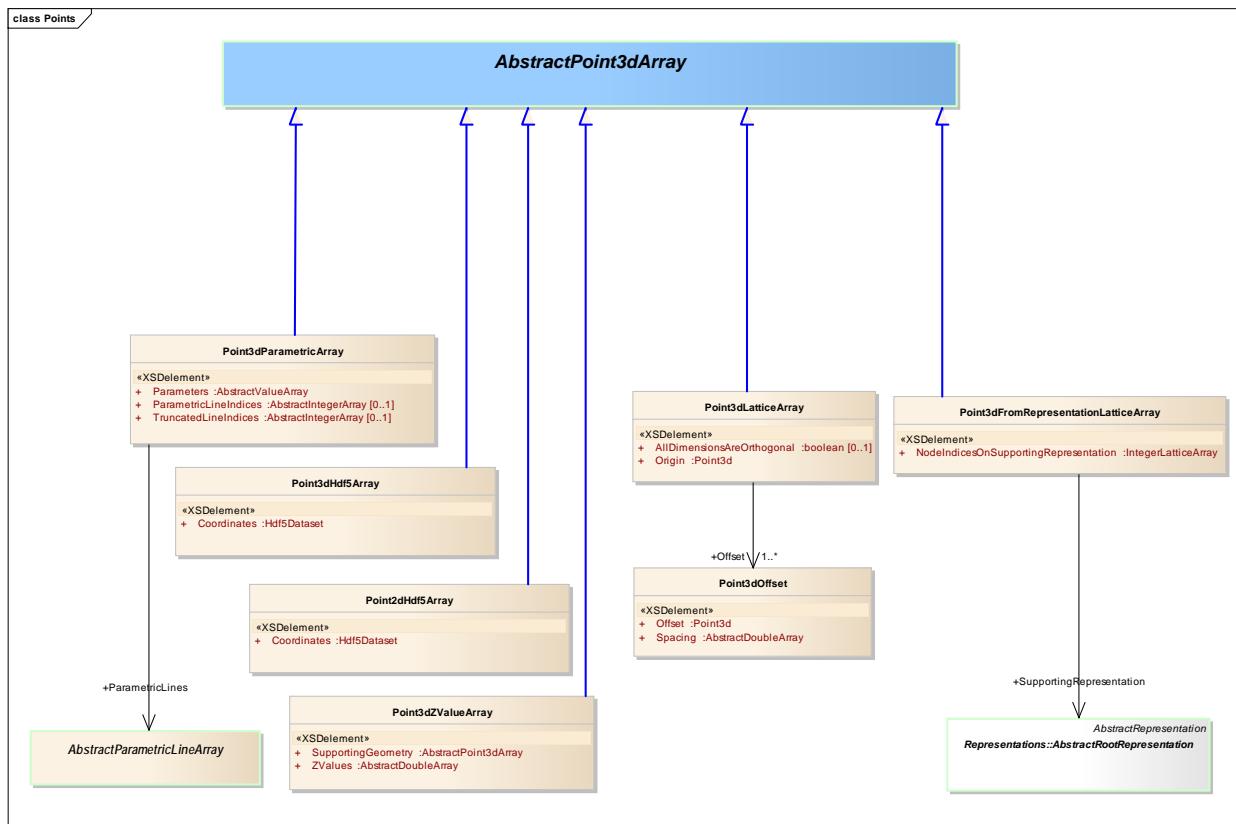


Figure 6-3. Class diagram showing the specification of point geometry organized into an array of 3D points.

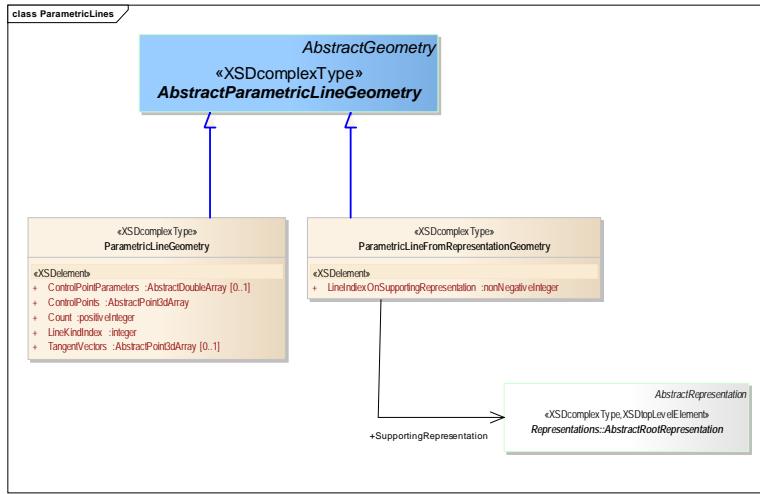


Figure 6-4. Class diagram showing the specification of parametric line geometry.

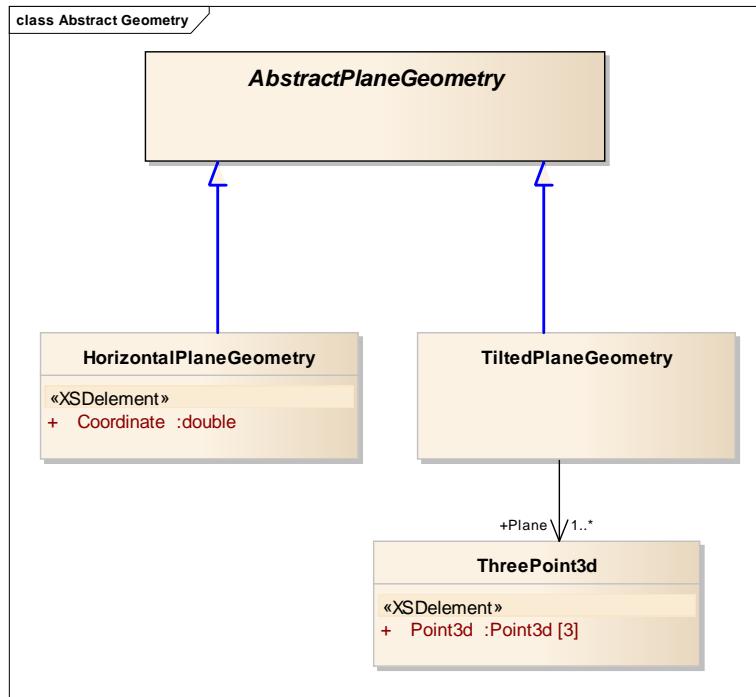


Figure 6-5. Class diagram showing the specification of plane geometry as horizontal or multiple tilted planes.

6.3 Parametric Points and Lines

A parametric line is essentially a vector-valued function. Given a parametric value P and the specification of a parametric line, what are the resulting values of $X(P)$, $Y(P)$, and $Z(P)$ (or $T(P)$) in the Local 3D CRS?

This is a parametric description of the points on the line. The use of a parametric line provides a more compact means of specifying point geometry. As importantly, it provides interpolated geometric information at higher resolution than the spacing of the control points on a piecewise linear line, which supports more accurate intersection calculations and geometric refinement.

Table 6 lists and describes the available types of parametric lines and the control points needed for each type of line.

Table 5—Description of RESQML Parametric Lines		
Parametric Line	No. of Control Point(s)	Description
Vertical	One control point: $(X, Y, -)$	Parametric value is $Z \Rightarrow (X, Y, Z)$.
Linear Spline 1 or more intervals	Two or more knots, each with a control point: $(P, X, Y, Z)_i, i=1, 2, \dots$	Piecewise linear interpolation in (X, Y, Z) as a function of the parametric value P .
Natural Cubic Spline 1 or more intervals	Two or more knots, each with a control point: $(P, X, Y, Z)_i, i=1, 2, \dots$	Piecewise cubic interpolation in (X, Y, Z) as a function of the parametric value P , subject to the additional constraints of continuous first and second derivatives at the knots, and vanishing second derivative at the edge knots.
Cubic Spline 1 or more intervals	Two or more knots, each with a control point and a tangent vector. Tangent vectors are defined as the derivative of position with respect to the parameter P .	Piecewise cubic interpolation in (X, Y, Z) as a function of the parametric value P , given the additional constraint of specified tangent vectors at the knots.
Z Linear Cubic Spline 1 or more intervals	Two or more knots, each with a control point: $(P, X, Y, Z)_i, i=1, 2, \dots$	Linear spline interpolation in Z as a function of the parametric value P . Natural cubic spline interpolation in (X, Y) as a function of the parametric value P .
Minimum-Curvature Spline 1 or more intervals (knots = stations)	One control point for the first knot. Two or more knots, each with a tangent vector. Tangent vectors are defined as the derivative of position with respect to the parameter P .	Piecewise minimum curvature interpolation in (X, Y, Z) as a function of the parametric value P , given the constraint of specified tangent vectors at the knots. Curvature is defined in the units of measure of the local 3d CRS, without unit conversion, so care must be taken when using mixed units.

With the exception of the vertical parametric line, the parameter itself has no specific interpretation, nor is there any requirement that the same parametric values be used along different lines in a parametric line array. However, it is important that the parametric values associated with the control points and the parametric values used for interpolation are consistent. When used to describe the geometry of a wellbore trajectory, the parameter is interpreted as the measured depth along the wellbore. In this case, the specification of the measured depth units of measure and datum location are part of the wellbore trajectory representation, and not the geometry itself.

NOTE: As a companion to this documentation on cubic splines, a spreadsheet (titled *20131223 RESQML Cubic Splines.xls/x*) is included with the download of the RESQML standard. The spreadsheet allows you to see the equations described here and experiment with different input values and examine the results.

6.3.1 Cubic Splines

There are many ways to describe a cubic spline. The current description draws heavily upon the description in *Numerical Recipes* (Press et al. 1992), but with some variations that are in use within our domain. For a spline, each interval ($i = 1, \dots, N$) is parameterized in terms of a parameter p , or equivalently a normalized parameter ξ , $\xi = (p - p_{i-1})/\Delta p$, $\Delta p = p_i - p_{i-1}$. The parameter takes on the values of $\xi=0$ and $\xi=1$ at the two knots that bound the interval. We may express a cubic spline in terms of

its function values and second derivatives, specified at the two knots. Spline derivatives are specified with respect to the parameter p , but need to be converted to a derivative with respect to ξ within an interval,

$\frac{d\vec{x}}{d\xi} = \Delta p \frac{d\vec{x}}{dp}$ and $\frac{d^2\vec{x}}{d\xi^2} = (\Delta p)^2 \frac{d^2\vec{x}}{dp^2}$, when using the following table of basis functions. Although the

derivatives with respect to the parameter p may be continuous at the knots, because of the scaling by Δp , the normalized derivatives with respect to ξ need not be. The interpolant for a cubic spline can then be written as a superposition of four basis functions (**Figure 6-6**).

Basis	$\phi(\xi)$	$\phi(0)$	$\phi(1)$	$\phi''(0)$	$\phi''(1)$
1	$1 - \xi$	1	0	0	0
2	ξ	0	1	0	0
3	$-\frac{1}{6}\xi(1-\xi)(2-\xi)$	0	0	1	0
4	$-\frac{1}{6}\xi(1-\xi^2)$	0	0	0	1

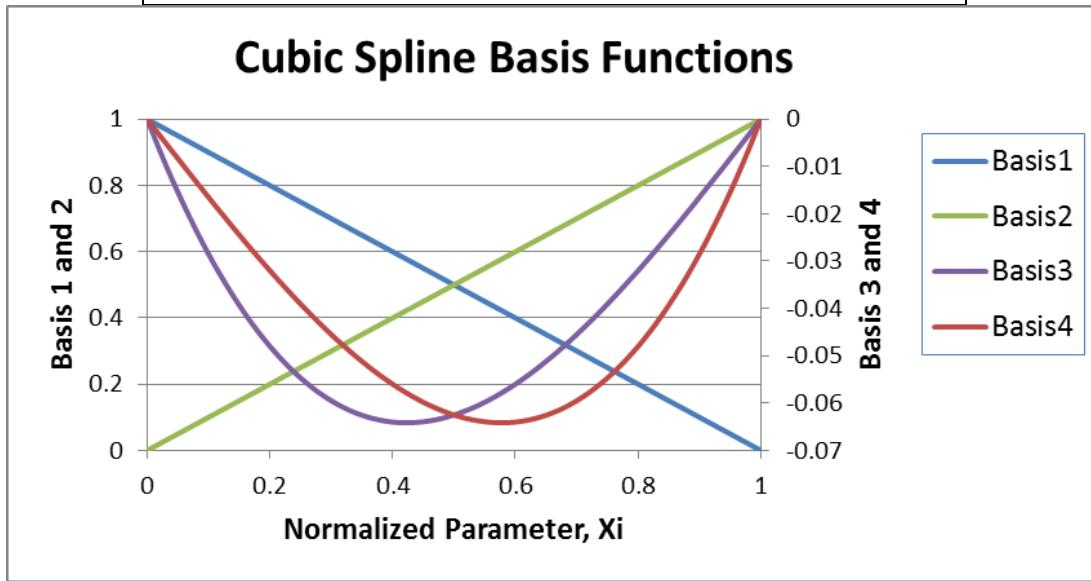


Figure 6-6. Cubic Spline Basis Functions.

Specifically, within an interval we have:

$$\begin{aligned}\vec{x}(\xi) &= \vec{x}_0 \cdot \phi_1(\xi) + \vec{x}_1 \cdot \phi_2(\xi) + \left(\frac{d^2\vec{x}}{d\xi^2} \right)_0 \cdot \phi_3(\xi) + \left(\frac{d^2\vec{x}}{d\xi^2} \right)_1 \cdot \phi_4(\xi) \\ &= \vec{x}_0 \cdot (1-\xi) + \vec{x}_1 \cdot \xi - \frac{1}{6}(\xi)(1-\xi) \cdot \left\{ \left(\frac{d^2\vec{x}}{d\xi^2} \right)_0 \cdot (2-\xi) + \left(\frac{d^2\vec{x}}{d\xi^2} \right)_1 \cdot (1+\xi) \right\}\end{aligned}$$

For instance, a function that takes on the values of a and b at the interval knots but with vanishing second derivatives would be given by $a \cdot (1-\xi) + b \cdot \xi = a + (b-a) \cdot \xi$. With non-zero second derivatives, a cubic equation would arise. Any cubic spline may be expressed in this fashion, and all cubic splines rely upon the specification of the control points at the knots. However, different cubic spline implementations arise depending upon the specification of the derivatives at the knots.

6.3.2 Tangential Cubic Splines

For tangential cubic splines, the control points and the tangential derivatives, $\frac{d\vec{x}}{dp}$, are specified at each

knot. The function is everywhere continuous and differentiable, but the second derivatives may be discontinuous at the knots. Using the cubic functional form within an interval, these four constraints may be used to determine the second derivatives at the interval edges.

$$\begin{aligned}\left(\frac{d^2\vec{x}}{d\xi^2}\right)_0 &= (6\Delta p) \cdot \left\{ \frac{(\vec{x}_1 - \vec{x}_0)}{\Delta p} - \frac{2}{3} \left(\frac{d\vec{x}}{dp} \right)_0 - \frac{1}{3} \left(\frac{d\vec{x}}{dp} \right)_1 \right\} \\ \left(\frac{d^2\vec{x}}{d\xi^2}\right)_1 &= (-6\Delta p) \cdot \left\{ \frac{(\vec{x}_1 - \vec{x}_0)}{\Delta p} - \frac{1}{3} \left(\frac{d\vec{x}}{dp} \right)_0 - \frac{2}{3} \left(\frac{d\vec{x}}{dp} \right)_1 \right\}\end{aligned}$$

Hence:

$$\begin{aligned}& \left\{ \left(\frac{d^2\vec{x}}{d\xi^2} \right)_0 \cdot (2 - \xi) + \left(\frac{d^2\vec{x}}{d\xi^2} \right)_1 \cdot (1 + \xi) \right\} \\ &= 6 \cdot (1 - \xi) \cdot \left\{ (\vec{x}_1 - \vec{x}_0) - \Delta p \cdot \left(\frac{d\vec{x}}{dp} \right)_0 \right\} - 6 \cdot (\xi) \cdot \left\{ (\vec{x}_1 - \vec{x}_0) - \Delta p \cdot \left(\frac{d\vec{x}}{dp} \right)_1 \right\}\end{aligned}$$

Notice that when written in this form, that zero values for Δp are permitted. This type of spline appears in many interactive graphics packages to provide a smooth interpolant with specified slopes at the knots.

6.3.3 Natural Cubic Splines

For a natural cubic spline, only the control points are specified at each knot. The function is everywhere continuous and differentiable, with continuous second derivatives at the knots. The requirement of a continuous second derivative is sufficient to uniquely define the derivatives. For the edge knots, by definition, the second derivative vanishes. For the internal knots, the second derivative is determined from a quadratic fit through each triple of knots, as shown in **Figure 6-7**.

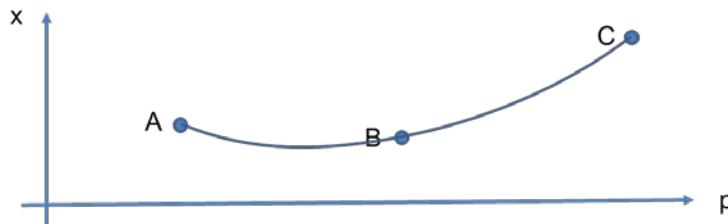


Figure 6-7. Second derivative at a knot.

The quadratic functional form that is used in the derivation, and the first and second derivatives at the center knot are as follows.

$$\vec{x}(p) = \vec{x}_B + (\vec{x}_A - \vec{x}_B) \frac{(p_B - p)(p_C - p)}{(p_B - p_A)(p_C - p_A)} + (\vec{x}_C - \vec{x}_B) \frac{(p - p_A)(p - p_B)}{(p_C - p_A)(p_C - p_B)}$$

$$\frac{d\vec{x}}{dp}(p_B) = \frac{(\vec{x}_B - \vec{x}_A)}{(p_B - p_A)} \cdot \frac{(p_C - p_B)}{(p_C - p_A)} + \frac{(\vec{x}_C - \vec{x}_B)}{(p_C - p_B)} \cdot \frac{(p_B - p_A)}{(p_C - p_A)}$$

$$\frac{d^2\vec{x}}{dp^2}(p_B) = \frac{2}{(p_C - p_B)} \left(\frac{(\vec{x}_C - \vec{x}_B)}{(p_C - p_B)} - \frac{(\vec{x}_B - \vec{x}_A)}{(p_B - p_A)} \right)$$

The quadratic fit shows that the best estimate of the slope at the intermediate knot is a weighted average of forward and backward differences, which reduces to a central difference for evenly spaced data. The last of these equations is used in the calculation of the interpolant for the natural cubic spline. Again, care must be taken in their normalization before using the table of basis functions.

6.3.4 Z Linear Cubic Splines

There is one interesting variation on the cubic spline in use in our domain: the so-called *Z Linear Cubic Spline*. As shown in **Figure 6-8**, the resulting interpolant is bounded in Z where a cubic spline need not be.

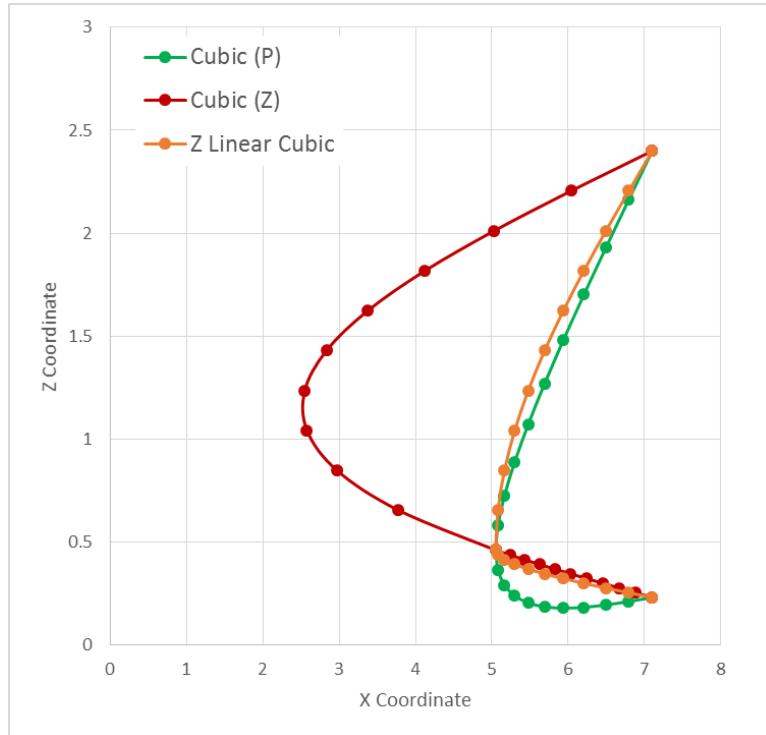


Figure 6-8. Z Linear Cubic and cubic splines over two intervals.

This variation mixes a linear spline interpolant in Z with a natural cubic spline interpolant in (X,Y). It always appears as a natural cubic spline. The figure also includes two variations on the cubic splines. The green curve corresponds to the standard natural cubic spline interpolation from Numerical Recipes. The red curve is a cubic spline as a function of the Z coordinate. The Z Linear Cubic Spline is a function of the parameter P, and has continuous derivatives at the knots when expressed as a function of P, but not as a function of Z.

6.3.5 Minimum-Curvature Splines

The minimum-curvature interpolation method, also known as the circular arc method, is used to describe the shape of a wellbore trajectory interpolated between stations (knots) on which the dip and azimuth are specified. The current exposition largely follows that of Taylor and Mason (1972), Zaremba (1973) and the more recent discussions by Sarawyn and Thorogood (2005), suitably generalized beyond the wellbore trajectory domain. There are a number of differences between this mathematical exposition and industry practice:

- The current calculation is performed in the local 3D projected CRS, and ignores curvature effects which may be important, especially for extended reach wells.
- The current calculation does not assume that the parameterization of the spline is with respect to the measured depth along the trajectory. As a consequence, just as with all of the other RESQML splines, the tangent vectors (derivative of position with respect to parameter) must be specified in the local 3D CRS, and need not be unit vectors specified solely by dip and azimuth.
 - In the context of a well trajectory representation, the local 3D CRS is usually not rotated with respect to the projected CRS. As a consequence, the local Y axis is aligned with the projected North direction.
- This generalization allows an interpolant to be defined with mixed units of measure. However, no unit conversions are implied so care must be taken in such cases.

The algorithm is posed as follows. Given an initial station for which we know position, measured depth (arc length), and tangent vector, and a next station for which we know measured depth and tangent vector, what is the position of the next station based upon a minimum-curvature interpolant? Translating from station to spline knot, from measured depth to parameter values, and from dip and azimuth angles to tangent vectors, we have the following expression for the minimum-curvature trajectory.

$$\vec{x}(\xi) = \vec{x}_0 + \Delta p \cdot \left\{ \left(\frac{d\vec{x}}{dp} \right)_0 \frac{(1 - \cos(\xi\psi))}{\psi \sin \psi} + \left(\frac{d\vec{x}}{dp} \right)_1 \frac{(\cos((1 - \xi)\psi) - \cos \psi)}{\psi \sin \psi} \right\}$$

Here, ψ is defined by $\cos \psi = \left(\frac{d\vec{x}}{dp} \right)_0 \bullet \left(\frac{d\vec{x}}{dp} \right)_1 / \left\| \left(\frac{d\vec{x}}{dp} \right)_0 \right\| \left\| \left(\frac{d\vec{x}}{dp} \right)_1 \right\|$. The next position is given by $\xi = 1$:

$$\vec{x}_1 = \vec{x}_0 + \Delta p \frac{(1 - \cos \psi)}{\psi \sin \psi} \left\{ \left(\frac{d\vec{x}}{dp} \right)_0 + \left(\frac{d\vec{x}}{dp} \right)_1 \right\}.$$

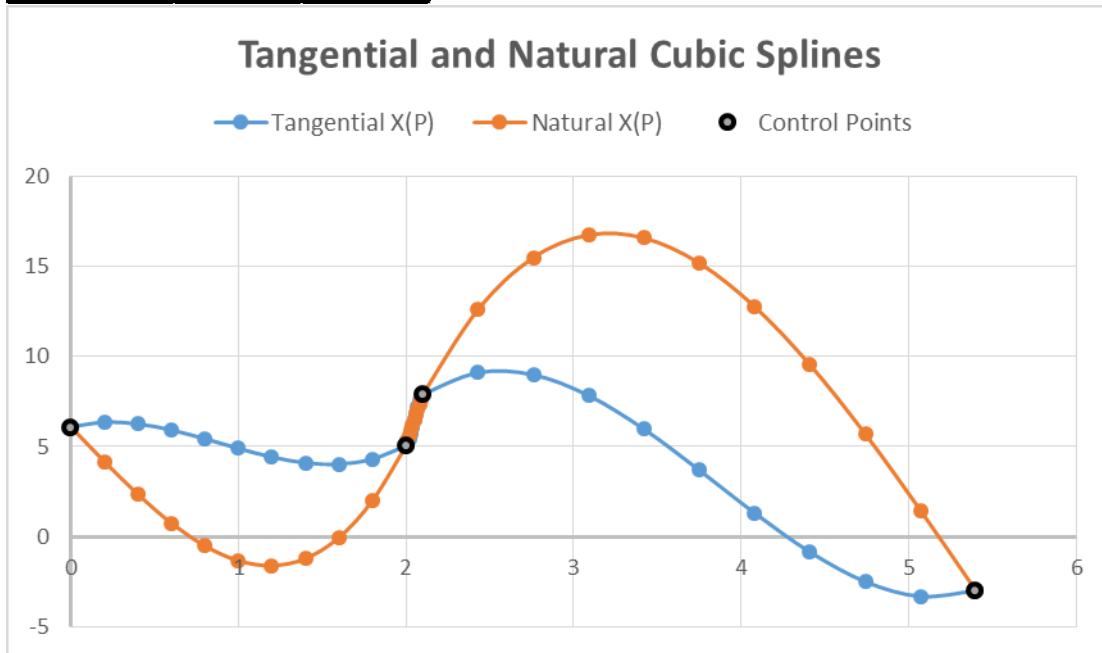
If the two tangents vectors are close to parallel then ψ will be close to 0, and some care has to be taken when evaluating these expressions. In that limit:

$$\frac{(1 - \cos(\xi\psi))}{\psi \sin \psi} \xrightarrow{\psi \approx 0} \frac{\xi^2}{2} \quad \frac{(\cos((1 - \xi)\psi) - \cos \psi)}{\psi \sin \psi} \xrightarrow{\psi \approx 0} \frac{1 - (1 - \xi)^2}{2} \quad \frac{(1 - \cos \psi)}{\psi \sin \psi} \xrightarrow{\psi \approx 0} \frac{1}{2}$$

Additional higher order terms in ψ may also be included to improve upon this approximation. Again, in the context of a wellbore trajectory representation, p is interpreted as the measured depth, Δp as the incremental measured depth, and the tangent vectors will be unit vectors.

6.3.6 Example: Tangential and Natural Cubic Splines

INPUT		
P	X	dX/dP
0	6.1	2.3
2	5.05	5
2.1	7.9	6
5.4	-3	3



Left ($X_i=0$)			Natural		Right ($X_i=1$)			Natural							Tangential	Natural	
P	X	dX/dXi_2	d2X/dXi2		P	X	dX/dXi_2	d2X/dXi2		X_i	P	BASIS1	BASIS2	BASIS3	BASIS4	X(P)	X(P)
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0	0	1	0	0	0	6.1	6.1
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.1	0.2	0.9	0.1	-0.0285	-0.0165	6.3532	4.170571
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.2	0.4	0.8	0.2	-0.048	-0.032	6.2596	2.351714
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.3	0.6	0.7	0.3	-0.0595	-0.0455	5.9194	0.754
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.4	0.8	0.6	0.4	-0.064	-0.056	5.4328	-0.512
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.5	1	0.5	0.5	-0.0625	-0.0625	4.9	-1.33571
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.6	1.2	0.4	0.6	-0.056	-0.064	4.4212	-1.60657
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.7	1.4	0.3	0.7	-0.0455	-0.0595	4.0966	-1.214
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.8	1.6	0.2	0.8	-0.032	-0.048	4.0264	-0.04743
0	6.1	-44.7	0		2	5.05	55.5	110.5714		0.9	1.8	0.1	0.9	-0.0165	-0.0285	4.3108	2.003714
0	6.1	-44.7	0		2	5.05	55.5	110.5714		1	2	0	1	0	0	5.05	5.05
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0	2	1	0	0	0	5.05	5.05
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.1	2.01	0.9	0.1	-0.0285	-0.0165	5.1649	5.330209
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.2	2.02	0.8	0.2	-0.048	-0.032	5.3912	5.612718
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.3	2.03	0.7	0.3	-0.0595	-0.0455	5.7013	5.897064
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.4	2.04	0.6	0.4	-0.064	-0.056	6.0676	6.182785
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.5	2.05	0.5	0.5	-0.0625	-0.0625	6.4625	6.469416
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.6	2.06	0.4	0.6	-0.056	-0.064	6.8584	6.756493
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.7	2.07	0.3	0.7	-0.0455	-0.0595	7.2277	7.043554
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.8	2.08	0.2	0.8	-0.032	-0.048	7.5428	7.330134
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		0.9	2.09	0.1	0.9	-0.0165	-0.0285	7.7761	7.615771
2	5.05	13.9	0.276429		2.1	7.9	-13.7	-0.18708		1	2.1	0	1	0	0	7.9	7.9
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0	2.1	1	0	0	0	7.9	7.9
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.1	2.43	0.9	0.1	-0.0285	-0.0165	9.1095	12.6162
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.2	2.76	0.8	0.2	-0.048	-0.032	8.984	15.49887
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.3	3.09	0.7	0.3	-0.0595	-0.0455	7.8325	16.75173
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.4	3.42	0.6	0.4	-0.064	-0.056	5.964	16.57849
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.5	3.75	0.5	0.5	-0.0625	-0.0625	3.6875	15.1829
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.6	4.08	0.4	0.6	-0.056	-0.064	1.312	12.76868
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.7	4.41	0.3	0.7	-0.0455	-0.0595	-0.8535	9.539554
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.8	4.74	0.2	0.8	-0.032	-0.048	-2.5	5.699247
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		0.9	5.07	0.1	0.9	-0.0165	-0.0285	-3.3185	1.451487
2.1	7.9	-164.4	-203.726		5.4	-3	144.6	0		1	5.4	0	1	0	0	-3	-3

Figure 6-9. Example tangential and natural cubic splines over three intervals.

6.4 References

Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P. (1992). Numerical Recipes, Cambridge University Press

Sawaryn, S. J., & Thorogood, J. L. (2005). A Compendium of Directional Calculations Based on the Minimum Curvature Method. Society of Petroleum Engineers. doi: <http://dx.doi.org/10.2118/84246-PA>

Taylor, H. L., & Mason, M. C. (1972). A Systematic Approach to Well Surveying Calculations. Society of Petroleum Engineers. <http://dx.doi.org/10.2118/3362-PA>

Zaremba, W. A. (1973). Directional Survey by the Circular Arc Method. Society of Petroleum Engineers. doi: <http://dx.doi.org/10.2118/3664-PA>

7 Properties

This chapter provides an overview of how properties are attached to representations, and in one particular case, how geometry is attached. **Figure 7-1** below shows the Properties class diagram.

NOTE: Many of these property-related data objects first developed by the RESQML SIG have now been moved to Energistics *common* (CommonTypes) so they are available for all MLs. For more information, see the *Energistics common Technical Reference Guide* and UML model.

7.1 Overview of How it Works

A **property** is a group of values (array of values) at different locations in the model. Inside a single property, each individual value is attached to a single type of element in the representation—either its topological elements, such as nodes or cells, or bigger elements such as the entire representation or large parts of it, through subrepresentations. For more information, see Chapter 5 (page 48).

Each property is also associated with a single **property kind**, which provides global semantics about the meaning of these values (for example, porosity, permeability, saturation, etc.). Properties can also be associated with **facets**, which provide additional context for the values. For example, RESQML has a facet for condition of acquisition, which indicates if a temperature or pressure value is a surface reading or a reservoir reading (for more information, see Section 7.2.3 (page 70)).

It is very common in subsurface/earth modeling workflows to follow the evolution of some properties through time or to consider them together as a group of properties. Each property inside the group can be attached to the same or different representations and the same or different property kinds. To capture this information, RESQML provides the notion of:

- time series (Section 7.5 (page 72))
- property sets (Section 7.4 (page 71))

Each continuous property is associated with a unit of measure (UOM) providing the context for individual values. However, a property's UOM must be compatible with the UOM of its corresponding property kind (see 7.2.1 (page 69)). Each property kind should be derived from a given **quantity class** (in the Energistics common schema; see the *Energistics common Technical Reference Guide*).

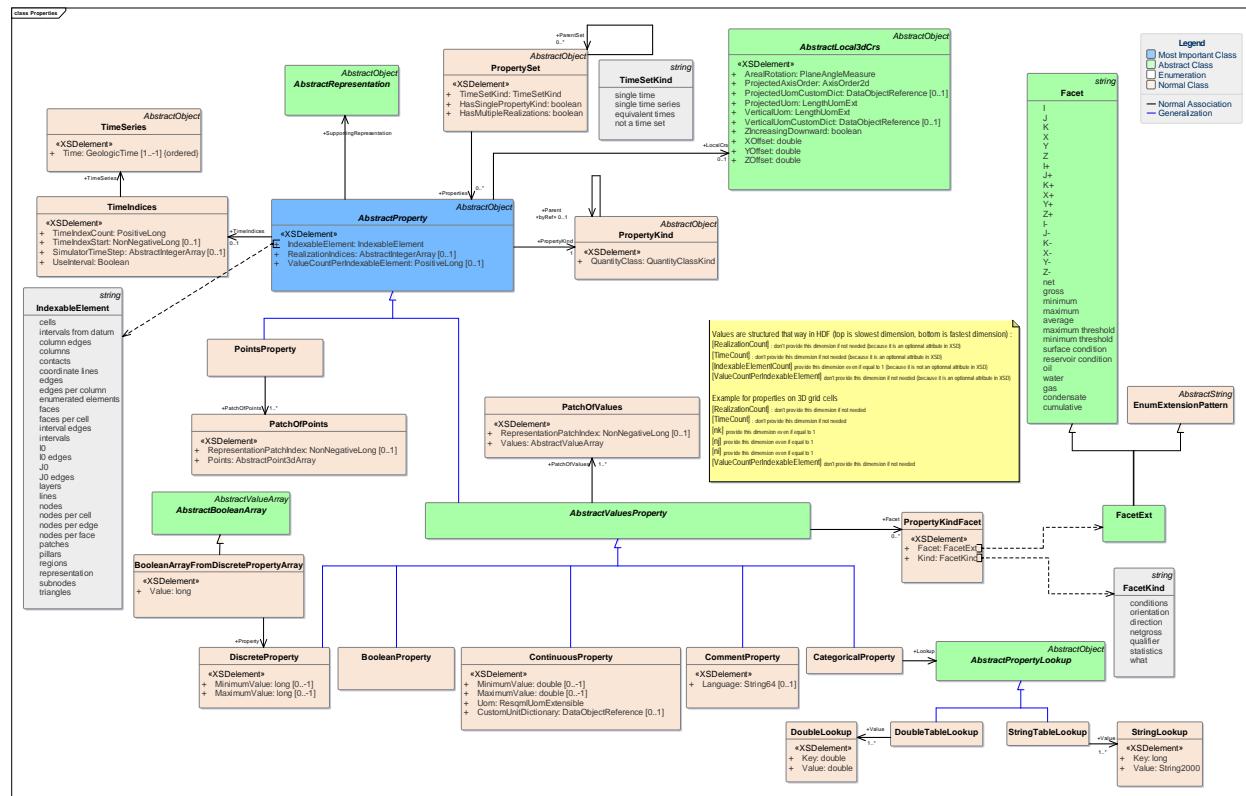


Figure 7-1. Properties class diagram.

7.2 Property

In RESQML, all subsurface or surface values tied to specific topological locations (or indexable elements) in a model are carried by any concrete data object that derives from abstract property (contained in the properties package) attached to one or more representations. A representation can have several properties and provides the indexable elements for these lists of values (for more information on indexable elements, see Section 5.2 (page 49)).

In some cases, geometric information also must be "attached" to a specific location in a model, but the treatment of geometric points is almost identical to that of property values. For more information, see Section 7.2.4 (page 71).

This section provides an overview of how properties and geometric information are attached to RESQML models.

The value locations are based on three elements:

- The representation supporting the property values.
 - The type of elements on which the values are stored.
 - One index that uniquely specifies a given element. For a given representation type, each element type is associated with an indexing scheme that allows you to identify any particular element in the representation. The indexing scheme provides a range (0... number of elements) that defines the order inside an array (for more information, see Section 5.2.2 (page 50)).
 - When local values are scalar, the array is 1D (in this case, count = 1).
 - Else, the array is 2D and (1) the slowest (first) array dimension is the number of elements in the representation, while (2) the fastest (last) dimension is the number of values per element. The list

of element types available is given as a finite list, but this list may be restricted according to the type of representation.

The storage of the value array can be fully explicit inside an HDF5 file, or implicit. Implicit storage can use constant or linear equations based on indexing. The list of values is separated in a patch of values for each patch of the representation. The patch of values contains an index which directs the mapping to the corresponding representation patch.

The stored values are either of type double, integer, character, Boolean or an index that maps to either a string or a table. Each data type corresponds to a specific XML type (**Table 7**) based on the abstract property value type.

Table 6—XML Types Corresponding to Data Types

Type	Corresponding Data Type/Description
BooleanProperty	Contains Boolean values. Used to capture a choice between 2 and only 2 possible values/states for each indexable element of a data object, for example, identifying active indexable elements. A zero value means “false”, everything else means “true”.
ContinuousProperty	Contains double values; most common type of property used for storing rock or fluid attributes. Stores min and max values so that the value range can be known before accessing all values. Also contains a unit of measure that can be different from the unit of measure of their property type, but must be convertible into this unit.
DiscreteProperty	Contains discrete integer values; typically used to store any type of index. Stores min and max so that the value range can be known before accessing all values.
CategoricalProperty	Contains discrete integer values. However these index values are just a proxy for the most complex types, such as string or table. Associated with a lookup structure: String lookup associates indices to a string in a table of strings. Example of use: storage of facies properties, where a facies index is associated with a facies name. Table lookup associates indices to a cell inside a table of values. Example of use: storage of indices associated with empirical tables, such as PVT table.
CommentProperty	Contains strings of character. Used to capture comments or annotations associated with a given element type in a data object, for example, including comments on specific location on a well path. The language (e.g., English, French, etc.) used in the comment is included in property values.

7.2.1 Property Kinds

Property kinds (**Figure 7-2**) carry the semantics of the list of values. For example, they are used to identify if the values represent porosity, length, stress tensor, etc.

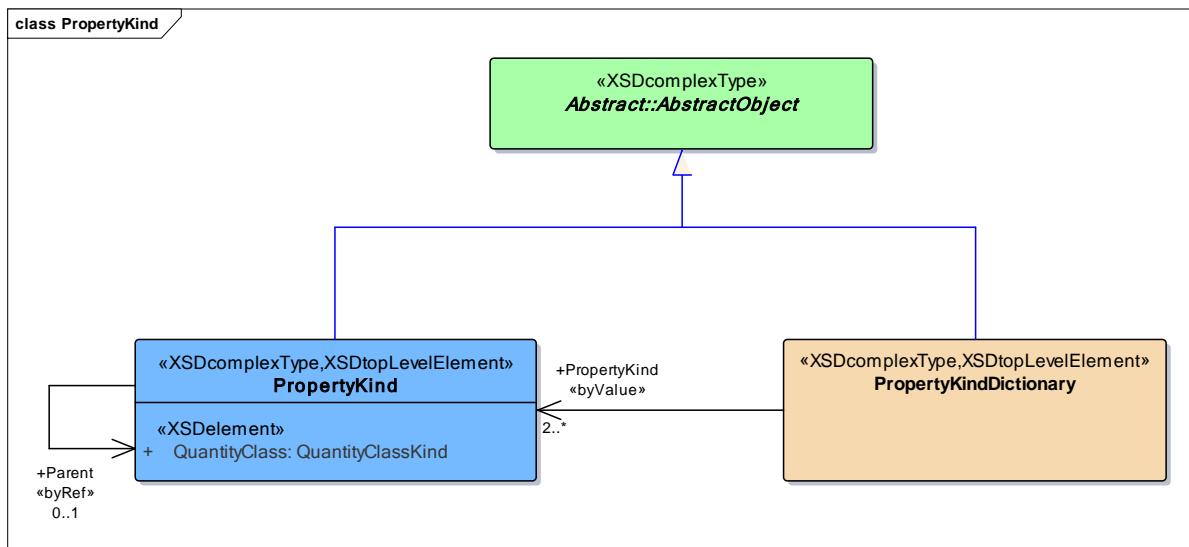


Figure 7-2. Property kind class diagram.

RESQML delivers RESQML "well-known" kinds by means of a property kind dictionary (see CommonTypes in Energistics common). This dictionary represents a finite list of well-known kinds (for example, length, volume, and permeability) which must be used as often as possible. This enumeration is based on the quantity class in the Energistics common package, which also constrains the list of allowed units of measure for the property values (see 7.2.4, page 71).

7.2.2 Extending the Property Kinds List

During a RESQML-enabled data exchange, if the writer needs to exchange a property kind that is not present in this list of well-known kinds, then he can create his own company- or user-specific kind to extend the list of available property kinds. In this context, the following conventions MUST be observed:

- The user-specified kind must always be based on another property kind (association "parent" in Figure 7-2), well-known or not (e.g., a custom kind may be based on another custom kind, which is ultimately based on a well-known kind).
- To identify the company or group that defines the property kind, the "originator" located in the "Citation" group of the kind must be used. For example, mycompany.com:myproduct
- A reference quantity class (Energistics common package) must be provided and requires the associated property value unit to be compatible through conversion.
- All of this information must be explicitly defined in the files being transferred.

7.2.3 Facets

In addition to property kinds to define semantics, a facet is a mechanism that gives some context about the nature of the property. Each property kind can be associated with as many property kind facets as needed.

NOTE: Recommended best practice is to use facets instead of creating a specialized local property kind.

Facets can represent any of the following:

- *Conditions* of acquisition, which, for example, allows distinguishing between surface conditions and reservoir conditions.
- *Side* indicates on which side of a surface the property applies, for example, it can indicate plus or minus.

- *Direction*, which, for example, can indicate vertical or horizontal permeability.
- *NetGross*, which, for example, indicates the difference between net and gross volumes.
- *Statistics* can be used to indicate if values are minimum, maximum, average, etc.
- *What*, which indicates the element that is measured, for example, in a mineral concentration measurement, the mineral name.
- *Qualifier* is used to capture any other context that doesn't fit into one of the above types.

7.2.4 Mapping Simulator Key Words to Property Aliases

To fully identify the kind of properties, flow simulators usually expect enumerated keywords as property titles. Because RESQML does not constrain anything about the title, it is mandatory--even in a flow simulation context--to always describe a property using a property kind and optional facets.

However, to be kind to a software reader, RESQML also allows a property to define one or more aliases, which can be used to specify the expected keywords for a particular simulator that is identified as the authority of the alias(es). These aliases can be used as shortcuts to quickly get the kind of a property in a flow simulation context.

NOTE: The alias is only a shortcut; the RESQML property kind and facets construct contains the mandatory information. For example, you cannot use the same alias for a different combination of property kind and facets.

7.2.5 Units of Measure

The RESQML-recognized units of measure are listed in the Properties package in the ResqmlUom enumeration. Each property kind by means of its quantity class is associated with a representative UOM (see the "baseForConversion" element) and a list of UOMs. This list of UOMs represents all the allowed UOMs for the property values associated to this property kind. Often, the representative UOM corresponds to SI units, but it is not required. The UOM of a given property is decided at the property value level and it must be consistent with the list of UOMs of its quantity class.

7.3 Geometrical Information Stored as a Property

Sometimes it is necessary to store geometrical information about the elements of a representation, for example, for former locations, height, distance, etc. Specific examples include descriptions such as a depth of the oil-water contact, or throw of a fault from the footwall to the hanging wall.

To use property capabilities for this type of data requires a coordinate system, which is why an optional link between properties and a local coordinate system is available in the data model.

When spatial locations need to be stored, a specific kind of property is used: **points property**, which is based on an array of points instead of an array of values.

7.4 Property Set

Property values are often collected to indicate that they should be considered together as a set, for example, to show they are part of the same simulator output. To provide this type of grouping, RESQML provides a **property set**.

Property sets have these characteristics:

- Sets are not exclusive; a property (group of values) can belong to several sets.
- Sets can be related through a parenting relationship. For example, a parent group representing the entire simulator output can be made up of several smaller groups corresponding to each time step.
- Sets can contain semantic information; for example, you can specify if all the contained property values correspond to a single property kind, or if they correspond to multiple realizations, at similar or equivalent times.

7.5 Time Information: Time Step and Time Series

Properties can be associated with optional time information. Two types of time information are available:

- **Time Step.** Directly related to the time step of a flow simulator, which is an index that is only defined in the context of a specific simulation data set.
- **Time Series.** A series of ordered time stamps (a time produced by software when something significant happens, such as saving a file). A series can itself be the continuation of another parent time series, with which it can overlap. A property value references an entry in a time series through its time index.

If you export the results of a simulation run, all property values representing the same property over time must have the same name and refer to the same time series.

7.5.1 Time Index, Time Series and Geologic Time

RESQML includes several time concepts, which are listed and described here. The time usage for representations within RESQML, e.g., for properties and geometry, uses a time index into a time series (**Figure 7-3**). The use of a time series as a top-level data object allows a RESQML reader to determine all of the data times in use within a data set, without needing to interrogate individual properties or geometries. This approach also simplifies the treatment of properties in reservoir simulation where we may have computed many properties during a simulation run, but all of which share the same time series.

- **Time Series.** Stores an ordered list of times, for example, for time-dependent properties, geometries, or representations. It is used in conjunction with the time index to specify times for RESQML.
 - TimeSeriesParentage. Indicates that a time series has the associated time series as a parent, i.e., that the series continues from the parent time series.
- **Time Index.** Index into a time series. Used to specify time. (Not to be confused with time step.)
- **Geologic Time.** XML dateTime, with an optional year offset to capture very long time intervals.

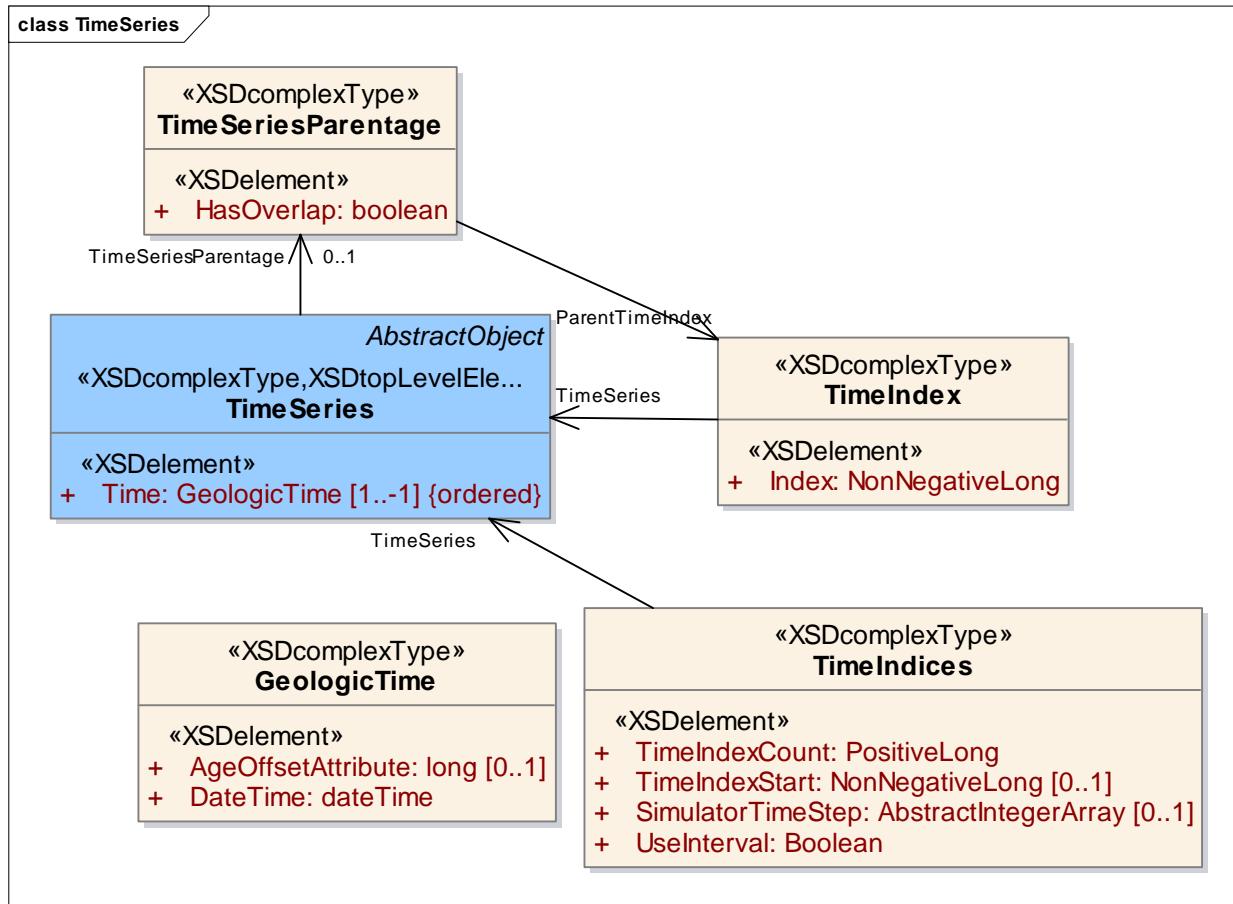


Figure 7-3. RESQML property time treatment.

7.5.1.1 Examples

Figure 7-4 shows an example of how the time series is used to support grouping of properties at identical or related times.

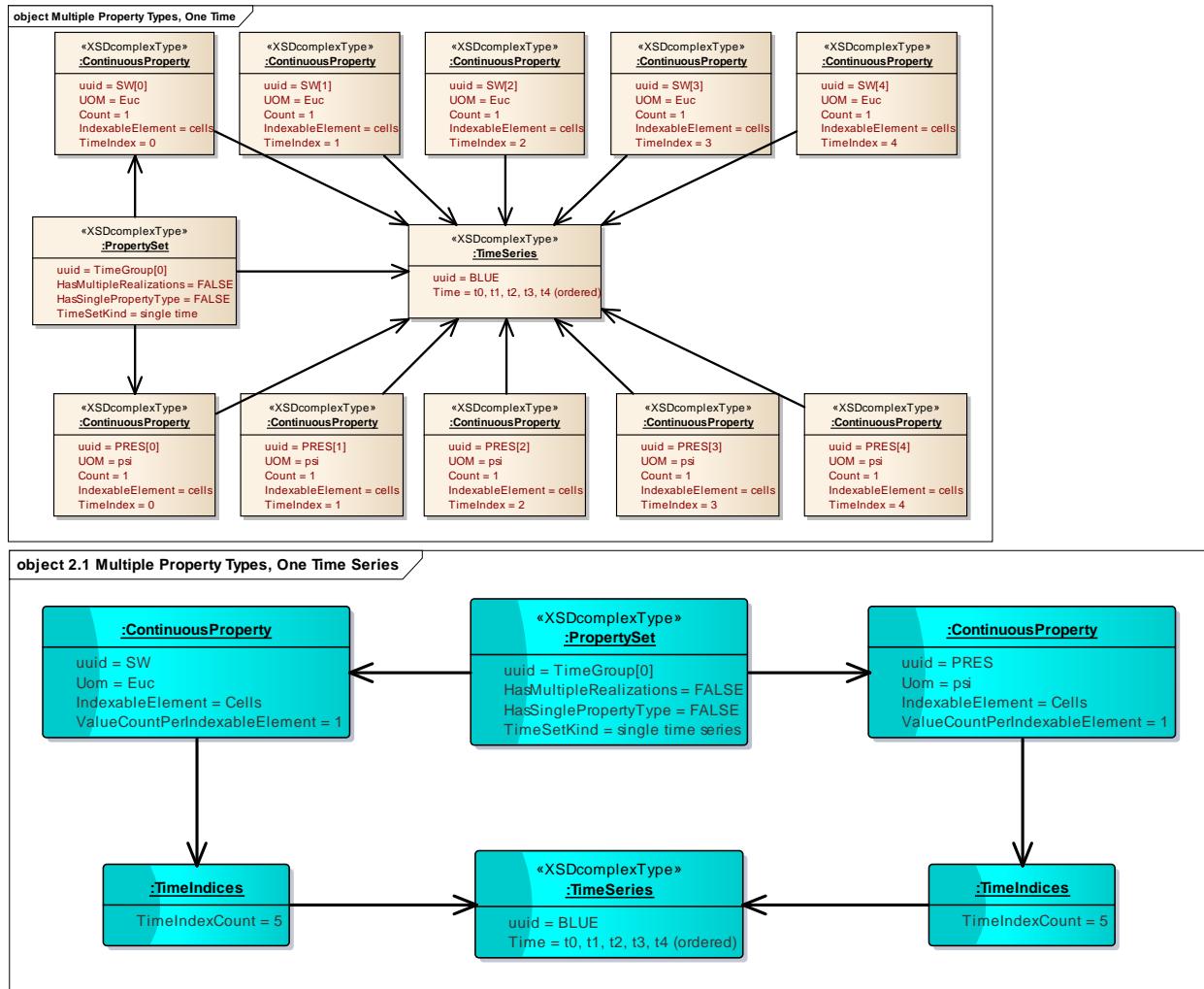


Figure 7-4. RESQML property set time group examples.

In the first example, a property set is constructed at one time from two properties. In the second example, the property set is constructed from one property type at multiple times. In both of these examples, time is referenced into the time series object. The use of time series parentage allows one set of times to be appended to another, e.g., to support reservoir simulation restarts for to explore a variety of reservoir development scenarios.

Geologic time is used in features and interpretations to specify time as an XML dateTime, with an optional year offset to capture very long time intervals (**Figure 7-5**).

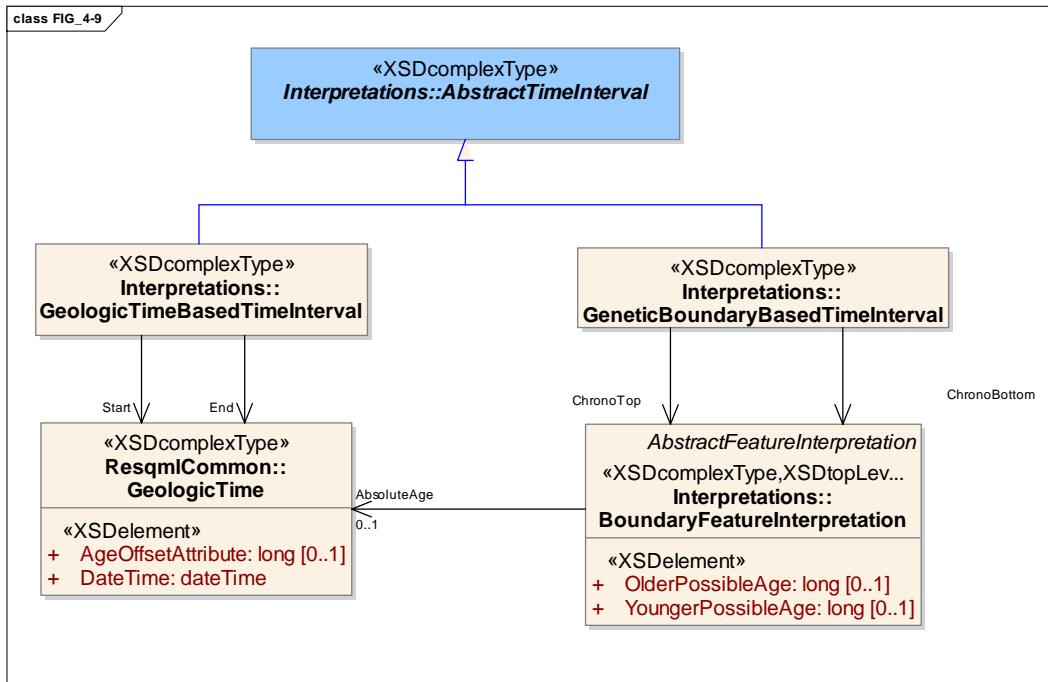


Figure 7-5. Geologic time usage.

7.6 Property Series

The property series data object makes it possible in RESQML to capture the evolution of property values through time and/or multiple realizations of values during stochastic processes.

7.6.1 Use Cases

In RESQML you can associate a single scalar value or a vector of values to a location in space. However, some use cases require capturing the evolution of these values through time and/or multiple realizations of these values during stochastic processes.

Examples include:

- Capturing the oil production for a well interval through time.
- Transferring the results of a sequential Gaussian simulation or other geostatistical results.

In these cases, instead of providing a single value or a 1D array for each location, we need to provide one of the following:

- One value for each location and each time.
- One value per location per realization index.
- One value per time per realization, which is the most advanced case of time-dependent stochastic processes.

In RESQML, this data is added by extending the property model to include property series. The dimensionality of the property series depends on the initial format of the property data:

- If the values were initially represented by an array, time and realization add one or two axis dimensions to this array.
- If the values were scalar values, they each become a 1D or 2D array.

The property series inherits from the abstract values property, which itself inherits from the abstract property. The abstract property includes optional elements that can be used to associate a single time, a

single time step, or a single realization index with a property. If the property series is being used to store values for multiple times, multiple time steps, or multiple realizations, then these “single” optional elements should not be used. If they are provided, then they should be ignored.

7.6.2 Description

Figure 7-3 shows the RESQML properties model with the property series.

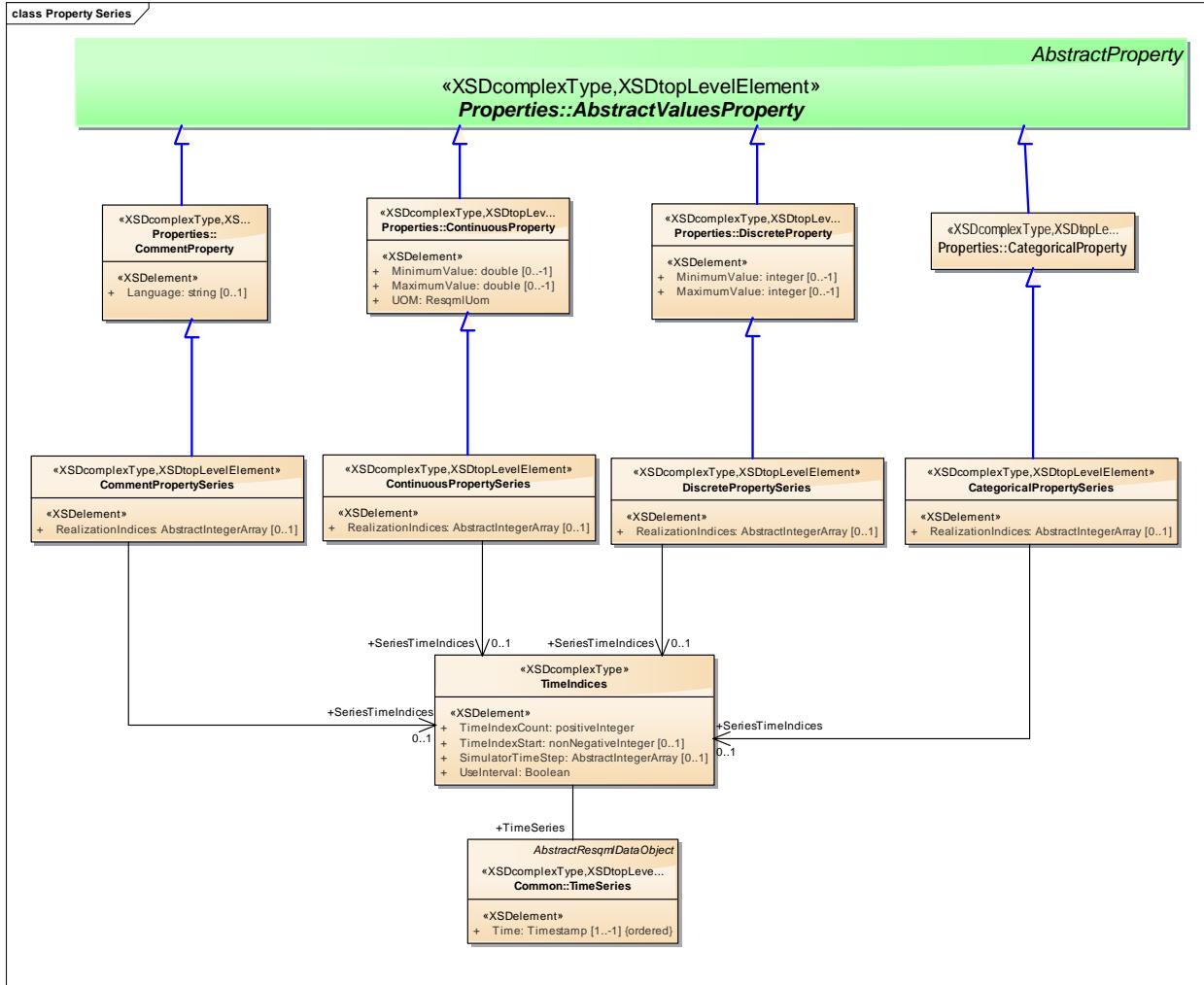


Figure 7-6. UML model of properties with property series.

The property series types are used to capture the two additional dimensions while extending property types. Each property type (continuous, discrete, categorical, comment) has its own property series type.

All property series types optionally contain:

- **An array of realization indices, which defines the realization dimension.** In stochastic simulations, a realization result or a group of results is often associated with one single index, which often reflects their order in the sequence in which they were created. The realization indices are capturing all the indices used to define the realization’s extra axis. In the trivial case—where all results are provided—an integer lattice array can be used.
- **A link to time indices, which defines the time dimension.** The first element of the time indices indicates the number of time entries to consider. If the Boolean flag named “use interval” is true, then

the actual number is this count minus one. The actual time values are provided by a link to a time series and a starting index.

7.6.3 Indexing

Because the values in property series are dependent on some additional axes, it is important to define the influence of the new axes on the value indexing and the place of these new dimensions in the HDF dimension orders.

For property series (and only for property series), values are structured this way in HDF (top is the slowest dimension, bottom is the fastest dimension). The property model results in some attributes that are optional and some that are required in the schema (XSD):

- **RealizationCount:** Optional, don't provide this dimension if the property series is not related to several realizations (no stochastic simulation).
- **TimeCount:** Optional, don't provide this dimension if the property series is not related to several time information.
- **IndexableElementCount:** Required, always provide this dimension even if equal to 1.
- **ValueCountPerIndexableElement:** Optional, don't provide this dimension if the property values are scalar ones. Provide this dimension if the property values are vectorial ones.

Additionally:

- The time count for property series is only related to the series time indices. It is not related to the inherited time index, which should be ignored if also present.
- The realization count for property series is only related to the realization indices. It is not related to the inherited realization index, which should be ignored if also present.

7.6.3.1 Property Series Example

Here are some dimensionality examples for a property series on 3D IJK grid cells (left is the slowest dimension, right is the fastest dimension):

- **Case A:** The property series is a scalar one and is not related to any time or realization information:
[nk][nj][ni]
- **Case B:** The property series is a scalar one and is only related to a time information:
[TimeCount][nk][nj][ni]
- **Case C:** The property series is a scalar one and is only related to a realization information:
[RealizationCount][nk][nj][ni]
- **Case D:** The property series is a scalar one and is related to a time and realization information:
[RealizationCount] [TimeCount][nk][nj][ni]
- **Case E:** The property series is a vectorial one and is related to a time and realization information:
[RealizationCount] [TimeCount][nk][nj][ni][ValueCountPerIndexableElement]

8 Earth Model

This chapter provides an overview of earth model features and earth model interpretations and how they may be used and constructed in RESQML.

This chapter explains some basic concepts and provides some extended examples (see Section 8.3 (page 103) for creating and updating an earth model interpretation—following the feature/interpretation/representation/property knowledge hierarchy defined in Chapter 4—beginning with structural and stratigraphic features, then adding related interpretations and representations, including related geometry.

To ensure an accurate update of the earth model when more information (new data) is acquired during the lifecycle of a prospect, the concept of earth model interpretation is the most important.

The goal is to transfer the information built up from each step of the modeling process, which includes the semantic and numeric information necessary to describe a subsurface area for development and/or reservoir management activities.

8.1 What is a RESQML Earth Model and How Can it Be Used?

The RESQML earth model has these main parts:

- model feature
- earth model interpretation (formally, a *feature-interpretation*, but referred to as *interpretation* for brevity)

8.1.1 Model Feature: Definition and Use

RESQML uses this definition of **model**: An explicit description of the relationships between geologic features, such as rock volume features (e.g., stratigraphic units, geobodies, phase units) and boundary features (e.g., genetic, tectonic, and fluid boundaries).

In RESQML, a model feature is an entry point to all RESQML elements that have been associated together into a RESQML organization. It is through a model (organization) feature that a software reader can access all the information about the raw data acquired, the interpretations of these raw data, and all the organization interpretations built on them.

8.1.1.1 Use of the Model Feature

To be used effectively along a multi-software workflow, it is strongly recommended that, at the beginning of any data transfer, the user declares both:

- a model feature
- a first-associated earth model interpretation

The creation of the model feature and a first interpretation of it drives the user to begin thinking about how to organize information for data exchange and "initializes" the flow of information.

So, the first (and more important) usage of the model feature is to create a UUID reference on which all the results of operations relative to one prospect can be "linked" and retrieved independently of software files, database releases, or proprietary and legacy data stores.

This means that, at the very beginning of a prospect life cycle, it might be useful (though not mandatory) for an asset team to create a model feature, which can be the "container" (organizing basis) for a consistent set of information.

NOTE: The model does not actually "contain" its contents, but is referenced to them using data object references. The individual features of the model can also be exchanged individually, during prospect appraisal and development.

8.1.2 Earth Model Interpretation: Definition and Use

An earth model interpretation is an organized collection of subsurface feature-interpretations, which are integrated in their mutual relationships.

A given earth model feature can have many associated earth model interpretations. (In fact, all features in RESQML may have multiple associated interpretations.)

BUSINESS RULES: In RESQML v2.1, each earth model interpretation consists of one stratigraphic column, one or more structural organization interpretations, and one or several rock fluid organization interpretations.

- For a 4D model that evolves through time, several structural organizations are needed.
- For a model that comprises several reservoirs, several rock fluid organization interpretations are needed.
- Currently, the data model can represent stratigraphic succession all issued from the same paleo-geographic domain and corresponding to one single stratigraphic column.

NOTE: It may be possible in a future version of RESQML to introduce into an earth model several stratigraphic columns corresponding to several different paleo-geographic domains.

- These interpretations must be consistent with one another.

NOTE: Currently, this business rule cannot be enforced in the schema, so the reading software must "trust" that the writing software has followed this rule.

Each of the component feature-interpretations (for example, horizon interpretations, fault interpretations, geologic unit interpretations, and contact interpretations) that make up the organization interpretations may also be associated together, which is explained in Section 8.2 (page 80).

8.1.2.1 Use of Earth Model Interpretations

An earth model interpretation has these main uses; it provides a:

- "Snap shot" at a given time stamp to capture a consistent set of information at a specific step of the geomodeling/simulation business process. The objective is to mark this step or to transfer a complete and consistent set of information from one software package to another after a specific step.
- Way to consistently update, stage by stage, this consistent model (from the snap shot forward) with new data and information about the prospect—and only have to transfer the new or changed data.
- Way to archive one given step or the global history of the prospect creation and its evaluation.

From an IT perspective, an earth model interpretation can be considered an instance of one model feature (**Figure 8–1**).

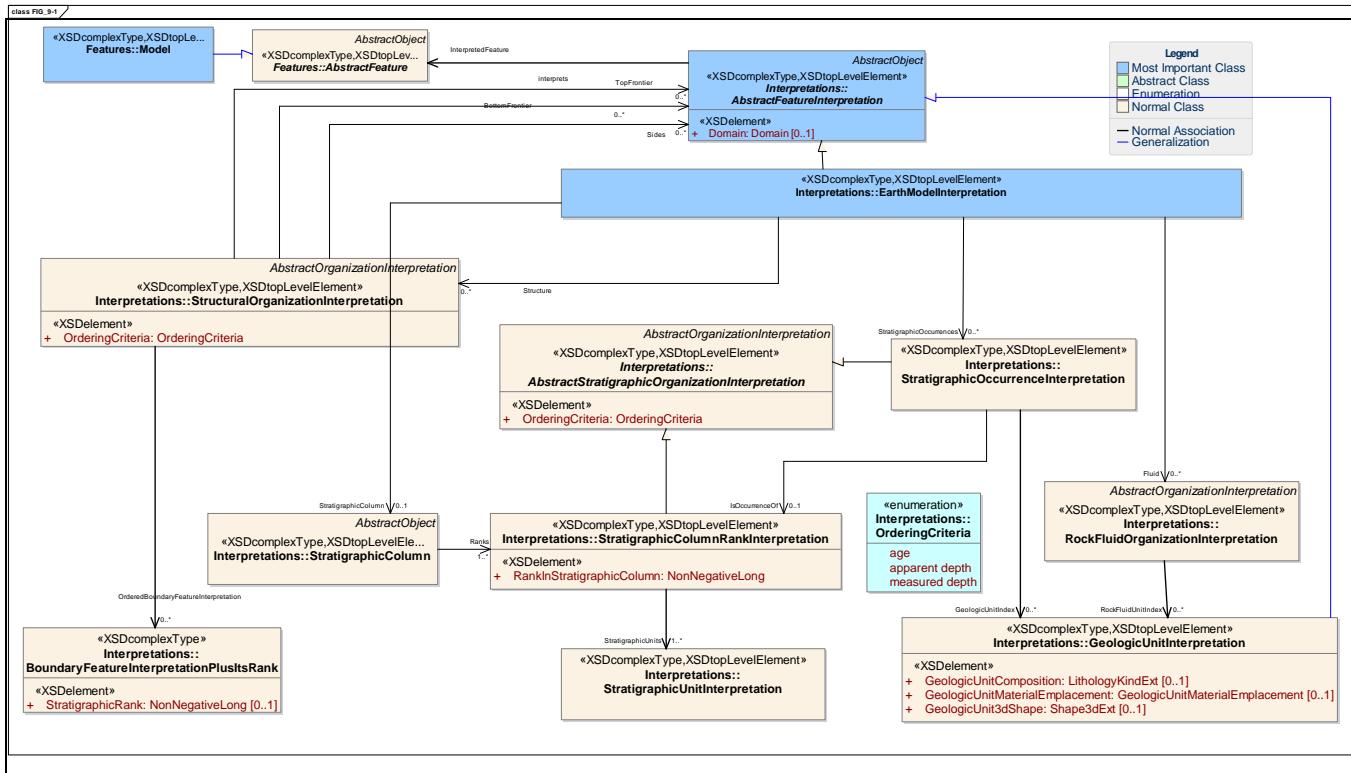


Figure 8–1. Earth model feature and interpretation in the RESQML UML model.

8.2 Features and Components of an Earth Model Interpretation

This section describes the individual geological and technical features and the individual and organization interpretations that can be used to build an earth model interpretation. **Figure 8–2** is a simple diagram showing the objects that make up an earth model interpretation.

First, we define the features, which must exist before any interpretation: individual and model features (see Section 8.2.1). Then we explain how some interpretations of these individual features can be associated into structural, stratigraphic, and rock fluid organization interpretations (see Section 8.2.2), and how the relationships between the individual feature-interpretations are specified by defining the contact interpretations (see Section 8.2.2.2) between the individual feature-interpretation.

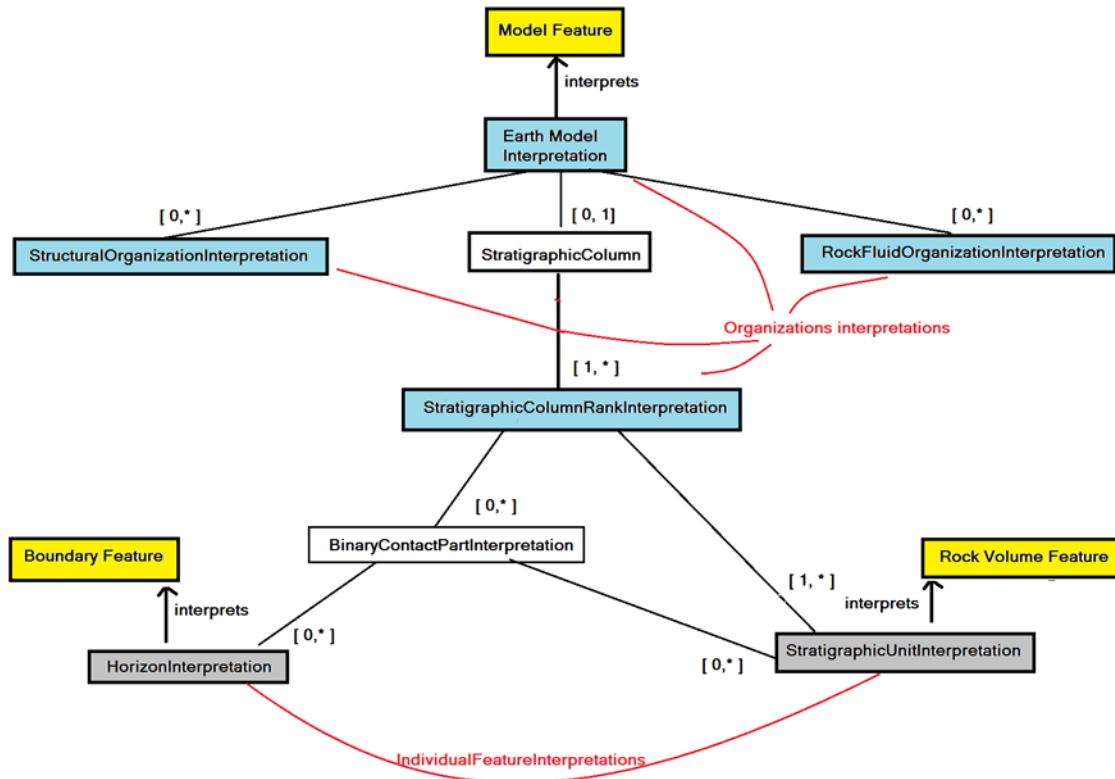


Figure 8–2. Simple diagram showing example features and interpretations used to set up an earth model interpretation and its stratigraphic column.

8.2.1 Features

The abstract feature is the parent class of all the features used for exchanging structural and stratigraphic information (**Figure 8–3**).

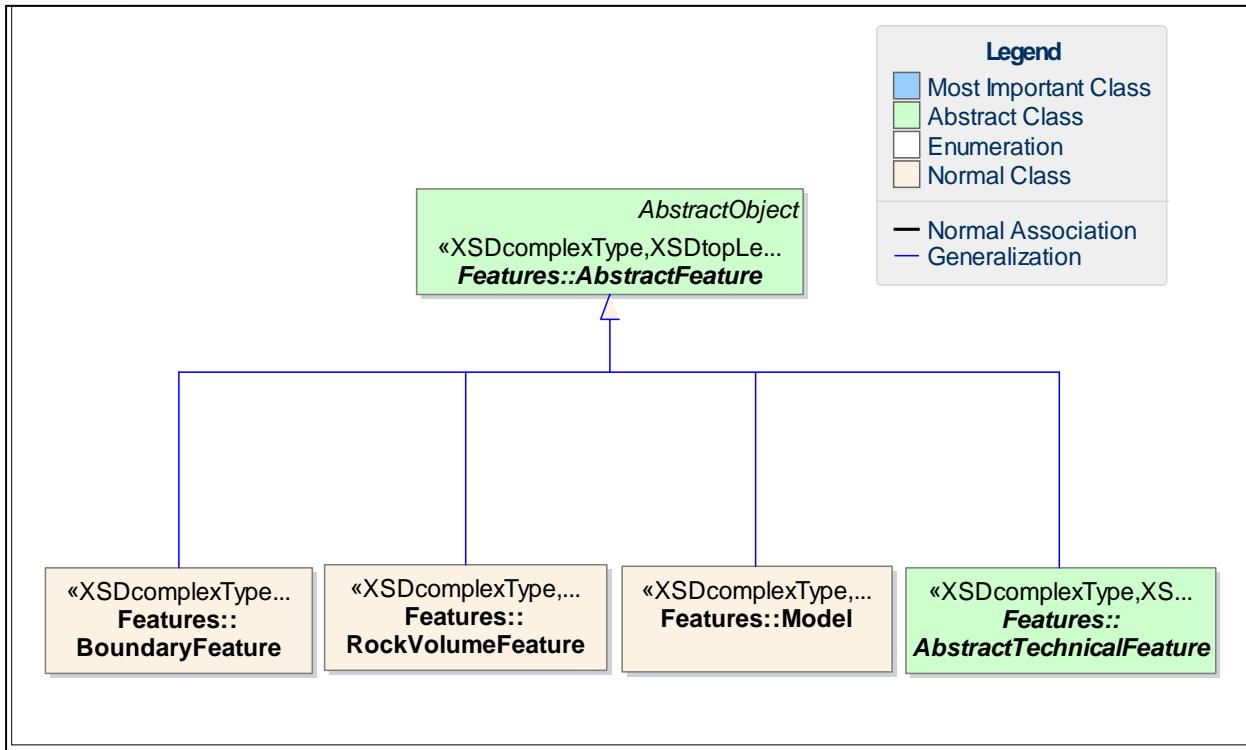


Figure 8–3. Abstract feature and geological and technical features.

RESQML has two main types of features: geological and technical (technical features do not have interpretations). (For definitions of geological and technical features, see Section 3.1.3 Figure 8–3 shows the main organization of the feature class, which includes:

- **These geological features:** (Note: To simplify the data model, an abstract geological feature concept is NOT explicitly expressed in the RESQML hierarchy):
 - **Boundary Feature:** A geological surface object that is meaningful for geologists. Examples of boundary features include: horizons (stratigraphic boundaries), geobody boundaries, faults or fluid boundaries that are present before production.
 - **Rock Volume Feature:** A continuous portion of rock material with definite rock boundaries. It is a volume object. Some of these rock volume features are “static”, while some others are “dynamic”. Reservoir fluids are dynamic because their properties, geometries, and quantities may change over time, during field production.
 - **Model (Organization) Feature:** The explicit description of the relationships between geologic features, such as rock volume features (e.g. stratigraphic units, geobodies, phase unit) and boundary features (e.g., genetic, tectonic, and fluid boundaries). In industry, this concept is commonly referred to as an ‘earth model’.
- **These abstract technical features (Figure 8–4):**
 - **Wellbore features** describe wells in three dimensions and associated WITSML wellbore references. For more information on wells, see Chapter 12.
 - **Streamline features** specify the vector field on which the streamlines are based. Streamlines are commonly used to trace the flow of phases (water/oil/gas/total) based upon their flux at a

specified time. They may also be used for trace components for compositional simulation, e.g., CO₂, or temperatures for thermal simulation. The flux enumeration provides support for the most usual cases with provision for extensions to other fluxes. For more information on streamlines, see Chapter 11.

- **Frontier feature** describes the studied domain used to build the volume of interest for the model area.
- **Seismic features** can define a set of seismic lattices to describe multiple seismic 3D surveys or a set of seismic 2D lines to characterize 2D seismic sections. For more information on seismic, Chapter 13.

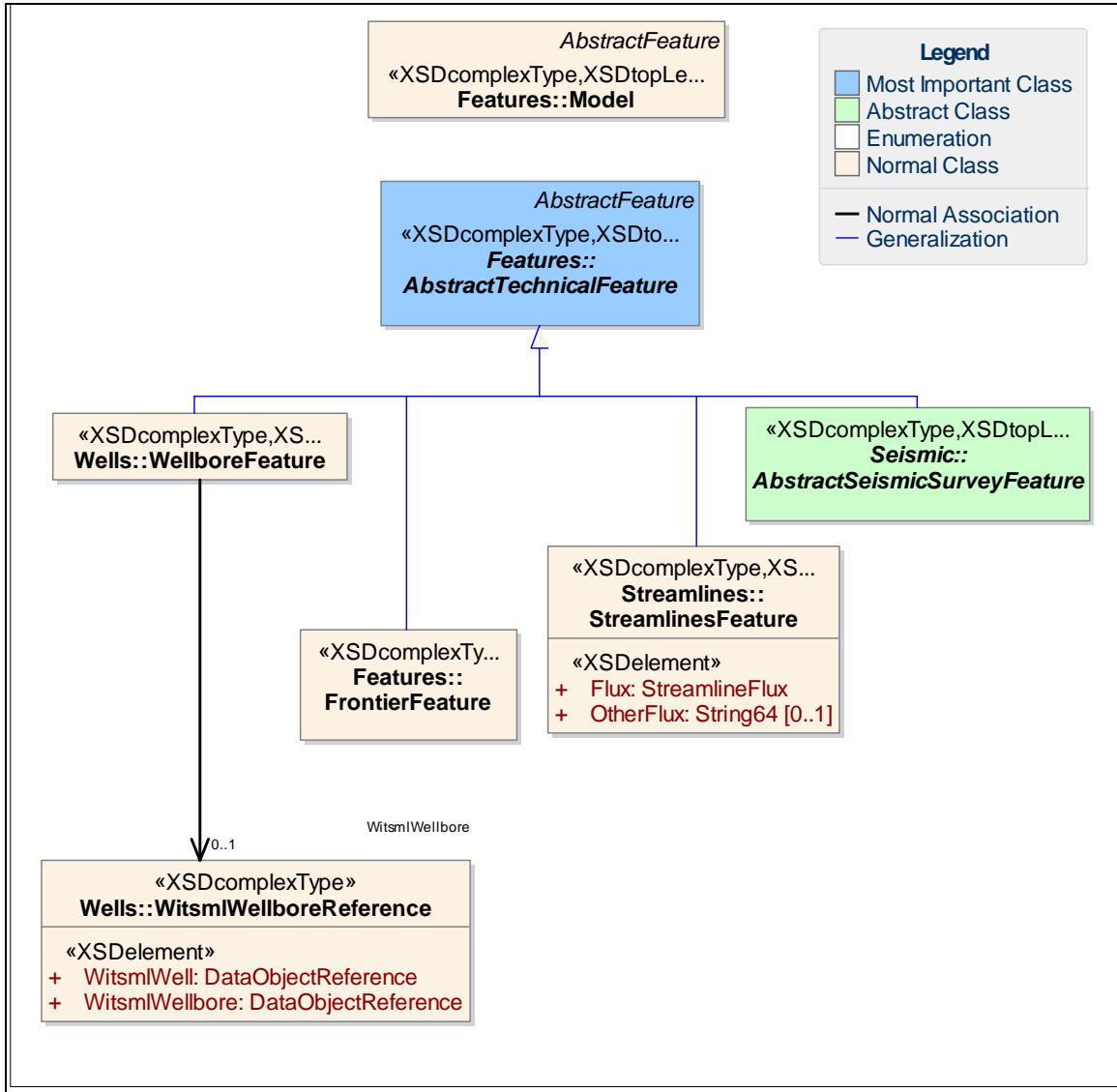


Figure 8–4. Technical features.

In RESQML, all individual features can be assembled into these types of organization features:

- Structural organization (surface)
- Stratigraphic (volume)
- Fluid (volume)

- Earth model (association of the previous organization features in this list)

8.2.2 Structural and Stratigraphic Organization Interpretations

Interpretations are used to both characterize a “natural object” with some geology-related information and to designate a set of different representation types (e.g., for a boundary: 3D point sets, 2D grids, triangulated surfaces) corresponding to an object that was validated by one user at a given time (as described in the citation object attached to the interpretation).

For example, if a user is changing the number or the geometrical location of points in the representation of a previously recorded interpretation, he MUST create a NEW interpretation and, if possible, explain why he performed this operation (i.e., changed the geometrical points).

Conversely, if a user uses the data that exists in a representation linked to an existing interpretation to create a new representation that is essentially unchanged (e.g., from a 3D points set he creates a triangulated surface), he MUST NOT create a new interpretation but must link this new representation to the existing interpretation.

However, individual geological object interpretations are only related to individual geological objects (geological units, horizons, faults); organization interpretations can sometimes integrate some “generic” interpretations of technical data objects (like frontiers or wellbore trajectories).

Most of the information contained as attributes or enumerations in individual interpretations or organization interpretations can help users to understand how the topology and the geometry of the geological objects and organization representations should be built (or should have been built if a representation is already associated with a given interpretation).

An abstract interpretation is the parent class of all specialized interpretations of a linear, surface, or volume of geologic data objects. In an interpretation, we must find all the necessary information needed to build up a realization of a representation of these data objects.

An interpretation must be declared in a given domain (depth, time (for seismic) or mixed), which allows the users to specify which kind of data were used for building the interpretation.

8.2.2.1 Individual Feature-Interpretations

Figure 8–5 is a class diagram of individual feature-interpretations.

8.2.2.1.1 Geologic Feature-Interpretations

RESQML has two categories of individual interpretations of geological features, which include:

- **Boundary interpretation (surface).** A boundary interpretation may be one of these kinds: fault, horizon, or geobody boundary. Each of these has characteristics (enumerations), which may influence how its representation is built (**Figure 8–6**).
- **Geologic unit interpretation (volume),** which is characterized by these 3 attributes:
 1. Its GeologicUnitComposition: valid values include the ones defined in the LithologyKind Enum of Energistics *common* (see the *CTA Overview Guide* or *Energistics common Technical Reference Guide*), which are a description of minerals or accessories that constitute a fractional part of a lithology description and are listed here for easy reference:

Lithology Kinds from Energistics common				
alkali feldspar rhyolite	alkali olivine basalt	amphibolite	andesite	anhydrite
anorthositic rock	anthracite	aplite	arenite	argillaceous
arkose	basalt	basanite	bauxite	bituminous coal
blueschist metamorphic rock	boninite	breccia	carbonate ooze	carbonatite
chalk	chert	clay	claystone	coal
conglomerate	dacite	diabase	diamictite	diorite
dioritoid	doleritic rock	dolomite	dolomitic	eclogite

Lithology Kinds from Energistics common				
exotic alkaline rock	feldspar	feldspathic arenite	fine grained igneous rock	foid dioritoid
foid gabbroid	foid syenitoid	foidite	foiditoid	foidolite
foliated metamorphic rock	fragmental igneous rock	gabbro	gabbroic rock	gabbroid
glauconite	gneiss	granite	granodiorite	granofels
granulite	gravel	greenstone	gumbo	gypsum
halite	hornfels	igneous rock	impact generated material	impure dolomite
impure limestone	intrusive rock (plutonic)	iron rich sedimentary rock	kalsilitic and melilitic rocks	komatiitic rock
latitic rock	lignite	lime boundstone	lime framestone	lime grainstone
lime mudstone	lime packstone	lime wackestone	limestone	marble
marl	metamorphic rock	mica schist	migmatite	monzogabbro
mud	mudstone	mylonitic rock	no description	no sample
ooze	ophiolite	organic bearing mudstone	peat	pegmatite
peridotite	phaneritic igneous rock	phonolite	phonolitoid	phosphate
phosphate rock	phyllite	porphyry	potassium and magnesium salts	pyroclastic breccia
pyroclastic rock	pyroxenite	quartz arenite	quartzite	rhyolite
rock salt	sand	sandstone	sandy	sapropel
schist	serpentinite	shale	siliceous ooze	silt
siltstone	skarn	slate	spilite	syenite
syenitoid	sylvite	tephrite	tephritoid	tholeiitic basalt
tonalite	trachyte	trachytic rock	trachytoid	travertine
tuff	tuffite	ultrabasic	undifferentiated	unknown
wacke				

2. Its mode of emplacement (intrusive/non-intrusive)
3. Its 3D shape (sheet, dyke, dome, mushroom, channel, delta, dune, fan, reef, wedge).

Use of all three of these attributes allows full characterization of geologic units. Examples:

- Sand, non-intrusive, sheet = Sandstone unit
- Clay, intrusive, dyke= mud dyke
- Granite, intrusive, dome = granitic batholith
- RESQML geologic units have these two main categories and the following specific values:
 - **Geobody interpretation**
 - **Stratigraphic unit**, which can have these deposition modes:
 - Proportional between top and bottom
 - Parallel to bottom
 - Parallel to top
 - Parallel to another boundary

It is the relationships between these individual feature-interpretations that define the contact interpretations (Section 8.2.2.2 (page 89)) and the organization interpretations (Section 8.2.2.3 (page 97)).

8.2.2.1.2 Technical Feature-Interpretations

Wellbore interpretation is a specific interpretation applied to a technical feature (the wellbore feature).

- **Wellbore interpretation (line).** A wellbore feature can contain different wellbore interpretations, which correspond to planned and/or drilled trajectories. A Boolean attribute "isDrilled" is used to indicate drilled versus planned wellbores. For more information, see Sections 12.2.1 and 12.2.2.

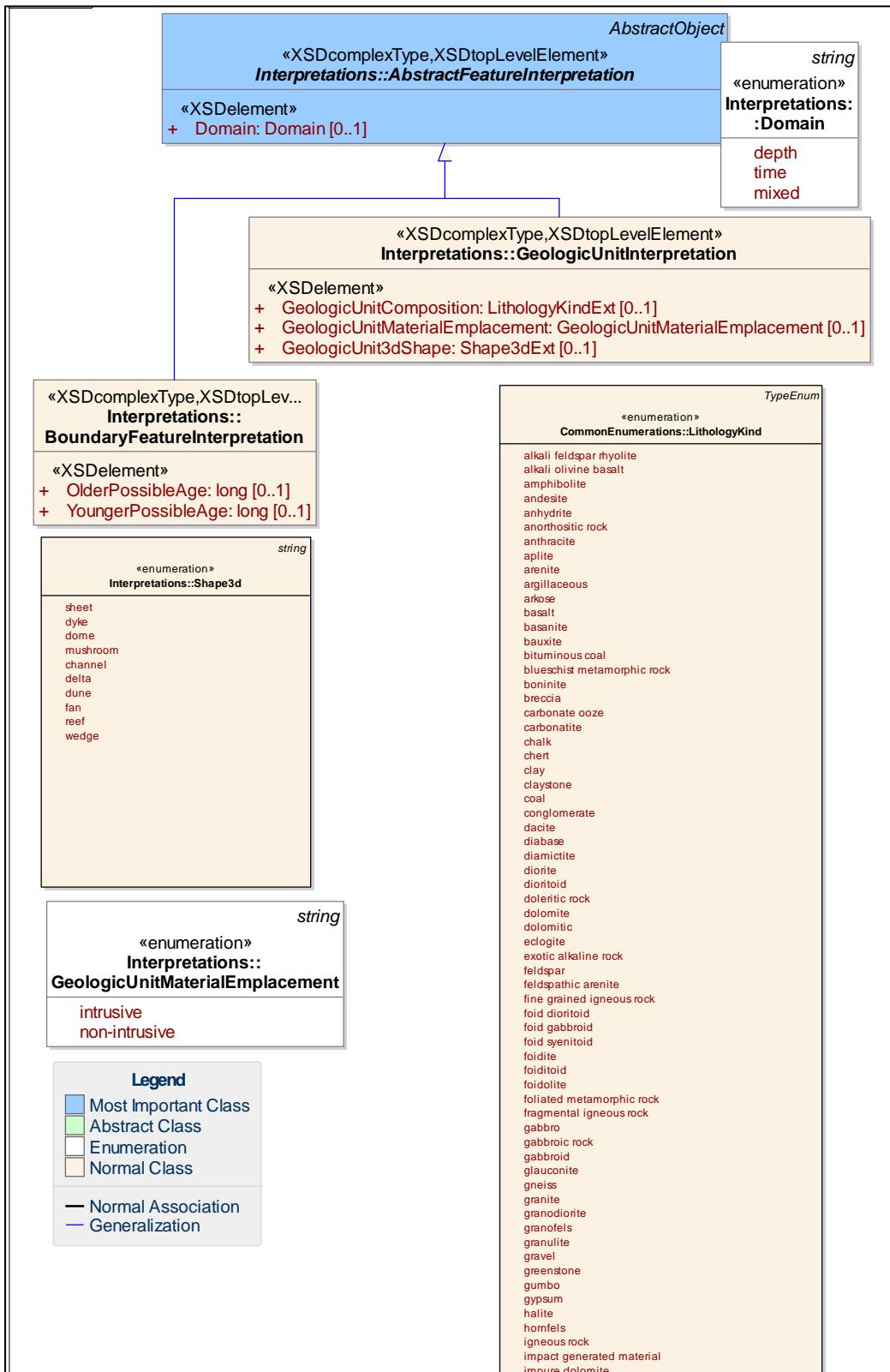


Figure 8–5. Individual Interpretations (NOTE: The lithology kind enums have been truncated so the diagram would fit on the page. All values are listed in a table above or see the common UML model.)

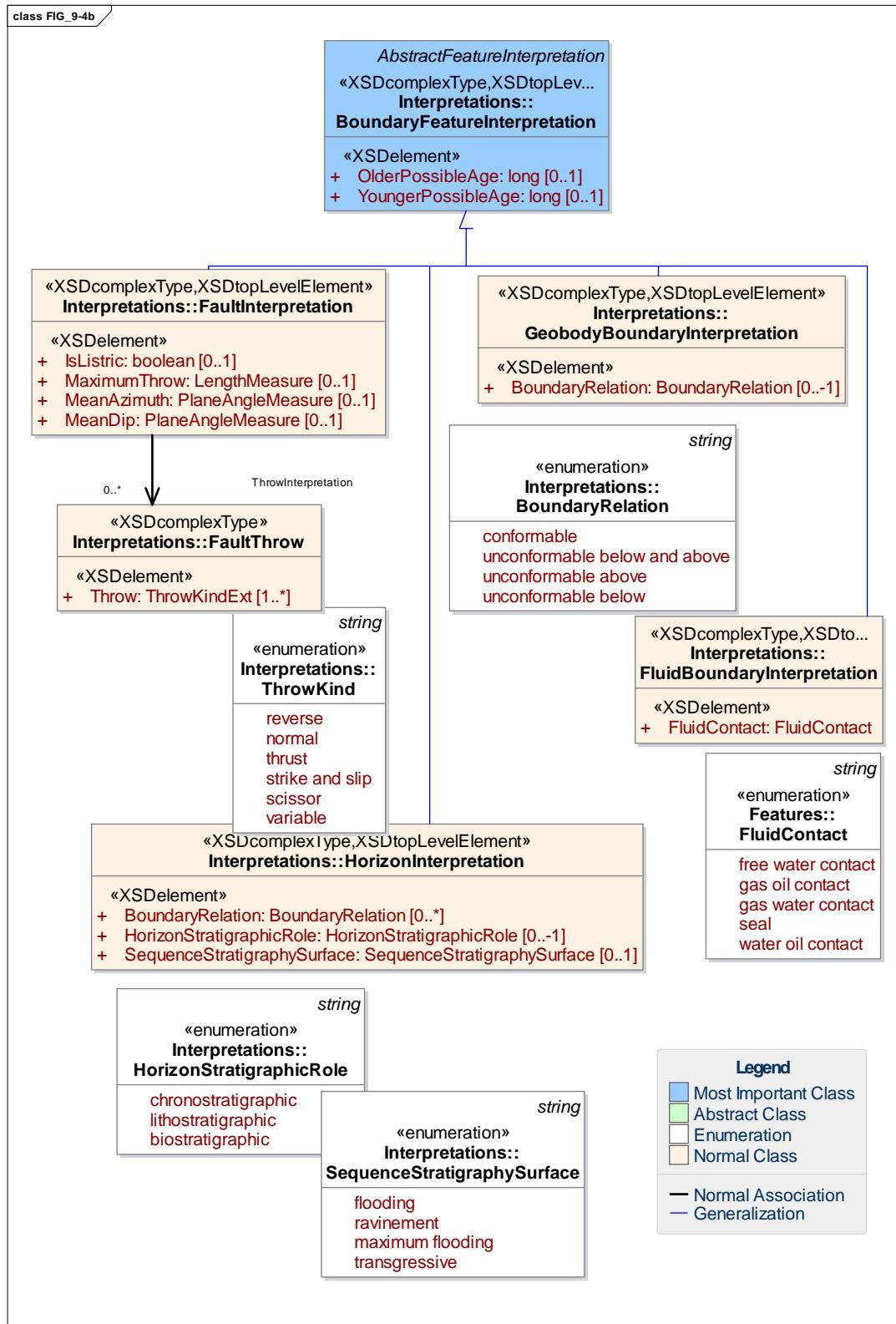


Figure 8–6. Boundary interpretation.

To specify which kind of stratigraphic horizon you are considering, use the enumeration horizon stratigraphic role. The adjectives chronostratigraphic, lithostratigraphic, biostratigraphic simply specify various kinds of stratigraphic horizons in view of the specific property that is considered (chronostratigraphic: geological age, lithostratigraphic: lithology, biostratigraphic: fossil association). Horizon stratigraphic role should be considered as a business term. Here the word “role” simply stands for kind.

In the class fault interpretation, fault is a general term for designating a discontinuity having a tectonic origin. Fault thus designates true faults but also thrust surfaces. A thrust surface is registered as a fault interpretation whose fault throw kind is “thrust” and that has the attributes: is listric = 0, maximum throw = 0.

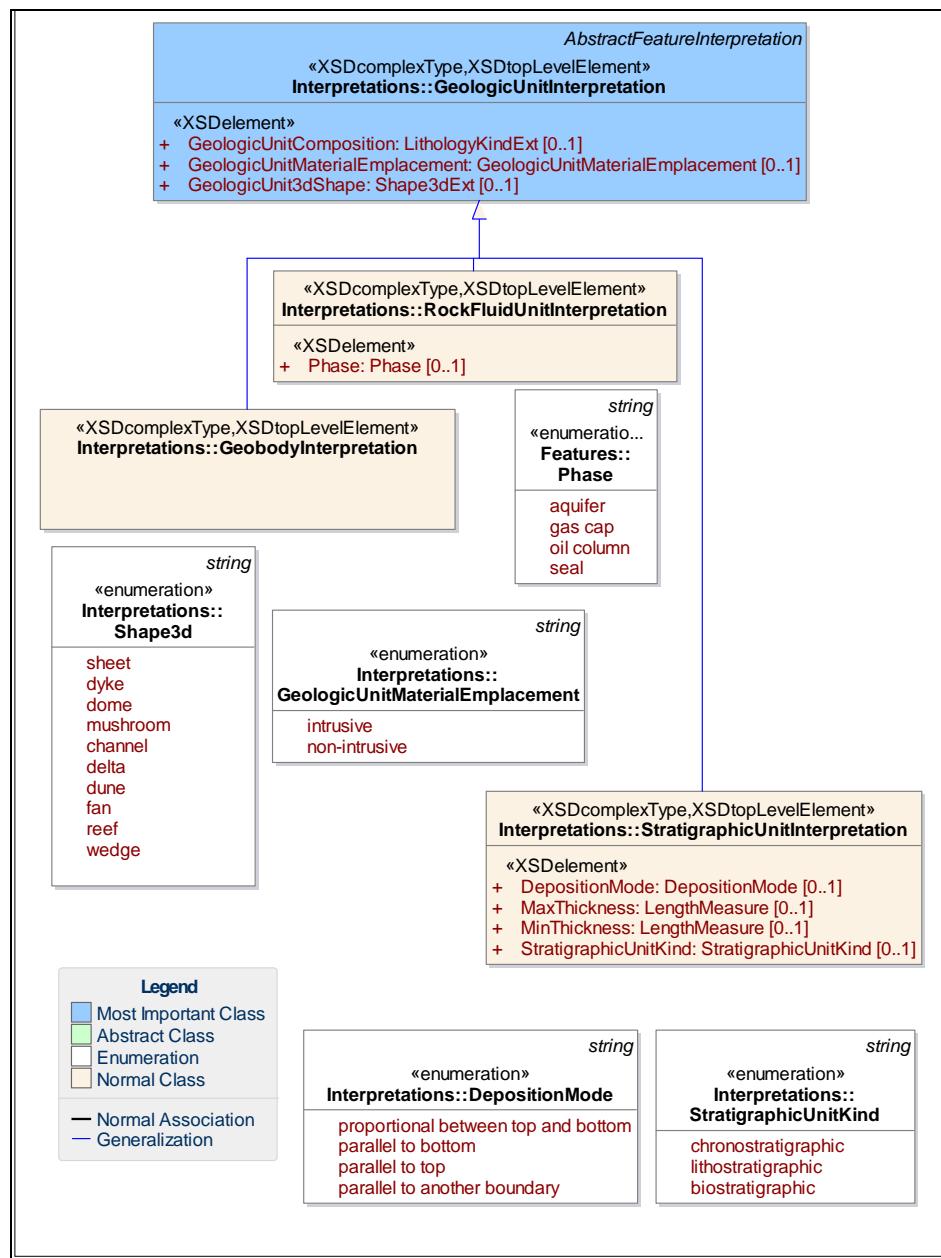


Figure 8–7. Geologic interpretation.

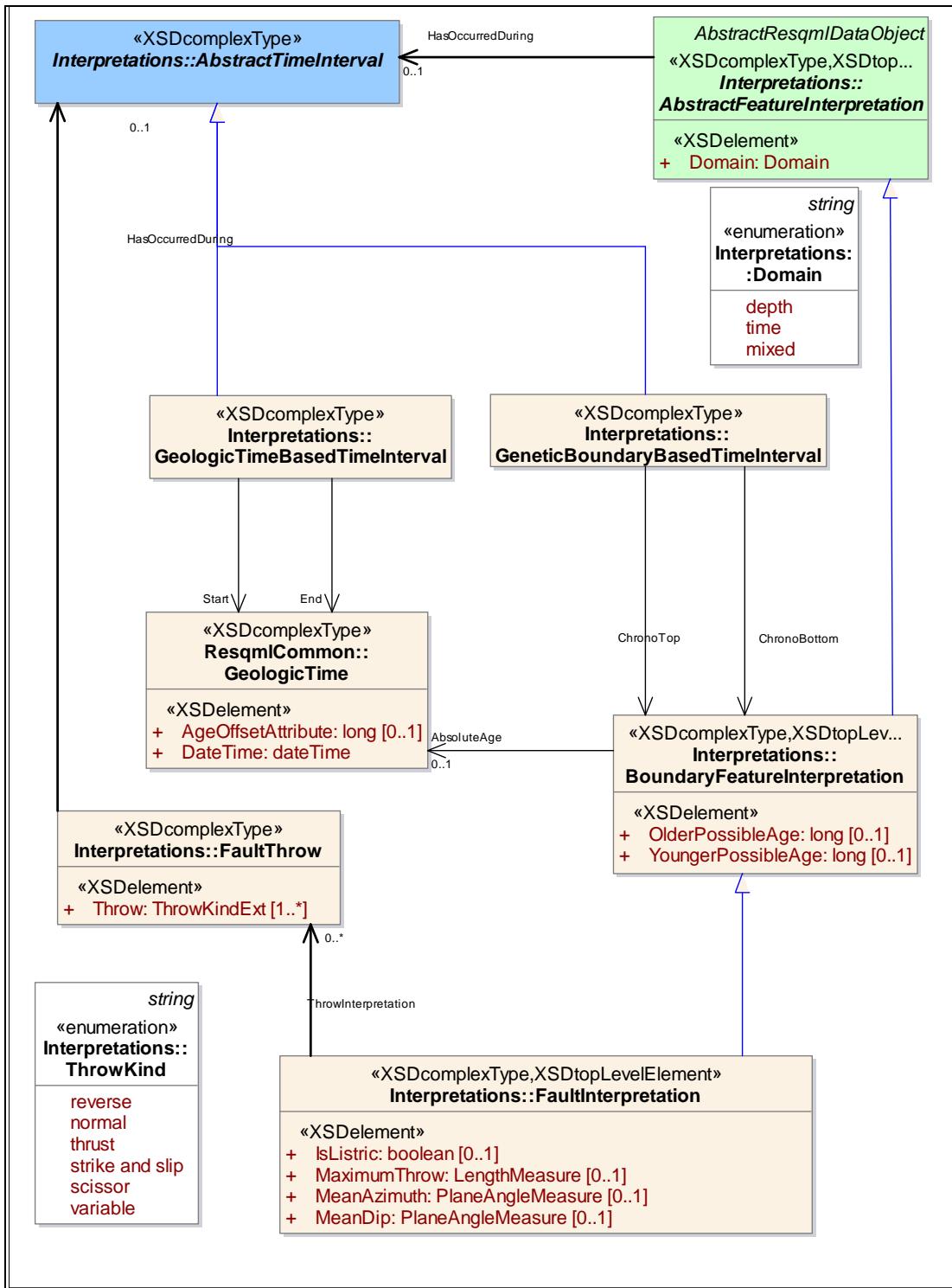


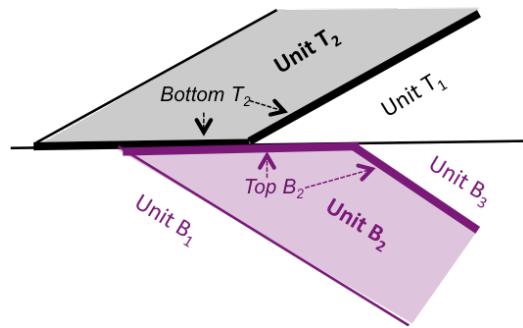
Figure 8–8. Associations between time intervals and geologic feature interpretations.

8.2.2.2 Contact Interpretations

To create consistent organization interpretations in RESQML, you must specify how the individual feature-interpretations are related or “linked together.” In the subsurface, these links occur as geological contacts. Geological contacts describe the relationships between geologic object interpretations and/or

technical features. They may be classical contacts (two objects in contact with each other) or may also denote other relationships such as crossing, splitting and even non-crossing (conformity).

A stratigraphic unit is likely to have two contacts; one on its top and one at its bottom, and in the case of unconformities, these contacts are likely to be composed of different parts. A contact interpretation may thus be expressed by one or several contact interpretation parts (**Figure 8–9**).



The contact Top unit B₂ exists in the two binary contacts (= Contact parts)

- Contact_Part T₂-B₂
- Contact_Part B₃-B₂

The contact Bottom T₂ exists in the two binary contacts:

- Contact_Part T₂-B₂
- Contact_Part T₂-T₁

Figure 8–9. Contacts and contact parts

The contacts in RESQML are mainly surface contacts (between two units), linear contacts (between two boundaries), and can be also be nodal contacts (between linear contacts) (**Figure 8–9**).

The organization interpretation (Section 8.2.2.3 (page 97)) gathers the individual representations and the contact interpretations, but a preliminary step is required to define these contacts.

Contact interpretations exist only in the context of an organization interpretation. This means that each contact interpretation has a unique index in the context of only one organization interpretation but not in "the universe" (no UUID). Therefore, contact interpretations cannot be shared by organization interpretations.

8.2.2.2.1 Contact Interpretation Part

A contact interpretation part is built on relationships between two feature-interpretations or between two already existing contact interpretation parts (**Figure 8–10**). **Figure 8–11** shows the contact element references. Further explanation follows the figures.

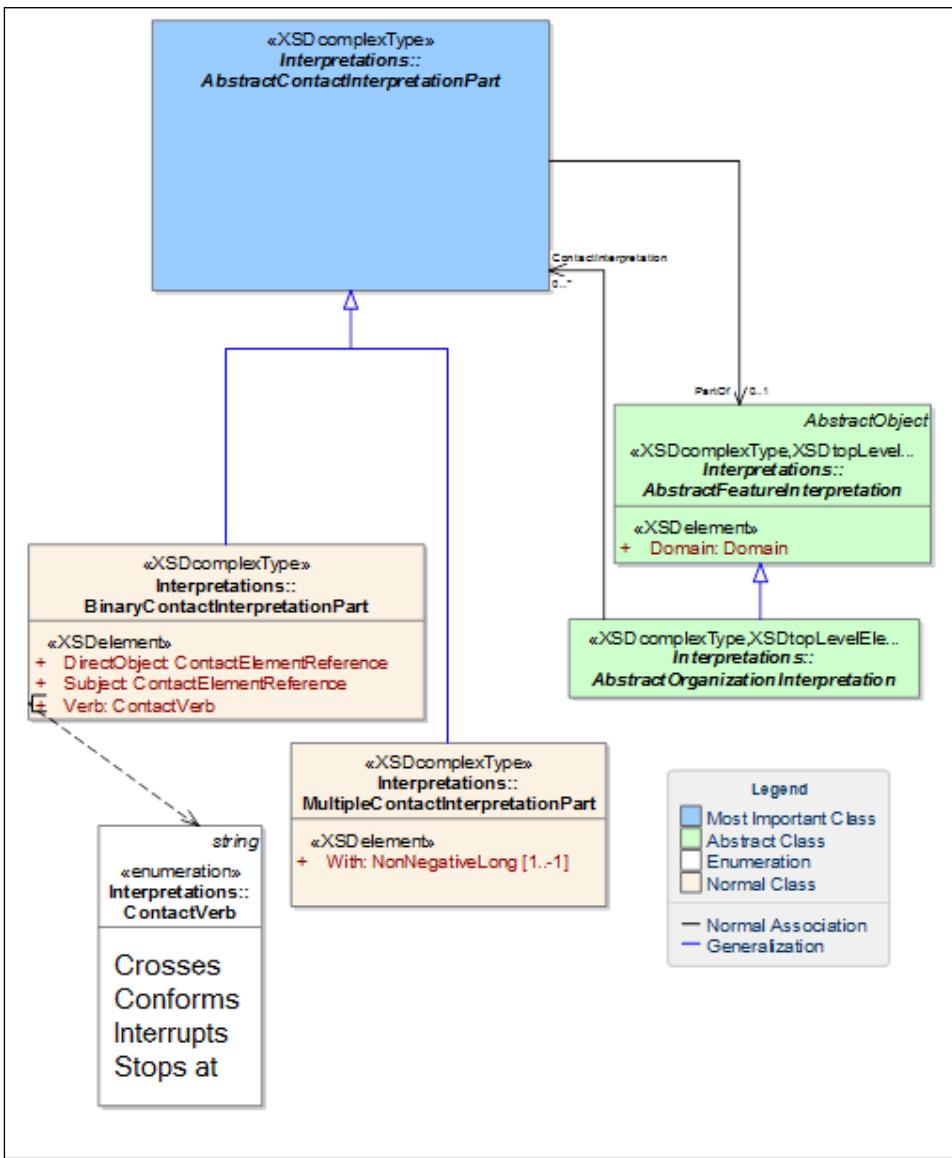


Figure 8–10. Contact Interpretations.

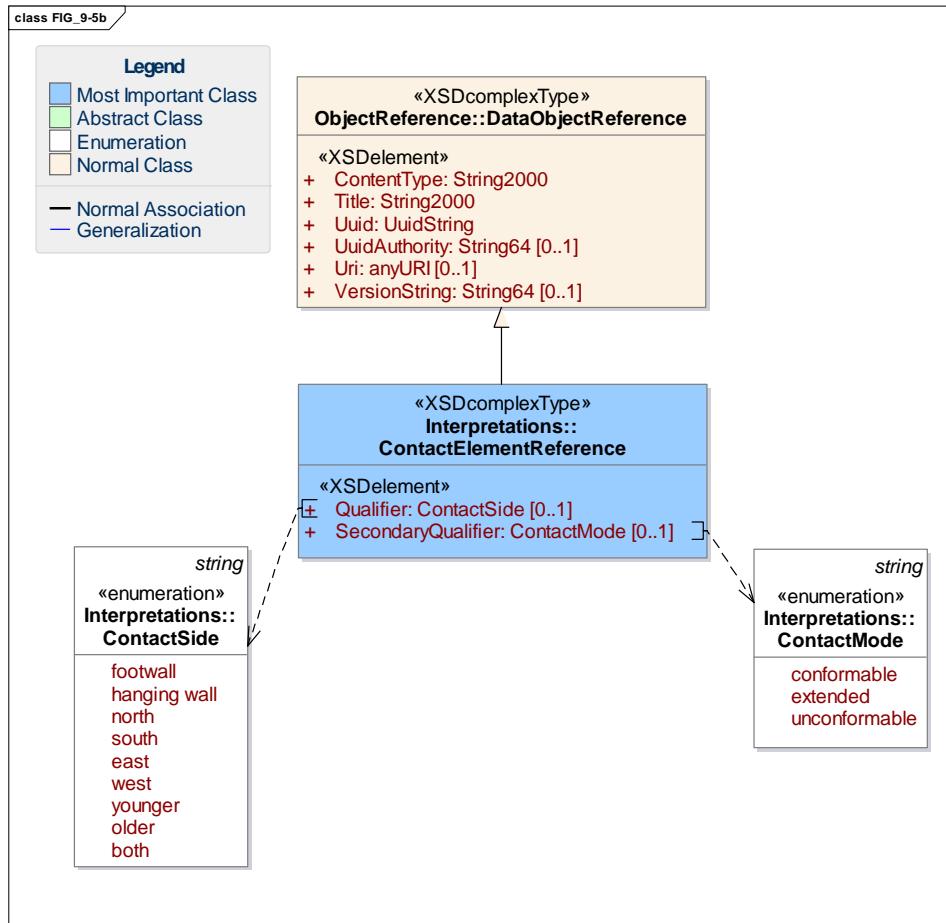


Figure 8–11. Contact element reference.

A binary contact interpretation part is used to specify a relationship between a subject and an object in the following way:

SUBJECT	Subject Qualifier 1	Subject Qualifier 2	VERB	DIRECT OBJECT	Direct Object Qualifier 1	Direct Object Qualifier 2
----------------	------------------------	------------------------	-------------	----------------------	------------------------------	------------------------------

In addition to this format, we observe this rule: Age (Subject) \leq Age (Direct Object)

Contact part relationships describe the construction of nodal, linear, or surface contacts.

Qualifiers 1 and 2 are used like this:

Type of Contact interpretation part:	Used Qualifier	Qualifier Meaning
boundary to boundary	Qualifier 1	Contact Side
unit to unit	Qualifier 2	Contact Mode

Examples:

- boundary to boundary

SUBJECT	Subject Qualifier 1	Subject Qualifier 2	VERB	DIRECT OBJECT	Direct Object Qualifier 1	Direct Object Qualifier 2
FAULT 2	both sides	<i>not used</i>	stops	FAULT 1	hanging wall side	<i>not used</i>

- unit to unit

SUBJECT	Subject Qualifier 1	Subject Qualifier 2	VERB	DIRECT OBJECT	Direct Object Qualifier 1	Direct Object Qualifier 2
UNIT 2	<i>not used</i>	conformable	stops	UNIT 1	<i>not used</i>	unconformable

The information conveyed by the verb and the qualifiers appears to be redundant in some cases. However, this verbose mode was kept to respect the practices of the various corporate users of RESQML.

Individual representation elements are used to describe how the contacts were built (what modeling business rules were used) for the construction of the binary contact interpretation. Practically, these individual representations are known by their contact element references (that are data object references). This is a way to define which boundary interpretation or geologic unit interpretation you want to use to define the contact interpretation semantics (the SUBJECT and the DIRECT-OBJECT to be referenced in the contact element reference)).

In the case of frontier feature—which has no interpretation—RESQML (v2.*) lets us define a “generic interpretation” so that this construct can work, which results in an expression like “fault interpretation ‘stops at’ generic interpretation of frontier feature”.

With RESQML, it is possible to describe the following kinds of contact relationships. Here the word **to** represents the verb (verb options are listed at the end of this section).

- generic interpretation of frontier feature **to** generic interpretation of frontier feature
- horizon interpretation **to** generic interpretation of frontier feature
- geobody boundary interpretation **to** generic interpretation of frontier feature
- horizon interpretation **to** horizon interpretation
- horizon interpretation **to** geobody boundary interpretation
- geobody boundary interpretation **to** horizon interpretation
- geobody boundary interpretation **to** geobody boundary interpretation
- horizon interpretation **to** fault interpretation
- geobody Boundary Interpretation **to** fault interpretation
- geologic unit Interpretation **to** generic interpretation of frontier feature
- geologic unit Interpretation **to** geologic unit interpretation
- fault interpretation **to** generic interpretation of frontier feature
- fault interpretation **to** horizon Interpretation
- fault interpretation **to** geobody boundary interpretation
- fault interpretation **to** fault interpretation

Or to explain this in a simpler way:

- Boundary interpretation of any type (horizon interpretation, geobody interpretation, tectonic boundary (fault) interpretation) **to** boundary interpretation of any type
- Geologic unit interpretation **to** geologic unit interpretation
- Any boundary interpretation or geologic unit interpretation **to** generic interpretation of frontier feature
- Generic interpretation of frontier feature **to** generic interpretation of frontier feature

The contact element reference (Figure 8–11 above) is used to reference the interpretation of a geological boundary or a geological unit. Its qualifier includes these options:

For boundary interpretation contacts, the qualifier is Contact_Side and includes these options:

- for fault interpretation contacts:
 - foot wall (side)
 - hanging wall (side)
 - north (side)
 - south (side)
 - east (side)
 - west (side)
 - both (sides)
- for horizon interpretation contacts:
 - younger (side)
 - older (side)
 - both (sides)

In this case we do not need a contact mode: For geological unit interpretation contacts, the second qualifier is Contact_Mode and includes the options:

- conformable
- unconformable
- extended

In this case we do not need a contact side: A unit is considered to be conformable when one of its boundaries coincides with the contact part between itself and the opposite unit.

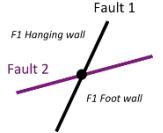
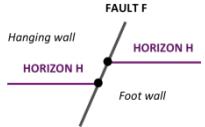
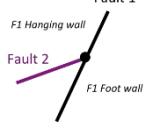
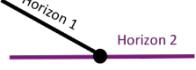
The verbs correspond to a topological/geometrical action that must be realized to calculate the corresponding contact representation. The verb options are:

- Crosses
- Interrupts
- Stops

NOTE: Some RESQML-enabled readers only use the verbs “crosses” and “stops”, and replaces “interrupts” with “stops.”

8.2.2.2.1 Practical Application: Boundary-to-Boundary Contact Interpretation Parts

Table 7 shows examples of how to define boundary-to-boundary contact interpretation parts for cases illustrated in column one.

Table 7: Boundary-to-Boundary Contact Interpretation Parts					
Case	Subject	Contact side	Verb	Direct Object	Contact side
	Fault 1	both sides	crosses	Fault 2	both sides
	Fault F	hanging wall	interrupts	Horizon H	both sides
	Fault F	foot wall	interrupts	Horizon H	both sides
	Fault 1	hanging wall	stops	Fault 2	both sides
	Horizon 2	older side	stops	Horizon 1	both sides
	Horizon 2	younger side	stops	Horizon 1	both sides

8.2.2.2.1.2 Practical Application: Unit-to-Unit Contact Interpretation Parts

We define the concept of unit-to-unit contact part because the contact between two units can be different on different locations in the prospect. This concept of contact part is thus specifically designed to deal with these complex use cases. However, in many cases only one contact part exists between two units.

Describing unit-to-unit contact interpretation parts is a bit more complex than the method described above for boundary-to-boundary contact interpretation parts. This additional complexity is because RESQML represents separately the structural organization and stratigraphic column attached to one earth volume. This design was chosen so that a 2D object (structural organization) and 3D object (stratigraphic columns) could be considered separately (see Section 8.2.2.3).

However, when a stratigraphic column has unconformities, the units in contact are not only separated by a plain boundary surface, but by an actual geological object as shown in **Figure 8–12**. In the case of a relation “unconformable above and below” (case 3 of Figure 8–12) plays the role of an actual geological unit and may be considered as such in the stratigraphic column. In view of the adopted syntax of subject/verb/direct_object, we do not represent the geological unit contacts globally, but we consider the relationship (conformable/unconformable) of each of the units with the contact surface . You can say it is a geological unit having a zero thickness.

Because we also impose the rule: Age (Subject) \leq Age (Direct Object), the relationships between units are completely defined without using the terms “unconformable below” or “unconformable above (baselap)”, which correspond to a global description of the relationships between units.

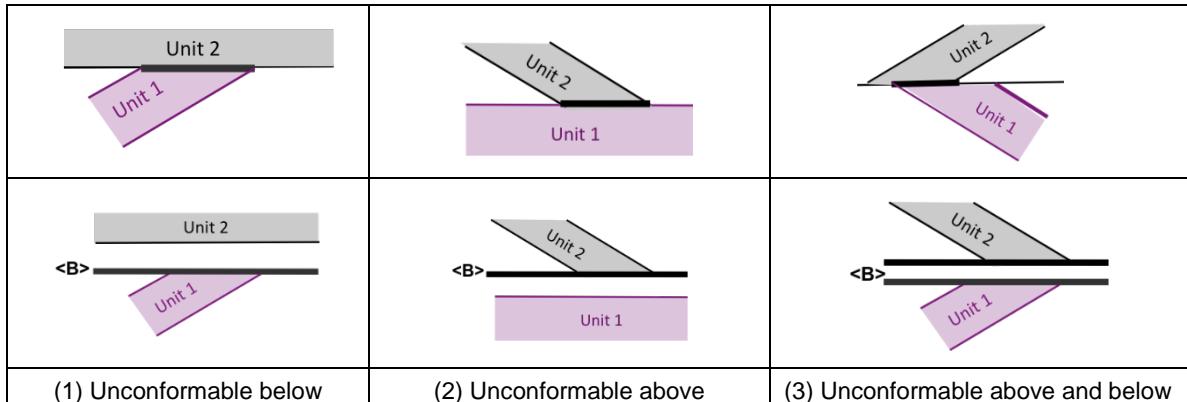
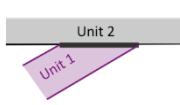
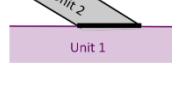
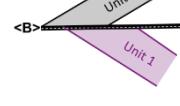


Figure 8-12. Unconformities within a stratigraphic column. In the bottom boxes, . symbolizes a geological object which may be an erosion surface in cases (1) and (3) and/or to a fossil sea bottom corresponding to an interruption of sedimentation in cases (2) and (3).

As shown on **Table 8**, the verb that is used for designating unit to unit contact interpretations is “stops” in all the cases. The four possible cases are thus identified by the qualifiers conformable / unconformable applied to each of the two units that are in contact.

Table 8: Unit-to-Unit Contact Interpretation Parts

	Subject	Contact Mode	Verb	Direct Object	Contact Mode
	Unit 2	conformable	stops	Unit 1	conformable
	Unit 2	conformable	stops	Unit 1	unconformable
	Unit 2	unconformable	stops	Unit 1	conformable
	Unit 2	unconformable	stops	Unit 1	unconformable

8.2.2.3 *Organization Interpretations*

Technically, an organization interpretation:

- is typically composed of one stack of its contained elements
- typically contains:
 - contacts between the elements of this stack among themselves
 - contacts between the stack elements and other organization elements
- may be built on other organization interpretations

The class feature interpretation set is used to specify collections of unordered interpretation features. For example, this class can be used to specify fault sets in a structural organization interpretation.

RESQML distinguishes these types of organization feature-interpretations—structural, stratigraphic, rock fluid, and earth model—which are described further below.

8.2.2.3.1 Structural Organization Interpretation

A structural organization interpretation (Figure 8–13) contains the boundary interpretations (and the frontiers associated as generic interpretations of frontier features) that are present in a given volume of rock and their contact interpretations, which includes the contact kinds listed in Section 8.2.2.2 (page 89).

A structural organization interpretation instance corresponds to a complete description of the architecture of the surfaces present within an earth volume. It includes an ordered list of implicit indices (the implicit value of each index is given by its order in the list). Each index contains the stratigraphic rank of the boundary interpretation or of the fault interpretation set attached.

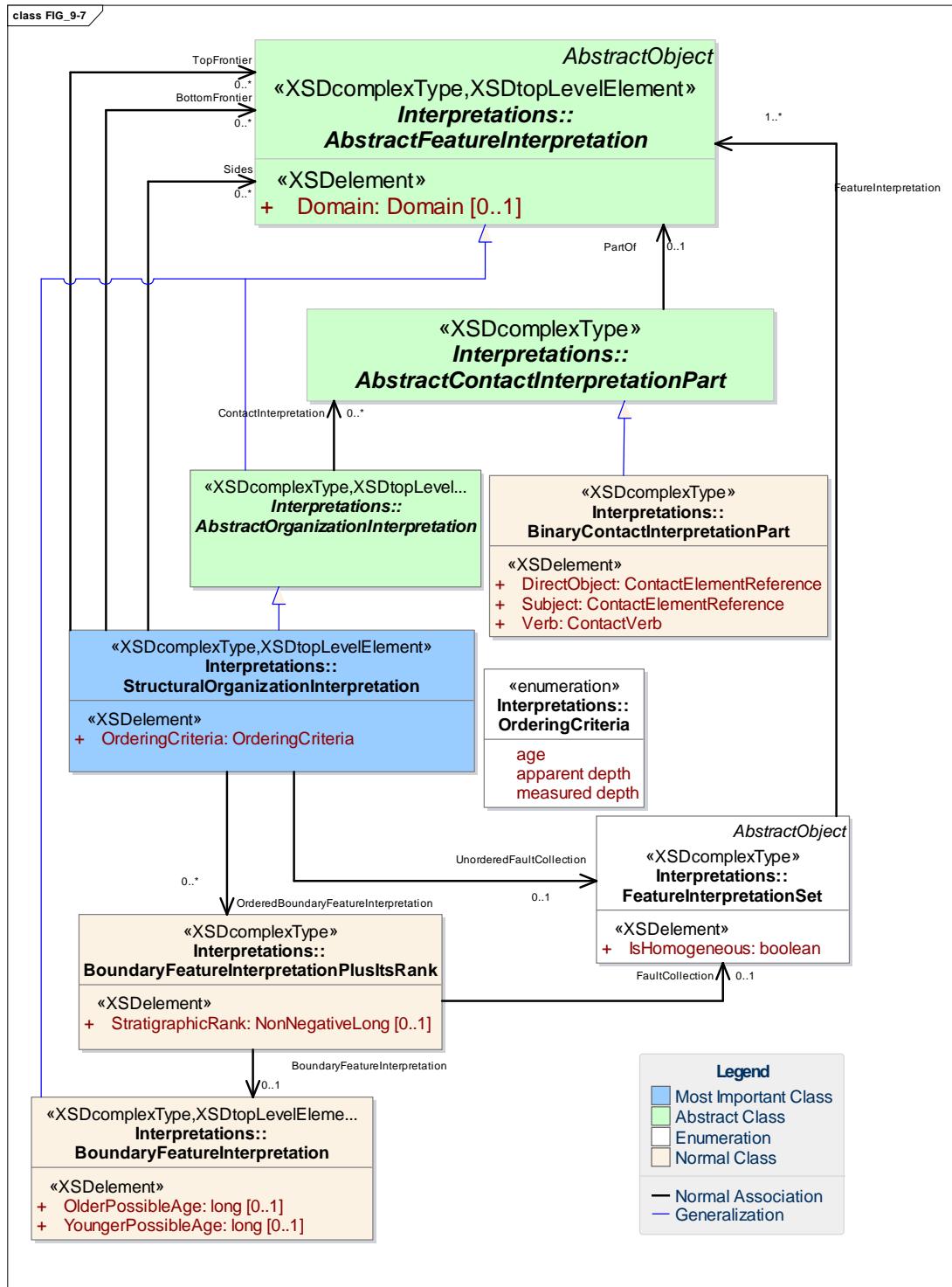


Figure 8–13. Structural organization interpretations.

A structural organization can be ordered by age, apparent depth or along measured depth. In the case when it is ordered by age, it must be organized in stratigraphic ranks. A structural organization consists of one list of boundary feature interpretations and/or a feature interpretation set (mainly a set of unordered

faults). To specify a fault network, we must add some binary contact interpretation parts based on fault to fault relationships.

Figure 8–14 shows an example of a succession of events related to an earth model. This example displays, in addition to the deposition of various sedimentary units, a faulting event consisting in three faults, which have the same age and constitute a fault network. This fault collection interpretation set is inserted into the following sequence of events:

/Younger Erosion: Rank 0 // Top of younger Deposition: Rank0 // Interpretation set3 faults: Rank1// Top of Intermediate deposition Rank1 //Top of older Deposition: Rank 1//Bottom of older deposition Rank0.

Figure 8–15 shows the corresponding structural organization and also the feature interpretation set corresponding to the three faults.

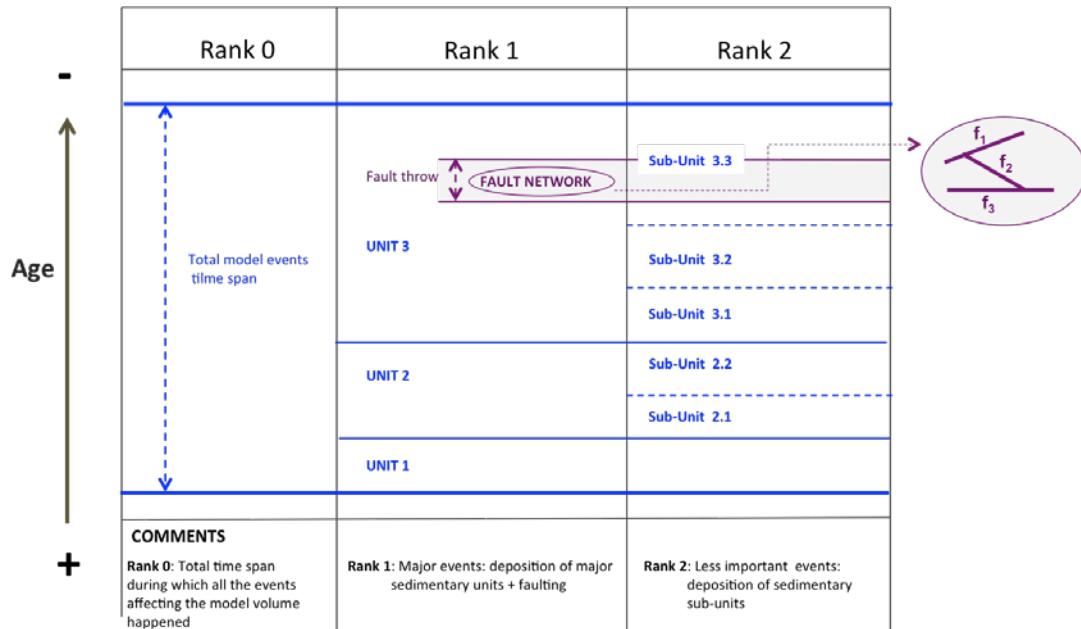


Figure 8–14. Example of a succession of geologic events.

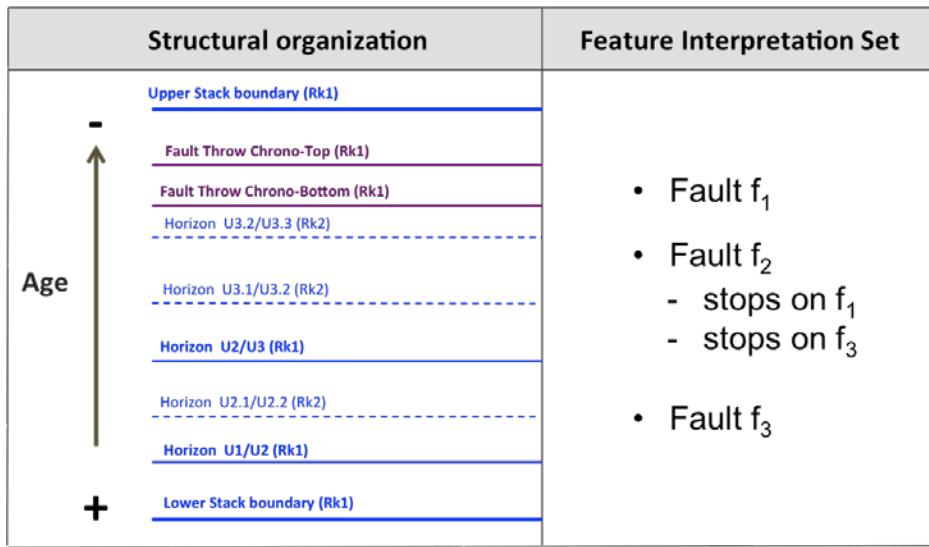


Figure 8–15. Structural organization and feature interpretation set corresponding to the example of Figure 8–11.

8.2.2.3.2 Stratigraphic Organization Interpretation

Stratigraphic organization interpretation defines the relationships between stratigraphic units. **Figure 8–16** shows this class along with the child classes it comprises, which are:

- **Stratigraphic column rank interpretation column** is a global hierarchy containing an ordered list of stratigraphic unit interpretations ordered by age or in apparent depth.
- **Stratigraphic occurrence interpretation** designates a local succession of geological units arranged according to an ordering criterion, which are present along a well, on a 2D map or on a 2D section, or on a part of the global earth volume. Ordering criterion may be age, or apparent depth or measured depth along a well trajectory. We can note that this term “stratigraphic occurrence interpretation” is not generic enough because some geobody interpretations or rock fluid unit interpretations could be intersected by a well. In fact we should have named this class “geologic occurrence”.
- **Stratigraphic column** corresponds to the type of assemblage commonly designated by geologists as a stratigraphic succession or simply as “stratigraphy”. It is a global hierarchical level that can be made up of several ranks of stratigraphic units, the one which are ordered by age.
 - Practice: As I have no direct link from a stratigraphic organization interpretation to a stratigraphic column, how can I link my stratigraphic column to a stratigraphic organization interpretation?
 - Answer: In fact, the link exists in the other direction. A stratigraphic column can gather several stratigraphic column rank interpretations, which are all of them abstract stratigraphic organization interpretations. This construction was designed this way to avoid circular links. At the end, the more important data object in this construction is the stratigraphic column and this stratigraphic column should belong to an earth model interpretation.

8.2.2.3.3 Rock Fluid Organization Interpretation

Rock fluid organization interpretation (**Figure 8–16**) is composed of rock fluid unit interpretations. Each of the rock fluid unit interpretation corresponds to a fluid phase.

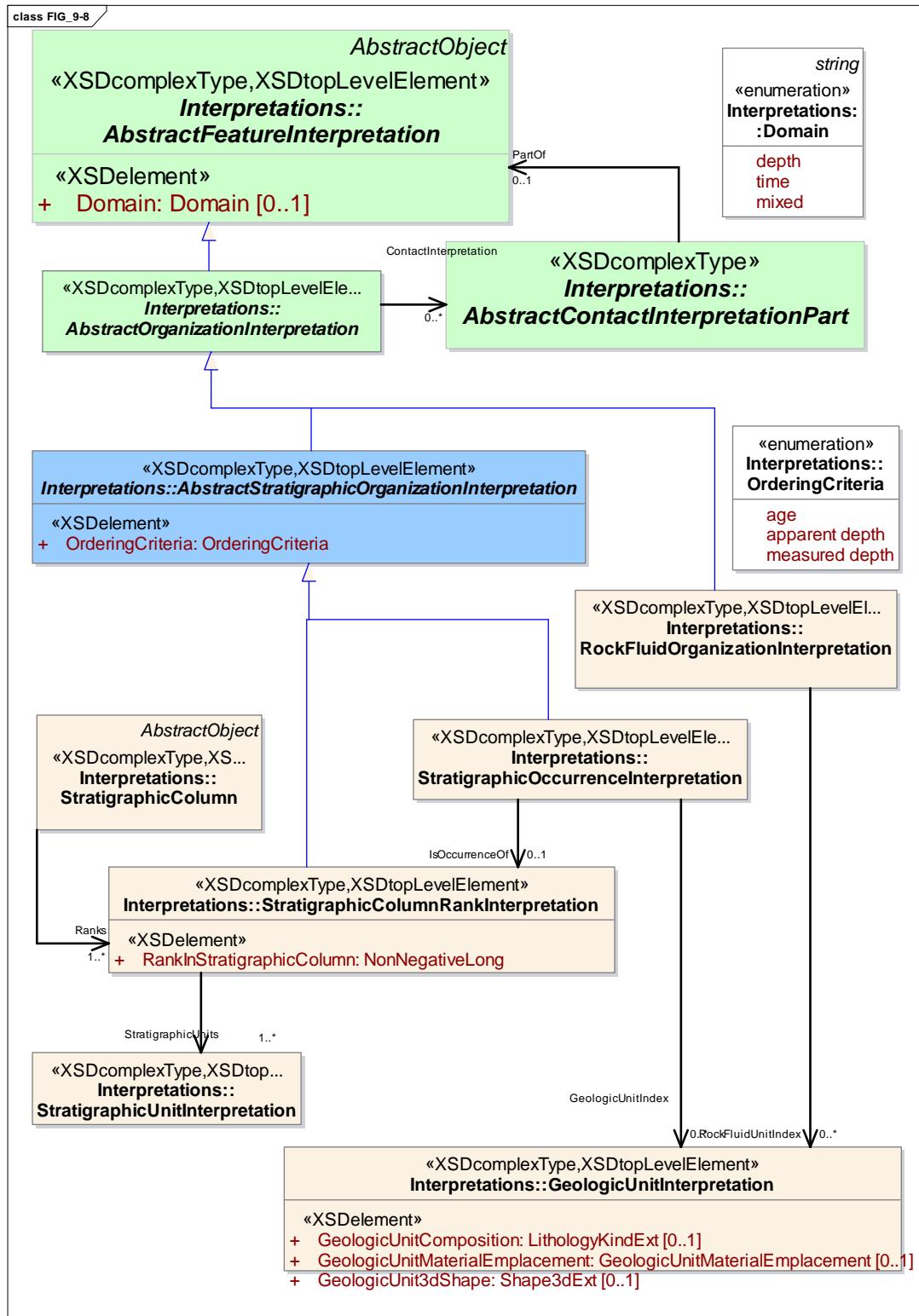


Figure 8–16. Stratigraphic and rock fluid organization interpretations.

8.2.2.3.4 Earth Model Organization Interpretation

Earth model organization interpretation (Figure 8–1 (page 80)) refers to a “trusted” interpretation of a geological site at a specific stage of the modeling process; “trusted” means that this interpretation should be geologically and topologically consistent. Additionally, for an earth model organization interpretation to be considered trusted, the rock fluid organization interpretation must be based on the stratigraphic organization interpretation (the stratigraphic Column) and this stratigraphic organization interpretation must be built on a structural organization interpretation. Thus an earth model rests on various kinds of organizations describing the stratigraphic, structural, and rock fluid characteristics of the model.

8.3 Examples: How to Use RESQML in the Life Cycle of an Earth Model

This section walks through several examples that show how RESQML can be used to facilitate the earth modeling process.

- **Example 1: Creating an Earth Model** (Section 8.3.1 (page 103)) is a step-by-step example that shows how to create an earth model in RESQML. It begins with two slightly different seismic interpretations by team leads Paul and Peter. The asset team selects Peter's interpretation and develops it to a non-sealed surface framework, performing tasks and exchanging data between various RESQML-enabled software.
- **Example 2: Possible Next Steps: How to Reuse/Update a Complete Earth Model for Static Property Modeling and Flow Simulation** (Section 8.3.2 (page 113)) extends the first example; the asset team decides to look at Paul's interpretation. The section explains some possible next steps to update the earth model created in the first example.
- **Example 3: Strategy to Re-Engineer an "Ancient" 3D Grid** (Section 8.3.3 (page 115)) shows how RESQML can help in reading and updating an "ancient" grid from an old project.

NOTE: There are different ways to implement RESQML into software. Typically, a developer may only be concerned with the RESQML format when the user explicitly chooses an option to save to this format or export to RESQML. However, in this example, to help map RESQML constructs to an earth modeling business process, the RESQML concepts are discussed in parallel throughout the example.

8.3.1 Example 1: Creating an Earth Model

To best show how RESQML works, this example uses different RESQML-enabled software for different steps in the process (in reality, some software packages could do all or most of these steps). The software used in this scenario includes those listed in the table below. The table also shows the names of each EPC file (also called an Energistics package) which contain all the data objects that each software reads and writes. (Note: Energistics packages are transient; they exist only for a single transfer. For more information about Energistics packages, see Section 3.1.8).

Software "Name"	Type/Role of Software	Package Read	Package Written
Software A	Seismic interpretation		EPC_0
Software B	Surface modeling	EPC_0	EPC_1
Software C	Surface structural framework assemblage	EPC_1	EPC_2
Software D	Sealed surface structural framework modeling	EPC_2	EPC_3
Software E	Stratigraphic volume framework modeling	EPC_3	EPC_4
Software F	3D grid modeling software	EPC_2 or EPC_3	EPC_5
Software G	Fluid flow simulation software	EPC_5	EPC_6
Software H	Gridding software with editing functions	EPC_5	EPC_6

Ideally, it is best to begin thinking about the data exchange at the beginning of the earth modeling process.

The first exchange begins after the realization of step 1, seismic interpretation (see the next section). At the end of this step 1, the users record the results of the seismic interpretation.

8.3.1.1 Step 1: Prospect Creation after Seismic Interpretation

Two teams of geophysicists and geologists, one led by Paul the other by Peter, are each working with the same data on the same task: to obtain the best fit between the seismic image (seismic reflector) and two 2D grids for Horizon1 and Horizon2, and the best fit between the seismic image (noise area) and (a set of lines) for Fault1 (**Figure 8–17**, left). The teams are each using seismic interpretation Software A. (In reality, the geologists might be using log analysis software, but for simplicity of the example we have both groups using the same Software A.)

Software A uses an EPSG code for geographic localization and a user has interpreted a seismic cube and several seismic 2D lines. Horizon1 and Horizon2 are two reflectors corresponding to well-known boundary features in this survey. These well-known boundary features are the Horizon1 and Horizon2 features, which the team creates in the software. By definition, each RESQML top-level feature has a UUID (for this example: HORIZON_1_UUID, HORIZON_2_UUID, FAULT_1_UUID) In Software A, Paul and Peter picked these reflectors, and for each reflector, a 2D grid (Grid2D) was defined.

To manage their results in Software A, they name them: Horizon1_Paul, Horizon2_Paul, Horizon1_Peter, Horizon2_Peter, Fault1_Paul, Fault1_Peter, Figure 8–17, right).

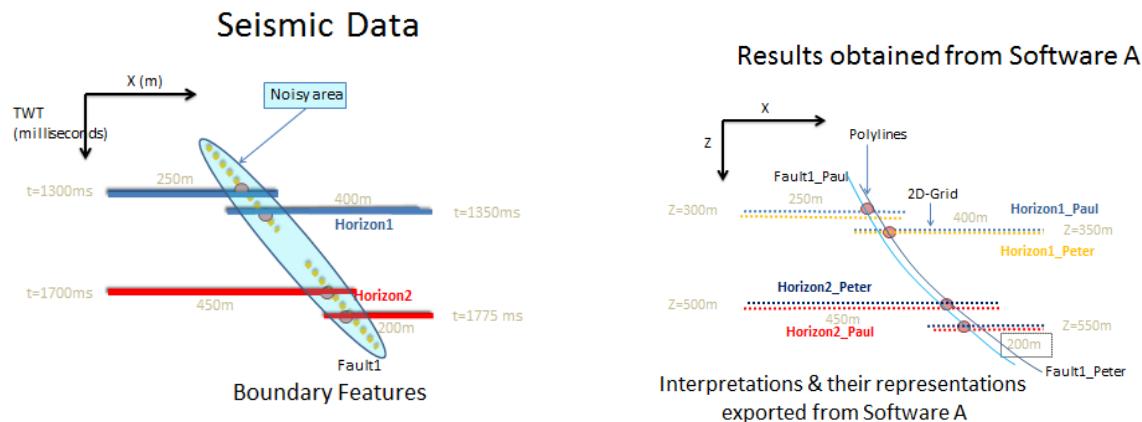


Figure 8–17. Data to be transferred from Software A.

Each of their interpretations is fine, but the results are not identical, with two different interpretations of seismic reflectors and faults lines. They have calculated the potential volume between Horizon1 and Horizon2 and they have different estimations, so they want to export the data from their respective interpretations to Software B, surface modeling software, where they can create triangulated surfaces for more precise results and comparison.

8.3.1.1.1 Preparing for Data Exchange

When working in the context of an asset team and the full life cycle of an earth model, and before beginning to exchange data in the workflow, we must give some thought to how to organize data for iterative exchanges throughout the life cycle. This organization is important so that in the future, we can retrieve the hypothesis used during the successive modeling steps.

First, create an earth model feature in Software A with the title: MY_EARTHMODEL_Feature (UUID_EMF, Citation_EMF) (Figure 1–1 (page 80)). This example does not yet use the earth model feature (because we have not yet created any earth model interpretations). However, in support of long-term organization for full life cycle and iterative workflows, the model feature corresponding to this earth model was created first.

Then, because the team has created boundary features, each with a UUID (for this example: HORIZON_1_UUID, HORIZON_2_UUID, FAULT_1_UUID), the team can attach one or more corresponding interpretations to each feature, using a data object reference to indicate the UUID of the

feature that is being interpreted. (For more information, about features and interpretations see Chapter 4.)

This reference mechanism allows enrichment of the model, step by step. For example, because of this referencing ability, at a given time stamp, we can export Peter's interpretations. In a few weeks, after more work has been done, Paul's interpretation can also be exported and are associated with the same features by UUID.

So we have a prospect with three individual features: (Horizon_1, Horizon_2 and Fault_1). Each of these features has two interpretations (one by Peter and one by Paul), and for each interpretation, we can attach one or more representations. The representations that can be exported from Software A are:

- For each horizon interpretation, one 2D grid based on a point lattice of 2 dimensions (4 2D grids, all together with only one point lattice 2D).
- For each fault Interpretation, one polyline set.

8.3.1.1.2 Organization Interpretations Set Up

With the existing information, we can create one RESQML structural organization feature and two structural organization interpretations (one for Paul and one for Peter). In this example, we will create two earth model interpretation instances (which reference the model feature we created above), with each one containing a structural model interpretation:

- Paul's earth model interpretation instance contains one (Paul's) structural organization instance, which contains: Horizon_1_Paul, Horizon_2_Paul, Fault_1_Paul.
- Peter's earth model interpretation instance contain one (Peter's) structural organization instance, which contains: Horizon_1_Peter, Horizon_2_Peter, Fault_1_Peter

A set of contact interpretations based on Paul's individual boundary interpretation contacts can also be attached to Paul's structural organization interpretation, with corresponding representations.

This is the setup point of the interpretation management.

After this step, these data objects can be stored in an Energistics package named EPC_0 and transferred to Software B or can be delivered on request from a Web service mechanism, which can retrieve these different elements using their respective UUIDs.

8.3.1.2 Step 2: Structural Modeling

8.3.1.2.1 Creating a New Surface Representations

Software B is used to do structural modeling (surface modeling). The information in EPC_0 is used to consistently transfer to Software B (and potentially to additional software packages) all the acquired representations and populate Software B's internal data model (**Figure 8–18**).

A user can manage all of this information in Software B with only one constraint: Software B MUST keep the RESQML UUIDs (i.e., as an attribute) of the data objects it instantiates.

If Software B can apply surface modeling on 2D grids and polyline sets, it can access the detailed information about the topology and the geometry of these representations and can create triangulated surfaces and define contacts as polylines (**Figure 8–19**).

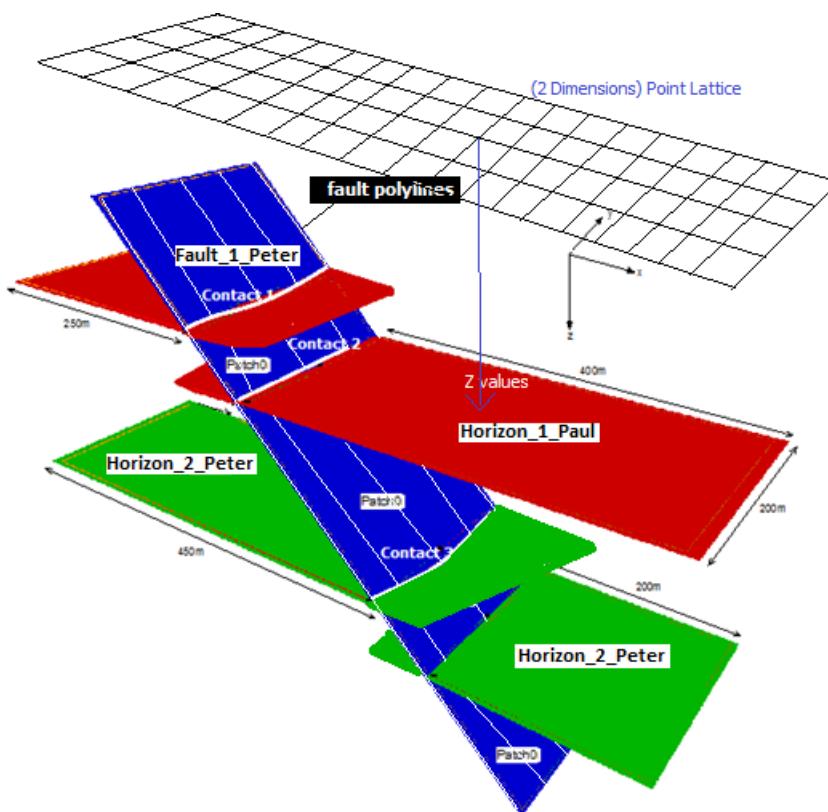


Figure 8-18. Data objects transferred from Software A.

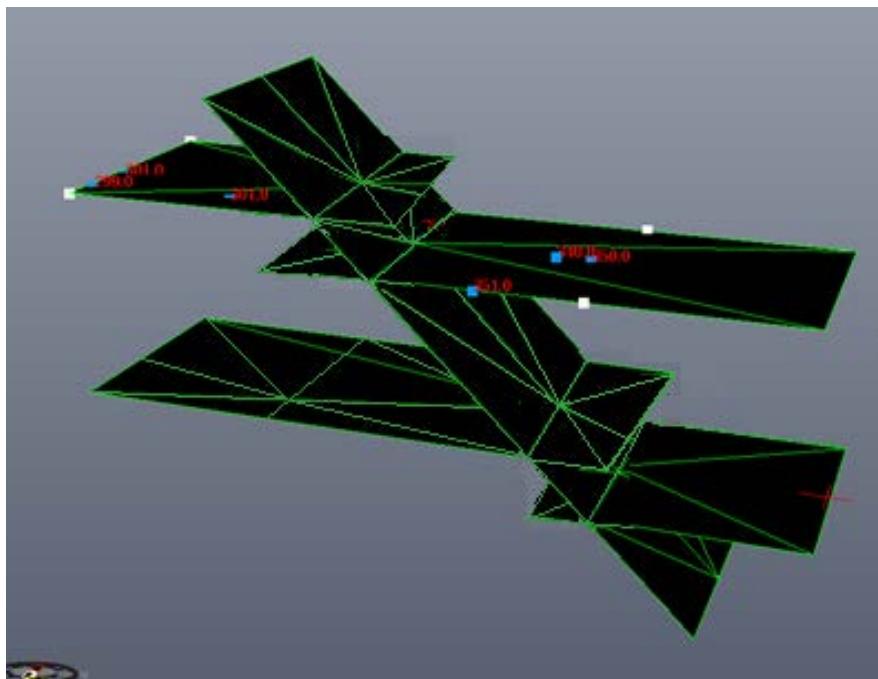


Figure 8-19. Triangulated set surface representations created in Software B.

In this case, after having created these new triangulated representations for both Paul's and Peter's interpretations, Software B adds to the previous RESQML data by linking these new representations to the existing interpretations using the UUID and data object reference mechanism discussed above.

Software B can export only the triangulated surface sets stored in an Energistics package named EPC_1, which references the elements in EPC_0.

8.3.1.2.2 Creating a Structural Organization Interpretation and a Non-Sealed Surface Framework Representation

At this step, the asset team must compare both interpretations and decide whether to follow Paul's hypothesis or Peter's. For this example, we'll follow Peter's less optimistic hypothesis.

Software C imports Peter's structural organization interpretation and all of his individual interpretations/representations with the objective to build a first non-sealed surface framework representation with this set of interpretation and representation data objects.

The first task is to complete Peter's structural organization interpretation. To do this, we need to order the horizon interpretations (by age or apparent depth) and we need the contact relationships between the horizons and faults.

For this example, we define a Rank1 containing only the Horizon_1_Peter interpretation and a Rank2 containing Horizon_1_Peter and Horizon_2_Peter interpretations.

Each of these horizon interpretation is represented by two patches; the fault Interpretation is represented by one patch, and the contact patches are represented by a succession of (x,y,z) nodes (**Figure 8–20**). (For more information on patches, see Section 5.3)

In fact, very often, no specific representation will be created during this sub-step; Software C will export an EPC_2 containing a consistent working set on which only one representation is chosen for one interpretation and only one interpretation of a feature contributes to building its structural organization interpretation.

This EPC_2 can be used as a starting point by 3D grid builder software to produce a reservoir grid or by a more sophisticated structural modeler to build a sealed surface framework (Software D, in this example).

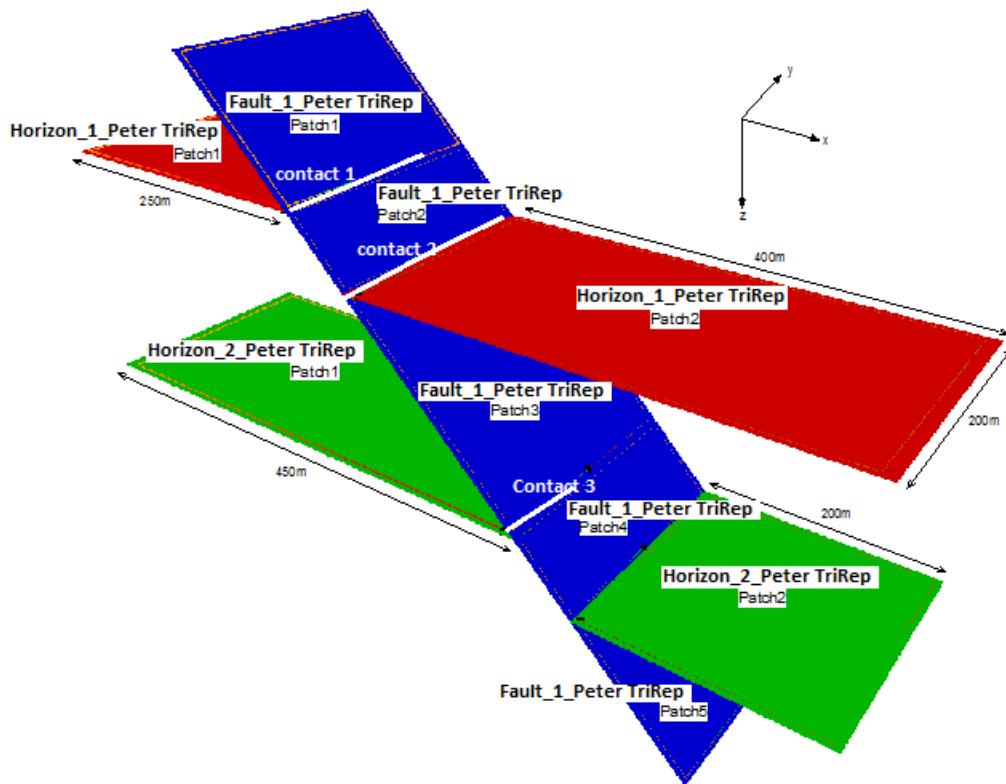


Figure 8–20. Structural organization interpretation represented by a non-sealed surface framework.

8.3.1.2.3 Creating a Sealed Surface Framework Representation in a Structural Organization Interpretation

The previous EPC_2 contains the horizons, faults, and contact interpretations defined during the seismic interpretation tasks, gathered in a consistent manner into a structural organization interpretation (which we defined in the first step (Section 8.3.1.1.2 (page 105)). This EPC_2 also contains their representations, which can exist on different supports (2D grid, triangulated surfaces, polyline sets, 3D points).

The objective of the next sub-step is to obtain a true boundary representation (BREP) of this structural organization interpretation.

Software D imports EPC_2 and uses it to define in detail the topological association of individual nodes of the different supports to build a topologically consistent sealed surface framework (**Figure 8–21**). For each node belonging to one support, the geometry is defined only once.

To build such a sealed surface framework this Software D must realize “clean” intersections between faults and faults, horizons and faults, and horizons and horizons, at least. As an added benefit, this Software D associates the nodes of the intersection lines together. In this case, this framework is sealed because the patches are limited by the intersection lines, and the nodes of the intersection lines could be the same on both sides of the intersection (**Figure 8–22** and **Figure 8–23**).

Because all the connected patches share their nodes while intersecting, the intersection line is defined as a subrepresentation of the patches, and the geometry of the nodes of the intersection are defined only once in the patch representation. (For more information on subrepresentations, see Section 5.4).

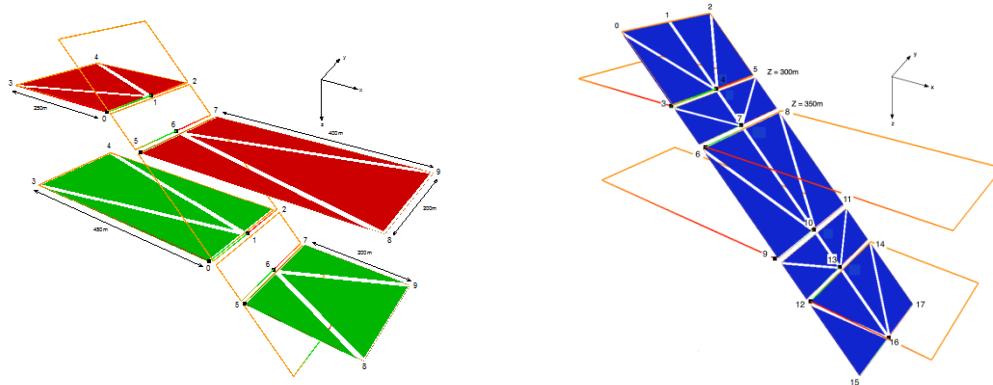


Figure 8–21. Detailed topology needed to obtain a sealed surface framework.

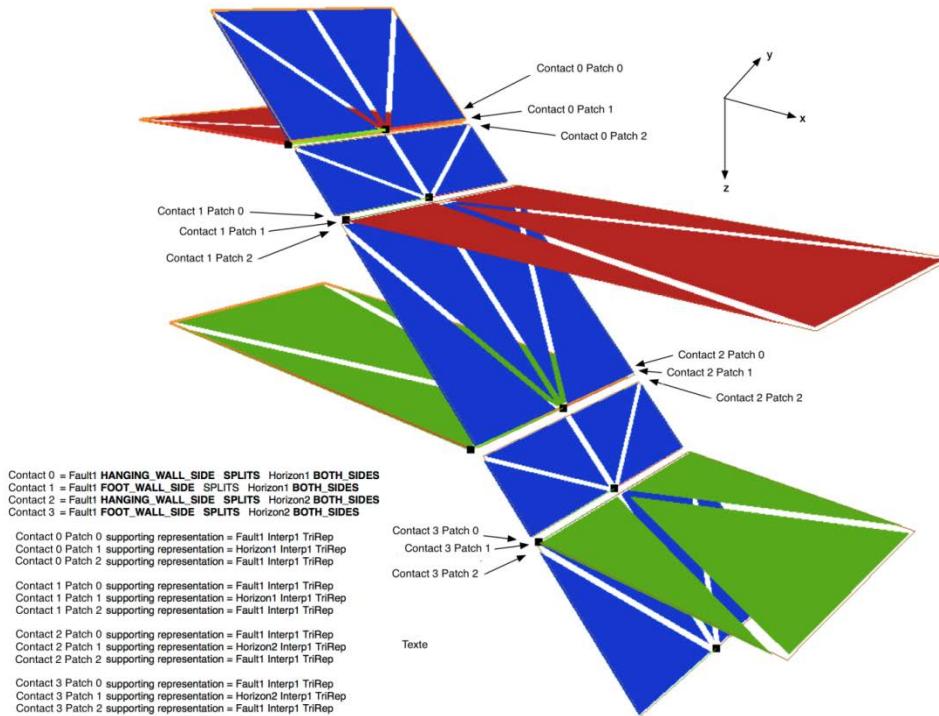


Figure 8–22. Sealed surface framework. For the previous colocated points, see Figure 1–23.

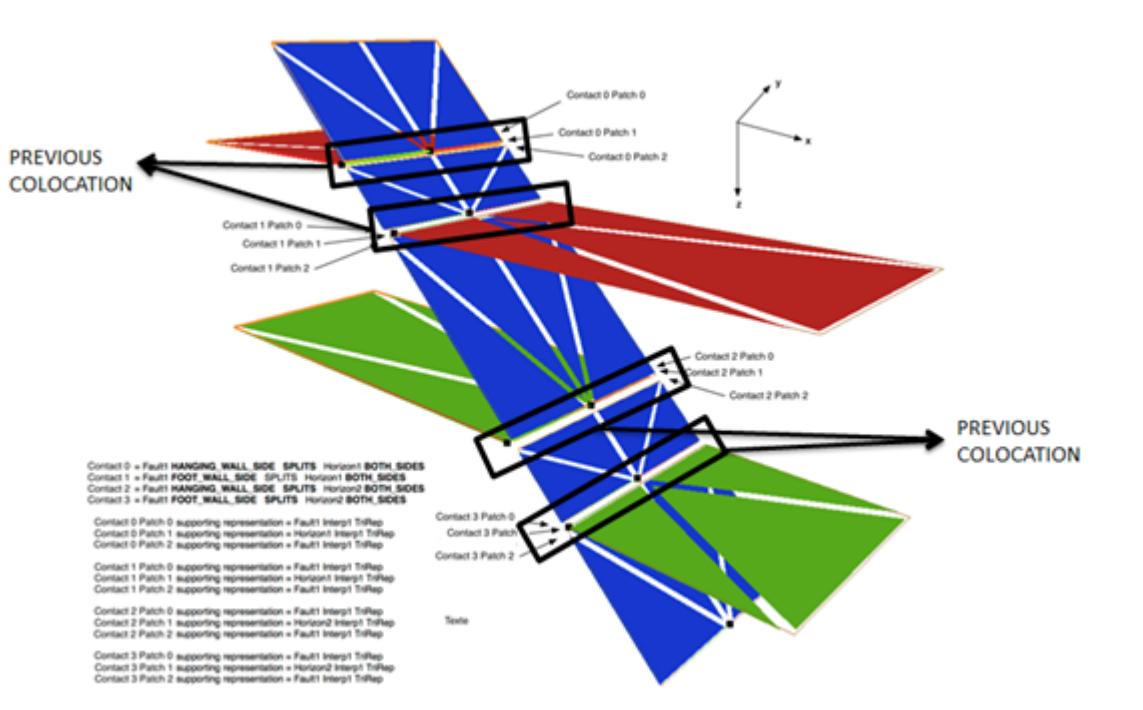


Figure 8-23. Sealed surface framework with previous colocation identified.

The objective of this sealed framework is to avoid geometric differences between two equivalent nodes, difference which could be generated by different 3D gridding software packages that have different geometrical functionality. (For more information on equivalent nodes and colcation in RESQML, see Section 5.5).

This sealed surface framework is an important part of the sealed volume framework, which is discussed in Section 8.3.1.3 below.

Software D exports an EPC_3 package with a lot of new consistent representations, which can be used as a base directly by a 3D grid builder or to set up a stratigraphic organization, which is discussed in the next section.

8.3.1.3 Step 3: Stratigraphic Modeling

Now it's time to build volumic representation, which can be done with a BREP or a 3D grid representation.

8.3.1.3.1 Creating a Stratigraphic Organization Interpretation and a Sealed Volume Framework Representation

Software E imports EPC_3 and is used to define an interpretation of a volume for our model (**Figure 8-24**).

Stratigraphic Column

Rank1 Rank2

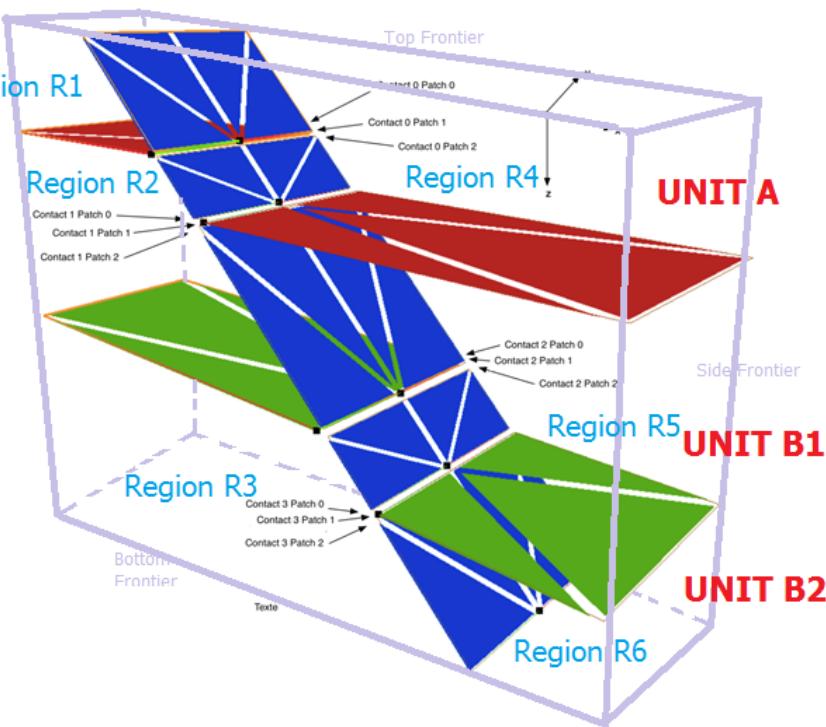
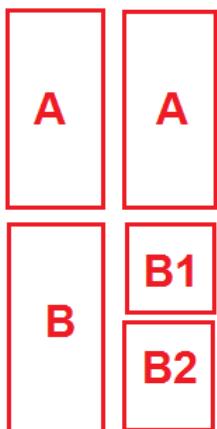


Figure 8–24. Example of a sealed volume framework attached to a stratigraphic organization interpretation.

In Software E, one stratigraphic organization interpretation can be created for each rank of the stratigraphic column. The asset team chooses the rank. In this example, they choose Rank2.

First, we must create the geologic (stratigraphic) units features that are used in Rank2: Unit A, UnitB1, and Unit B2. In the RESQML model, these units are associated with an upper and a lower genetic boundary.

We can also add some interpretation characteristics to these units. For example, Peter was qualifying the Unit A interpretation with a sedimentary siliciclastic composition and Unit B1 and Unit B2 as carbonate composition.

Software E declare as geologic unit features all the stratigraphic unit interpretations that must be identified, which includes Unit_A interp & Unit B interp for Rank1 and Unit_A interp + Unit B1 interp & Unit B2 interp for Rank2. This interpretation constitutes the stratigraphic column, which is attached to Peter's earth model interpretation.

The contact interpretations are defined between these units and follow the rules implied by the deposition mode defined for each stratigraphic unit.

By specifying all these relationships between surface and volume objects, an application can generate a sealed volume framework export, EPC_4, with an earth model interpretation containing a structural and a stratigraphic interpretation with a sealed surface framework and a sealed volume framework (**Figure 8–25**).

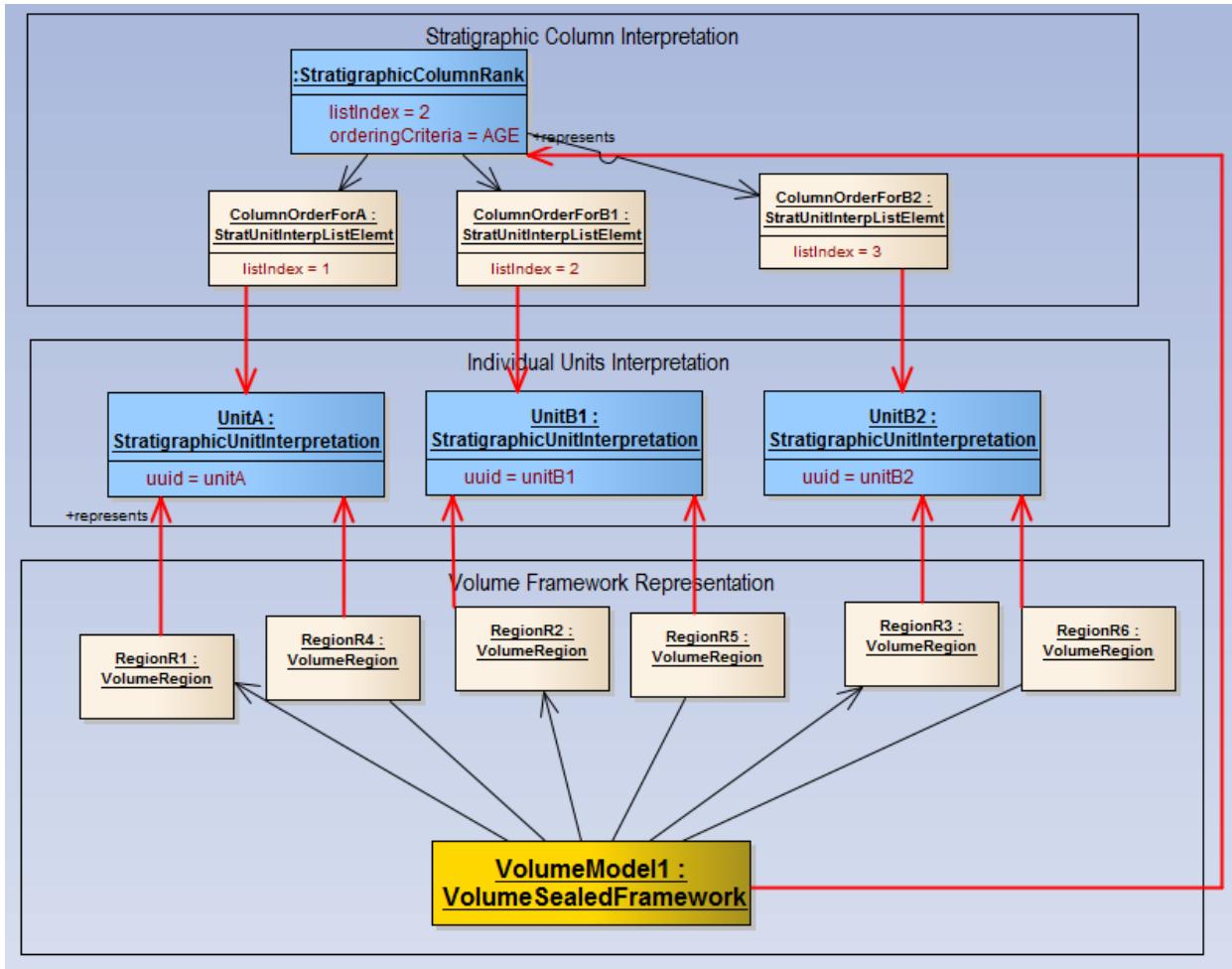


Figure 8–25. This instance diagram shows how a volume framework and a stratigraphic column are associated as shown in Figure 8–24.

8.3.1.3.2 Creating a Stratigraphic Organization Interpretation and a Grid Representation

Software F is "classic" package used to build 3D grid representations. To begin, Software F can import either of these:

- EPC_2, the structural interpretation with a non-sealed surface framework
- EPC_3, the structural interpretation with a sealed surface framework

In either case, the first step with Software F is to follow the same methodology as Software E: Define a stratigraphic organization interpretation by choosing a specific rank (Rank2 in this example) on which to build the 3D grid, and declare as geologic unit features, all the stratigraphic unit interpretations that must be identified (Unit_A interp & Unit_B interp for Rank1 and Unit_A interp + Unit_B1 interp & Unit_B2 interp for Rank2). This definition specifies the stratigraphic column, which will be attached to Peter's earth model interpretation.

Software F can also directly import this information from EPC_4.

Then, Software F can import from EPC_2 or EPC_3 Peter's boundary interpretations (Horizon_1_Peter and Horizon_2_Peter) and use them to build up its 3D grid (**Figure 8–26**).

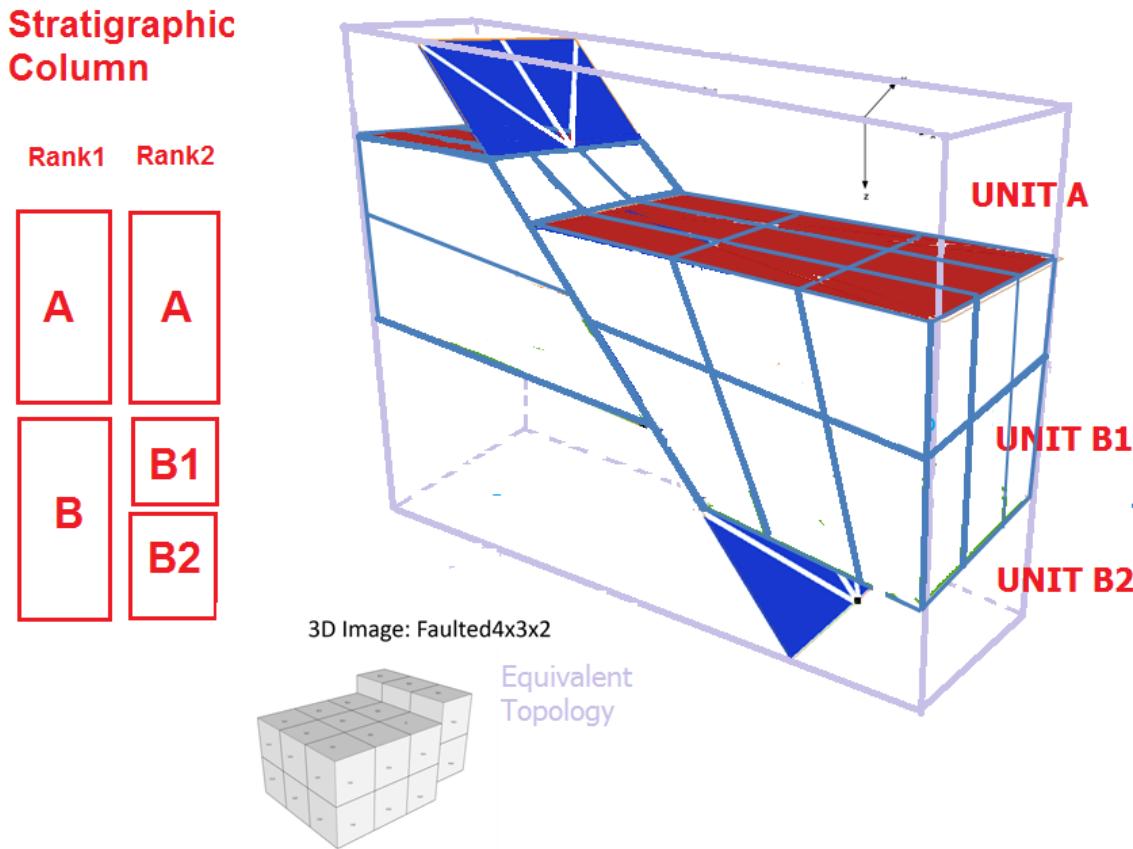


Figure 8–26. Example of a 3D grid representation, which represents a stratigraphic organization interpretation part.

The faulted 4X3X2 grid detailed in Section 10.17.2 can be used here to represent the Unit B1_Interpretation.

Commonly, the 3D grid defined in the Chapter 10 represents a complete rank of stratigraphic organization interpretation.

Software F can then create EPC_5 that contains a description of the stratigraphic organization interpretation in terms of stratigraphic column rank interpretation and the grid representation that is linked to it.

8.3.2 Example 2: Possible Next Steps: How to Reuse/Update a Complete Earth Model for Static Property Modeling and Flow Simulation

The asset team has done the modeling (static and dynamic); the results were not what they wanted. The team decides to start working on Paul's interpretation instead of Peter's.

8.3.2.1 Elements Contained in EPC_5

After all the previous tasks described above, EPC_5 should contain the following data objects:

- One Model Feature: MY_EARTHMODEL_Feature (UUID_EMF, Citation_EMF)
- Three boundary features:
 - HORIZON_1, HORIZON_2, FAULT_1
- Three unit features:
 - UNIT_A, UNIT_B, UNIT_B1, UNIT_B2

- On which the horizon and fault each have two interpretations

Five organization interpretations:

- Peter's earth model interpretation (Peter_EARTH_MODEL_interpretation) belonging to MY_EARTHMODEL_Feature
- Paul's earth model interpretation (Paul_EARTH_MODEL_interpretation) belonging to MY_EARTHMODEL_Feature
- Peter's structural organization interpretation belonging to Peter_EARTH_MODEL_interpretation
- Paul's structural organization interpretation belonging to Paul_EARTH_MODEL_interpretation
- Peter's stratigraphic column interpretation belonging to Peter_EARTH_MODEL_interpretation
- Peter's earth model interpretation contains both his structural organization interpretation and his stratigraphic column interpretation.

Paul's earth model organization interpretation contains only Paul's structural organization interpretation.
(In this example, we did not create a stratigraphic column interpretation for Paul.)

We were working on Peter's interpretation and as a result, we were focusing on horizon and fault interpretation contacts between these interpretations, which are incorporated into Peter's structural and stratigraphic interpretation.

We have these representations linked to interpretations by RESQML data object references:

For all the individual interpretations:

For horizon interpretation, we have:

- Grid 2D representations with their 2D point lattice (4)
- Triangulated surface representations (4)
- Subrepresentation (by interval edge) of the 3D grid (Horizon_1 : interval edge =0, Horizon_2: interval edge=1)

For fault Interpretation, we have:

- Polyline set representation (1)
- Triangulated surface representation (1)
- Subrepresentation (by list of faces) of the 3D grid (Fault_1: face)

For unit interpretation, we have:

- Subrepresentation (by interval) of the grid 3D representation (Unit B1: Interval 0 & 1).

For organizations, we have these representations:

- A non-sealed surface framework
- A sealed surface framework
- A sealed volume framework
- A (4X3X2) 3D grid

8.3.2.2 Strategy that can be Used to Update the Earth Model

Software G (fluid flow simulation software) imports EPC_5 and uses the data to calculate reserve estimations and fluid flow simulations. For this job, Software G mainly uses the 3D grid representation, which represents Peter's earth model interpretation, and produces static and dynamic properties attached to this 3D grid representation, which it saves in EPC_6.

However, the results are not convincing and the asset team would like to look at the model using Paul's structural interpretation instead of the current one.

Two strategies can be used to update the earth model, depending on the magnitude of the changes of Paul's interpretation:

- (A) If the changes are significant and we need to rebuild the grid from the structural model, we use the data contained in EPC_2 and re-execute steps 2 and 3 to generate a 3D grid based on Paul's hypothesis. We have all the necessary information to do this work.
- (B) If the topology does not need to be modified, we can use the data in EPC_5.

The basic workflow is:

- Create a new earth model for Paul's data containing a structural and stratigraphic column interpretation.
- Compare Paul's representations to Peter's.
 - If there is not an important difference between Paul's and Peter's representations (the deformation does not modify the topology of the 3D grid), then create and associate a "clone" of this 3D grid representation with Paul's stratigraphic organization interpretation.

In this case, using Software H (gridding software with editing functions), we replace the XYZ values of the nodes of the previous 3D grid-based representations of horizons to fit with the other representations of the horizons (Paul's representation) and then, update the 3D grid by constraining it on this new position of the nodes while minimizing the deformation for all other nodes.

8.3.3 Example 3: Strategy to Re-Engineer an “Ancient” 3D Grid

A geologist would like to use new RESQML-enabled software to re-evaluate an old study. She is using the software listed in the following table.

Software "Name"	Type/Role of Software
Software I	Retro Engineering RESQML Explorer
Software J	3D grid updater

Software I must export the rebuilt representation into a structural EPC package, and then the 3d Grid updater can take this package and do the work,

While examining the old data, she finds only a report describing the geology, a 3D grid representation (an old GRDECL file) and a collection of wellbores.

With the Software I, the geologist must create an earth model feature, an earth model interpretation, and associate it with this last stratigraphic organization. The geologist can import the GRDECL grid and declare that this 3D grid is a representation of this stratigraphic interpretation.

If she wants to produce more information—for example, she wants to update the earth model interpretation containing structural interpretation and the stratigraphic column interpretation—she can create the Horizon_1, Horizon_2, Fault_1 and Unit_B1 feature, and the corresponding interpretations: Horizon_1_Interp, Horizon_2_Interp and Fault1_Interp and UnitB1_Interp.

The representations are:

Horizon_1_Interp = subrepresentation of GRDECL grid: interval edge =0

Horizon_2_Interp = subrepresentation of GRDECL grid: interval edge =2

Fault1_Interp = subrepresentation of GRDECL grid: face =2,4,5,6, N

Fault1_Interp = subrepresentation of GRDECL grid: pillars =2,4,5,6, N

Unit_B1_Interp = subrepresentation of GRDECL grid: interval = 0,1

A structural organization interpretation also could be created with: Horizon_1_Interp, Horizon_2_Interp and Fault1_Interp.

These representations can be reprocessed and transformed into triangulated surfaces (if needed).

Then, after having acquired new well marker tops, these representations can be fitted to these markers, and the process of rebuilding the structural frameworks, stratigraphic framework, and reservoir grids (as described above) can be done by Software J.

8.4 References

Perrin, M., Rainaud, J-F, et al. 2013. *Shared Earth Modeling: Knowledge driven solutions for building and managing subsurface 3D geological models*. p123. Paris: Editions Technip IS

9 Structural & Stratigraphic Representations

The representation level of the RESQML knowledge hierarchy corresponds to a 3D modeling expression of a feature that was initialized at the beginning of a business process. For example, the same horizon feature-interpretation can have a 2D grid representation or a triangulated set representation.

This chapter explains information specific to structural and stratigraphic representations.

Representations share several key concepts, which are explained in Chapter 5 (page 48).

For information specific to other types of RESQML representations, see the table of contents in this document.

9.1 Introduction

An abstract representation is the parent class of all specialized digital descriptions of a feature interpretation or a technical feature (**Figure 9-1**).

Every representation may be based on a topology and may contain the geometry of its own digital description or may be based on the topology or the geometry of another representation. RESQML has two main categories of representations:

- **Individual representations.** Each Individual representation is specialized by dimension (point, polyline, surface, volume) and represents only one individual geological interpretation (such as, horizons, faults, geological bodies, geological units, and fluid phase units).
- **Organization representations,** which are “sets” of individual representations and frameworks. The individual representations can be collected into a bag or can be associated topologically in frameworks, which are consistent assemblages of different representations to represent interpretations of organizations (such as earth models, structural organizations, stratigraphic organizations, stratigraphic columns, and fluid organizations).

Any of the derived classes of abstract representation, including the grids representations, can be used to describe structural and stratigraphic feature interpretations. Grid representations are discussed in Chapter 10 (page 136).

Because this class inherits from the abstract RESQML data object, all the derived top-level element classes must have an UUID and a citation object attached.

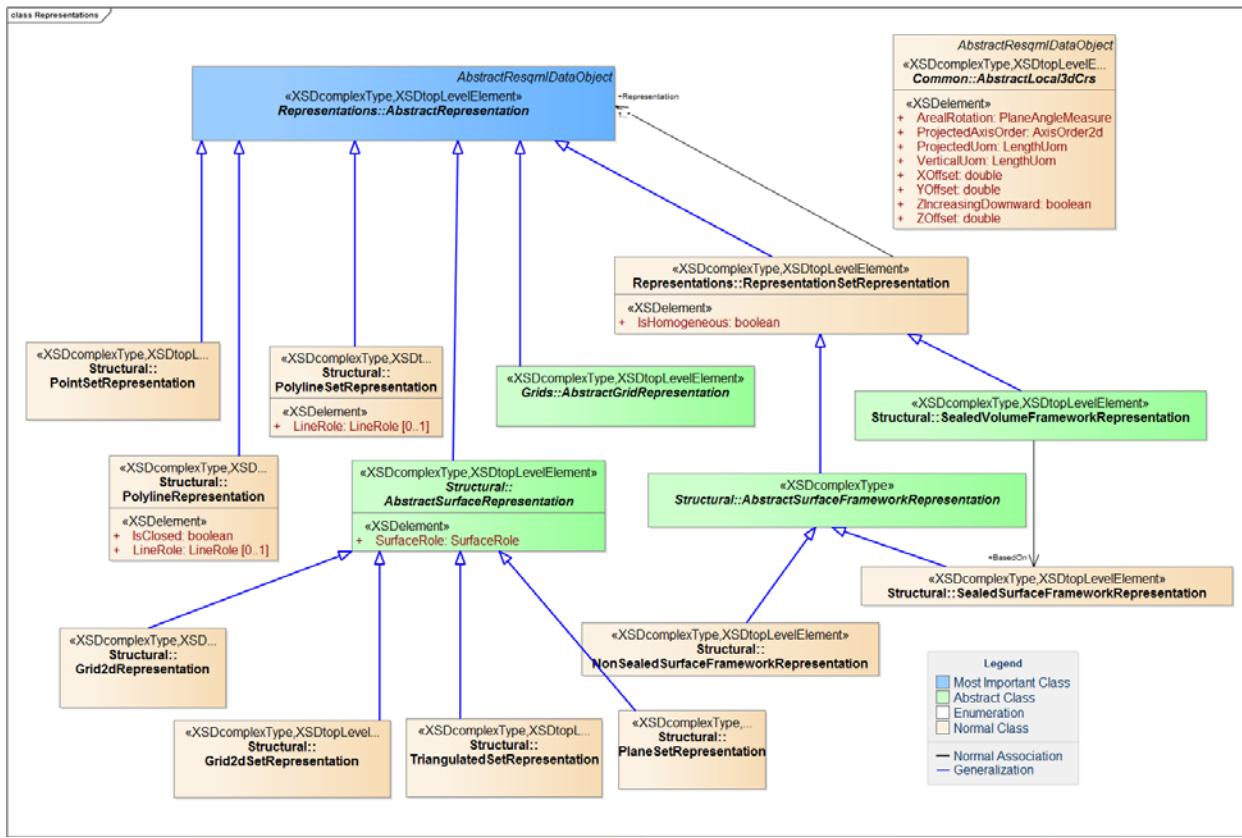


Figure 9-1. Overview of individual and organization representations in the RESQML UML model.

9.2 Individual Representations

Individual representations (which all inherit from abstract representation) are defined by their topology and their geometry. Individual representations may be based on either:

- A discrete definition of the geometry
- A parametric description of the geometry using parametric line descriptions

For discrete representations, the node is the elementary topological atomic element used for each individual representation based on a discrete geometry. Mainly each individual representation contains its topology, which defines how to associate these nodes to represent the individual representation as points, lines, surfaces or volumes.

The topological relationships within a representation may either:

- **Explicit:** For example, a triangle patch, where we must define the node number of the points that are linked together to make up a triangle.
- **Implicit:** For example the index of each node is given only by its order in the array of 3D points of the point geometry (most of the cases).

Non-discrete representations, such as an infinite plane, are also available.

The UML model in EA is organized as shown in **Figure 9-2** and explained in the sections below the figure.

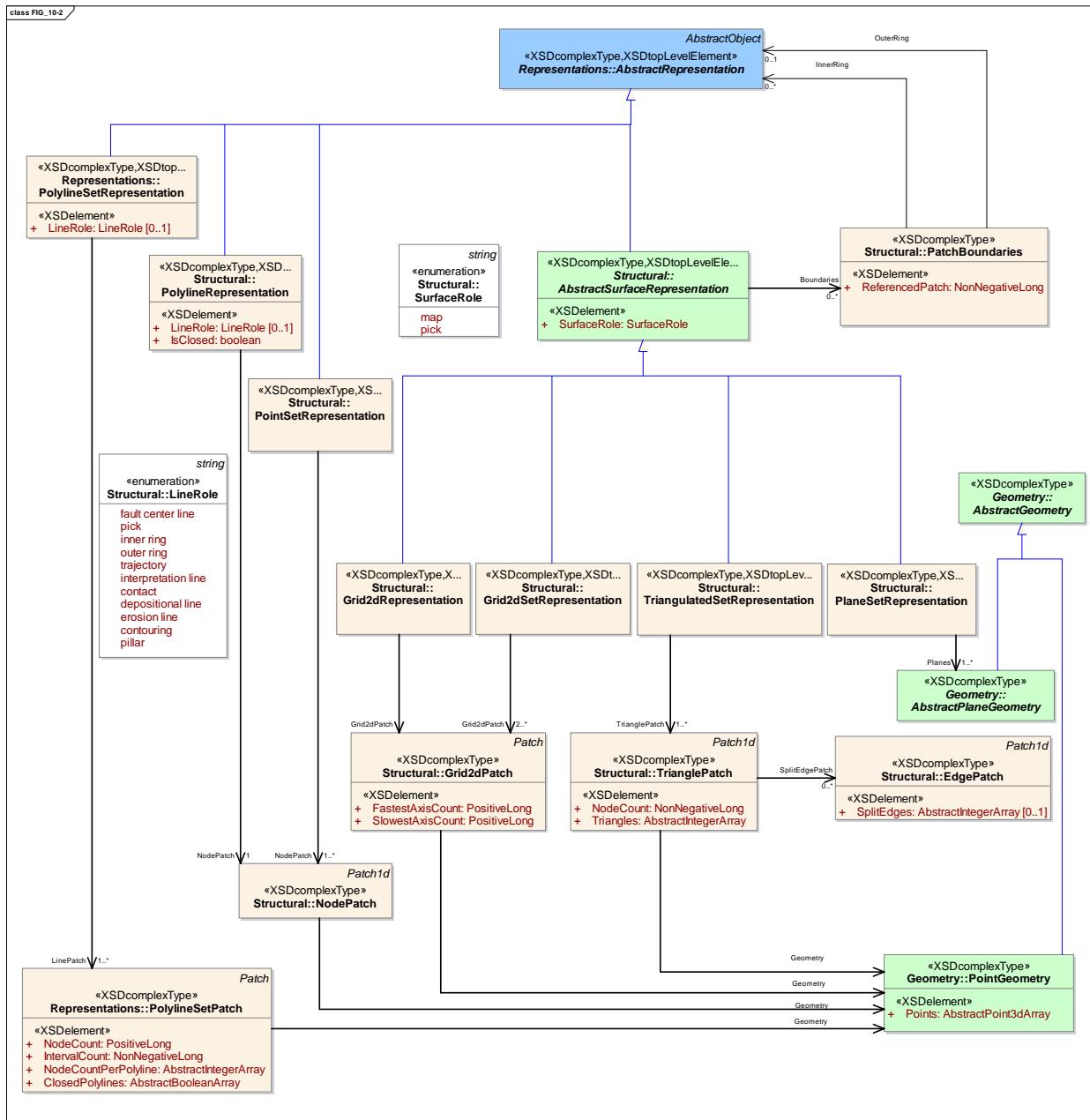


Figure 9-2. Individual representations.

9.2.1 Patch Management

The patch is the atomic element used to transfer individual structural representations. (For more information about patches, see Section 5.3.)

All classes with "patch" in their name (e.g. NodePatch, PolylineSetPatch, Grid2DPatch, etc.) implicitly have a patch index, which is used to uniquely sequence the elements of the patches with the same names.

This (implicit) patch index corresponds to the order in which the patches are stored in the XML instance.

RESQML uses this explicit method to avoid any ambiguity in data ordering among the elements. For example, the triangle indexing of a triangulated set representation consists of multiple triangulated sets, each having its patch index, which specifies the relative ordering of the triangulated patches.

9.2.2 Points

A point set representation is a set of nodes, with no topological information. It can contain one-to-many point set patches of nodes (node patch). Each set of nodes corresponds to one node patch, which has a count value equal to the number of nodes. Each node patch is a 1D patch, which must know the number of nodes contained in the abstract point 3D array, which is instantiated in a point 3D HDF5 array.

9.2.3 Polylines

9.2.3.1 Polyline Representations

A polyline representation is made of a single polyline or "polygonal chain," which may or may not be closed; it can be specified as closed by setting the Boolean value to true. Each polyline representation has an implicit topology. This topology consists of attaching the nodes in sequence by following the order of the nodes contained in the abstract point 3D array, which is an instance of a point 3D HDF5 array. The number of nodes of each polyline representation is given by a "count" value.

For a closed polyline (Boolean = true) the first and the last point are identical (the X,Y,Z value are repeated).

9.2.3.2 Polyline Set Representations

For better data transfer performance (more compact datasets), polylines and polygonal chains can be assembled into polyline set representations. In sets, each single polyline embedded into the set representation has an implicit topology. This topology consists of attaching the nodes in sequence by following the order of the nodes contained in a point 3D HDF5 array, which inherits from the abstract point 3D array.

A polyline set representation has two pieces of explicit topological information:

- The number of nodes for each polyline
- A Boolean to indicate if a polyline embedded into the set is closed or not
- When the "closed polyline" Boolean is true, the first and the last point of the given polyline are identical (the X,Y,Z value are repeated).

Note that the topological relationships between the polylines within in a set are NOT explicit. Reader software can read them in the order in which they are stored, but that order may not be important.

Polyline or Polyline Set Roles. The software writer can specify a role for these polylines from this list: fault center line, pick, inner ring, outer ring, trajectory, interpretation line, contact, depositional line, erosion line, contour line, or pillar.

9.2.4 Surfaces

The abstract surface representation is the parent class of all structural surface representations. For surfaces, the concept of patch is used to help the reader software to associate information that may be delivered in several parts. Each patch is a part of a representation, and all the patches together represent one interpretation of a geological object. These surfaces may consist of one or more patches; these patches may be bounded by an outer ring and several inner rings.

A surface may have two roles:

- Map, a representation support for properties.
- Pick, representation support for 3D points picked in 2D or 3D.

A surface may have patch boundaries for some referenced patches as inner and/or outer rings. The inner and outer rings can consist of any polyline representation, polyline set representation, or (linear) subrepresentation of an abstract surface representation.

The writer must give in the referenced patch the “implicit” patch index corresponding to the order in which the surfaces are stored in the XML instance.

RESQML has these main ways to represent surfaces:

- Triangulated set representations (see Section 9.2.4.1).
- Grid 2D and grid 2D set representations, including lattice (see Section 9.2.4.2).
- Plane set representations (see Section 9.2.5).

9.2.4.1 Triangulated Set Representations

The triangulated set representation contains one-to-many triangle patches (**Figure 9-3**). (For an example of a triangulated representation that includes geometry and topology, see **Figure 9-4**.) The patch 1D count specifies the number of triangles in each triangle patch, the node count gives the number of vertices.

The indices of all the nodes on which the geometry is defined are implicit. For each node, this index corresponds to the order of the nodes in an array of 3D points starting with the first; the array is typically stored in an HDF5 dataset. The topological arrangement of these nodes in triangles is described in triangle patches for each patch in the set.

Triangulated set representations also follow these guidelines:

- Split edges can be indicated in an edge patch instance (to complete the topological arrangement).
- The geometry of the nodes is given by a point 3D array, typically stored in an HDF5 dataset.
- The topology association of the triangles is also typically stored as an HDF5 dataset.

Business Rules. The patch construction is a powerful and important concept in RESQML. For use of patches with triangulated set representations, software writers MUST observe the following business rules.

BUSINESS RULE: Within a patch, all the triangles must be connected.

The patch contains:

- The Triangles stored in a Hdf5Array describing the topology of the triangles.
- The nodes stored in a Hdf5Array of 3D Coordinates attached to a Local3D CRS .

BUSINESS RULE: Each triangulated patch must have its proper number of nodes, which are used to set up all the triangles of the patch. If a particular node is shared between two patches, the location of this node must be repeated twice (or more, as used). This case results in two (or more) nodes with two (or more) different indices, but with the same X,Y,Z value.

To express the association of nodes that belongs to several patches, the writer must use the subrepresentation and subrepresentation identity mechanism. (For more information, see Section 6.3 (page 51).)

BUSINESS RULE: The writer must ensure that each patch is non-manifold, that is, where two or more triangles contact at one point (see <http://en.wikipedia.org/wiki/Manifold>).

BUSINESS RULE: The writer must ensure that a patch is correctly oriented. This means that all the triangles have the same local orientation.

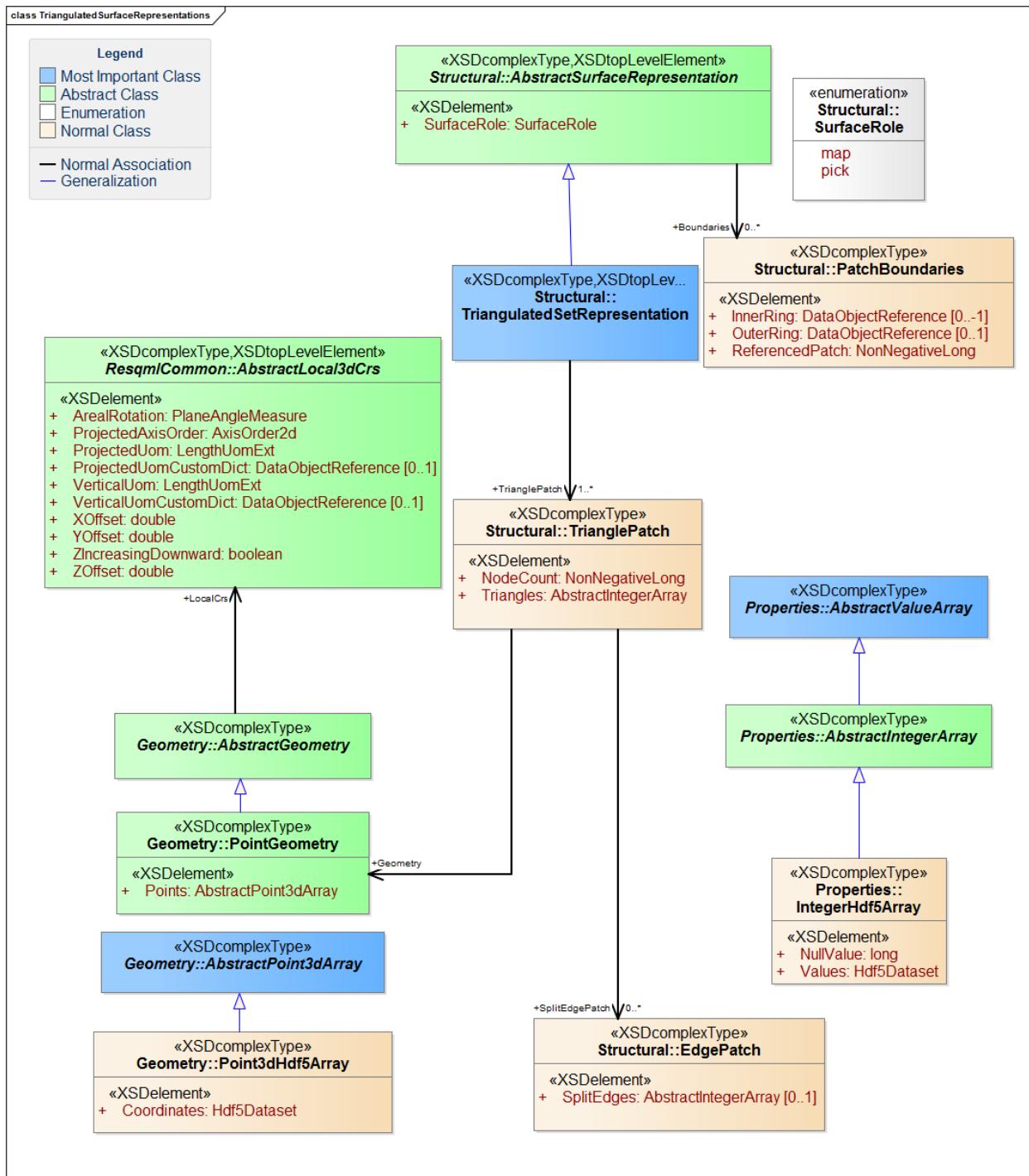


Figure 9-3. Triangulated surface representations.

(Triangulated) Surface Representation (Topology & geometry)

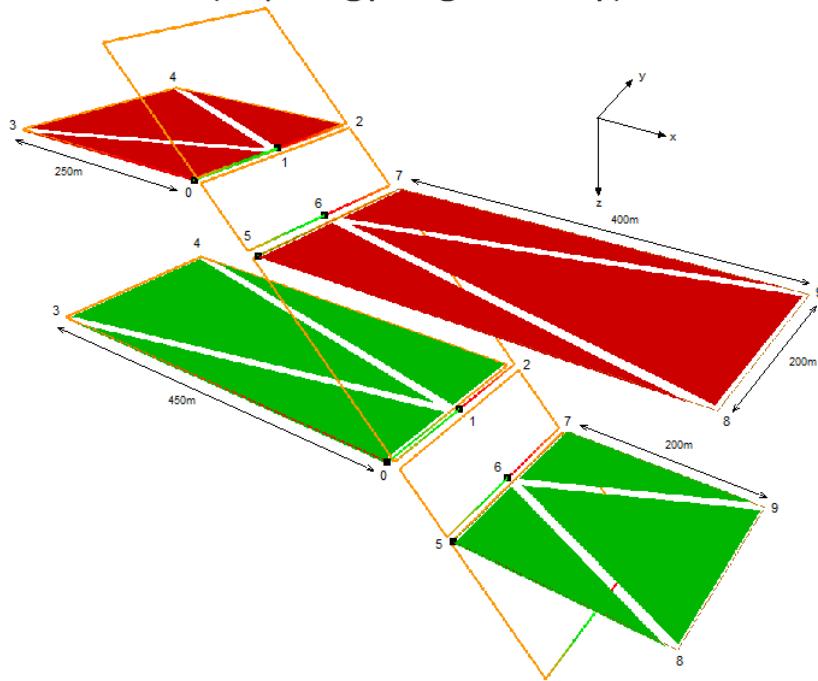


Figure 9-4. Example triangulated surface representation: topology and geometry.

9.2.4.2 Grid 2D Representation and Grid 2D Set Representation

Figure 9-5 shows the UML model for both of these representations, which are explained below.

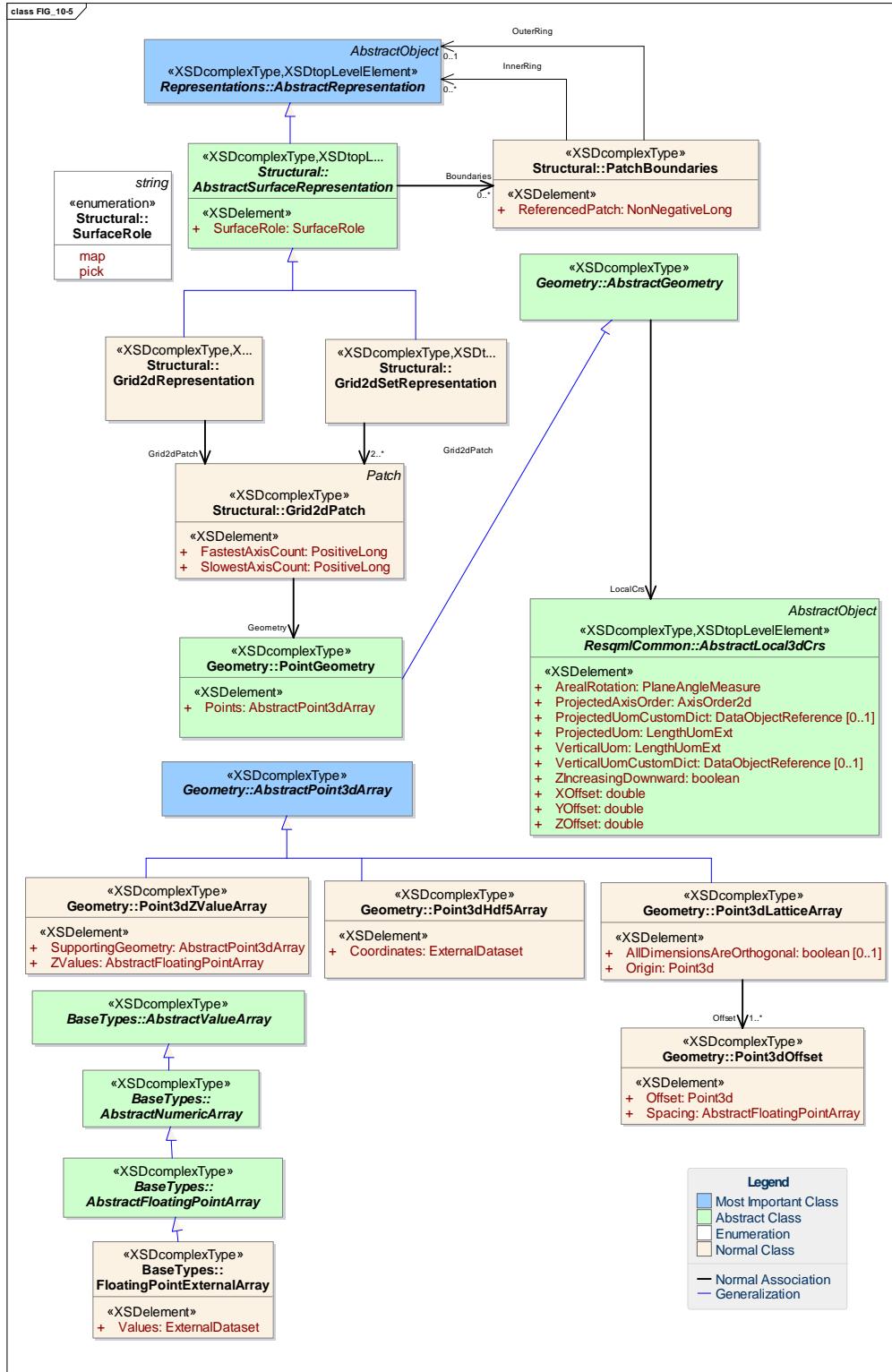


Figure 9-5. Grid 2D and grid 2D set surface representations.

A grid 2D representation is composed of only one patch located on one lattice (defined for 2 dimensions; for more information on lattices, see Section 9.2.4.3 below). The grid 2D set representation is composed of several patches, which correspond to one unique representation for which the different patches can be located by diverse lattices.

The geometry can be accessible three ways:

1. If no point lattice is provided, geometry is directly accessible on a completely defined array of explicit X,Y,Z 3D points (point 3D HDF5 array).

For an array of X,Y,Z 3D points without a lattice, an explicit HDF5 dataset is used. The number of points is implicit, given by the dimension of the HDF5 dataset.

2. Indirectly accessible because the geometry can be obtained by the association between an array of Z values of 3D points (Point3DValues) plus a (2D) point lattice defined as the “supporting geometry” on these Z values.

For an array of Z values of 3D points (Point3DValues) and a lattice, the nodes are aligned in only one 2D array of 3D points:

- To explore the array of Z values corresponding to these patches (one for grid 2D representation, several for grid 2D set representation), the fastest axis corresponds to the number of nodes in the second direction (N2), with index I2, and the slowest axis corresponds to the number of nodes in the first direction (N1), with index I1. The data order follows the index order: I1 + I2*N1.
- In this case, the topology association of the nodes is implicit (each I,J point is connected to I-1,J-1; I+1,J+1; I-1,J+1; I+1,J-1) and uses a 1D index, with values from 0 to N1*N2-1 for a structured 2D grid.

3. If a 3D point lattice (with explicit X,Y,Z location of each point) is given, the geometry is directly accessible.

9.2.4.3 How to Use a Point3D Lattice to Complement the Geometry of a 2D Grid Representation

The lattice itself defines the X,Y location of each point. Using this method is very efficient and allows the selection of an offset and a spacing in each direction.

To implement such a lattice:

- The writer may declare that all directions are orthogonal (which allows the reader to avoid some unnecessary checking).
- The writer must define an origin for this lattice.
- The dimensionality of the lattice is given by the number of “offsets” defined. If you write an instance with one offset only, your lattice will be one dimensional.
- For a multi-dimensional lattice, the ordering of the offsets follows their dimension, i.e., the first offset in the XML instance is the first dimension of the lattice array.

For a grid 2D representation, you must declare two instances of point 3D offsets. The dimension of each is the number of offset points declared for each point 3D offset.

To implement, you must know that a lattice of N offset points is described by a spacing array of size N-1. The offset between points is given by the spacing value multiplied by the offset vector. For example, the first offset is 0. The second offset is the first spacing offset. The third offset is (first spacing + second spacing) offset, etc.

The spacing may be regular, in which case, a count and only one spacing value is given. In other cases, the number of spaces between points is inferred from the dimension of the offset.

9.2.5 Plane Set Representation

Figure 9-6 shows the UML model for the plane set representation, which is used to represent things such as fluid boundary interpretations (such as water-oil contacts) or frontiers (boundaries) that limit a volume

of interest. Like other RESQML surface representations, these planes can have one outer ring and several inner rings to limit their extension.

Each plane can have either:

- a horizontal plane geometry (one Z coordinate).
 - a tilted plane geometry defined by three 3D points.

Guidelines for defining contacts:

- Use BoundaryFeature with a FluidBoundaryInterpretation with a PlaneSetRepresentation.
 - Do NOT use RockFluidOrganizationInterp to group them in a contact set; instead, use a RepresentationSetRepresentation.
 - Link the regions to the planes by using the OuterRing of the PatchBoundaries of the PlaneSetRepresentation (inherited from AbstractSurfaceRepresentation). NOTE: You should have the same number of PatchBoundaries as planes (pointing either to subrepresentations or to the entire representation).

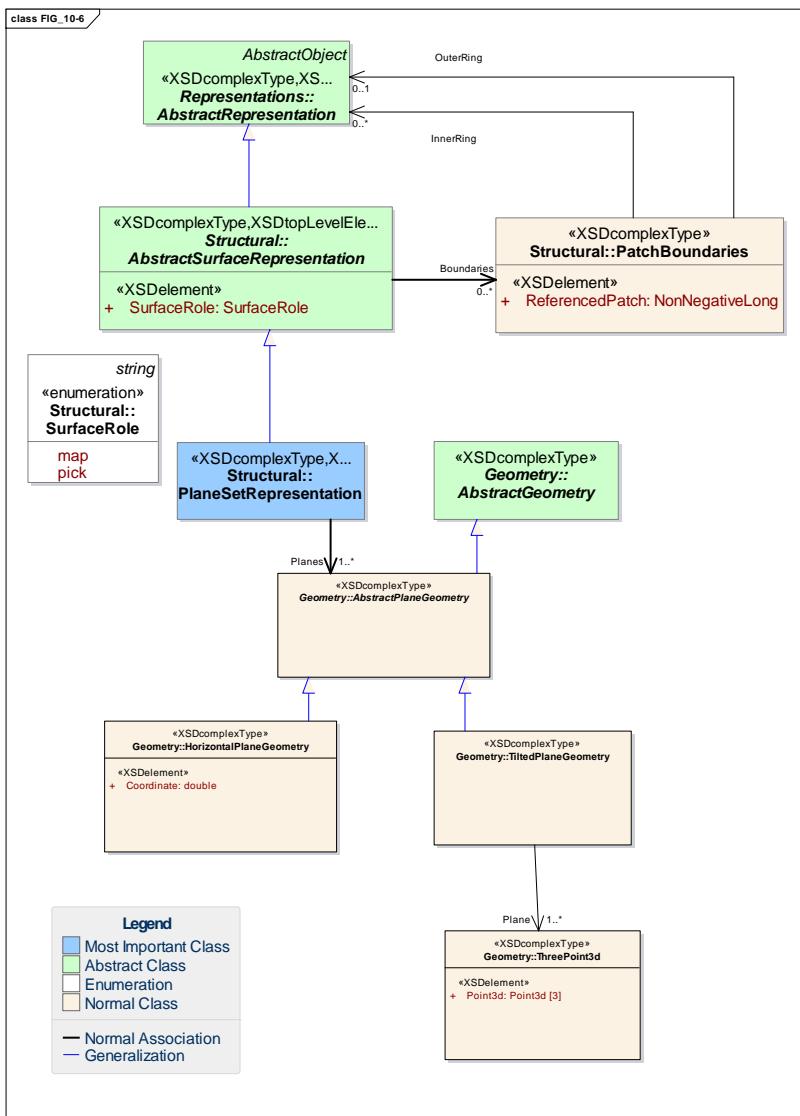


Figure 9-6. Plane set representation.

9.3 Organization Representations (Representation Sets and Frameworks)

Organization representations (Figure 9-7):

- Represent interpretations of earth models, structural organizations, stratigraphic organizations, stratigraphic columns, fluid organizations, etc.
- Are created by using representation set representations to collect together a "bag" of interpretations (either individual or other sets) or associate them topologically into a framework. (For more information on representation set representations, see Section 5.7.)

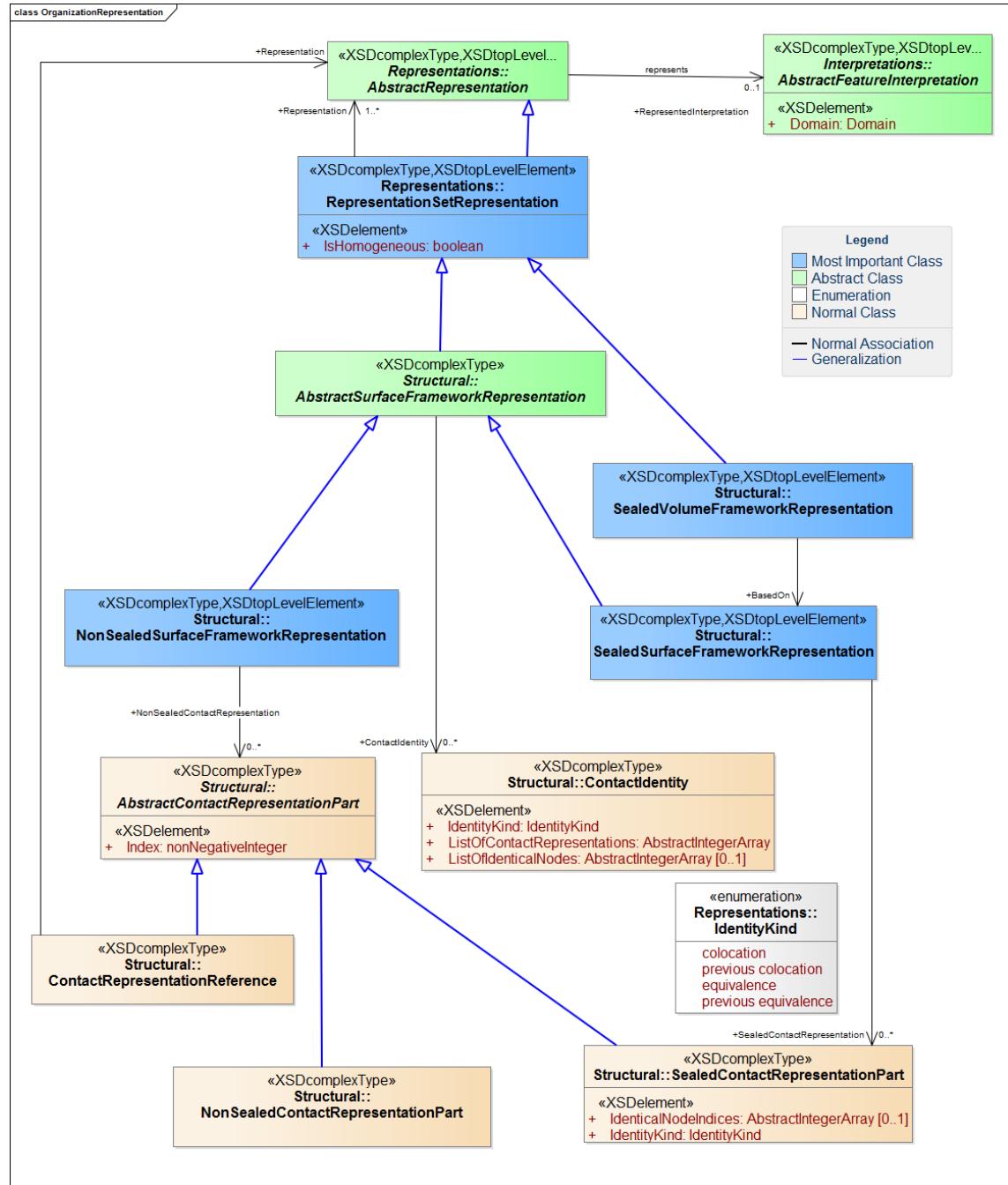


Figure 9-7. Organization representations (simplified diagram).

9.3.1 Representation Set Representation

Representation set representations (**Figure 9-8**) are used to carry an indexed assemblage of individual representations which will be used to set up representations of organizations Interpretations. An implicit representation index is given by the writer as an order in the “sequential” list of Representation UUIDs which will be gathered into the “XML instance” of a representation of an organization interpretation.

This implicit representation index is important because it will be used in the sealed surface framework and sealed volume framework to define the contact patches and the oriented macro faces. In fact the reader must create this index in its own code if he want to reuse it to establish the link between the individual representations and the contacts in the frameworks.

The representation set representation is also the parent class of the framework representations. Because the framework representations inherit from this class, they inherit the capability to gather individual representations into sealed and non-sealed surface framework representations, or sealed volume framework representations. In this context, the representation index is used to define the contact patches.

Other usages of representation set representation are presented in Chapter 5.

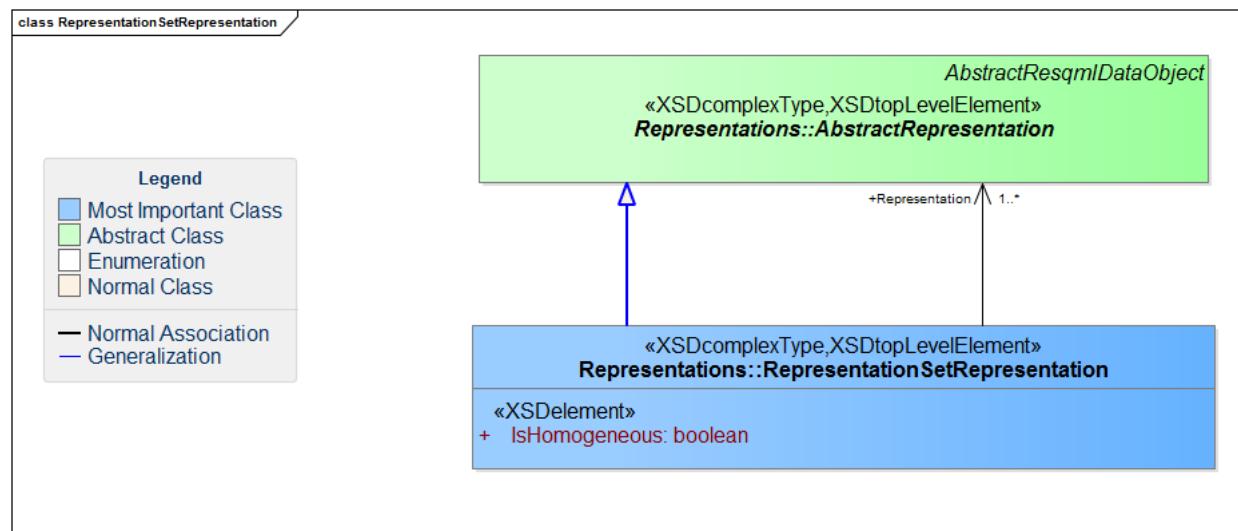


Figure 9-8. Representation set representation

9.3.2 Framework Representations

To create a framework representation, do the following:

- Group individual representations into a set.
- Describe the topological relationships between the individual representations by their contact representations.

These contact representations (in the UML diagram, contact representation parts) can be non-sealed or sealed. If you group only sealed contacts then the surface framework representation is itself sealed. Because a volume framework representation is a boundary representation (BREP), it must be sealed.

Each contact patch is a 1D patch containing count and a patch index, which can be reused to associate other patches together (example use case: “flower faults”). The patch contains a subset of topological elements of an existing supporting representation (identified by its representation index). The topological elements are given by an ordered list of nodes of this supporting representation (integer values).

9.3.2.1 Non-sealed Surface Framework Representation

Figure 9-9 shows the UML model of the non-sealed surface framework representation, which is explained below. **Figure 9-10** (page 130) shows a graphical example of a non-sealed framework.

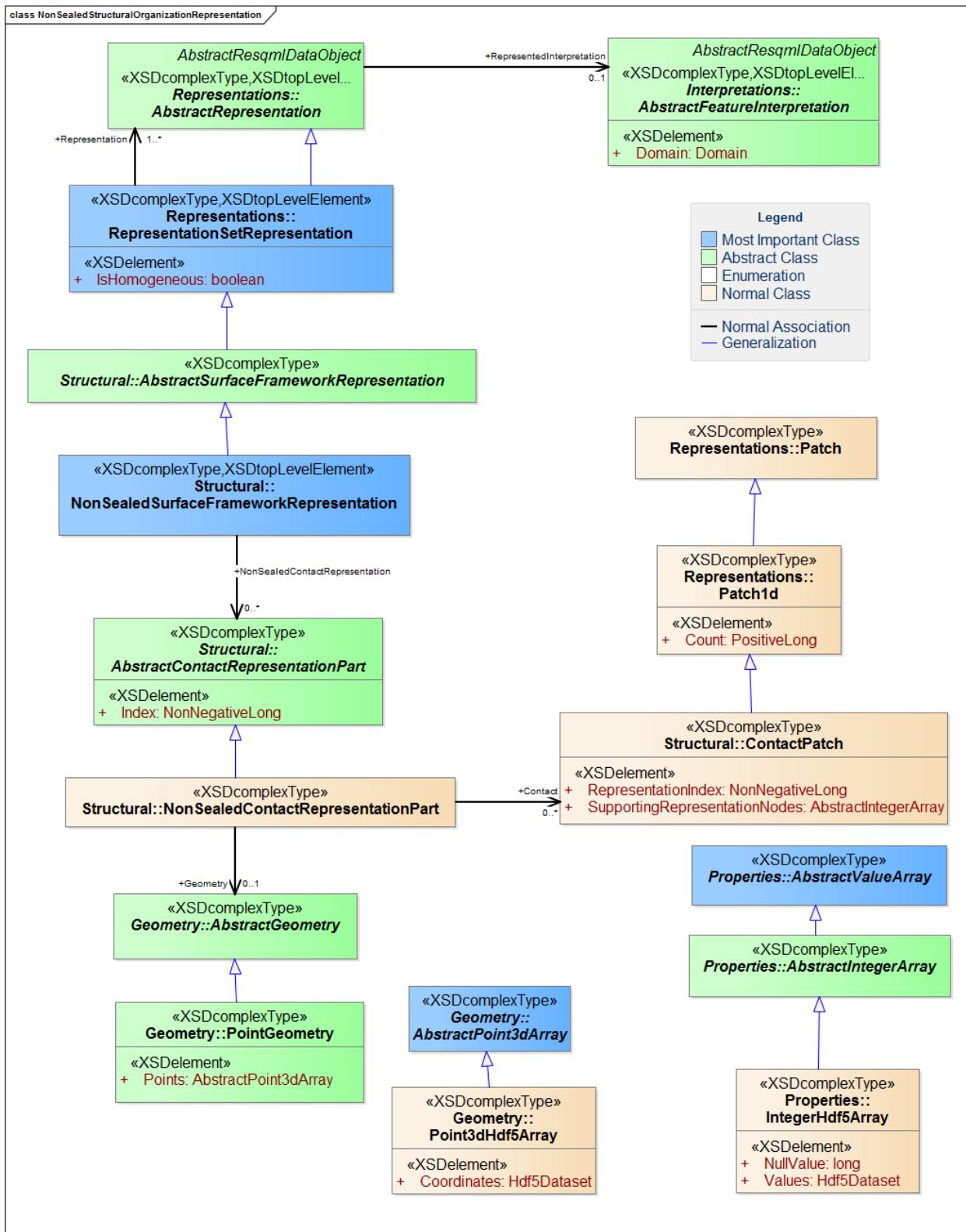


Figure 9-9. Non-sealed surface framework representation.

Figure 9-10 shows an example of a non-sealed framework. The objective of this representation is to gather, step by step, the structural information on boundaries and contacts between boundaries, all along the workflow, even if they are not all totally topologically consistent. A software package can read this information, and based on it, can set up a sealed surface framework representation.

The non-sealed surface framework contains line contact representations (surface/surface contact interpretations between boundaries) and representations of boundary interpretations (horizon, faults, and frontiers). There is no requirement to ensure topological consistency between these representations.

The non-sealed surface framework representation contains a list of representation list elements and a list of non-sealed contact representation parts. Each element of the list has one given “implicit” index given by the writer by the order of writing in the XML instance file.

Each element can have a geometry (one geometry is represented in Figure 9-9 by an abstract point3D array, which can be described by an X,Y,Z point 3D as a polyline. Or each element can be defined by a contact patch.

This construction allows users to include both sealed and non-sealed contacts.

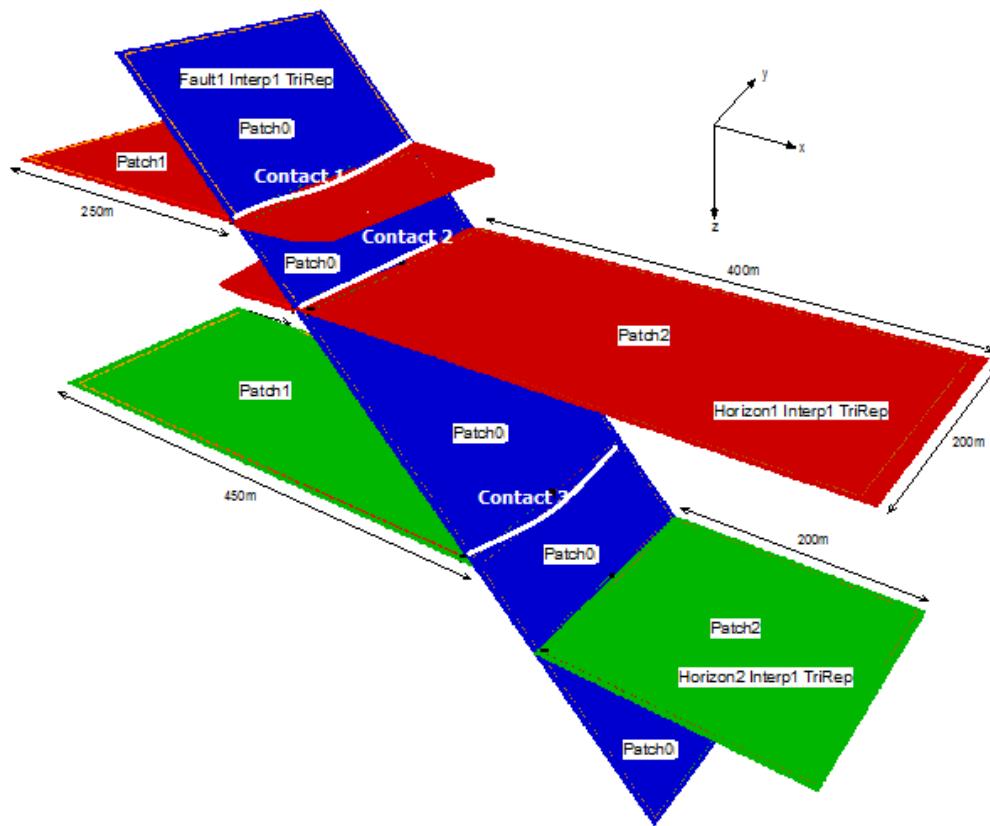


Figure 9-10. Example of a non-sealed surface framework. (The example scenario in Section 8.3 (page 103) is based on this framework.)

9.3.2.2 Sealed Surface Framework Representation

Figure 9-11 shows a UML model of a sealed surface framework representation. Figure 9-12 (page 133) shows a graphical example of a sealed surface framework.

Use this sealed framework to define a surface representation of a consistent structural organization interpretation. Using this representation as a starting point provides gridding software with a clean topology and geometry framework and avoids inconsistent geometry usage. When several nodes are topologically associated, they have absolutely the same geometry, even if they belong to different representations of "structural objects". Software can read this information and set up a sealed volume framework representation or a grid representation on this base.

A sealed surface framework representation contains line contact representations (surface/surface contact interpretations between boundaries) and representations of boundary interpretations (horizon, faults, frontiers) and associations between line contact representations (contact identity). These line contact representation are only defined by their topology (node index) into their supporting representation.

Because this sealed surface framework contains only topological information between the involved contacts and individual representations, the geometry of the sealed contact representation part belongs to the supporting representation nodes.

A complete sealed surface framework representation contains a list of representation list elements, a list of sealed contact representation parts (each element of the list having one given index) and a list of contact identities.

To define a sealed contact representation part and contact identity, use the same mechanism as the subrepresentation and (sub)representation identity (see Section 5.5).

A sealed contact representation part associates together two or more contact patches defined by their representation index. These sealed contact representation parts must have an identity kind of "colocated" or "equivalent." The number of identical node indices associated with a sealed contact representation part indicates which nodes (identified by their common index in all contact patches) of the contact patches are identical. If this list is not present, then it indicates that all nodes in each representation are identical, on an element-by-element level.

A contact identity associates already defined contact patches to add an identity kind ("previously colocated" or "previously equivalent") to this association of contact patches (see Section 5.5).

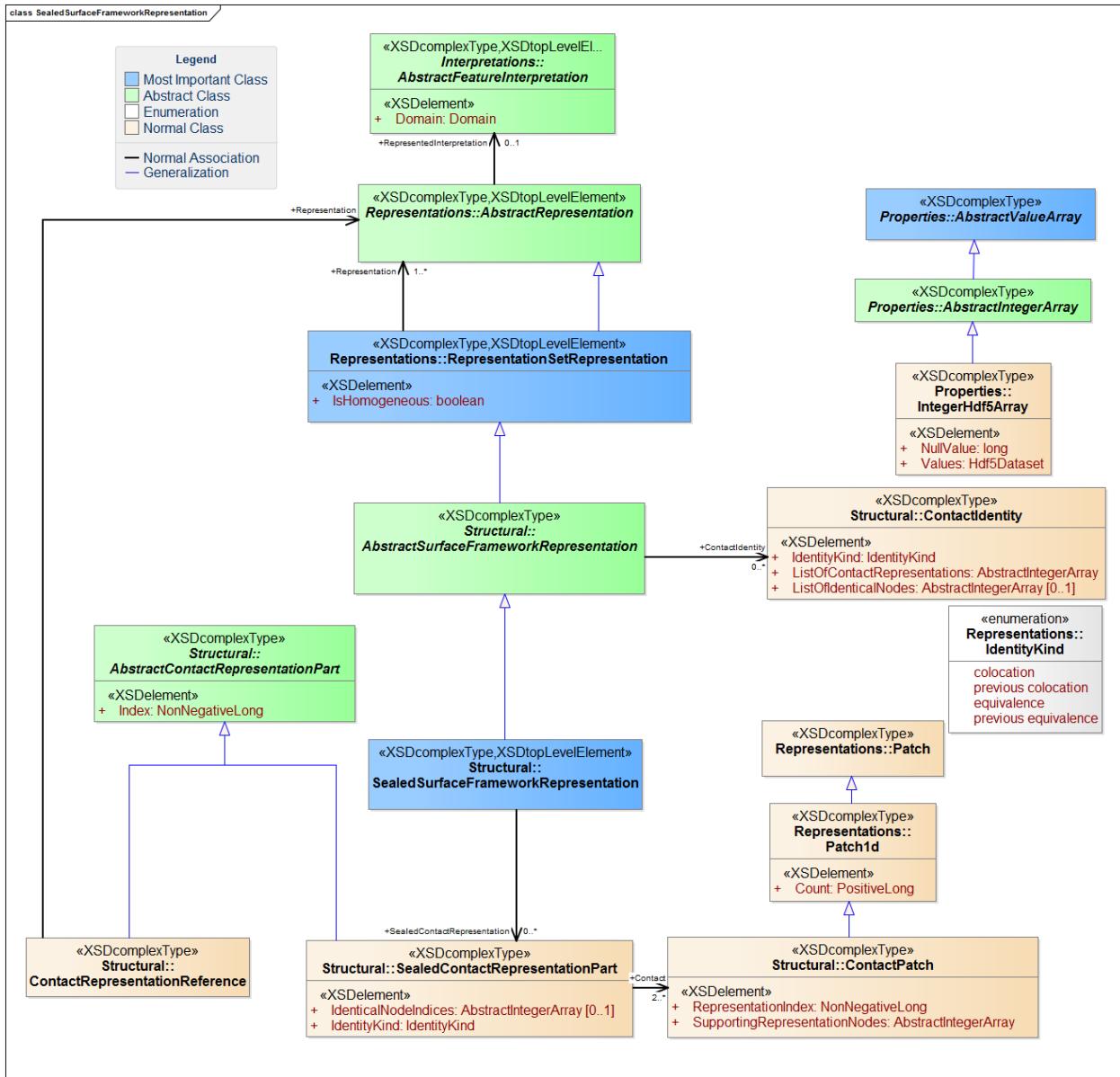


Figure 9-11. Sealed surface framework representation.

Contacts 3D Graphical view

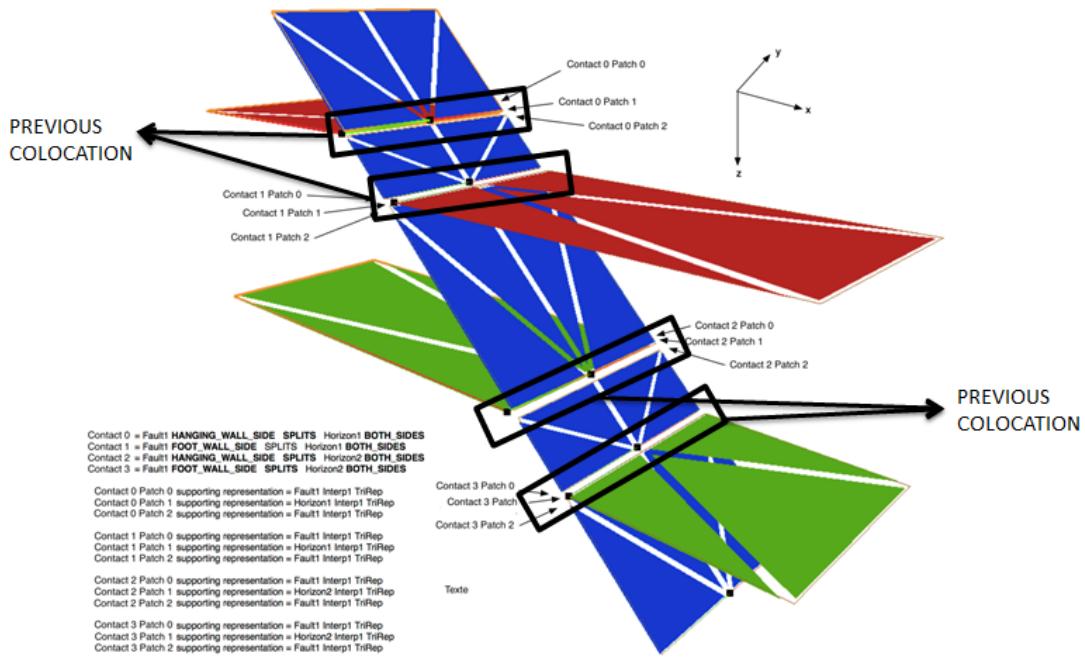


Figure 9-12. Example of a sealed surface framework.

9.3.2.3 Sealed Volume Framework Representation

Use this representation to define a volume representation of a consistent stratigraphic organization interpretation. Using this representation as a starting point gives gridding software a detailed clean topology and geometry framework and avoids inconsistent geometry usage. When several nodes are topologically associated, they have absolutely the same geometry, even if they belong to different representations of “structural objects”.

By delivering a sealed volume framework representation, a software package validates a consistent understanding of the geological stratigraphy interpretation on which property estimations, fluid flow simulation, and basin modeling can be conducted.

Note that the sealed volume framework representation is based on a sealed surface framework representation.

The sealed volume framework representation is described as a boundary representation (BREP): (<http://en.wikipedia.org/wiki/B-rep>). It contains surface contact representations (volume/volume contact interpretations between rock feature interpretations (stratigraphic units, geological bodies, fluid units) and volumes defined by shells).

Figure 9-13 shows the UML model for the sealed volume framework, which is explained below. **Figure 9-14** (page 135) is a graphical example of a sealed volume framework.

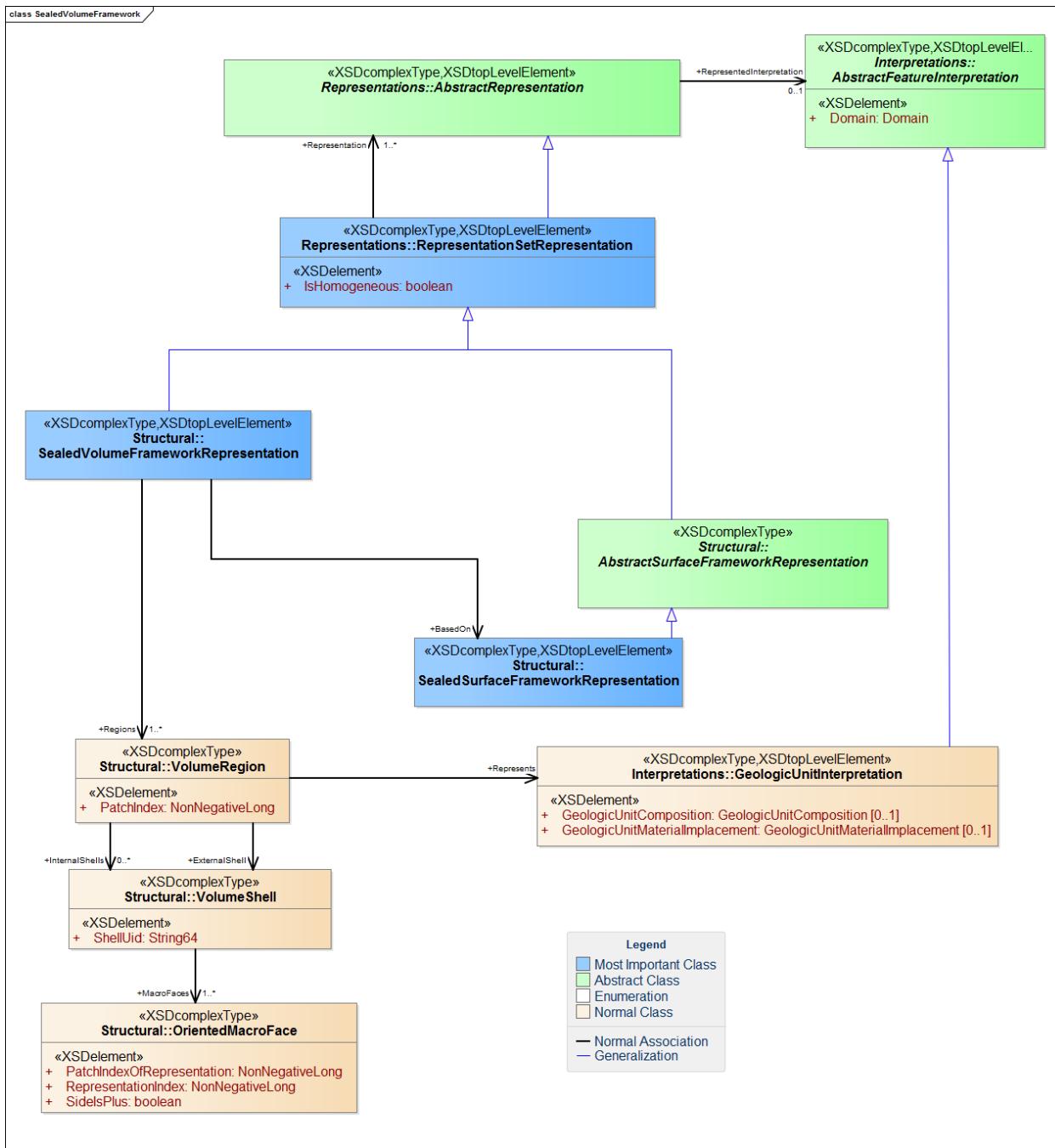


Figure 9-13. Sealed volume framework representation.

A sealed volume framework representation can be composed of several volume regions, each of which represents a rock feature interpretation (a geologic unit interpretation for the geosciML standard (<http://www.geosciML.org/>)).

Each volume region has one external volume shell and can have several internal volume shells.

A shell is composed of the sealed contact representation parts defined in the sealed surface framework representation on which this sealed volume framework representation is based, and by oriented macro faces. An oriented macro face is an element of a volume shell that is defined by a set of oriented faces belonging to boundable patches.

A macroface may describe a contact between:

- two structural, stratigraphic, or fluid units.
- one boundary feature (fault or frontier) and a unit.

This oriented macro face can be described by one patch of an individual representation. To indicate the right surface, the writer software must specify (by a representation index) from which representation patches are “extracted” and must specify which patch is involved by giving its patch index representation.

Usually, a macroface is a bounded open subset of a plane or a curved surface in 3D, delimited by an outer contour and zero and one or more inner contours describing holes.

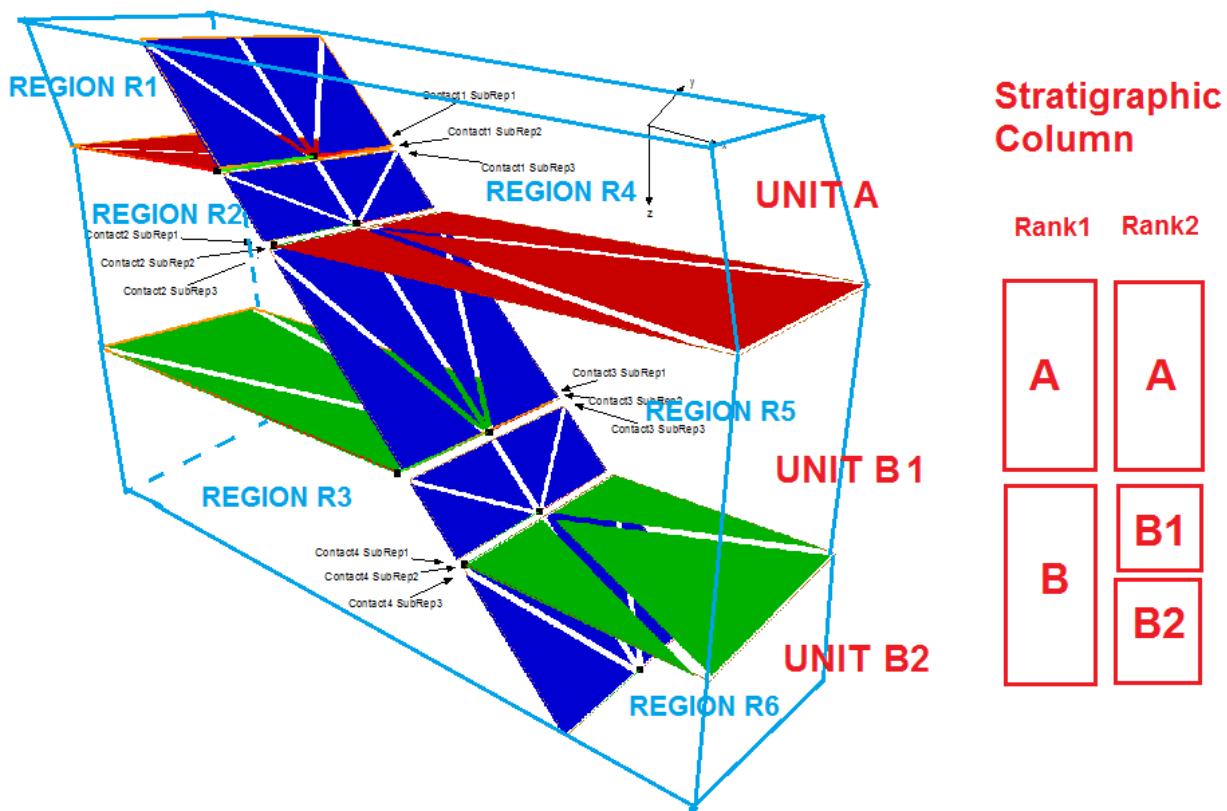


Figure 9-14. Example of a sealed volume framework.

10 Grids

A grid is a RESQML representation that provides a cellular discretization of space. A grid shares three important characteristics with all other RESQML representations:

- A description of the topology (indexing) and geometry of the grid representation.
- A grid, or a subrepresentation of a grid, may provide a representation of an interpretation of a RESQML geologic feature, most often an earth model or a structural organization.
- Properties may be attached to a grid representation, i.e., a grid supplies the topological support for properties.

For information on concepts shared across all RESQML representations, see Chapter 5.

RESQML also provides a closely related “grid connection set” representation, which is based on “cell-face-pairs”, for the purpose of describing the connections between grid cells, and a “blocked wellbore” representation to describe wellbore trajectories discretized on a grid.

Although it may seem reasonable to organize grids by the geometry of their cells, industry applications more naturally segregate grids by their topology, i.e., the dimensionality of the indexing of the cells. RESQML follows this approach and supports six distinct grid classes:

- Three grid classes are fundamental and will be recognizable to most practitioners.
- Three grid classes are combinations of these fundamental classes, and provide support for advanced variations in unstructured grids.

All grids support various extensions, such as higher order cell geometry, although some extensions may only exist for particular classes. For example, only IJK grids support radial grid cell interpolation. The corner point grid supported by many applications is a specific example of an IJK grid, although without all of the extensions now supported in RESQML. This chapter provides:

- A “quick start” to grids, which provides advice on the use of those features that are most basic for export/write and import/read of RESQML grids.
- Detailed sections that describe the objects that contribute to each of the grid classes and how they may be used.
- Examples of the various types of grids.

10.1 Grids: Quick Start

RESQML provides a rich data model for grids, created for the purpose of data exchange. However, because current and developing industry practice is actually quite diverse, the data model is extremely flexible, to the point where no specific application is expected to be able to import an arbitrary RESQML model. Hence, it is important to provide advice which emphasizes the most basic features of a grid. All of these items will be described in much more detail within the chapter, but basic guidance is provided here.

10.1.1 Topology

The primary organization of grids within RESQML is by the dimensionality and indexing of the grid cells. Fundamental classes are IJK (3 indices), column-layer (2 indices), and unstructured (1 index). This is an intrinsic characteristic which should be preserved on both export and import. Changes of indexing (swaps in IJK axis directions, IJK to column-layer, or column-layer to unstructured) are technically possible, but will sufficiently damage a round-trip workflow that such operations are considered to create a new grid, not to transfer an existing grid.

10.1.2 Geometry

With the exception of block-centered grids, which have no explicit geometry, the geometry of a grid is based upon the geometry of points. Points may be specified explicitly, e.g., by three (X,Y,Z) coordinates, or implicitly using parametric spline representations. The geometry of any grid is based upon a lowest

order description but with additive optional extensions. The latter is a deliberate design to facilitate grid transfers.

10.1.3 Feature Interpretations

Grid representations may support a number of geologic and technical feature interpretations. The geometry of a grid often provides a representation for a structural framework and may be deliberately aligned with the inline and crossline lines of a seismic survey.

With the inclusion of fluids, a grid may also represent a complete earth model. Subrepresentations of a grid based on its indexable elements provide feature interpretations of geologic elements, for example, of horizons and faults. Faults should be represented using the “cell-face-pairs” of a grid connection set representation, because this provides information about the local grid topology, which is otherwise difficult to reconstruct.

10.1.4 Properties

Properties may be associated with any or all of the indexable elements of a grid representation, or of the closely related grid connection set representation. The most basic usage is to associate properties with the cells of a grid or the nodes (connections) of a connection set representation. (For more information on properties, see Chapter 7.)

10.1.5 Advice on Grid Export

This section is intended to provide advice on aspects of grid export that experience has shown to be most variable from application to application. It is not an exhaustive list of important and useful grid features.

When exporting a grid, an application should especially preserve information that is difficult or impossible to reconstruct in the absence of the structural framework upon which a grid is often based.

- Include reference to the structural model interpretation from which the grid is derived.
- Export faults as grid connection set representations to support quantitative work, and export faults as a subrepresentation based on the pillars of a grid for qualitative work. (Stair-steps faults cannot be represented simply as pillars and are discussed in more detail, below.)
- Emphasize the use of “cell” and cell-face-pair “node” connection properties in preference to other property attachments.
- Take advantage of the parametric points because these include more information on the geometry of a grid than simply the position of the cell node vertices.
- For a grid with “layer gaps”, consider exporting a version of the grid with extra layers instead of gaps, because many geologic modeling applications do not support grids with such gaps.
- Take advantage of the redefined geometry representation, which allows one representation to be constructed from another. This may be used in a “belts and braces” approach to effectively export the geometry of a representation twice. For example, one representation may use the preferred description based on parametric points on spline curves to describe the coordinate lines of a grid, while another uses explicit (X,Y,Z) points. This ability should allow experimentation between different application vendors as the new standard comes into use.

10.1.6 Advice on Grid Import

- Preserve the topology of the grid by retaining the grid indices and dimensionality.
- Preserve the earth model or structural model interpretation upon which the grid is based.
- Import the lowest order grid geometry even if the higher order geometry or other extensions are not within the scope of the data model of the importing application.
- Be prepared to import parametric point geometry, because the parametric lines upon which they're based provide more information on the geometry of the grid than the positions of the cell node vertices.

Only one of the six grid classes now supported by RESQML has seen widespread historical use in the industry (corner-point grids). Lack of shared data standards for 2.5D unstructured column-layer grids and for fully unstructured 3D grids have limited their use and accessibility in the industry, and have impeded cooperation between different technical domains, e.g., flow simulation and geomechanical calculations. A period of experimentation and learning should be expected as the new RESQML data standards make their impact.

10.2 Example of Grids that can be Transferred with RESQML

Figure 10-1 shows the variety of grids that may be represented in RESQML, labeled A-F. Of these different grids, only image A, an IJK grid, could be transferred in RESQML v1. Grid B is an example of a 2.5D perpendicular bisector (PEBI) grid, which is represented as a RESQML unstructured column-layer grid. As with an IJK grid, there is a column-layer structure, but unlike an IJK grid, the columns are not structured.

The remaining grids are all examples with very different means of representing the grid geometry.

- In Grid C, each unstructured grid “cell” has been replaced by a pore volume and a point, and each inter-cell transmissibility has been replaced by a link between cells. This is an example of the “grid-free” representation used by the newest reservoir simulation applications.
- Grid D is an example of an IJK grid, but the grid cell geometry is specified by cell dimensions (DX, DY, DZ) instead of by the corner nodes of each cell. This is an example of one of the oldest reservoir simulation grid formats, which can now be transferred in RESQML.
- Grid E is an example of a polyhedral unstructured grid. The geometry of each cell is described as a polyhedra in space.
- Grid F is an example of “XYOT” truncated grid, which has an underlying IJK grid description, but includes split and truncated cells at fault block boundaries. The resulting cells may have more than six faces and more than eight corner nodes.

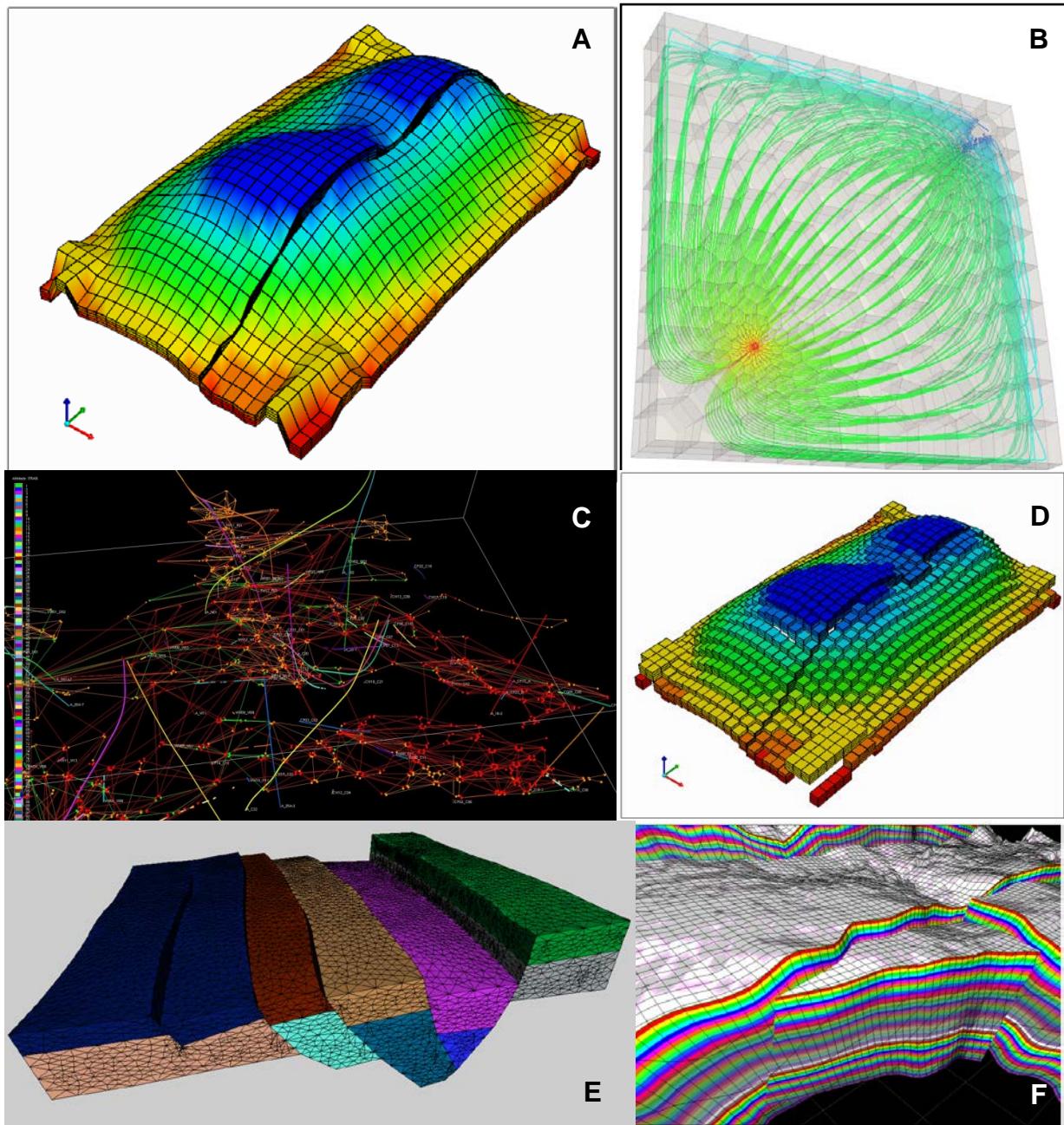


Figure 10-1. Grid examples that can be transferred with RESQML.

Sources: Figures used with permission or re-use policy from the respective sources. Further reproduction prohibited without permission. For full citations and copyrights, see Section 10.18. (A), (D) and (E): Contributed by RESQMLSIG members; (B) Zhang et al. (2011); (C) Martin (2010); (F) Lasseter (2004).

10.2.1 Local Grids

Complex grid patterns may also be developed by replacing portions of a grid locally by another grid (**Figure 10-2**). For example, this allows grids to have high resolution near wells, distorted to follow the geometry of the well, embedded in a coarse regular grid. Local grids were supported in RESQML v1 for corner-point grids. In RESQML v2+ they may be constructed for any of the grid classes.

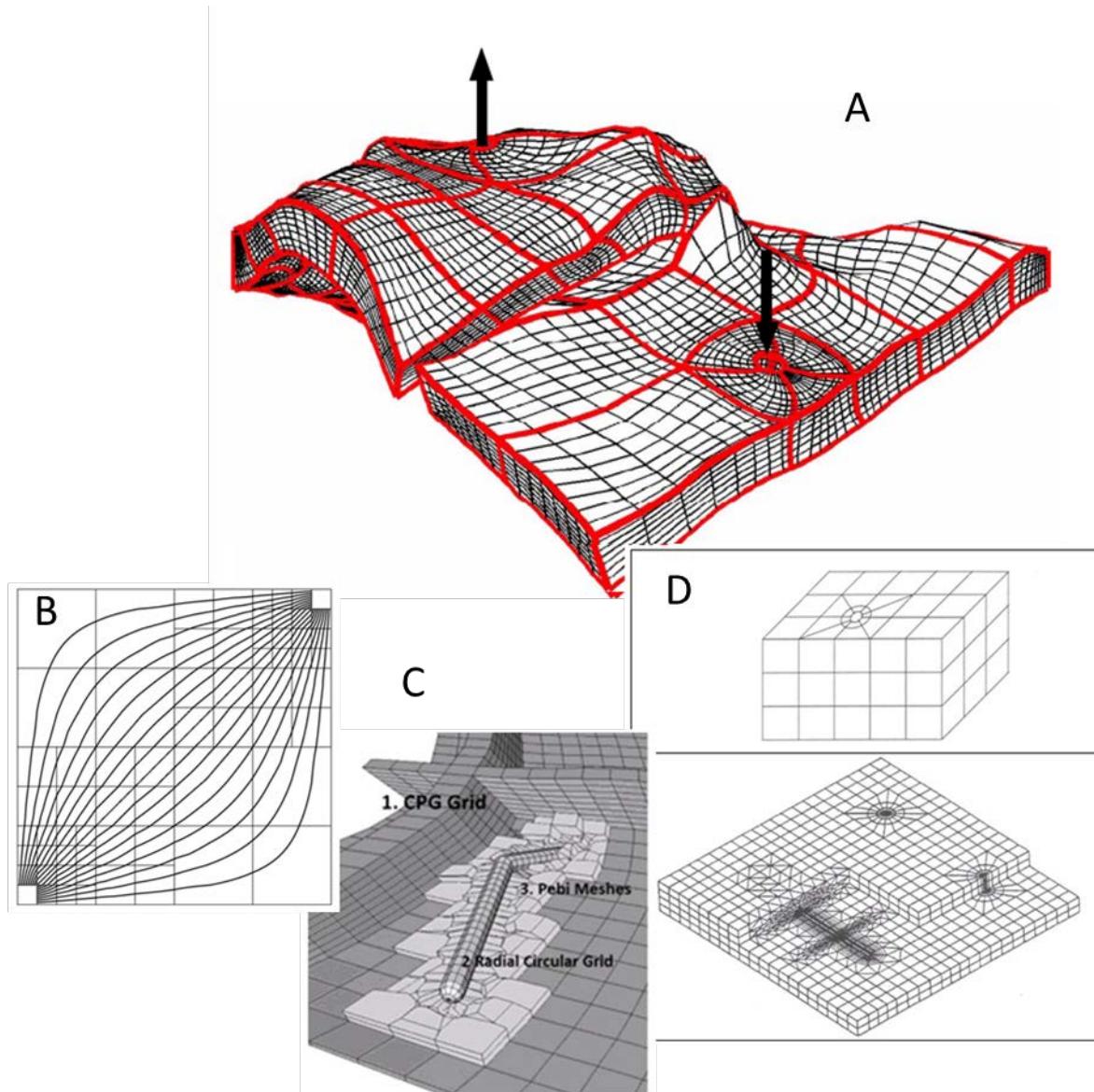


Figure 10-2. Local grid examples.

Sources: Figures used with permission or re-use policy from respective sources. Further reproduction prohibited without permission. For full citations and copyrights, see Section 10.18. (A) Jenny (2001) (B) Zhang et al. (2011) (C) & (D) Perrin & Rainaud (2013).

10.3 Grid Topology

The topology of each grid is organized according to the numbering of the cells of the grid.

Figure 10-3 shows the 6 RESQML grid representations and the 3 supporting abstract grid classes, which are explained below. The grid representation contains the topology (indexing) of the grid, the grid geometry, and grid-specific extensions, if any.

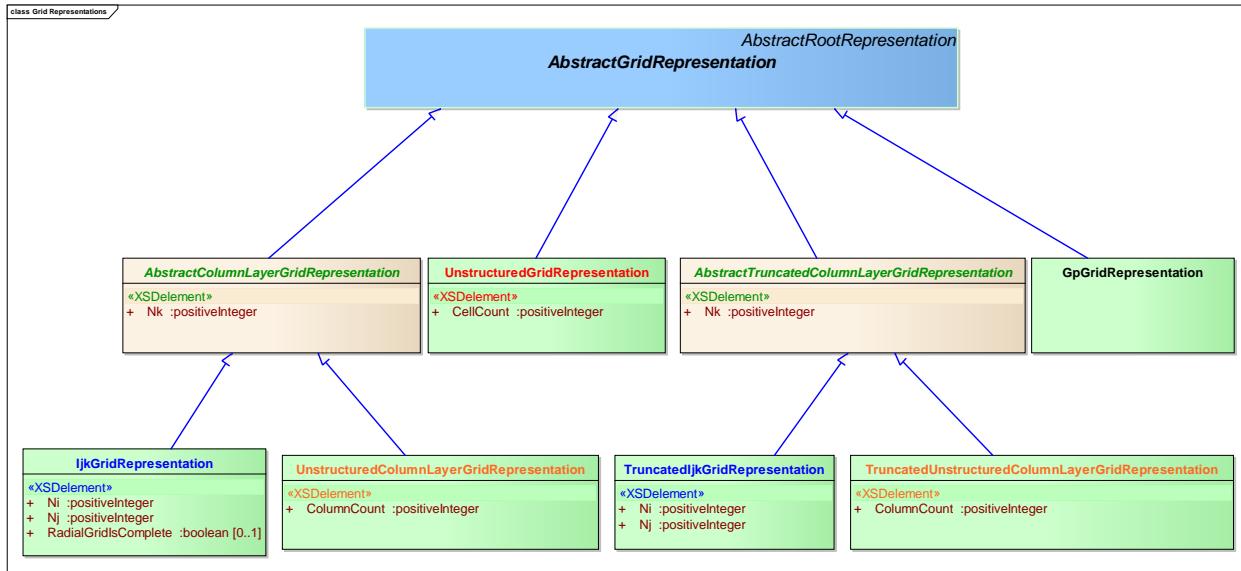


Figure 10-3. The six RESQML grid representations and the three abstract grid classes.

10.3.1 Fundamental Grid Classes

The three fundamental grid classes have 3, 2, or 1 grid cell indices.

- **IJK Grids** have 3 cell indices: NI, NJ, and NK.
 - I0 index = I-1, I=1...NI.
 - J0 index = J-1, J=1...NJ.
 - Layer index = K0 = K-1, K=1...NK.
 - Column index = I0 + NI*J0.
 - Cell index = I0 + NI*J0 + NI*NJ*K0.
 - The corner-point grid of RESQML v1 corresponds to a RESQML v2 IJK grid.
- **Unstructured Column-layer Grids** have 2 cell indices: #Columns and NK.
 - Column index = Column# = 0...#Columns-1.
 - Layer index = K0 = K-1, K=1...NK.
 - Cell index = Column# + #Columns*K0.
 - Unstructured column-layer grids are also known as 2.5D grids and PEBI grids, although there is no RESQML requirement to constrain the grid cell shapes to have perpendicular bisector (PEBI) cells.
- **Unstructured Grids** have a single cell index: #Cells.
 - Cell index = Cell# = 0...#Cells-1.

IJK grids and unstructured column-layer grids are both instances of column-layer grids. As such they each have a layer count, NK.

The (I,J,K) indices for an IJK grid, and the K indices for a column-layer grid have a special status compared to all other indices in the RESQML grid description. These indices may reference information external to a RESQML model, i.e., within a reservoir simulation data deck, which imposes a number of constraints. First, the ordering of the indices of a grid is not arbitrary, but is instead fundamental to the grid. So, if an application flips the order of an index, e.g., to change the parity of a grid, then the resulting grid is a new grid with a new GUID. RESQML may use the Representation Identity to indicate that these two grids are co-located in space, but they should be thought of as two different grids. Second, these indices are constrained by reservoir modeling domain usage to be 1-based. In contrast, all RESQML indices are 0-based. To prevent any confusion, this document and the schema reference the corresponding 0-based indices: $I_0=I-1$, $J_0=J-1$, and $K_0=K-1$.

Notation:

- # prefix and “count” are used inter-changeably in this document, i.e., #Objects = object count.
- # suffix and “index” are used inter-changeably in this document, i.e., Object# = object index.

With the exceptions of NI, NJ, and NK, just noted, elements which provide counts are consistently named “Count” within the RESQML schema. With the exception of the general purpose (GP) grid representation discussed below, the RESQML v2 schema constrains all grid element counts to be positive integers, i.e., degenerate grids are not allowed.

10.3.2 Truncated Cell Grids

Although not as common as the three fundamental grid types, the petroleum industry also uses “truncated” cell grids. These grids are based upon a column-layer fundamental grid, but with the extension that cells can be split and truncated. These splits are used to describe complex juxtaposition and cell shapes at fault block boundaries. The resulting cell indexing is that of the underlying grid, plus additional unstructured cells.

- **Truncated IJK Grids** have 3+1 cell indices: NI, NJ, NK and #UnstructuredCells>0.
- **Truncated Unstructured Column-layer Grids** have 2+1 cell indices: #Columns, NK and #UnstructuredCells>0.

For each of the truncated grids, the cell count is increased by #UnstructuredCells compared to the base column-layer grid. When stored, these additional “truncation cells” are stored in a separate 1d patch array so that the multi-dimensional array structure of the column-layer grid is retained for rapid data access, at least for the majority of cells.

10.3.3 General Purpose Grid

The general purpose (GP) grid is an unconstrained hybrid of any of the other grid types, and provides a grid description toolkit capability. It has been included in RESQML based on the recognition that industry practice in the representation of unstructured grids is still under development. It may have more utility as a research tool than as a data transfer standard. Currently no other industry standard exists to transfer such hybrid grids, so this is a unique capability.

10.3.3.1 Examples of Use

Example 1. A GP grid may be used to represent a discrete fracture model by combining an IJK grid, which describes the geometry of the matrix of a reservoir, with additional discrete fault “cells” for which volumes and transport properties are provided, but with no explicit geometry. The cell indexing of such a model would be 3+1, similar to the truncated IJK grid representation, but without any of the cell truncation description.

Example 2. Because the count of elements is not constrained in a GP grid, it supports degenerate grids. For example, an IJK general purpose grid with NK=0 will be very similar to a grid 2D representation. Similarly, a triangulated mesh could be represented as an unstructured cell GP grid with a cell count of 0, but with nodes and faces.

10.3.3.2 Indexing

Indexing of the GP grid is more complicated than for the other grids because there is no natural sequence or order in how the variety of grid elements within the grid representation may appear. Cell ordering for these hybrid grids is defined using a “patch” construction (For more information on patches, see Section 6.3 (page 51).) The unique index of each patch defines its relative order, i.e., if a GP grid combines multiple IJK grids into a single global grid, the patch indices will specify their order.

10.4 Grid Cell Geometry

RESQML has three grid cell geometries (**Figure 10-4**):

- Block-centered cells
- Column-layer cells
- Unstructured cells

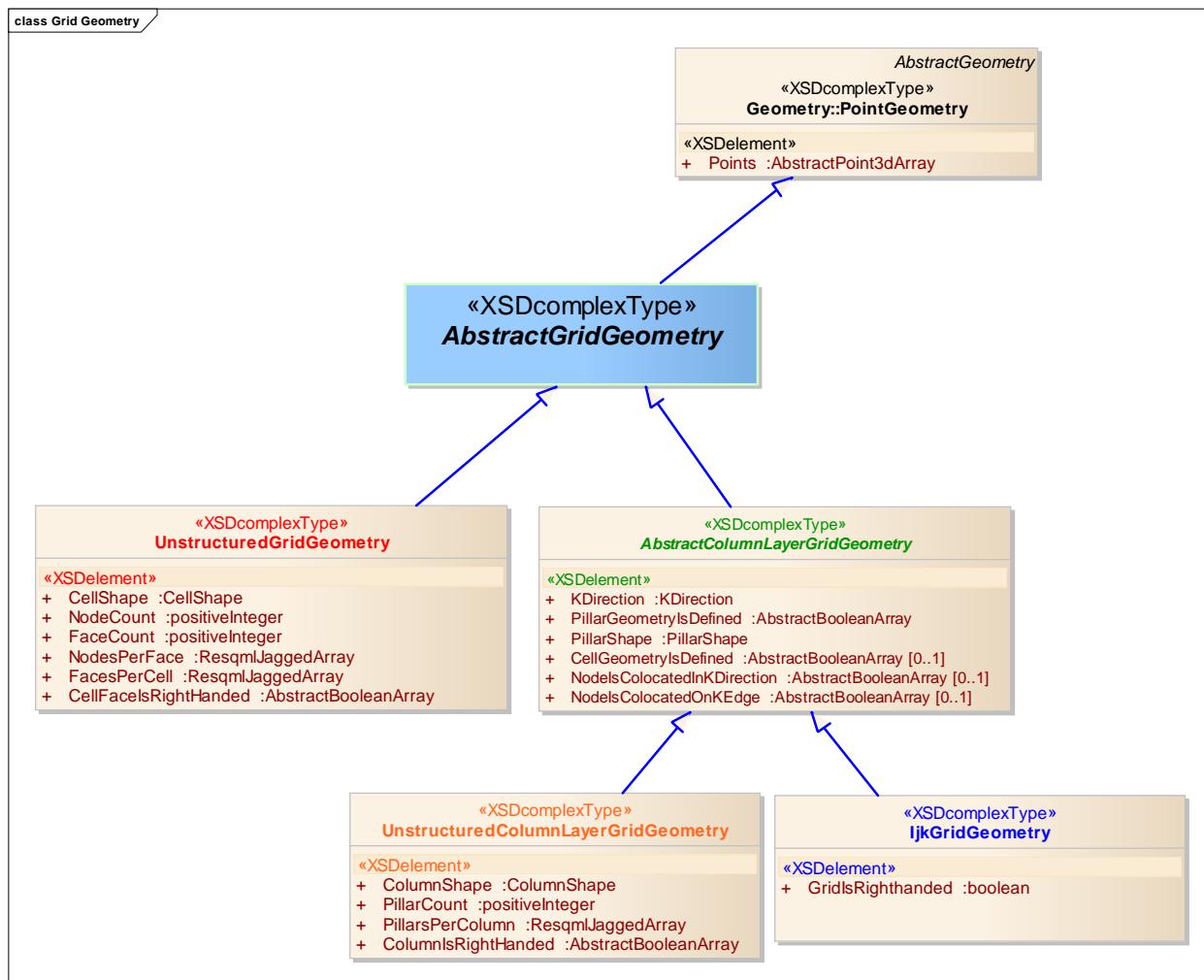


Figure 10-4. The three explicit grid geometries and the two abstract grid geometries, excluding extensions.

If the geometry is explicit, then the RESQML cell geometry is inferred from the cell corner nodes, to which points are attached. The two RESQML examples are the unstructured grid cells, for which cell nodes define the faces of the cells and the column-layer grid cells, for which cell nodes define the coordinate lines of the cells. The column-layer grids may be IJK (structured) or unstructured. The grid descriptions also consist of the topological relationships between cells, nodes, coordinate lines, faces, and/or the other

indexable elements of the grid. Additional points, to support higher order geometries, may also be associated with a grid, as described in Section 10.7.1.

10.4.1 Block-Centered Cells

Block-centered cells are cells with no corner nodes and no explicit geometry. In other words, the grid is indexed as described in Section 10.3 above, but no geometry is attached to any of the nodes of the cells of the grid.

This grid type is used to represent the newest reservoir simulation formats, but also the oldest. Instead of an explicit geometry, equivalent properties are specified on the grid, for example, pore volume and transmissibility, or perhaps equivalent cell dimensions: DX, DY, and DZ. Any grid class may be block-centered, with the exception of the two truncated grids, which are constrained to have explicit geometry.

10.4.2 Column-Layer Cells

Column-layer cells have their geometry described by nodes distributed along coordinate lines. All coordinate lines of a grid have the same number of nodes per coordinate line, although their locations may be colocated. IJK grids and unstructured column-layer grids are the two examples of column-layer grids.

Column-layer geometry supports two specific extensions for use in the description of complex reservoir structures:

- Nodes along a coordinate line may be split to provide additional nodes, which modify the geometry of the adjacent cells.
- Additional faces may be introduced to split and truncate any column-layer cell and create additional truncation cells. This extension is only available for truncated column-layer grids or for general purpose grids.

10.4.3 Unstructured Cells

Unstructured cells have their geometry described by nodes distributed on cell faces. The cells of the unstructured grids and the split cells of the truncated grids are all of this kind.

10.5 Grid Element Indexing

RESQML makes extensive use of indexing to attach properties to a representation, and to specify relationships between representations. For a complex object like a grid, with many different indexable elements, the grid indexing may also provide implicit topological relationships between elements, for example, between the columns and pillars of an IJK grid.

In most cases, the indexing of these objects is uniquely specified by their ordering in **Table 8**.

Table 9—Indexable Grid Elements

Indexable Element	IJK Grid	Unstructured Column Layer Grid	Unstructured Cell Grid
cells	$NI \times NJ \times NK (+) TruncationCellCount$	$ColumnCount \times NK (+) TruncationCellCount$	Count
columns	$NI \times NJ$	$ColumnCount$	---
layers	NK	NK	---
intervals	$NK + GapCount$ (layers + gaps interleaved)	$NK + GapCount$ (GapCount usually 0)	---
interval edges	$NKL = NK + GapCount + 1$	$NKL = NK + GapCount + 1$	---
hinge node faces	$NI \times NJ \times NKL$ (K faces)	$ColumnCount \times NKL$ (K faces)	Count
I0	NI	---	---
I0 edges	$NI_{IL} = NI+1$	---	---
J0	NJ	---	---
J0 edges	$NJL = NJ+1$ (usually) or NJ (if periodic)	---	---
faces per cell	$6 \times NI \times NJ \times NK$ (6 faces per cell)	Count (face order per cell: top + bottom + sides)	Count
nodes per cell	$4 \times 2 \times NI \times NJ \times NK$ (4 x 2 nodes per cell)	Count (node order per cell: top + bottom)	Count
edges per column	$4 \times NI \times NJ$ (4 edges per column)	Count (edge order follows faces)	---
pillars	$NI_{IL} \times NJL + SplitPillarCount$	PillarCount	---
coordinate lines	$NI_{IL} \times NJL + SplitPillarCount + SplitCoordinateLineCount$	PillarCount + SplitCoordinateLineCount	---
nodes	$CoordinateLineCount \times NKL$ (+ SplitNodeCount + TruncationNodeCount)	$CoordinateLineCount \times NKL$ (+ SplitNodeCount + TruncationNodeCount)	Count
column edges	$NI \times NJ + NI \times NJL + SplitColumnEdgeCount$	$UnstructuredColumnEdgeCount + SplitColumnEdgeCount$	---
faces	$NI \times NJ \times NKL + ColumnEdgeCount \times NK$ + SplitFaceCount + TruncationFaceCount	$ColumnCount \times NKL + ColumnEdgeCount \times NK$ + SplitFaceCount + TruncationFaceCount	Count
nodes per face	$4 \times FaceCount$ (4 nodes per face)	Count	Count
edges	$CoordinateLineCount \times NK + ColumnEdgeCount \times NKL$ + SplitEdgeCount	$CoordinateLineCount \times NK + ColumnEdgeCount \times NKL$ + SplitEdgeCount	Count
nodes per edge	$2 \times EdgeCount$ (2 nodes per edge)	$2 \times EdgeCount$	$2 \times EdgeCount$

x signifies a multi-dimensional array, indexed in the order shown, with the first index cycling fastest, etc.
 (+) signifies that this portion of the cell or node data is stored as a separate patch, e.g., 3D+1D or 2D+1D
 + signifies a one dimensional index constructed from multiple indices in the order shown
 + is in the order shown, and will over-ride patch indices, if any (Order exception: Interval index)
 GapCount always vanishes for Unstructured Column Layer Grids, but may be non-zero for GPGGrid patches

For example, the coordinate lines for a column-layer grid should always be numbered to first follow the pillars, with any additional split coordinate lines coming last. As with all other RESQML multi-dimensional arrays, the maximum array dimensionality should always be used. For example, the coordinate line nodes may always be stored as a 2D array, but for unfaulted IJK grids, they should instead be stored as a 3D array, since the coordinate lines are in that case two dimensional. When multiple multi-dimensional arrays are combined, for example, for column edges or for faces, then the storage can be only one dimensional.

The one exception to the sequential ordering specification is for intervals, which consists of the grid layers and the gaps between those layers. In this case, the interval ordering is an expanded version of the layer ordering, with the gaps interleaved between the layers. Specifically, if there are NK layers and $GapCount$ gaps between layers, there will be $NK + GapCount$ intervals. The intervals are, by definition, continuous: the bottom of one interval is the top of the next. As a result, the number of interval edges will be $NK + GapCount + 1$, as shown in **Figure 10-5**. Similar “object edge” constructions appear elsewhere in this table.

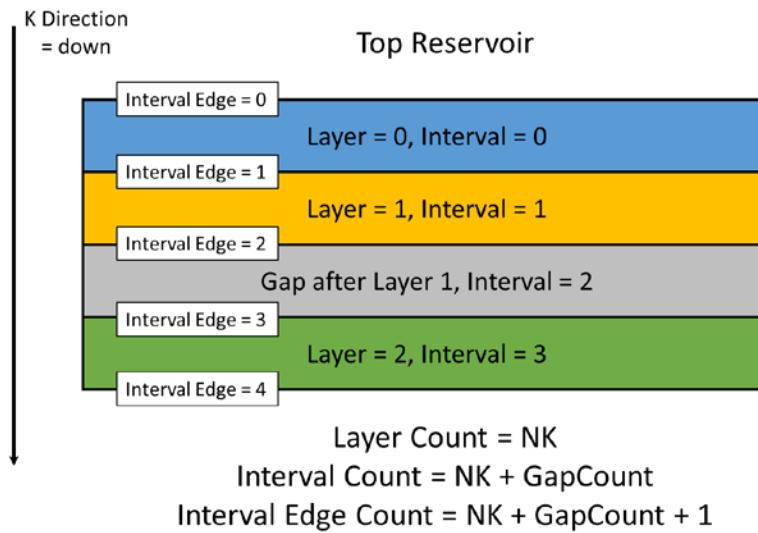


Figure 10-5. Layer, Gap, and Interval indexing for an example three layer grid with a gap between layers.

10.5.1 Indexes that Cannot be Uniquely Determined

Subnodes, split nodes, truncation cells, and general purpose grids are the objects whose index ordering cannot be uniquely determined in a RESQML instance using the relationships in the table. In these cases, a “patch” construction is used in which elements are provided a unique patch index, and then these elements are ordered according to their patch values (for more information on patches, see Section 5.3 (page 51)). For example, cell indexing is uniquely determined for each of the fundamental grid types shown, but when we work with a general purpose grid, multiple grid cell patches will combine to generate a single global cell index. When multiple elements of the same kind need to be combined, they are assigned a unique patch index, whose values determine the relative ordering of the data. This situation may arise even in the fundamental grids, when using subnodes for higher order finite element geometry or properties.

10.5.2 Multi-Dimensional Arrays

Although it may seem reasonable to always merge all objects of the same type into a single long array, this is not always advantageous. For column-layer grids, cells and nodes consist of multi-dimensional arrays, which should be stored in this fashion in HDF5 to facilitate hyper-slabbing and rapid data access. To retain this capability for the bulk of the model data, truncation cells and split and truncation nodes are stored as separate patches, although when indexed, the total count of cells and nodes each includes the multi-dimensional plus the one-dimensional counts. The schema is constrained to have a patch index of 0 for the grid representation itself, so that the relative patch ordering always starts with the multi-dimensional patch, i.e., 2D+1D nodes, not 1D+2D.

As with all RESQML representations, each element of a multi-dimensional array within a grid must have a well-defined 1D index (for more information, see Section 5.2. . For example for a two dimensional array ($N_1 \times N_2$) with indices $I_1=0,\dots,N_1-1$ and $I_2=0,\dots,N_2-1$, then the 1D index is $I_1+N_1*I_2$. For a three dimensional array ($N_1 \times N_2 \times N_3$) with indices $I_1=0,\dots,N_1-1$, $I_2=0,\dots,N_2-1$ and $I_3=0,\dots,N_3-1$, the 1D index is $I_1+N_1*I_2+N_1*N_2*I_3$. This ordering choice is sometimes called “fastest to slowest”, with the first index varying the fastest, and the last index varying the slowest. RESQML is not restricted to three dimensional arrays, for example, the nodes per cell on an IJK grid follow a 5D ($4 \times 2 \times N_1 \times N_2 \times N_3$) array indexing. (The equivalent array in RESQML v1 followed a 6D ($2 \times 2 \times 2 \times N_1 \times N_2 \times N_3$) array indexing, but unlike RESQML v2+, it was never necessary to reduce that multi-dimensional array index to a single 1D index.)

Examples of multi-dimensional arrays include the coordinate line nodes on a faulted grid, where $N_1=\text{CoordinateLineCount}$ and $N_2=\text{NKL}$. However, the dimensionality of an array may vary with context;

for example, the coordinate lines themselves may be either a 1D or a 2D array. In the special case of an unfaulted IJK grid, the coordinate lines are a 2D array indexed by NIL x NJL and the coordinate line nodes are a 3D array indexed by NIL x NJL x NKL.

10.5.3 Elements per Object

Unlike most of the elements in the table that have shared indexing across objects, the “elements per object” indexable elements are defined on a per object basis. For example, the “faces” indexable element enumerates all of the faces where a face shared between cells will only appear once in the enumeration. In contrast, the “faces per cell” indexable element counts a face each time it appears in a cell. For example, for an IJK grid there are $6 \times NI \times NJ \times NK$ faces per cell but far fewer faces. **Figure 10-6** is an example of the local cell face numbering for an IJK grid cell: 0...5. In more detail, to index face n of cell m, then the “faces per cell” index will be $6 \times m + n$, where m and n are both zero based indices. This enumeration is much easier to construct than that for the faces themselves, which depends upon the fault structure of the grid, and the split column edge indexing, not just the grid topology.

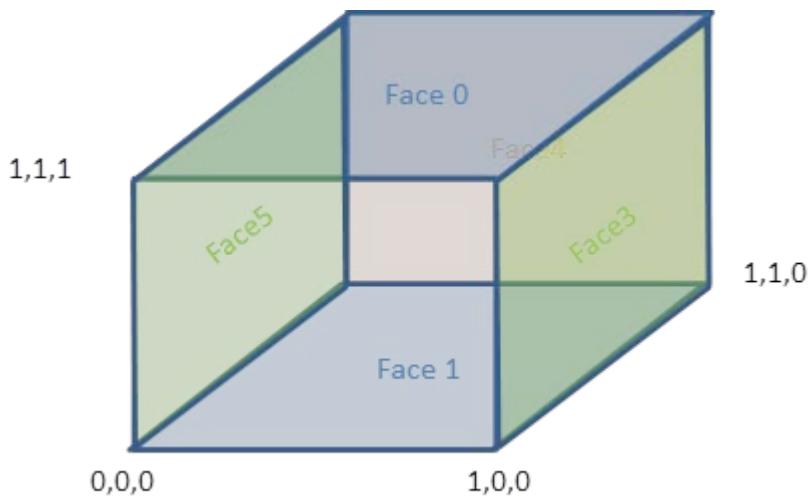


Figure 10-6. Local Faces per Cell indexing for an IJK Grid Cell.

For the unstructured elements, the ordering of these elements is part of the explicit description of the grid geometry. For column-layer elements, the ordering is implicit. By convention, the top ($\gamma=0$) and bottom ($\gamma=1$) faces always come first, followed by the side faces. Similarly, the top nodes per cell are numbered before the bottom nodes per cell.

10.6 Faulted Grids

Discretized representations of faults, using the following approaches, are a part of the minimum grid export specification. Two fault representations are recommended:

- A grid connection set representation, which is a collection of cell-face-pairs, is the recommended and preferred approach as it provides the detailed local fault topology required for flow simulation or for fault property characterization (transmissibility, throw, fault smear, etc.). Historically this information has often been lost on grid export, and has led to some of the largest discrepancies in reservoir performance prediction between different reservoir simulation applications.
- A subrepresentation of the grid with an ordered list of “pillars” as the indexable element is a more basic approach. We expect the pillars to be ordered from one (lateral) extremity of the fault to the other in order we define a single path for the fault representation. This approach works well to provide a qualitative description of the location of pillar-based faults, and will be natural to many geologic modeling applications. However, it cannot be used for XYZ stair-step faults, and it lacks the information on the local discretized topology of the fault, which is present in the grid connection set representation.

- Use of cell faces is NOT a recommended practice. Just as with the use of a pillar subrepresentation, this approach is lacking the detailed local fault topology. However, if required in order to transfer historical data sets that lack the information required for a grid connection set representation, this practice can be reproduced using a subrepresentation based on the “faces per cell” for a column-layer grid or the “faces” of an unstructured grid.

10.7 Additional Grid Geometry and Topology

Each of the RESQML grid geometries support a number of grid geometry extensions. Some, such as radial grids, have already been mentioned. Each of these extensions has its geometry specified as shown in **Figure 10-7** and described in **Table 9**. A few of these extensions, such as the radial origin polyline already have a defined indexing (count is NKL). However, most of the additional geometries require extensions to the topological description of a grid. For example, if we want to attach additional “hinge nodes” to some of the faces of an unstructured grid, the enumeration of the hinge node faces needs to be defined first. The grid geometry attachment has the option to take advantage of a patch index to remove any ambiguity in data ordering.

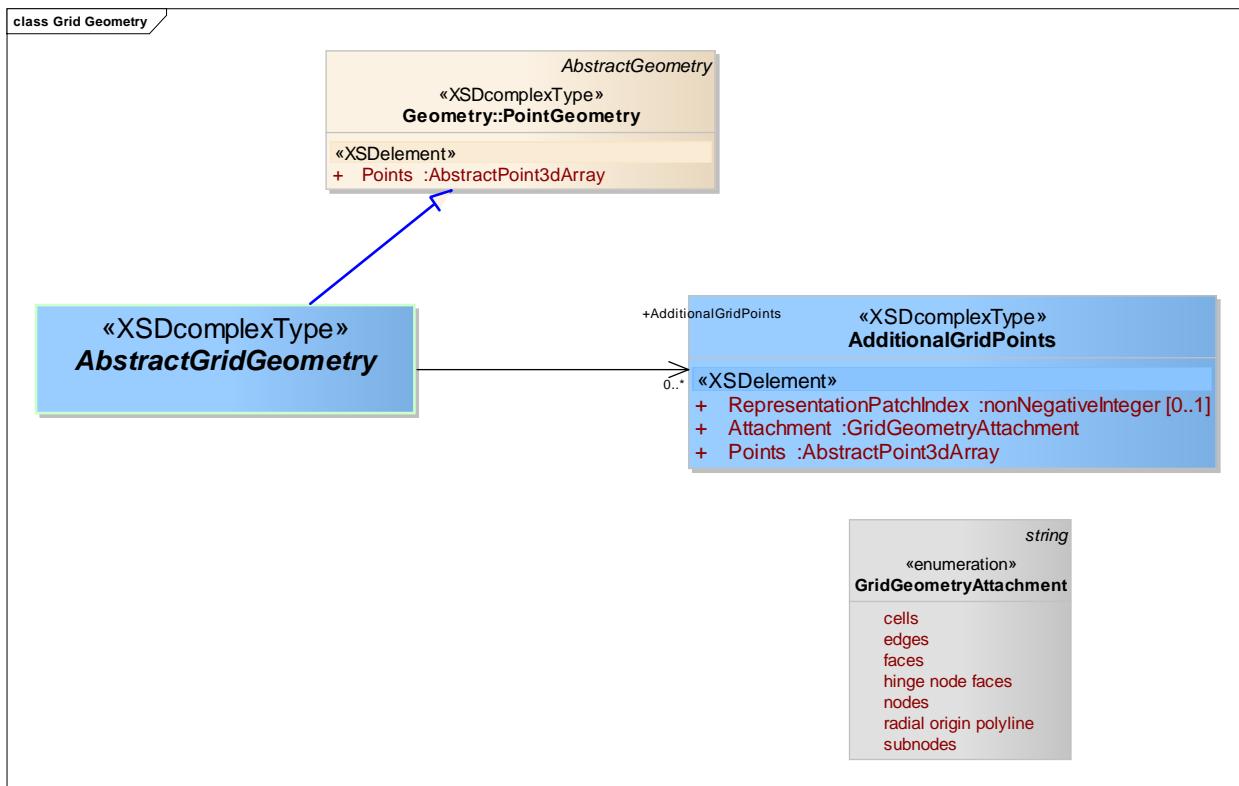


Figure 10-7. Additional grid geometry.

Table 10—Additional Grid Geometry Attachments

Grid Geometry Attachment	Description
radial origin polyline	Count = NKL. Only supported for IJK grids.
hinge node faces	Hinge node faces are the topological support for hinge nodes. A hinge node indicates that the cell face is replaced by a triangulated surface. Hinge node usage originally arose for PEBI grids to supply additional spatial resolution and to uniquely define the surface subtended by the cell face. For column layer grids, the hinge node faces are the K faces. For unstructured grids, the hinge node faces need to be explicitly enumerated.
nodes	Used to specify geometry for the split nodes and truncation nodes.
subnodes	This is a RESQML construction used to support higher order finite element grids by adding additional geometric control to distort the shape of the cells, faces, or edges of a grid.
cells	Identical to a cell subnode defined at the parametric center of a cell.
faces	Identical to a face subnode defined at the parametric center of a face.
edges	Identical to an edge subnode defined at the parametric center of an edge.

10.7.1 Higher Order Grid Geometry and Properties

Recently, the reservoir modeling industry has seen increased integration between reservoir fluid flow and geomechanical calculations, and a corresponding increase in the use of finite element grids. RESQML provides a new vendor-neutral finite element grid description which supports higher order grid geometry and properties. Higher order grid geometry may be attached to cell faces, cell edges, cells, or additional subnodes. This ability provides the geometric support for higher order finite element grids.

Subnode geometry is not sufficient to uniquely define the shapes or volumes of the resulting cells, as the latter depends on the method of interpolation between the nodes and subnodes, not just the nodal positions. A classification of the possible finite element shape interpolation schemes is not included in RESQML, although in some circumstances, an interpolation scheme may be inferred from the corresponding lower order grid description. This is in contrast to hinge nodes, where an explicit interpolation (triangulation) of the cell face is implied.

RESQML treats these geometries as additional information to an existing column-layer or unstructured geometry. This is an example of a deliberate strategy in which we start with a simple grid and then provide optional extensions. This provides guidance to a RESQML reader on how to ignore extensions that are not included within their internal data model, while still providing the possibility of some degree of data transfer. Instead of using flags to describe grid options, the extensions themselves are localized in specific data objects, and the RESQML reader may check for their existence and respond accordingly.

10.7.2 Finite Element Subnodes

Subnodes are used in RESQML to introduce additional degrees of freedom to represent either higher order finite element geometry or properties (**Figure 10-8**).

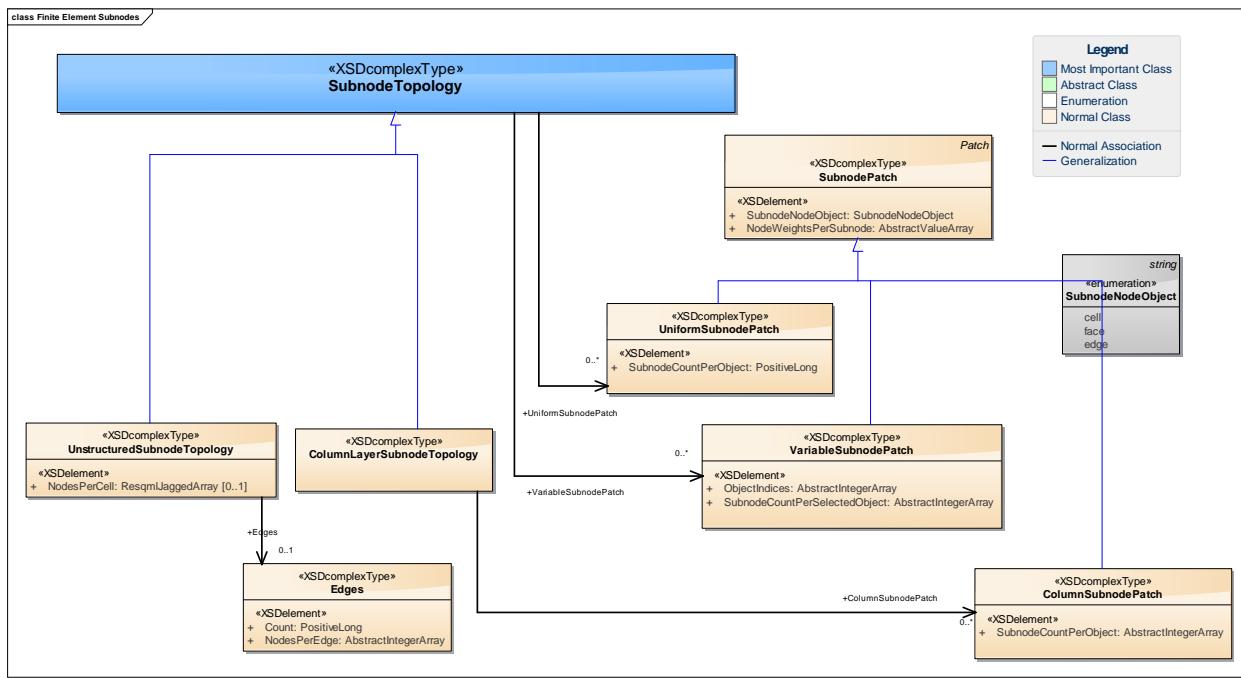


Figure 10-8. Finite element subnode specification.

A subnode is defined with respect to an ordered list of nodes specified by the reference subnode node object. For example, if there are N reference nodes, then a subnode is defined by N parametric weights. For visualization purposes, the inferred location for the subnode is at the weighted average location of all

of the nodes $\vec{x} = \sum_{\ell=1}^N w_\ell \cdot \vec{x}_\ell / \sum_{\ell=1}^N w_\ell$. When RESQML attaches a point to a subnode, it replaces this implicit position with an explicit position.

The additional geometry attachment also includes cells, edges, and faces. These attachments use implicitly defined subnodes with parametric weights equal to unity on the respective objects. This approach is useful, for example, to describe a face subnode defined at the parametric center of each face, irrespective of the number of nodes per face, which may vary from face to face.

The choice of subnode object controls the node count per object for each subnode. Before subnodes may be defined for an object, an ordered list of nodes must first be defined or otherwise known for the same object. The known ordered lists of nodes may vary from grid class to grid class. For example, an ordered list of nodes per cell can be inferred for any of the column-layer grids, but not for the unstructured grids. By construction, such an ordered list is always known for the face nodes, but need not have been defined for nodes per cell or nodes per edge.

The number of subnodes per object may be as uniform or as variable as the associated objects. For uniform subnodes, the count of subnodes is identical for each object. An example is an edge subnode describing a cubic spline, with two nodes per edge and two subnodes per edge, giving a total of four node weights that need to be specified. Column subnodes allow the number of subnodes to vary by column within a grid, while the variable subnodes object allows the number of subnodes to vary per object.

The continuity of the subnode geometry or property is controlled by the choice of subnode object, which may be cell, edge, or face. For example, if the choice is cell, then the geometry or a property associated with a cell subnode has no continuity to adjacent cells. This is how a discontinuous finite element basis is constructed. For continuous finite elements, either face or edge nodes are used, where the continuity of the basis follows the continuity of the object specified.

Subnodes are patches. Grid topology allows the definition of independent subnodes of multiple kinds. To provide more specificity on the subnode geometry or property attachment, subnodes are defined with patch indices, which should provide a unique specification to the attachment.

10.8 Local, Child and Parent Grids

Any RESQML grid may be a child of another grid. Specifically, a grid may inherit its geometry, topology and/or properties from a parent grid (**Figure 10-9**). A grid with explicit geometry may also be a child grid, for example, to support inheritance of properties while providing more spatial resolution of the reservoir grid geometry than was present in the parent representation. The “regrid” data object, described below, is used to specify the parent-child grid relationship. If both geometry and regrid information are provided by the writer, then the geometry must be considered more relevant than the regrid information.

10.8.1 Local Grids

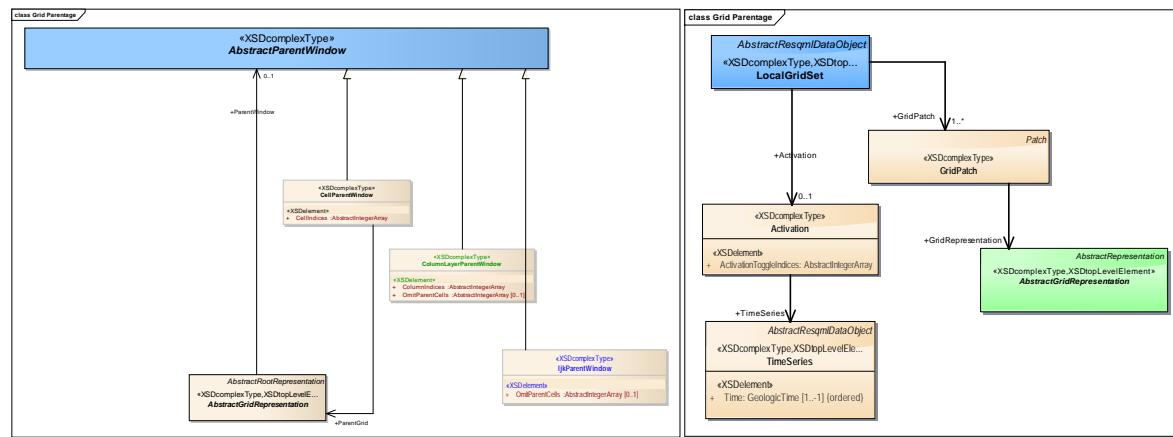


Figure 10-9. Child and local grids.

Key characteristics and differences between local grids and child grids:

- Parent-child grid relationships support multi-scale workflows in which more than one grid covers the same volume in space.
- In contrast, local grids are used to replace a portion of a parent grid by its child. Child grids that are listed as part of a local grid set are defined as a local grid.
- Multiple local grid sets are supported, and each set may include multiple local grids.
- **Time-dependent activation.** RESQML provides optional time-dependent activation and deactivation of local grids. Unless otherwise indicated, a specified local grid replaces a portion of a parent grid. However, if the optional activation object is present, then local grids do not replace a portion of the parent until activated. The state of the local grid may be toggled multiple times between inactive and active.

10.8.2 Parentage and Re-Gridding

A child grid is related to a parent grid by a parentage construction. The parentage construction consists of a parent window, which is a collection of parent grid cells, with potentially a re-gridding description between the parent and child grid. Re-gridding is a one dimensional operation in which the number and size of cells may be changed between the parent and child grids (**Figure 10-10**).

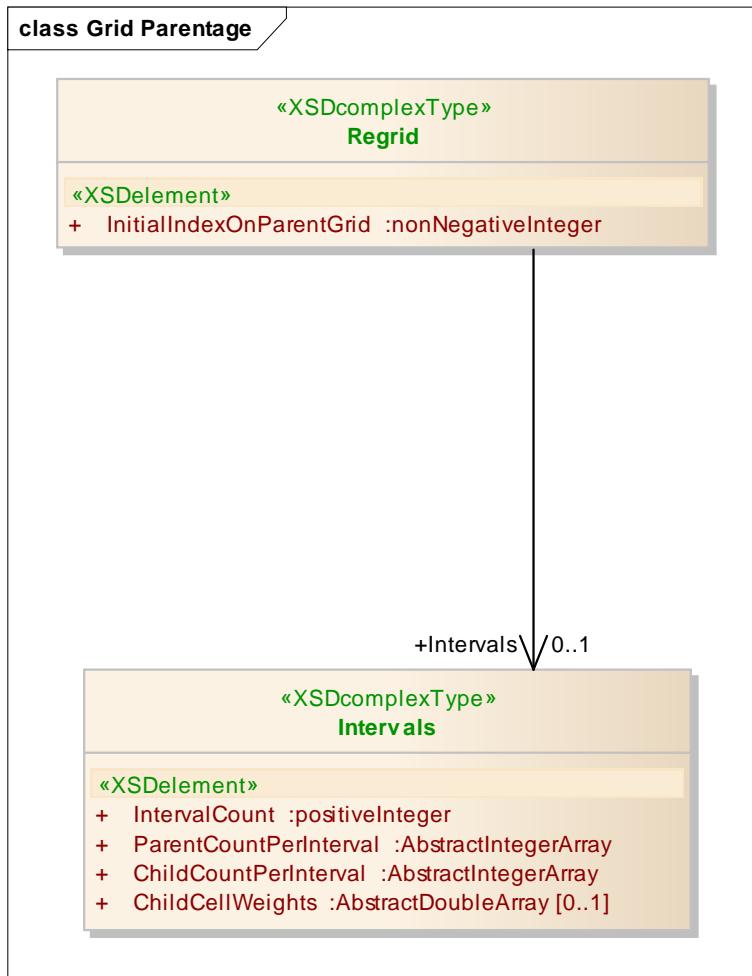


Figure 10-10. Regrid specification.

The regrid description is performed within intervals. The parent and child grids share cell faces at the interval boundaries, which allows the specification of implicit geometric or property relationships between the two grids. Within each interval, the number of parent and child grid cells is each specified. Observe these guidelines:

- For grid refinement, the number of parent cells in each interval is 1, and the number of child cells varies. Stated another way: so that a child cell does not span 2 (or more) parent cells, writers should set the parent count per interval to 1.
- For grid coarsening, it is the opposite, with the number of parent cells varying while there is only one child cell in each interval.

Regrid is not restricted to simple refinement or coarsening but instead supports mapping an arbitrary number of parent cells to an arbitrary number of child grids within each interval. By default, the child cells are of equal size within an interval. If uneven child cell sizes are required, they may be specified using the weights, which are proportional to the child grid cell sizes within each interval.

10.8.2.1 Example Regridding: Radial Grid Near a Well

Consider the refinement of a radial grid near a well. The global grid has dimensions of 10x1x3 and the child grid is a refinement of the innermost three cells on the second layer, with highest resolution near the well. The child grid also increases the angular resolution from a single 360° cell to four 90° cells.

I Regrid	Initial index on parent grid=0, Interval count=3, Parent count=(1,1,1), Child count=(4,2,2), Weights=(1,2,4,8,1,2,1,1)
J Regrid	Initial index on parent grid=0, Interval count=1, Parent count=(1), Child count=(4)
K Regrid	Initial index on parent grid=1, Interval count=1, Parent count=(1), Child count=(1)

For the I regrid, the weights indicate fractions of 1/15, 2/15, 4/15, 8/15 of the first parent cell, the weights indicate fractions of 1/3, 2/3 in the second parent cell, and the weights are equal indicating fractions of 1/2, 1/2 in the third parent cell. The J regrid has equal fractions of 1/4 and the K regrid indicates no refinement. Because the K regrid indicates 1:1 mapping, it need not be specified. For I, J, and K, because this is pure refinement, the parent count only takes on values of 1.

If the parent grid is an IJK grid, then the regrid may be in I, J, and/or K. If the parent grid is an unstructured column-layer grid, then the regrid may only be in K. No other grid cell geometries support a regrid geometry description. For other parent geometries, no intervals should be specified, and the child grid should have its geometry defined explicitly. However, the parentage construction may still consist of a window into the parent grid.

10.9 Grid Feature-Interpretations

Any RESQML representation, including a grid, may provide a representation of an interpretation of a feature (for more information, see Chapter 5 (page 39)). The most common application of this subsurface knowledge hierarchy is to indicate that different grid representations either share or have disparate interpretations of the features of an earth model, especially the reservoir stratigraphy and its fluid contacts. In addition to the generic relationships common to all RESQML representations, grids support several interpretation objects, which rely on the grid k or fluid contacts (if they are provided) or cell indexing of a grid to provide specific representations for a stratigraphic column or for a fluid phase unit, as shown in **Figure 10-11**.

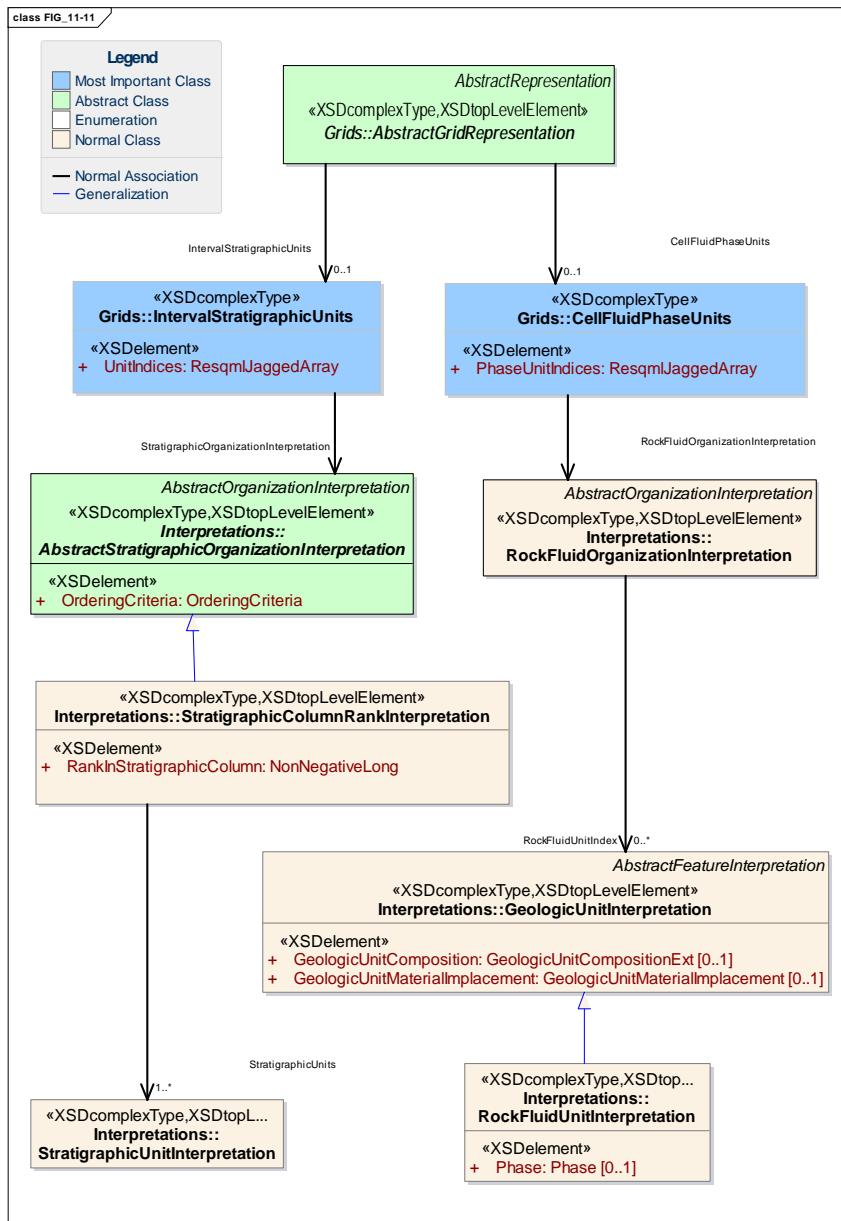


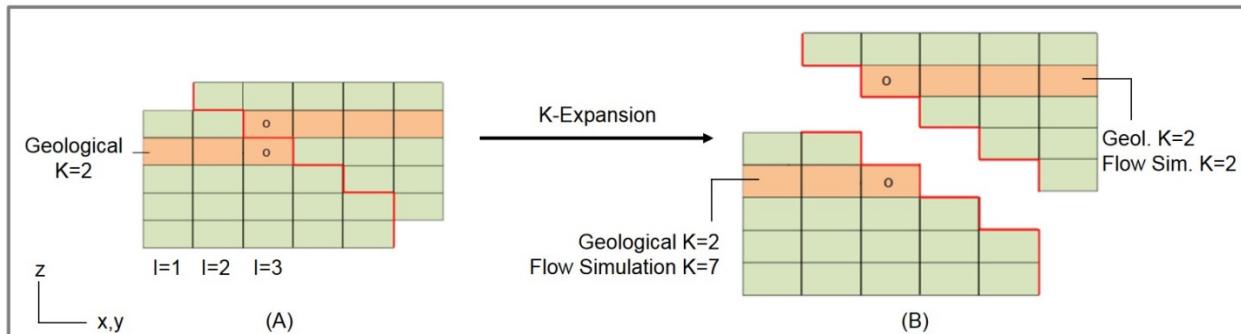
Figure 10-11. Specialized grid interpretation.

The interval stratigraphic column Interpretation takes advantage of a mapping between the intervals of a grid (layers + gaps) and a stratigraphic column. Implicit in the column are a definition of geologic units and

their horizons. Many grids have such a mapping, although exceptions may arise when we attempt to represent complex structures, especially with reverse faults. In such a case, a “K-layer” mapping to a stratigraphic organization may not exist. In this case, a mapping from the grid cells to the stratigraphy is available instead to provide a representation of the stratigraphic column interpretation. Similarly, the cell fluid phase interpretation provides a mapping from grid cells to a fluid organization, which specifies the hydrostatic fluids that are used to fill the model. These element level interpretation objects may also be applied to the intervals or cells of a wellbore frame or of a blocked wellbore. (For more information on wells, see Chapter 12 (page 213).)

When a given K layer in the grid corresponds to one or several stratigraphic units, the IntervalStratigraphicUnits element captures for each K the stratigraphic units associated with this layer. As it is possible for one K to lump together several stratigraphic units, a jagged array is required to capture this association. In a complex case when a single K unit does not follow the stratigraphy, a property with property kind “geologic k” is used to store indices that are used in the IntervalStratigraphicUnits element.

In **Figure 10-12**, where a K expanded grid is represented, Flow Simulation K is the natural index in the IJK grid, and Geological K will be exported as a property of property kind “geologic k”. These Geological K values are used as array indices for IntervalStratigraphicUnits unitIndices.



If the geological K is used, the cells marked with an 'o' will have the exact same indexing (I=3, J=0, K=2) (A). In order for the flow simulators to solve differential equations and describe fluid flow (Mass Balance, Darcy's Law), each cell must have an unique indexing. This can be met by considering each fault block separately (B).

Figure 10-12. Indexing when using geologic K. (Image courtesy of Paradigm.)

10.10 Unstructured Grids

10.10.1 Unstructured Geometry

New to RESQML at v2, are unstructured grids and their geometry, which include support for:

- Block-centered grids, one of the newest reservoir simulator data formats, which lack all geometry and which currently lack a standard industry representation.
- Unstructured grids with cell geometry, which provides a new vendor-independent specification of the grid geometry to complement the existing proprietary reservoir simulator formats and to support more general subsurface workflows.

Figure 10-13 shows how the geometry of an unstructured cell is defined by a list of (signed) faces, where each face is itself defined by a list of nodes.



Figure 10-13. Unstructured grid geometry.

Point geometry is attached to these nodes. The sign (or parity) of a cell face is positive if the face normal defined using a right hand rule is outwardly directed.

- For faces with three nodes, the geometry defined is a plane.
- For faces with four nodes, bi-linear interpolation from the unit square is most often used to define a smooth surface. However, some applications have been known to triangulate the surface instead and treat it as two or four planes.
- For faces with five or more nodes, there are a variety of interpolation schemes in the literature. RESQML transfers the node point geometry, but does not specify the method of interpolation other than for hinge nodes, which utilizes triangulation. When an interpolation method is well-known in the industry, e.g., tri-linear interpolation for corner-point cells, the RESQML reader is expected to be consistent with domain usage.

The description of the unstructured grid geometry also includes a cell shape specification. This enumeration is intended to be used by a RESQML reader to determine if a grid instance can be reconciled with an internal application data model. For example, many geomechanical calculations may be restricted to tetrahedral cell shapes.

10.10.1.1 Unstructured Geometry Extensions

Figure 10-14 shows that unstructured grid geometry supports two extensions: hinge nodes and finite elements.

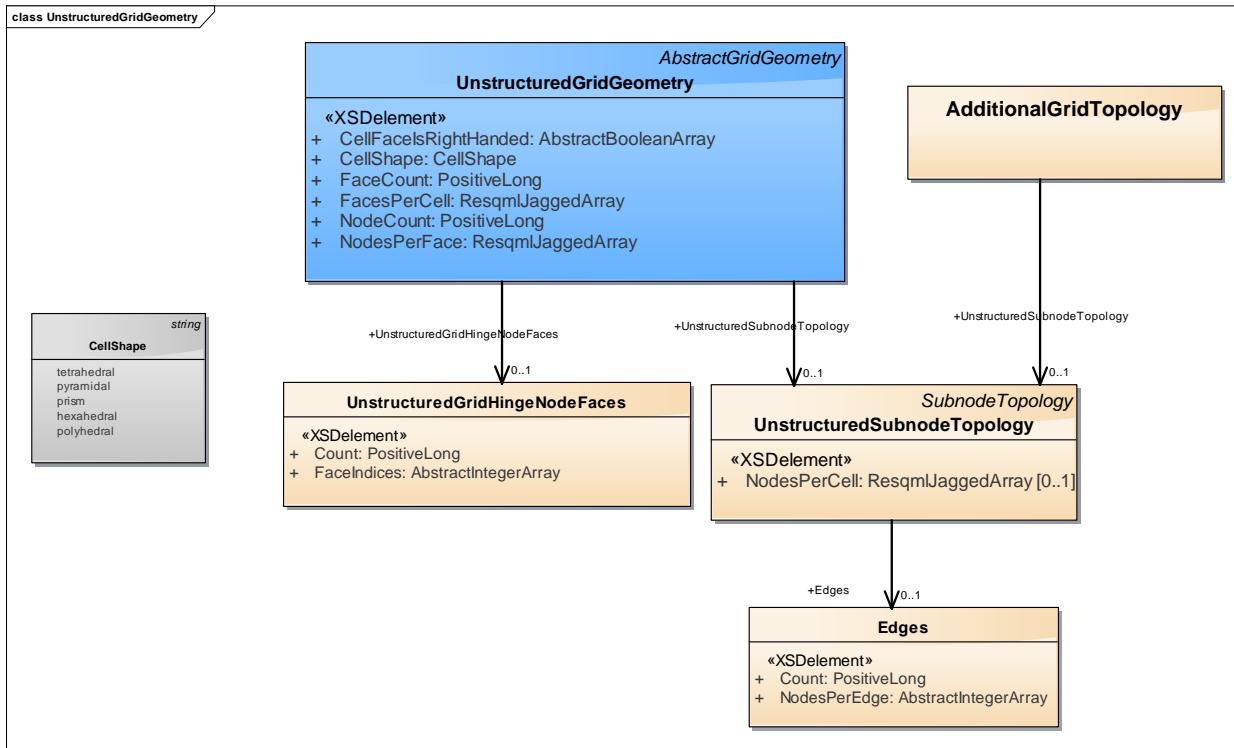


Figure 10-14. Unstructured grid geometry with extensions.

Unlike grids based on columns, the hinge node faces are not defined implicitly. Instead any cell face to which it is desired to associate hinge nodes must be specified as a hinge node face. Hinge nodes are an example of explicit interpolation on a face, in which the surface is replaced by a triangulated surface using the additional point geometry of the hinge node. The unstructured grid geometry also supports finite element extensions, as shown. The unstructured grid geometry supports the unstructured grid representation and also arises in the general purpose grid as a patch. A similar construction is used when defining the truncation cells for the truncated column-layer grids.

10.10.2 Unstructured Grid Representation

Figure 10-15 shows the unstructured grid representation, which supports:

- Block-centered grids, with no geometry. The topology of the grid is very simple; it is the number of cells, with no implicit relationships between objects.
- Grids with unstructured cell geometry, which have additional topological relationships between cells, faces, and nodes.

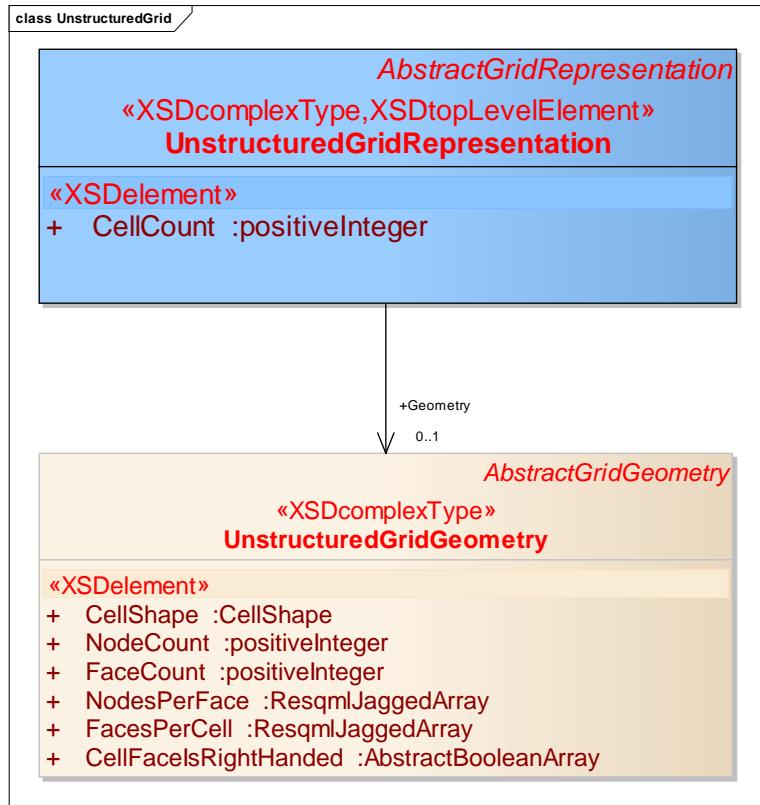


Figure 10-15. Unstructured grid representation.

10.10.2.1 Alternate Indexing Use Case

The unstructured grid representation is the only RESQML representation that allows an alternate cell indexing to the standard 0-based contiguous RESQML cell indexing. If used, this alternate cell indexing must be the only one considered for any related grid cell objects, such as grid connection set, subrepresentation, grid parentage, etc.

NOTE: Because this alternate indexing requires a lot of extra work for software readers to process, **its use is strongly discouraged** except if no other solution is acceptable.

RESQML has only one use case that requires use of alternate cell indexing. Assume you have a huge grid where only a tiny part is geometrically defined. In this situation, a likely approach might be to define the huge grid without any geometry and, for the tiny grid with geometry, define a subrepresentation. However, RESQML does not allow definition of geometry on a subrepresentation. The recommended solution is to create the tiny part of the grid as a RESQML unstructured grid, and use alternate cell indexing to describe the indices of the corresponding cells in the huge grid that has no geometry.

10.10.3 Unstructured Grid Indexable Elements

The unstructured grids have the simplest indexing of any of the grid types because all of the topological relationships between the grid elements are explicit. **Figure 10-16** shows the indexable elements organized into three categories: topology, geometry, or additional elements.

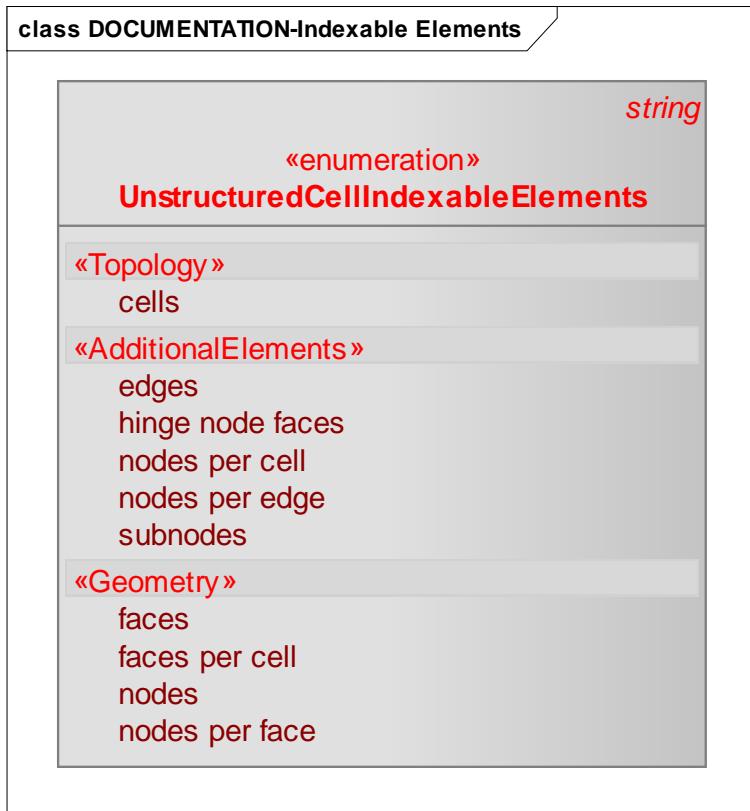


Figure 10-16. Indexable elements for an unstructured grid.

Element	Definition/Description of Use
Topology	Those elements whose indexing depends upon the grid index, which in this case is simply the cell count. For an unstructured grid, this index describes the enumeration of the cells of the grid.
Geometry	Those elements that need to be introduced to support the description of the geometry of the grid, which in this case are the nodes on the cell faces. For an unstructured grid, the faces and nodes are indexed according to the face count and the node count respectively. Two other indexable elements are also implicit in the unstructured cell geometry: <ul style="list-style-type: none"> • faces per cell are an ordered list of faces per cell, and • nodes per face are an ordered list of nodes per face. Both of these ordered lists are defined implicitly by the RESQML jagged array construction when defining the cell geometry. (For more information about jagged arrays, see Section 4.3.4 (page Error! Bookmark not defined.).)
AdditionalElements	Required to describe higher order geometry or properties. If hinge nodes are used as part of the higher order geometry of the cells, then hinge node faces must be defined. The remaining elements are part of the finite element construction.

Element	Definition/Description of Use
	<ul style="list-style-type: none"> • Edges and nodes per edge need to be specified before edge subnodes can be defined. • Similarly, nodes per cell need to be specified before cell subnodes can be defined. • Face subnodes require the specification of nodes per face, but their enumeration is already implicit in the construction of the unstructured cell geometry.

As with the column-layer grids, unstructured grids have a number of “object1 per object2” indexable element kinds, which may be used in favor of “object1” indices when the latter are more complicated. For example, for an unstructured grid, for a cell of the model with N faces as specified by the “faces per cell” element, the faces are numbered from 0 to N-1 in this order. Unlike the faces indexing, which depends upon whether faces are shared between cells, this simple enumeration is completely local to the cell. Faces per cell indices appear in the grid connections representation and the blocked wellbore representation.

Nodes per cell and faces per cell appear as part of the finite element subnode construction, but these should not be confused with the node, face and cell enumerations, each of which have their own indexing.

10.11 Column-Layer Grid Geometry

The column-layer cell geometry (**Figure 10-17**) is used to provide the topological support for nodes on coordinate lines. This provides the geometric description for the IJK and unstructured column-layer grids.

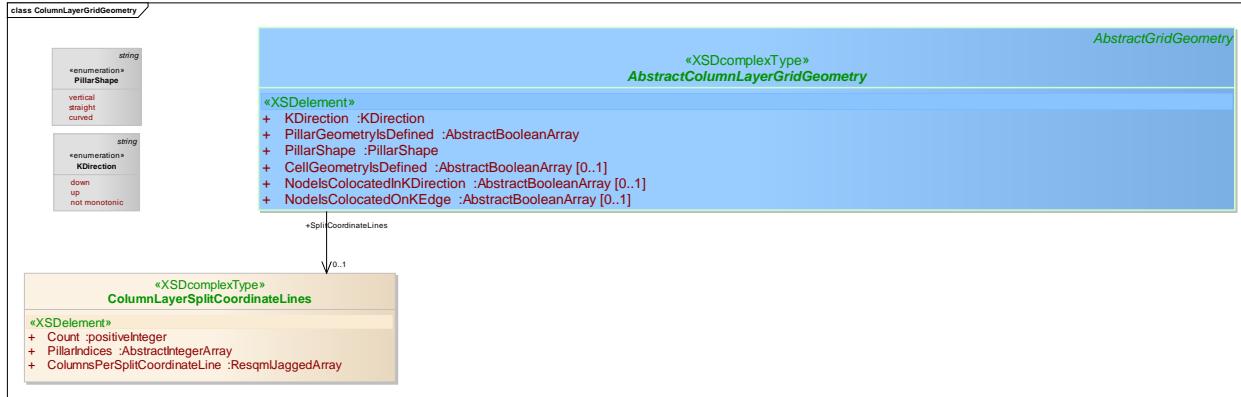


Figure 10-17. Column-layer cell geometry.

This geometry is based on pillars and columns, coordinate lines, and nodes.

10.11.1 Pillars and Columns

Pillars and columns describe the 2D topology of the grid.

- For IJK grids, each column has four pillars,
- For unstructured column-layer grids, each column is defined by a list of three or more pillars for each column.
- For an unfaulted grid, each pillar has one corresponding coordinate line.
- For faulted grids each pillar requires additional “split” coordinate lines.
- For a faulted IJK grid each pillar can have up to four coordinate lines per pillar.
- For faulted unstructured column-layer grids, each pillar may have as many coordinate lines as there are adjacent columns.
- The number of nodes per coordinate line is fixed at $NKL = NK + GapCount + 1$, and provides the topological support for the node geometry of the grid with an array of CoordinateLineCount $\times NKL$ points.

10.11.2 Coordinate Lines

RESQML provides support for parametric points and parametric lines. If the node points are defined parametrically, then each pillar must have a parametric line defined. The parametric point construction uses the mapping from coordinate line index to parametric line index, which is identical to the mapping from coordinate line to pillar.

- For an unfaulted grid, the count and indexing of coordinate lines is identical to that of the pillars.
- For an IJK grid, there are four pillars per column of the model.
- For an unstructured column-layer grid, there is an arbitrary number of pillars per column, which are specified explicitly as part of the grid description.
- For a faulted grid, the count of coordinate lines is increased by the number of split coordinate lines. The indexing of the coordinate line is defined by the “columns per split coordinate line” jagged array construction. The pillar index for each split coordinate line is also specified. With these two pieces of information, the topology and geometry of the faulted grid cells can be inferred from the specification of the unfaulted grid description.

- In order to have a well-defined index for the cell faces in the case of a faulted grid, a column edge index must be defined first. This is part of the faulted column-layer grid construction.

10.11.3 Additional Information Included in this Construction

This construction also includes additional geometric information on:

- the orientation of the coordinate lines (K direction)
- the shape of the pillars
- whether geometry has been defined for each cell and pillar
- colocation information on the coordinate line nodes

For the nodes along each coordinate line, there are #CoordinateLine x (NKL-1) Boolean values, which indicate whether the nodes are colocated. When describing the continuity of the nodes along each K edge, the count depends upon the topology of the grid. For example, for an IJK grid, the count will be 4 x NI x NJ x NKL, where the first index indicates that there are 4 edges for an IJ column. Ordering is shown in **Figure 10-18**, and is chosen to be identical to the natural ordering for an unstructured column-layer grid.

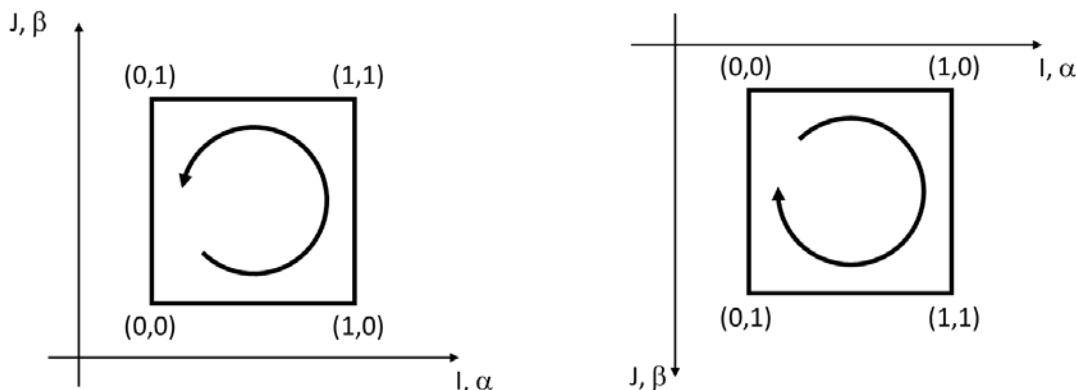


Figure 10-18. RESQML IJ node ordering.

- For an IJK grid, the RESQML IJ node ordering within a column of the grid starts at the $(\alpha,\beta) = (0,0)$ corner and then increases first α then β : $(0,0), (1,0), (1,1), (0,1)$. For the “nodes are colocated on K edge” element, the colocation direction follows this node ordering.
- For an IJ grid, there are 4 Boolean values indicating colocation between $(0,0)$ and $(1,0)$, between $(1,0)$ and $(1,1)$, and so on. As shown in Figure 11-17, this may appear to be either a clockwise or counter-clockwise direction when viewed in a spatial (XY) context, but at the indexing level the two diagrams are identical. The RESQML grid description includes mandatory parity information and K direction information, from which the spatial orientation may be inferred, but there is no direct specification of a clockwise or counter-clockwise data ordering.

10.11.4 Supported Extensions

The column-layer cell geometry supports two optional geometric extensions: split nodes and finite element subnode topology (**Figure 10-19**).

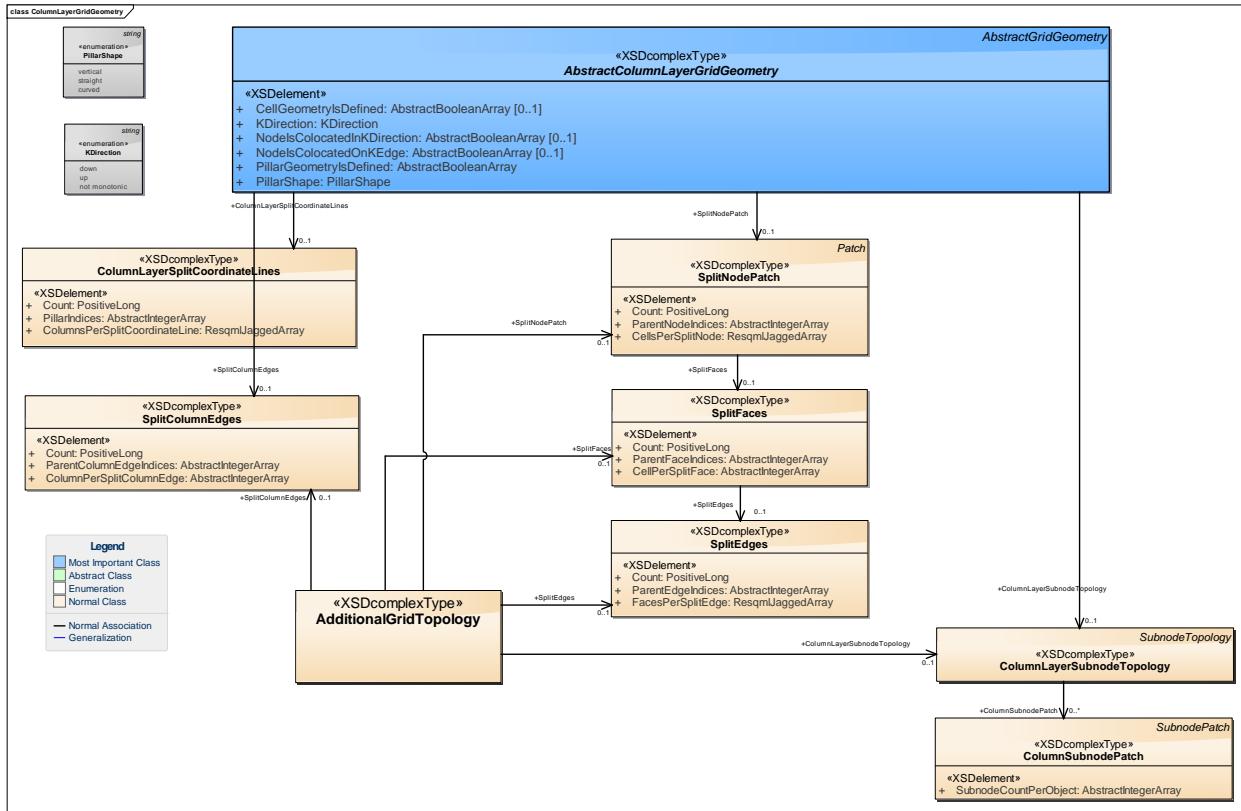


Figure 10-19. Column-layer cell geometry with extensions.

Split nodes introduce additional nodes and hence additional spatial resolution into a column-layer grid. They are used by some modeling applications, most often to improve the representation of stair-step reverse faults. After split nodes are introduced, additional split faces and split edges may also need to be defined. The additional grid topology object collects all of these extensions so that they are available for property attachment, even if they are not used for geometry extensions.

Subnodes are used as part of the finite element grid description. Again, both of these extensions are structured to localize these degrees of freedom, to provide guidance to a RESQML reader which may not support these features.

10.12 IJK Grids

10.12.1 IJK Grid Geometry

The IJK grid geometry is a column-layer grid geometry together with a mandatory specification of the grid parity (left handed or right handed) (**Figure 10-20**). Parity is described in Section 10.12.210.12.4 (page 165). In addition, IJK grids support areal gaps between columns of the grid through the introduction of additional “split” pillars. These are described using the IJ gaps object.

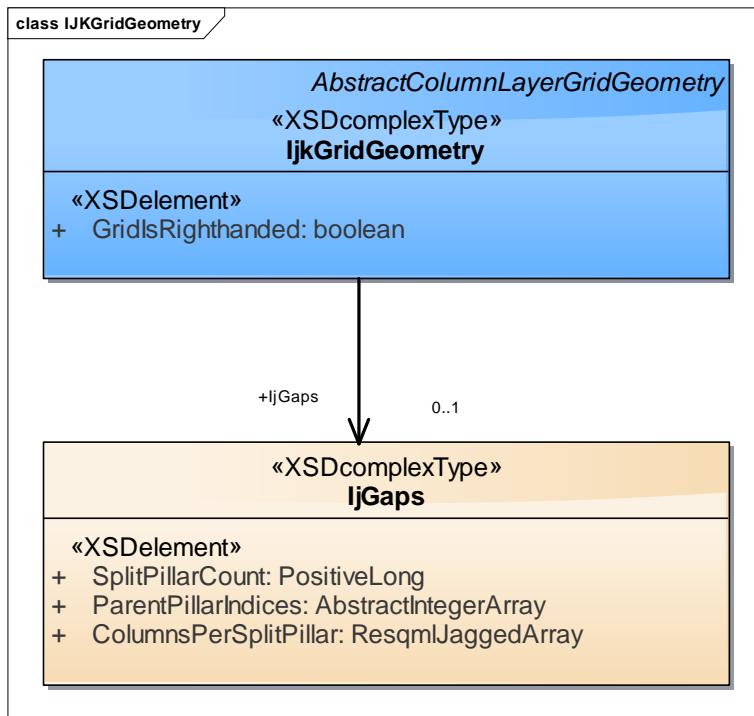


Figure 10-20. IJK grid geometry.

10.12.2 IJK Grid Representation

An IJK grid representation is the most common example of a grid with column-layer cell geometry. To match industry usage, this grid type supports several extensions not present within the unstructured column-layer grid or the unstructured grid, specifically, IJ gaps, K gaps, and radial cell interpolation (**Figure 10-21**).

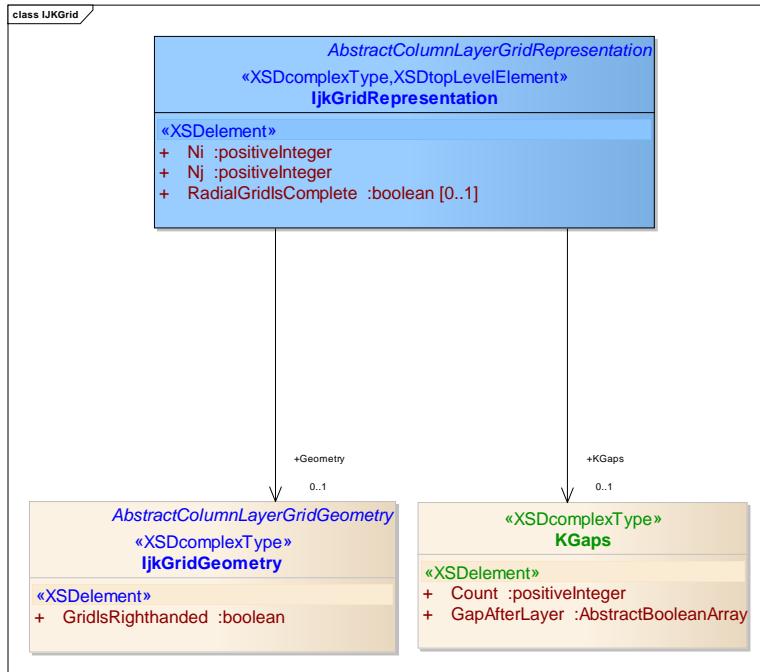


Figure 10-21. IJK grid.

The IJK Grid has seven primary indices, all of which may be obtained from the IJK grid objects shown in **Figure 10-21**:

- NI, NJ, NK, as previously described
- #Intervals = NK + GapCount
 - GapCount is the number of K gaps in the model. In RESQML, unless stated otherwise, the bottom of one layer is contiguous with the top of the layer below it. As a consequence, the number of nodes along a coordinate line needed to describe the cell geometry is NK+1, not 2*NK, which would be required if no assumptions were made about continuity.
If gaps are introduced between layers, then GapCount>0, #Intervals>NK, and the number of nodes also increase. Eclipse GRDECL/GRID and VIP/Nexus CORP data formats, for example, make no assumptions about the continuity of cell geometry. Each would have GapCount = NK–1, #Intervals = 2*NK. This indexing may be preserved within RESQML, if desired, using the gap count.
- NIL, NJL and NKL are used to define the number of the edges (or “lines”) of the cells.
 - NIL = NI + 1 (always)
 - NJL = NJ + 1, except for “complete” (periodic) radial 360° grids, in which case the first and last cell faces are identical and NJL = NJ
 - NKL = NK + GapCount + 1

The first three of the seven indices appear explicitly within the schema, as shown in **Figure 10-21**. The others are inferred from the optional IJK grid objects, as just described.

IJK grids support a radial interpolation option. For radial grids, the r=0 origin point must be specified independently for each interval edge of the model. This geometry is attached through the additional grid geometry attachment kind of “radial origin polyline”. The number of nodes on this polyline must match the coordinate line nodes and hence has a count of NKL. The use of an explicit center point allows arbitrary inclinations for the radial grid, for example, to represent horizontal radial (or elliptical) grids aligned with horizontal wells.

The IJ gaps object is used to describe gaps between columns of the model. i.e., to describe fractures with finite volumes instead of simply fault surfaces. Grids of this type may be described using Eclipse GRID and VIP/Nexus CORP data formats, but not Eclipse GRDECL. This option does not modify the indexing of the cells or columns, but it does increase the number of pillars and coordinate lines.

IJK grid geometry inherits from column-layer geometry, and provides the topological support for the coordinate line nodes of the grid. In addition, it carries several elements that describe the grid geometry:

- Of these, the grid parity (\pm) is mandatory. Its value is needed for the calculation of cell volumes and cell face transmissibility. The parity is defined as the sign of the triple cross product obtained from the three cell tangent vectors.

$$\pm = \text{Sign}(\vec{t}_1 \times \vec{t}_2) \bullet \vec{t}_3 = \text{Sign}(\vec{n}_3 \bullet \vec{t}_3)$$

$$\text{Cell Volume} = \pm \iiint_{\text{Cell}} (\vec{t}_1 \times \vec{t}_2) \bullet \vec{t}_3$$

$$\text{Face Transmissibility} = \pm \iint_{\text{Face}} (\vec{n}_f \bullet \vec{t}_f) / t_f^2$$

These are the geometric components to cell volume and face transmissibility, and do not include the additional physical properties, e.g., porosity, permeability or viscosity. The parity calculated from this triple product is uniform on a well-formed grid, although local defects may occur. Typically, negative cell volumes or negative transmissibilities are set to zero within reservoir modeling applications.

- The other elements provide information on whether geometry is degenerate (node colocation) and are optional.

10.12.3 IJK Grid Indexable Elements

Figure 10-22 lists the indexable elements for IJK grids.

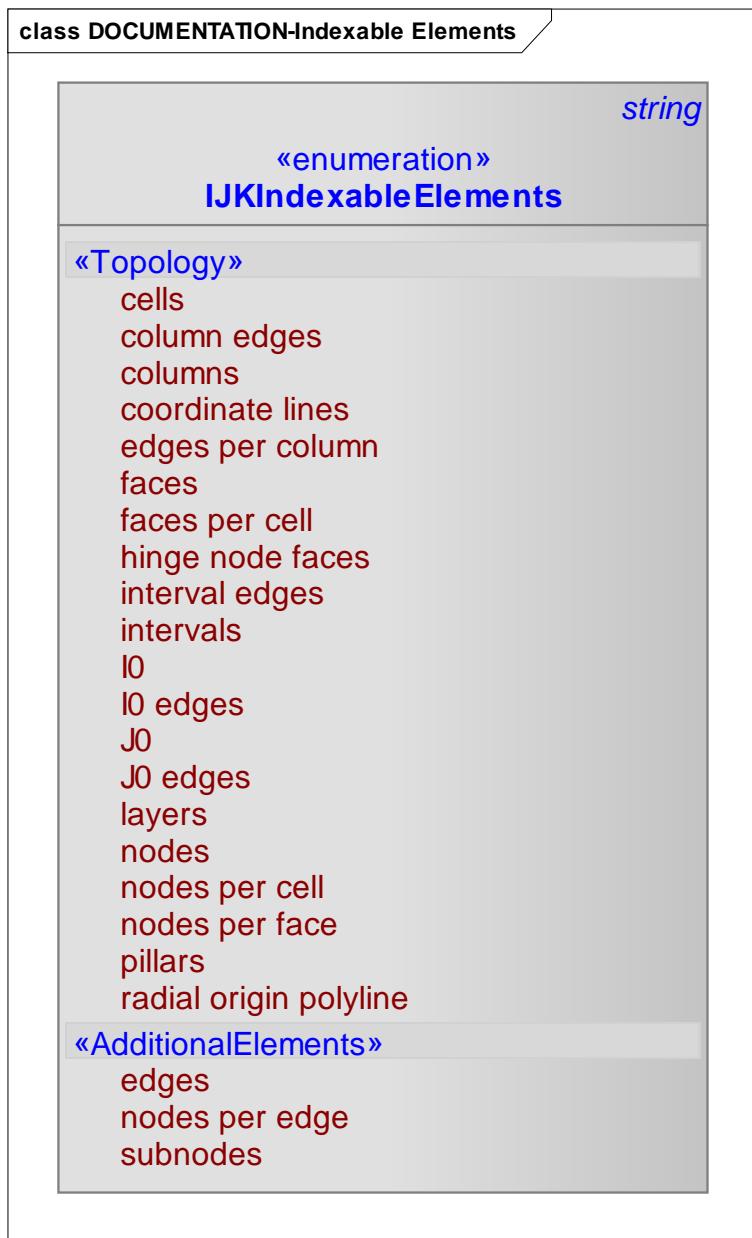


Figure 10-22. Indexable elements for an IJK grid.

RESQML v1 introduced six indices for the corner-point grid: ($\alpha, \beta, \gamma, I, J, K$), which are also part of RESQML v2. The (I, J, K) indices have already been described.

The (α, β, γ) indices each take on the values (0,1). They are aligned with the (I, J, K) directions, respectively, and are used to index the nodes and faces within a cell. In v2, they also contribute to the enumeration of many of the other grid elements. When designing the IJK grid indexing we made two choices, which are intended to simplify implementation:

- Consistent grid indexing between block-centered (no geometry) and nodal grids.

- Consistent grid indexing between IJK and unstructured column grids using a shared column-layer indexing and geometry.

As an example of the enumeration and the use of the indices to implicitly specify the grid topology, consider the relationship between columns and pillars shown in **Figure 10-23**.

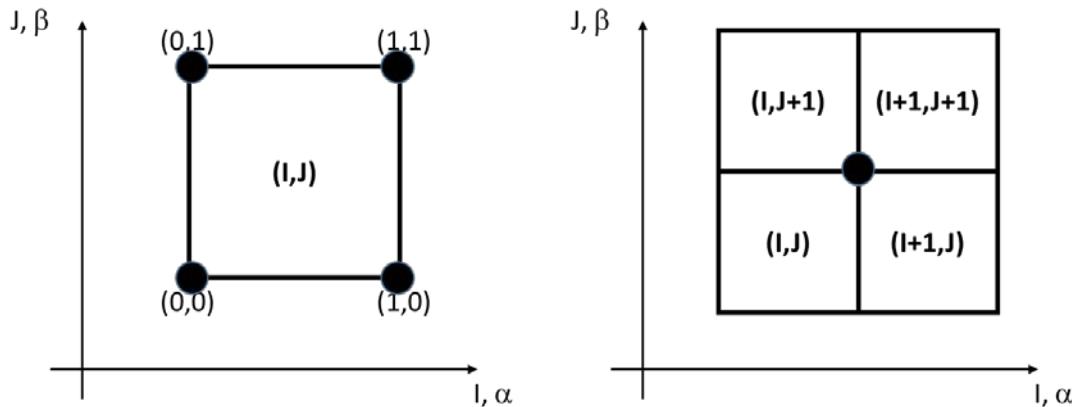


Figure 10-23. Columns and pillars of an IJK grid.

Their indices are:

$$\text{Column index} = I_0 + NI^*J_0$$

$$\text{Pillar index} = (I_0 + \alpha) + NIL^*(J_0 + \beta)$$

Again, $I_0=I-1$, $J_0=J-1$ and also $NIL=NI+1$. The form of the indexing in these equations determines the topological relationship between the columns and the pillars. For one column, indexed by (I, J) , there are four adjacent pillars indexed by $(\alpha, \beta) = (0, 0), (1, 0), (1, 1)$, and $(0, 1)$. With the form of the pillar indexing, we obtain the same index for I and $\alpha=1$ as for $I+1$ and $\alpha=0$. As a consequence, the four columns (I, J) , $(I+1, J)$, $(I+1, J+1)$ and $(I, J+1)$ implicitly share the same pillar. Similar relationships exist in a cross-sectional view of a grid, for example, between the layer intervals and the interval edges.

The indexing for the IJK grid has more implicit relationships between the grid elements than any other grid class. By choice—as much as possible—the IJK grid indexing and the unstructured column-layer grid indexing are identical. Specifically, the following node order is followed for IJK grids: $(\alpha, \beta) = (0, 0), (1, 0), (1, 1), (1, 0), (0, 0)$. With this specification, the indexing for the IJK grid is identical to that of a corresponding unstructured column-layer grid. However, unlike the unstructured column-layer grid, the relationship between column and pillar is implicit.

Element indexing is described using the following indices:

- $I_0=0\dots NI-1$, $J_0=0\dots NJ-1$ and $K_0 = 0\dots NK-1$
- $\alpha = 0, 1$, $\beta = 0, 1$ and $\gamma = 0, 1$
- $L=0\dots NKL-1 = NK + \text{GapCount}$

The first three indices correspond to the (I, J, K) indices of a corner-point grid, except that they are now explicitly 0-based. The next three (node) indices have not changed their usage since RESQML v1. The interval index, L , includes both layers and gaps, and is useful for indexing grids with K gaps.

A number of indexable elements that depend upon NI and NJ , and which are not present within the unstructured column-layer grid, arise for IJK grids. Specifically, I_0 and J_0 and the I_0 and J_0 edges each have one dimensional indices.

10.12.3.1 Special Consideration: Column Edge

The only indexable element where the commonality in indexing between the IJK grid and the unstructured column-layer grid is not obvious is the column edge.

- For the unstructured column-layer grid, the indexing of the shared column edges is explicit.
- For the IJK grid, the indexing is implicit and first follows the I column edges ($N_{IL} \times N_J$) and then the J column edges ($N_I \times N_{JL}$). The I faces and J faces do not appear as explicit indexable elements but are instead part of the range of elements of the faces. The edge and face indices correspond to the enumeration of shared edges and faces. This ordering is a RESQML choice. The column edge indexing needs to be defined to support face indexing.

There are a number of “object1 per object2” indexable element kinds for IJK grids, which may be used in favor of “object1” indices when the latter are more complicated. For example, for an IJK grid, there are 6 “faces per cell”: 0=top, 1=bottom, 2-5 are side faces following the IJ circular node ordering, described previously. This is a very simple enumeration, versus the “faces” indexing, which depends on the grid faulting and column edge enumeration. Faces per cell indices appear in the grid connections representation and the blocked wellbore representation.

10.12.4 IJK Grid Origin

A frequently-asked question is: How do I determine the location of the origin of an IJK grid in the field of view? Generally, this is a function of the data, not of the schema, but certain inferences can be made from the grid parity and from the specification of the K direction. A number of observations can be made:

- In practice, almost all grids have their origin on the left of the field of view, which specifies the direction of the I axis. Origins on the right, are possible, but are extremely rare in practice.
- The K direction can be used to determine if the origin is on the upper face of a model (K direction = down) or on the lower face of a model (K direction = up).
- The direction of the J axis is determined uniquely by the grid parity, given the directions of the I and K axes.

Calculation of the Jacobian for volume and the transmissibility for flux, are based on the triple-cross product of the cell or grid tangent vectors. To ensure that they are correctly calculated, they also need to know the parity of the 3D coordinate reference system.

10.12.5 IJK Cartesian and Radial Cell Interpolation

RESQML provides support for radial cell interpolation as part of the IJK grid representation. Unlike reservoir simulation software, which provides special keywords to allow specification of grids directly in (r, θ, z) coordinates, RESQML continues to specify cell nodes as (x, y, z) points. This specification allows reasonable interchange with geologic model vendors, the majority of whom do not provide support for radial grids. However, when the “radial grid is complete” Boolean element is included in a RESQML grid description, together with the radial origin polyline geometry, it then implies that radial interpolation should be used to describe the cell shape.

- Specifically, for a Cartesian corner-point cell, the cell volume is defined by the tri-linear interpolant of (x, y, z) in (α, β, γ) between the 8 corner nodes.
- In contrast, for a radial corner-point cell, the cell volume is defined by the tri-linear interpolant of (r^2, θ, z) .

This choice follows from the transformation of unit volumes: $dx dy dz = r dr d\theta dz = \frac{1}{2} dr^2 d\theta dz$. In other words, the interpolation describes a radial (cylindrical) cell shape. The I (or α) coordinate (dimension: N_{IL}) is always in the radial direction, and the J (or β) coordinate (dimension: N_{JL}) is always in the angular direction. For a fixed value of γ , we have a 2D radial coordinate system.

- To define r from (x, y, z) , we need the coordinates of an origin at that same value of γ . The definition of the radial grid includes the specification of NKL origin points, which may be interpolated linearly in γ .

- To define θ from (x,y,z) , we may use the usual trigonometric functions.

Some care must be taken to ensure that the branch cut does not lie within the cell. For the special case of complete 360° grid cells with $NJ=1$, care must be taken to have strictly monotonic values for θ . This construction is sufficiently general to represent simple radial grids with vertical coordinate lines, or more complex radial grids that may conform to a complex horizontal or undulating well trajectory.

Radial IJK grids may have a slightly different topology from Cartesian IJK grids. If a radial grid is complete, then it is periodic and covers 360° . Consequently the last J cell face is identical to the first, and $NJL=NJ$ instead of the usual $NJL=NJ+1$.

Local radial grids use a modified interpolation method on the outermost ring(s) of cells ($I=NI$). To ensure geometric consistency with the parent grid, the α range is reduced so that the radial cell shape remains consistent with the cell volume interpolation method of the parent grid cell. Specifically, a circular arc between two node points on the local grid normally extends beyond a linear interpolant on the parent grid. The extent of the radial cell in the α direction is reduced so as not to exceed the linear interpolant.

10.13 Unstructured Column-Layer Grids

10.13.1 Unstructured Column-Layer Grid Geometry

The unstructured column-layer grid geometry is an extension of the column-layer grid geometry used to describe the geometry of unstructured column-layer grids. As with the column-layer cell geometry, it provides the topological support for nodes on coordinate lines.

Unlike the IJK grid geometry, the unstructured column-layer grid geometry must include explicit topological relationships between the columns of a grid, and geometric grid parity information, as shown **Figure 10-24**.

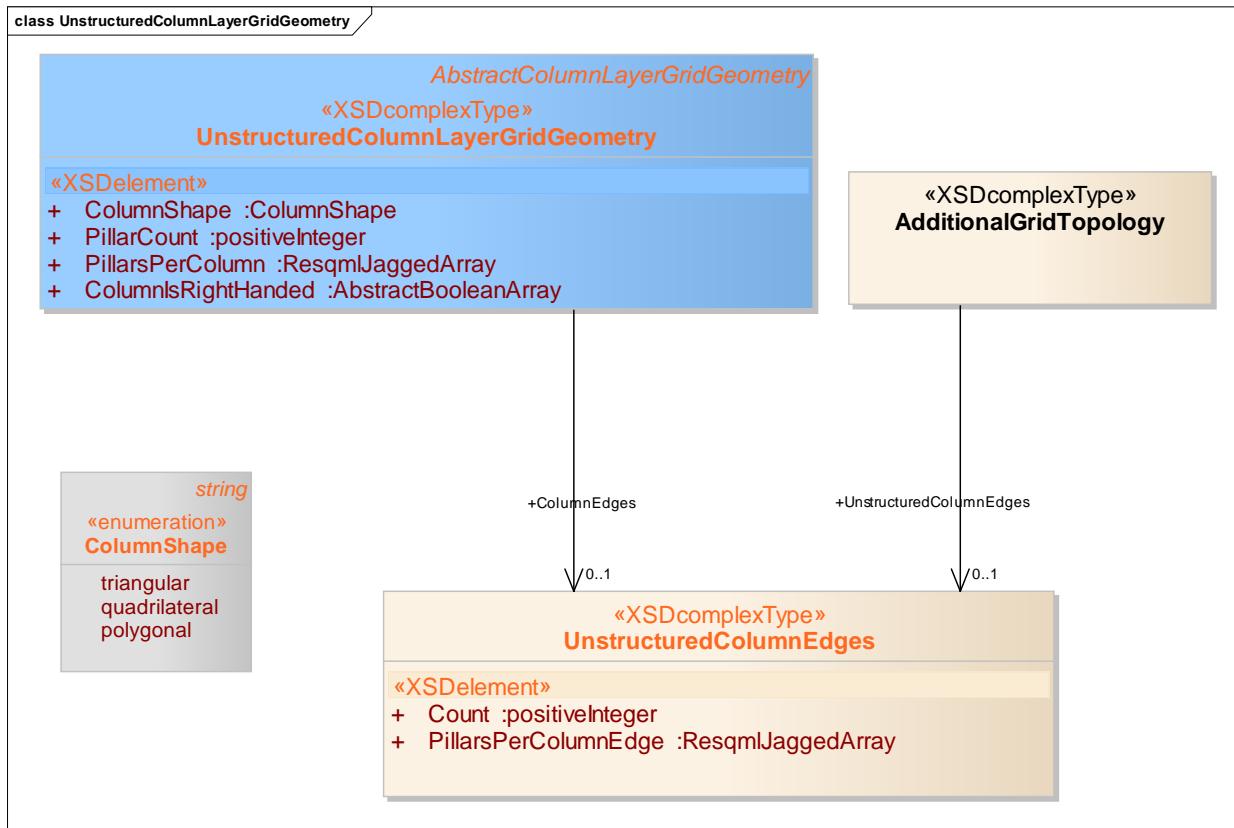


Figure 10-24. Unstructured column-layer cell geometry with extensions.

Specifically, this information is required: the number of pillars, the specification of which pillars are used to delineate which columns, and the parity of each column of the model. In contrast, for IJK grids, much of this information is known implicitly from the IJ indexing, or is uniform on the grid.

As with the column-layer cell geometry, if faces are to be indexed, then column edges must be defined. Neither faces nor column edges are required as part of the description of the grid geometry. However, this optional definition is retained as part of the additional grid topology. In addition, the description includes information on the column shapes for the grid. The shapes are defined topologically, i.e., by the number of pillars used to define each column.

The unstructured column-layer grid geometry supports the unstructured column-layer and the truncated unstructured column-layer grid representations. It also is used in the general purpose grid as a patch.

10.13.2 Unstructured Column-Layer Grid Representation

The unstructured column-layer grid is based on the unstructured column-layer cell geometry, and hence the column-layer cell geometry. In a 2D sense, these grids are not structured, and all column and pillar adjacency information is explicit. However, just as with IJK grids, there is a well-defined global layering within these grids, which holds for all columns.

The grid description for an unstructured column-layer grid (**Figure 10-25**) has many of the same elements as the IJK grid. However, unlike an IJK grid, RESQML provides no optional extensions, because none arise in current industry practice. If such a requirement does arise in the future, then the general purpose grid description is available for use.

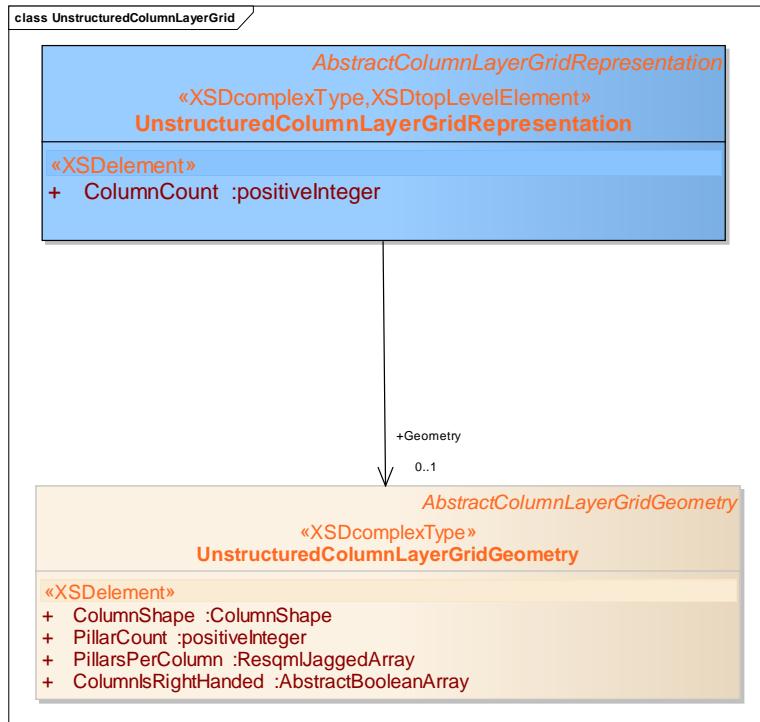


Figure 10-25. Unstructured column-layer grid representation.

10.13.3 Unstructured Column-Layer Grid Indexable Elements

The unstructured column-layer grids have both implicit and explicit topological relationships between the grid elements and a corresponding mixture of implicit and explicit indexing. As shown in **Figure 10-26**, the indexable elements can be organized into three categories: topology, geometry, or additional elements.

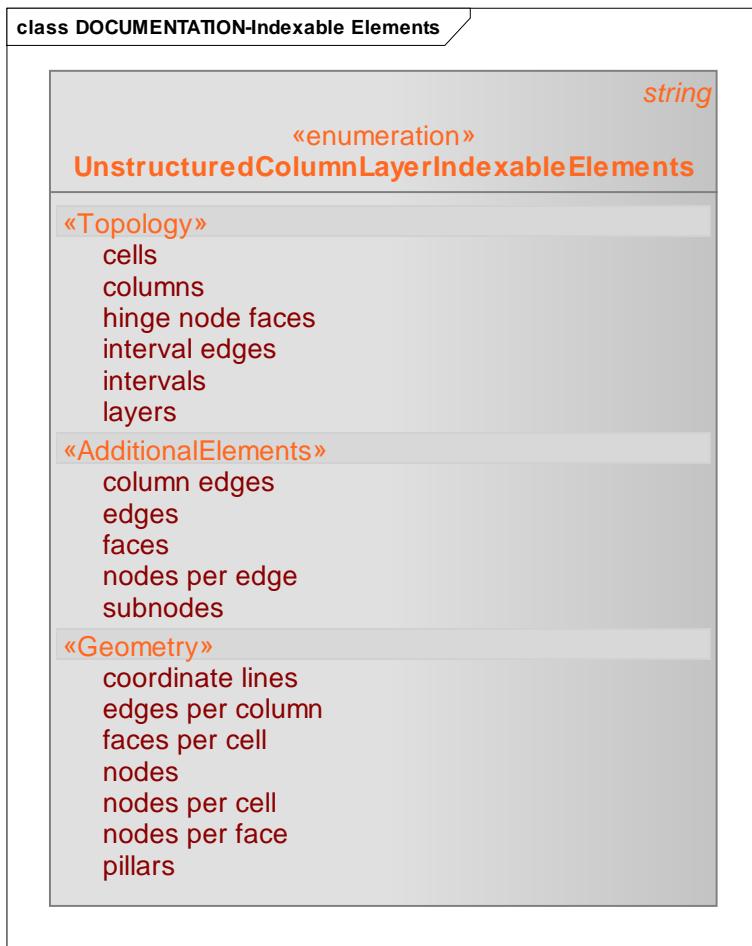


Figure 10-26. Indexable elements for an unstructured column-layer grid.

Element	Definition/Description of Use
Topology	<p>Those elements whose indexing only depends upon the grid indices, which in this case are the ColumnCount and NK.</p> <p>Element indexing is described using the following indices:</p> <ul style="list-style-type: none"> • Column index = Column# = 0...ColumnCount-1 • Layer index = K0 = 0...NK-1 and $y = 0,1$ • Interval index = L = 0...NKL-1 <p>For the unstructured column-layer grid representation, the GapCount=0 and the interval and layer indexing are identical. However, when this topology is re-used as part of a general purpose grid, gaps are supported and these two indices need not be identical.</p>
Geometry	<p>Those elements that need to be introduced to support the description of the geometry of the grid, which in this case are the nodes on the coordinate lines of the grid. The pillar count is explicit, and the CoordinateLineCount = PillarCount + SplitCoordinateLineCount. Both are 1D indices. The nodes always have a 2D index constructed from the number of coordinate lines with NKL nodes per coordinate line.</p> <p>Faces per cell, nodes per cell and nodes per face are indexed in an implicit fashion, which is in contrast to the unstructured grid, where indexing is explicit. By RESQML convention, for all of these elements, the $y=0$ face or nodes are indexed first, followed by the $y=1$ face</p>

Element	Definition/Description of Use
	or nodes. The areal ordering of the elements follows from the explicit definition of columns in terms of pillars.
Additional Elements	Faces are the most commonly used, and their definition requires the specification of the column edges. Neither are required for the specification of the column-layer grid cell. They may be required for either property indexing or for higher order finite element geometries. The face count is a one dimensional index constructed from an ordered combination of the column edges, the columns, NK and NKL: FaceCount = ColumnCount x NKL + ColumnEdgeCount x NK, in this order. Faces should not be confused with faces per cell. If a face is shared between two cells, then it appears only once as an enumerated face, but twice as an enumerated face per cell.

As with IJK grids, there are a number of “object1 per object2” indexable element kinds for unstructured column-layer grids, which may be used in favor of “object1” indices when the latter are more complicated. For example, for an unstructured column-layer grid, for a column of the model with N sides, there are N+2 “faces per cell”: 0=top, 1=bottom, 2-(N+1) are side faces following the explicit pillars per column ordering. This is a very simple enumeration, versus the “faces” indexing, which depends on the grid faulting and column edge enumeration. Faces per cell indices appear in the grid connections representation and the blocked wellbore representation.

10.14 Truncated Column-Layer Grid Representation

RESQML supports two truncated column-layer grid representations, which may be based upon either 1) IJK or 2) unstructured column-layer grid geometry. These grids are very similar to the underlying fundamental grids but with the extension to allow cells to be truncated and split along the fault surfaces of a structural framework. The truncated cells have more faces than in the underlying grid, and the portions of the cell volumes that are split create additional split cells. The resulting grid cell topology has 3+1 indices for truncated IJK grids and 2+1 indices for truncated unstructured column-layer grids.

Figure 10-27 shows the grid representation. Its elements are similar or identical to those already described. Unlike other grid classes, explicit grid geometry is required and the truncated grids do not support geometry-free, block-centered representations.

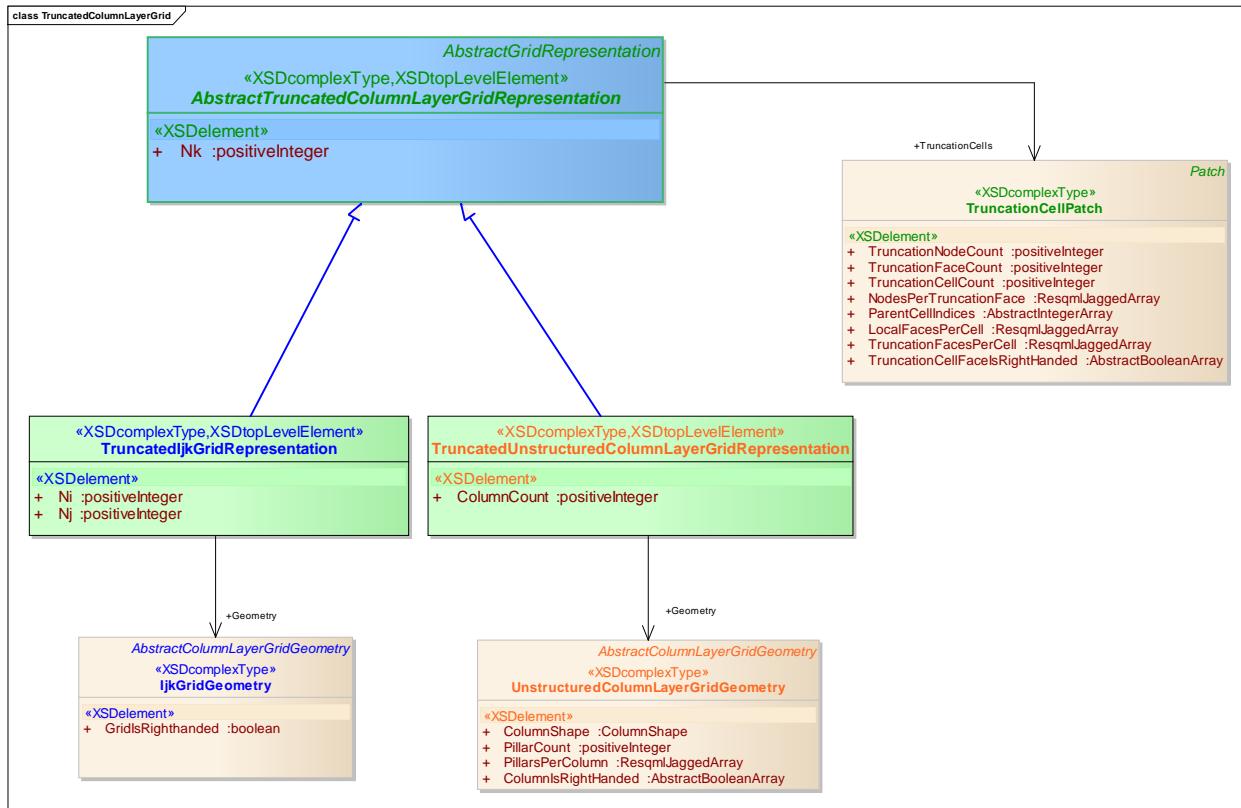


Figure 10-27. Truncated column-layer grid.

10.15 General Purpose (GP) Grid Representation

The general purpose grid representation (**Figure 10-28**) is supplied as a research tool for unstructured grids. It consists of all the grid objects described elsewhere in the schema, combined in a general way. It supports block-centered, explicit geometry, implicit geometry, and higher order grid geometries. Because the general purpose grid allows objects to be mixed in arbitrary combinations, patch topology must be defined for each of the component grid types.

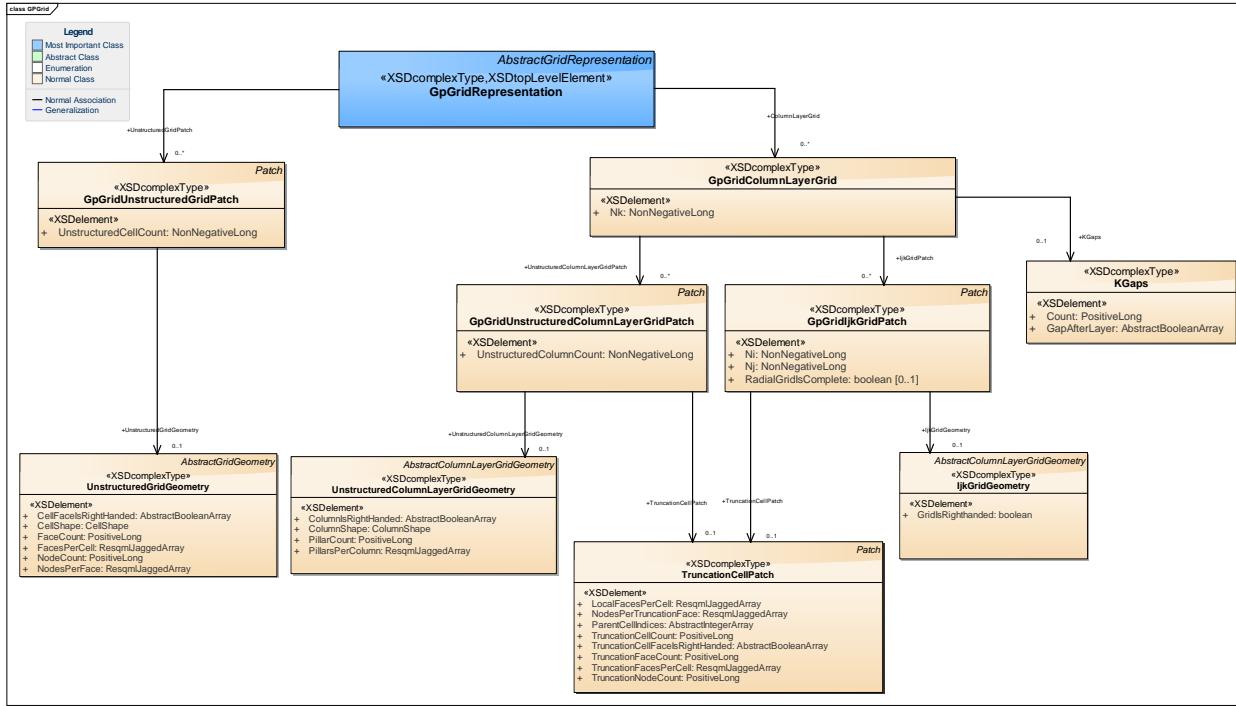


Figure 10-28. General purpose grid representation.

10.16 Grid Connection Set Representation

Grid connection sets are RESQML representations (**Figure 10-29**) used to represent some cell-to-cell connections on grid. A main use case is describing the cell-to-cell connections where the fluid flow is impacted due to a fault, which includes the concept of non-standard adjacency (NSA) also known as non-neighbor connectivity (NNC). As with any other representation, grid connection sets may be used to support properties or relationships. Unlike a grid, geometry as such is never attached to a connection, although geometric properties, e.g., cell face overlap area, may be attached to a connection.

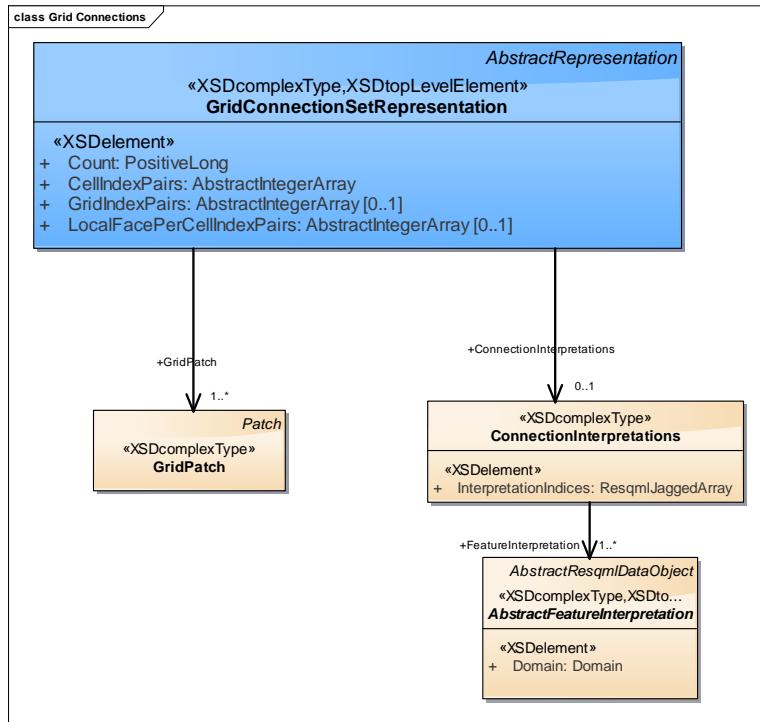
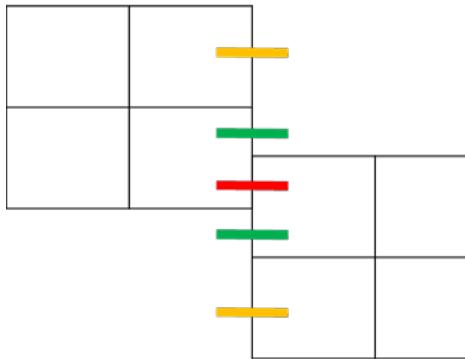


Figure 10-29. Grid connections.

Grid connection sets are the preferred RESQML method for representing the impact of faults on fluid flow within a grid, and are preferred over the use of grid cell-face subrepresentations (The use of cell-face-pairs is more complete than single cell-faces, which are missing a corresponding second cell face identification, and only provide an incomplete representation of the topology of a fault.) RESQML allows a couple of options for specifying grid connection sets; recommended best practice is to specify a single grid connector set that contains all faults and other important cell to cell connection. To describe more complex structures (e.g., faults or geobody boundaries, potentially merging at below the spatial resolution of a grid) you can associate zero or more interpretations with the individual connections (explained in more detail below).

BUSINESS RULE: A single cell-face pair should not appear within more than a single grid connection set. This rule is designed to simplify the interpretation of properties assigned to multiple grid connection sets, which might otherwise have the same property defined more than once on a single connection, with no clear means of resolving the multiple values.

The current construction provides a list of connections between cell pairs; for requirements and options for NSA/NNC, see **Figure 10-30**). There is no requirement that the two cells are on a shared grid, and so the grid index may need to be provided for each cell. The grid index object is used to convert a list of grid UUIDs to local grid indices, by pairing a (locally) unique index with a reference to a grid. Because directional information is often needed for the use of a connection, the local face per cell index is also included. This is a local index, which, for example, would take on the values of 0...5 for an IJK grid cell.



— **Non-standard adjacencies (NSA) aka non-neighbor Connectivities (NNC).** REQUIRED; must be defined in a grid connection set.

— **Partial connectivities between cells.** (Cell connectivity is ensured by red connection). OPTIONAL

— **Non-cell connectivities.** Provides additional information about the fault along the cells. RECOMMENDED

Figure 10-30. Required, recommended and optional practices for specifying NSA/NNC.

The connection interpretations provides the feature-interpretations, and hence the feature's name. Notice that the interpretation data structure allows more than one interpretation to be associated with a single connection, which may be required depending upon the spatial resolution of a grid. In engineering terms, the implications are that the transmissibility multiplier for a connection may need to be computed as the product of transmissibility multipliers for each fault or feature-interpretation.

The connection interpretations object may be used to associate feature-interpretations with individual connections. The ordered list of feature-interpretations defines a 0-based index for each interpretation, which are then referenced from the interpretation indices array so that each connection has zero, one, or many associated interpretations.

10.17 Grid Examples

This section contains several simple grid examples. Each example consists of:

- One or more class diagrams, which show the objects used to describe the grid.
 - One or more instance diagrams, which show the attribute and element data values.

In each diagram, the blue object is the grid representation. Green objects identify the top level objects referenced in the construction.

10.17.1 Eclipse GRDECL File as an IJK Grid

An Eclipse GRDECL grid consists of an $NX \times NY \times NY$ corner-point grid described by these keywords (**Figure 10-31**):

- COORD defines straight coordinate lines, which correspond to the RESQML pillars.
 - ZCORN provides the parametric Z-values, which are used to specify the corner nodes for each cell, using the COORD data as a linear lookup from Z to XY. Adjacent columns share pillars (no IJ Gaps), but the top and base of cells need not be continuous: a GRDECL grid supports NZ-1 K Gaps.
 - ACTNUM is an array that is used to specify whether the geometry and/or properties of a cell are defined.
 - Properties are attached to the cells of the grid.

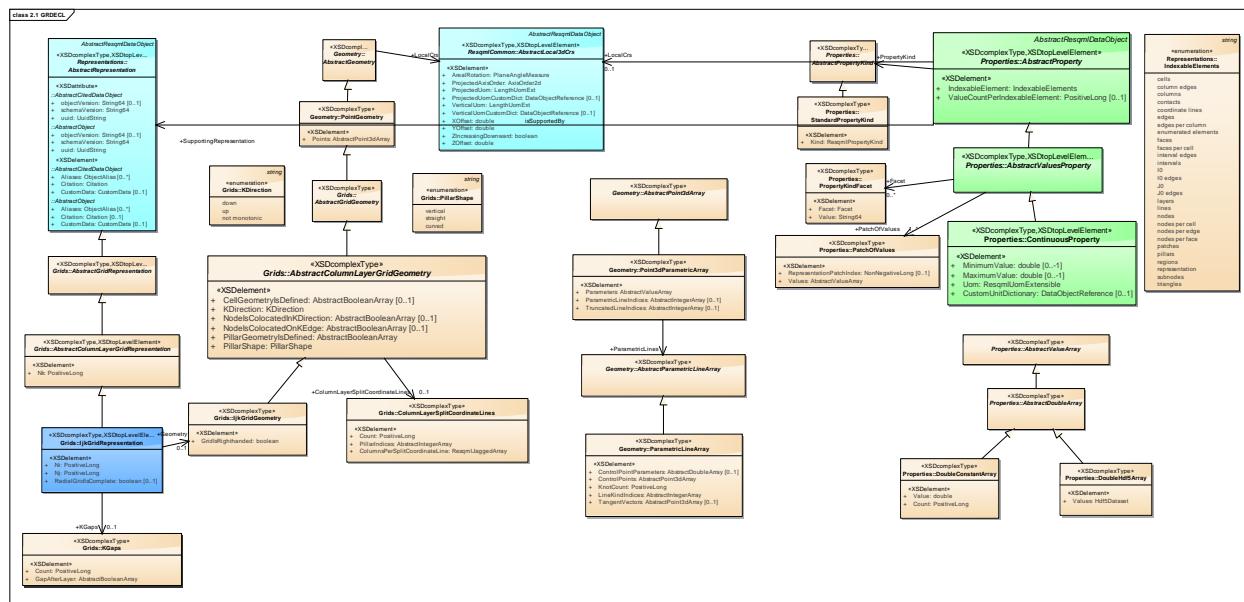


Figure 10-31. Class diagram for an Eclipse grid, including geometry/topology, parametric lines/points, and properties.

This figure is split into its major components in **Figure 10-32**, **Figure 10-33** and **Figure 10-34**, which are explained below.

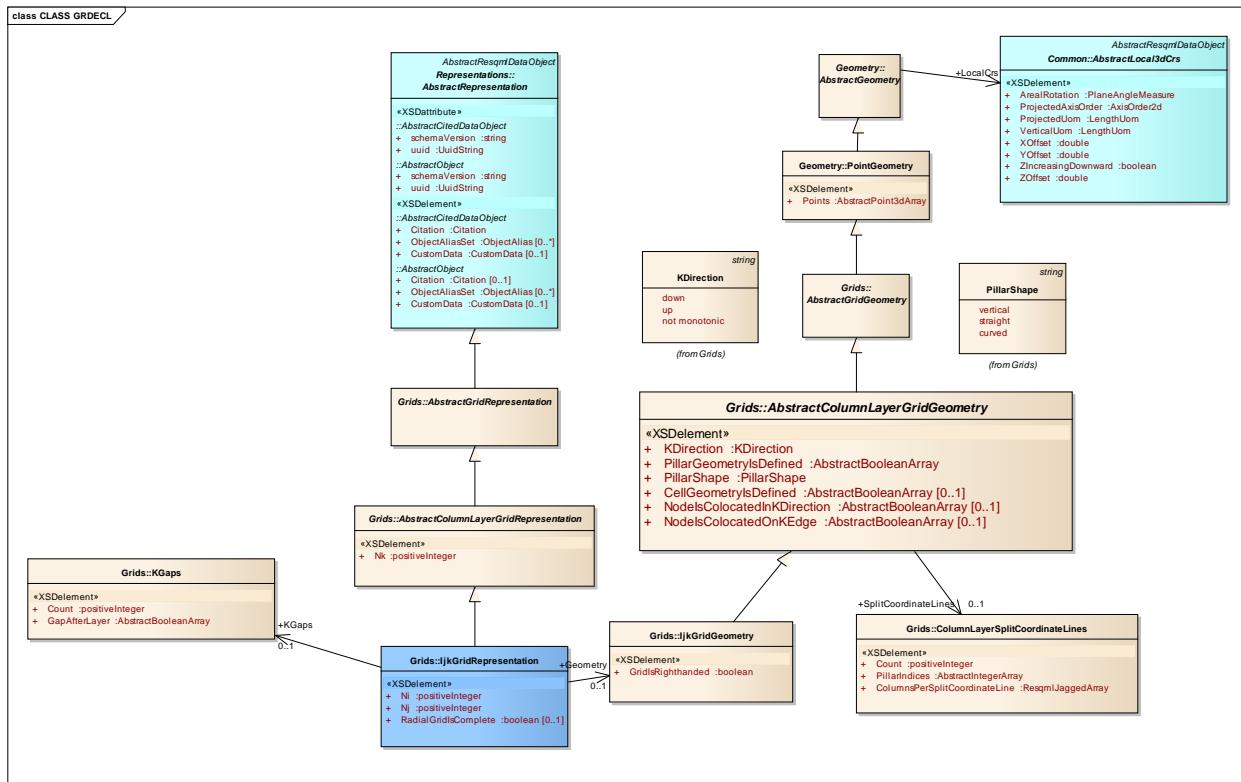


Figure 10-32. Class diagram for the geometry of an Eclipse grid.

The IJK grids have three indices and inherit a UUID as an abstract data object. An Eclipse grid implicitly assumes that all cells are faulted and includes redundant geometric information when they are not. In contrast, RESQML only introduces split coordinate lines and split coordinate line nodes when faults are present, if any. However, in this example, no analysis of the data within the GRDECL file has been performed, and so the cell geometry is stored as if each cell were faulted.

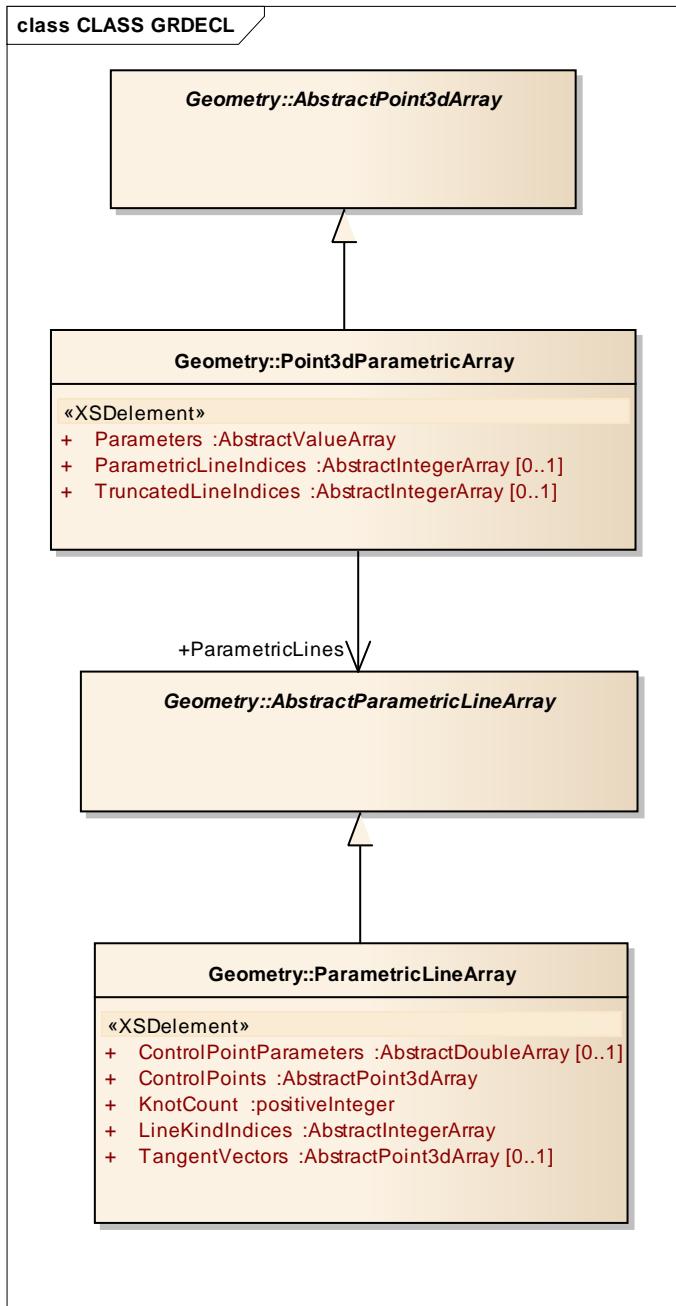


Figure 10-33. Class diagram for the parametric points and lines of an Eclipse grid.

The COORD data from the GRDECL file are used to define a $(NX+1) \times (NY+1)$ array of straight parametric lines, i.e., linear parametric lines with two control points per line. The control points are the XYZ values from the COORD data and the control point parameters are the COORD Z values. The ZCORN data from the GRDECL file are used to describe the cell corner nodes. The array of parametric points will be referenced for all of the $4*NX*NY$ coordinate lines, which themselves reference the $(NX+1) \times (NY+1)$ parametric lines, once for each pillar.

There is one subtlety in this construction. The parametric line indices array in the points 3D parametric array does not need to be specified because it is identical to the pillar indices array in the column layer split coordinate lines. This is an example of the mapping from array index of the parametric points to the

parametric line index already being known in context so that it does not need to be specified a second time. The optional parametric line indices array is available for when this mapping is not otherwise known.

A GRDECL file assumes that Z is monotonic and uses the Z coordinate to provide a linear lookup to X and Y. In contrast, the parametric representations in RESQML parameterize the XYZ coordinates to an arbitrary parameter, P. This parameterization removes all restrictions on the shape of the reservoir, for example, to support overturned reservoirs. This flexibility also allows for simplification in the grid description for specific layering schemes. For example, a parametric value that varies between 0 and NK may be used to naturally describe a proportional layering scheme.

Property treatment is common to all RESQML representations. Properties are attached to the IJK grid representation and follow the indexing of the selected attachment kind, which in this case are cells. The abstract class of values may be instantiated in several ways, including a constant array or an explicit array of values stored as an HDF5 dataset.

Interestingly, the ACTNUM keyword of the GRDECL file does not have a standard interpretation in the industry; its usage varies from application to application. For some applications, a zero value of ACTNUM implies that all properties have been nulled for that cell while for other applications a zero value implies that both properties and geometry are null. To avoid any ambiguity, RESQML does not use "ACTNUM" as such. Instead, RESQML uses:

- an optional explicit Boolean array CellGeometryIsDefined, which indicates exactly that.
- an optional standard property kind called "active". This property kind allows, by means of the instantiation of a Boolean property attached to the grid, to indicate that an indexable element should (or should not) be taken into account.

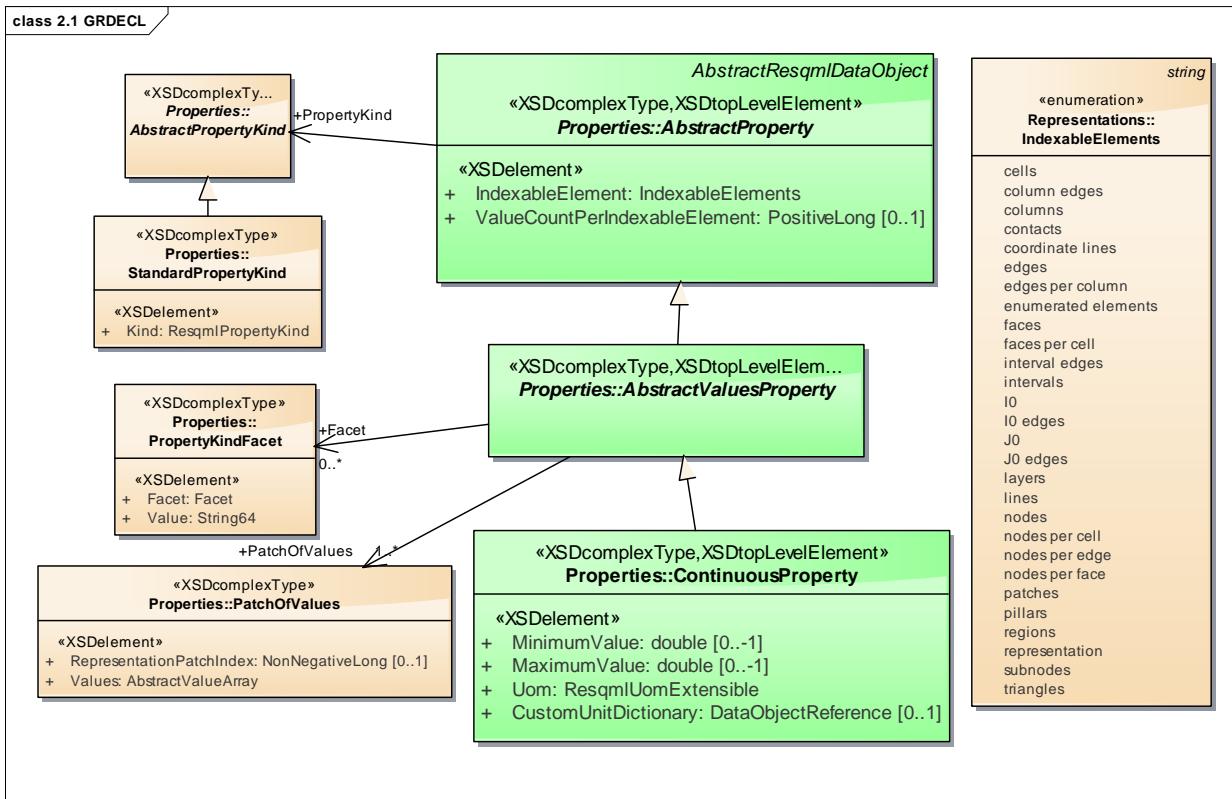


Figure 10-34. Class diagram for the cell properties of an Eclipse grid.

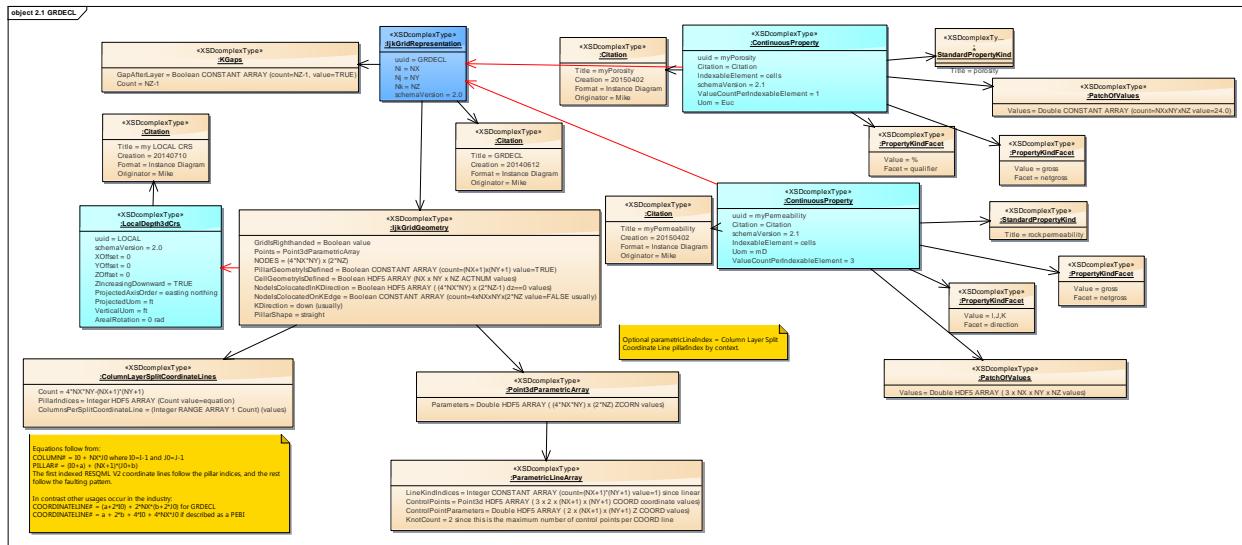


Figure 10-35. Instance diagram of an Eclipse GRDECL file, including properties.

Grid name (UUID) is GRDECL with indices NI=NX, NJ=NY and NK=NZ. The remaining indices are NIL=NX+1, NJL=NY+1, NKL=2*NZ and #Intervals=2*NZ-1. Properties are attached to the IJK grid representation by reference, with an attachment kind of cells. One property (porosity) shows the use of a constant array. The other property (permeability) shows the use of the property count to store the three directional permeability arrays as a single 4D array. The use of property facets allows more details about the property values to be stored, e.g., that these are both gross properties, that the porosity is expressed in percentage, and that the three permeabilities are in the I, J, and K directions.

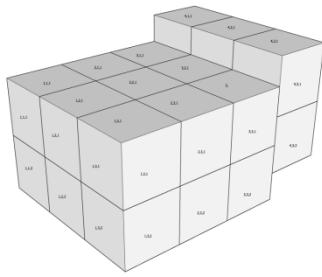
The IJK grid contains the geometry of the coordinate line nodes, and in this case, the data from the ACTNUM array within the omit cells object. The nodes are defined parametrically using the COORD parametric lines. Because the model is faulted, the coordinate lines need an array, which is used to implicitly describe which columns of the model follow the split coordinate lines and which do not. This is the array of pillar indices. This array also indicates how to map from the 4*NX*NY coordinate lines to the (NX+1) x (NY+1) pillars, and is used as part of the parametric points construction.

Coordinate lines may be indexed using a 4D set of indices: (α, β, I, J). The Eclipse GRDECL file and RESQML each reduce this index to a single 1D index, or equivalently choose an order for the coordinate lines. RESQML v2 specifies that the index of each unsplit coordinate line is identical to its corresponding pillar index. Each split coordinate line is defined explicitly by the column per split coordinate line jagged array and the corresponding pillar indices. In contrast, GRDECL does not distinguish between split and unsplit coordinate lines and will have a different, and longer, list of coordinate lines than will RESQML.

10.17.2 4x3x2 Faulted IJK Grid

Figure 10-36 and **Figure 10-37** show an example of a simple grid with regularly spaced vertical coordinate lines and a single fault. The grid has no gaps in IJ or in K.

3D Image: Faulted4x3x2



$DI = 100 \text{ ft}$
 $DJ = 100 \text{ ft}$
 $DK = 50 \text{ ft}$

 Top = 1000 ft
 in West
 Top = 980 ft
 in East

2D Image: Faulted4x3x2

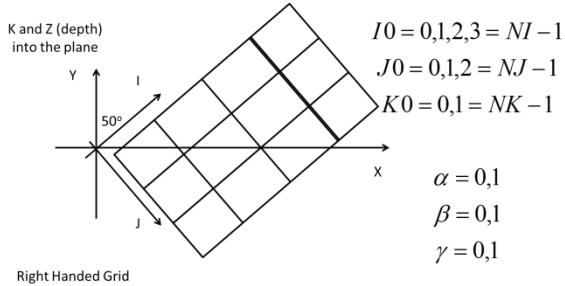
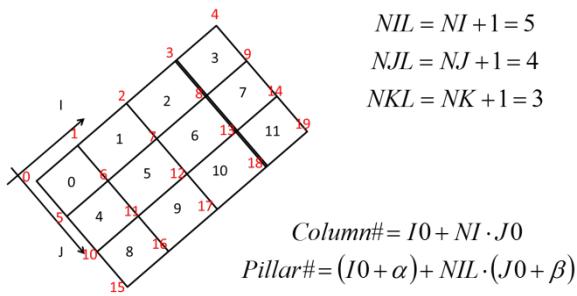
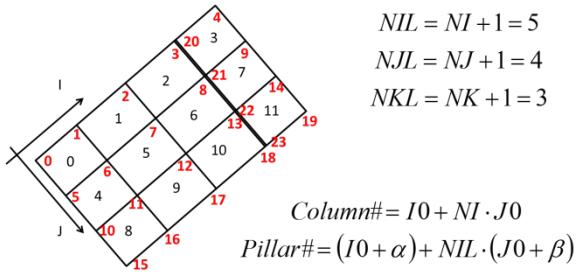


Figure 10-36. 4x3x2 faulted corner-point grid.

Column and Pillar Indices: 4x3x2



Coordinate Line Indices: 4x3x2
(Other Choices Are Possible)



ColumnPerSplitCoordinateLine:
List of Lists

- Array 1 = Offsets for each I Split Coordinate Line
– 1 3 5 6
- Array 2 = Columns Per Split Coordinate Line
– (3) (3 7) (7 11) (11)
– Parentheses have been added for readability

Figure 10-37. Indexing of the elements for the 4x3x2 faulted corner-point grid.

In **Figure 10-37**, the split coordinate lines are specified using a jagged array construction, which consists of two arrays. The first provides the offset into the second array for each element and the second array provides the values. There are 20 pillars in this grid and 4 split coordinate lines. The first 20 coordinate lines correspond to the pillars, and the remaining lines are the split coordinate lines. Because there are no IJ gaps in this grid, the split coordinate lines share the same pillars as the unsplit coordinate lines.

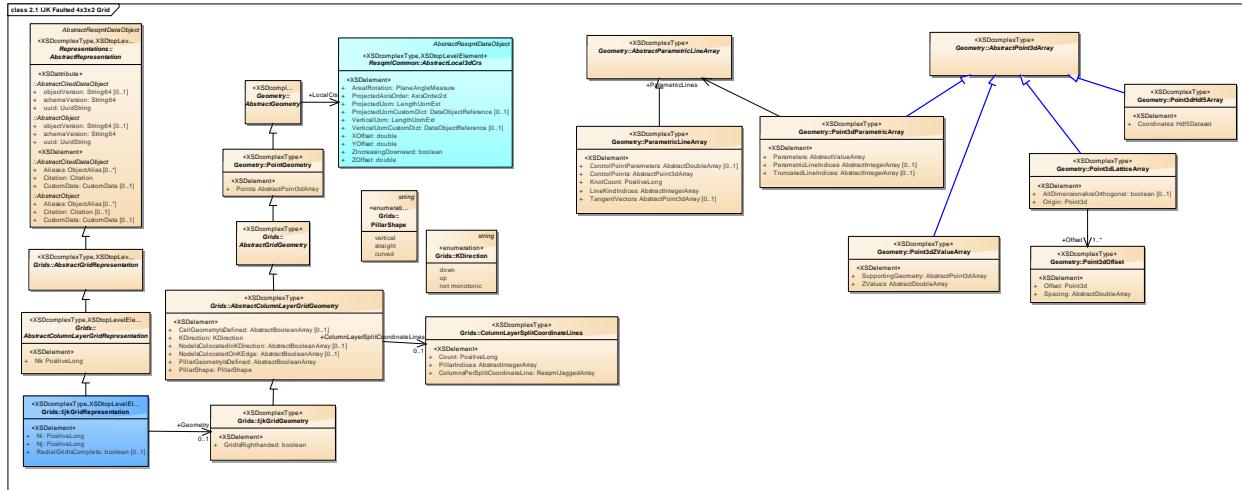


Figure 10-38. Class diagram of an IJK Grid Representation of the 4x3x2 Faulted corner-point grid

The class diagram for this grid (**Figure 10-38**) is similar to that of the GRDECL grid (**Figure 10-31**), although with fewer options. The grid description has neither IJ nor K gaps, the grid spacing is regular with vertical pillars, and all cells have their geometry defined. Property attachment is not part of this example. Because of the regularity of this grid, we can demonstrate implicit parameterizations for the nodes. For this example, values for the cell dimensions and nodal positions are available.

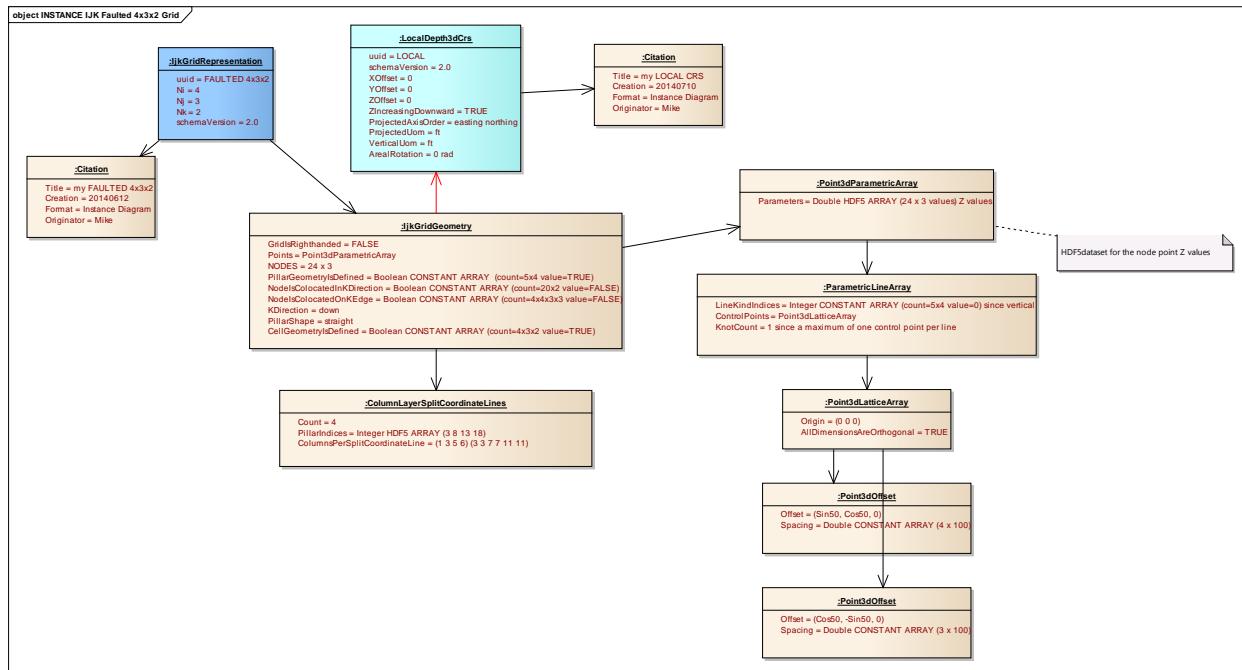


Figure 10-39. Instance diagram for an IJK Grid Representation of the 4x3x2 Faulted corner-point grid.

The parametric line indices, or equivalently the pillar indices, provide the mapping from the coordinate lines to the parametric lines. In this case, the parametric lines are vertical with a regular XY spacing. They are specified using the array of lattice points 3D for an orthogonal lattice. The Z coordinate of each node is given explicitly on a 24x3 array. For this simple grid, the values are (1000 1050 1100) or (980 1030 1080) per coordinate line, depending on which fault block of the grid the line is positioned within.

10.17.3 Fault Representations on the 4x3x2 Faulted IJK Grid

There are two recommended ways of creating a representation of a fault given a grid. If the fault is a pillar fault, i.e., follows the pillars of a structure through the structure without stair-stepping, then the fault may be represented using a pillar-based subrepresentation of the structure, which in this case is a faulted grid. This places the fault in the grid, but does not explicitly specify the topological support of the fault on the grid. The latter can be constructed using a grid connection set representation, which is based upon all of the cell-face-pairs of the grid which intersect the fault. **Figure 10-40** is a class diagram which contains both the grid subrepresentation, the grid connection set representation, and the fault feature and interpretation. **Figure 10-41** (page 188) is the instance diagram for the same objects.

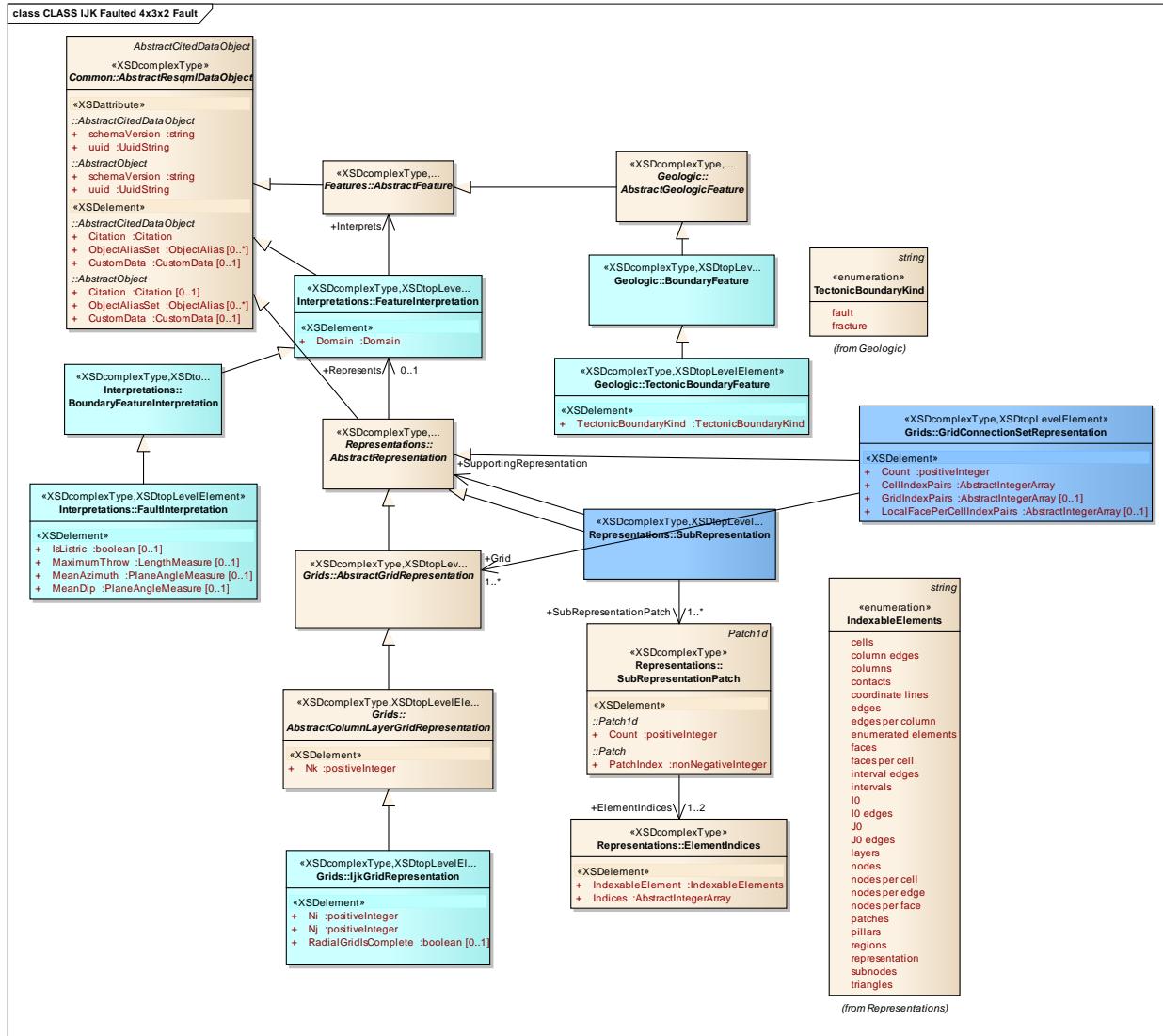
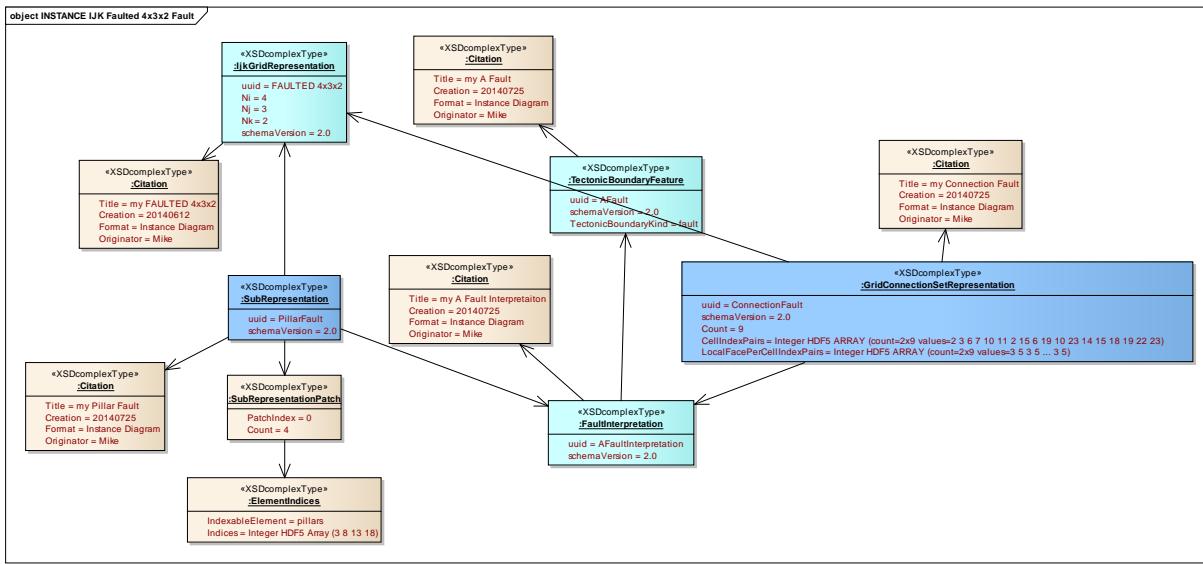


Figure 10-40. Class diagram for a subrepresentation and a grid connection set representation which provide the recommended representations for a fault.



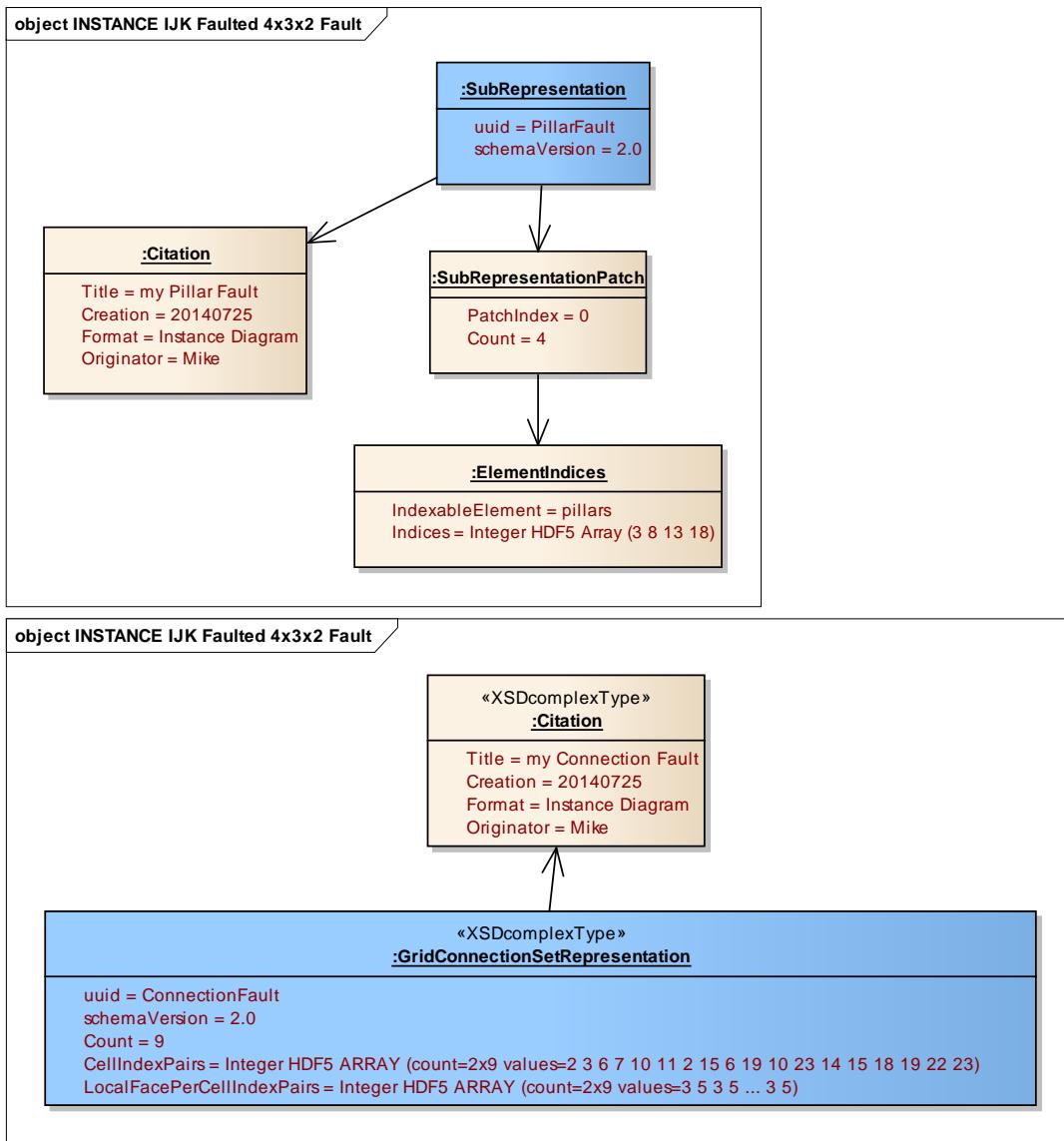


Figure 10-41. Instance diagram for a subrepresentation and for a grid connection set representation for the fault of the Faulted 4x3x2 grid (a) Overview (b) Subrepresentation (c) Grid connection set representation.

In the instance diagram, it can be seen that the pillar-based subrepresentation provides a very simple fault representation, based upon the four pillars that support the fault. The grid connection set representation is a much more detailed representation, with, in this case, 9 connections. For instance, the first connection is from the J+ face of cell (I,J,K) = (1,3,1) to the J- face of cell (I,J,K) = (1,4,1). Cell indices are 2 and 3. Local cell face indices are 3 for J+ and 5 for J-. These are listed in detail in the instance diagram.

10.17.4 Association between IJK Grid and Horizons

The following example (**Figure 10-42**, **Figure 10-43**, **Figure 10-44**) is used to demonstrate how to specify a relationship between the interval edges of a grid and the horizons of an earth model. In this specific example there are three stratigraphic units (Unit A, Unit B, Unit C) bounded by four horizons (H1, H2, H3, H4). The reservoir grid has been built only for Unit B, which in this case, is the only unit with appreciable productivity. This example shows how to use a stratigraphic column to express the relationship between the top and base of the reservoir grid and horizons H2 and H3, which bound Unit B. For the grid this involves constructing a subrepresentation of the two interval edges of the grid. For the stratigraphic column this involves constructing a column of stratigraphic rank two, consisting of Unit B and horizons H2 and H3. A more complete stratigraphic column of rank one also exists, consisting of all three units and all four horizons. However, the rank one stratigraphic column is not referenced directly, other than implicitly through the shared horizon interpretations and the statement of the differing ranks of the stratigraphic columns.

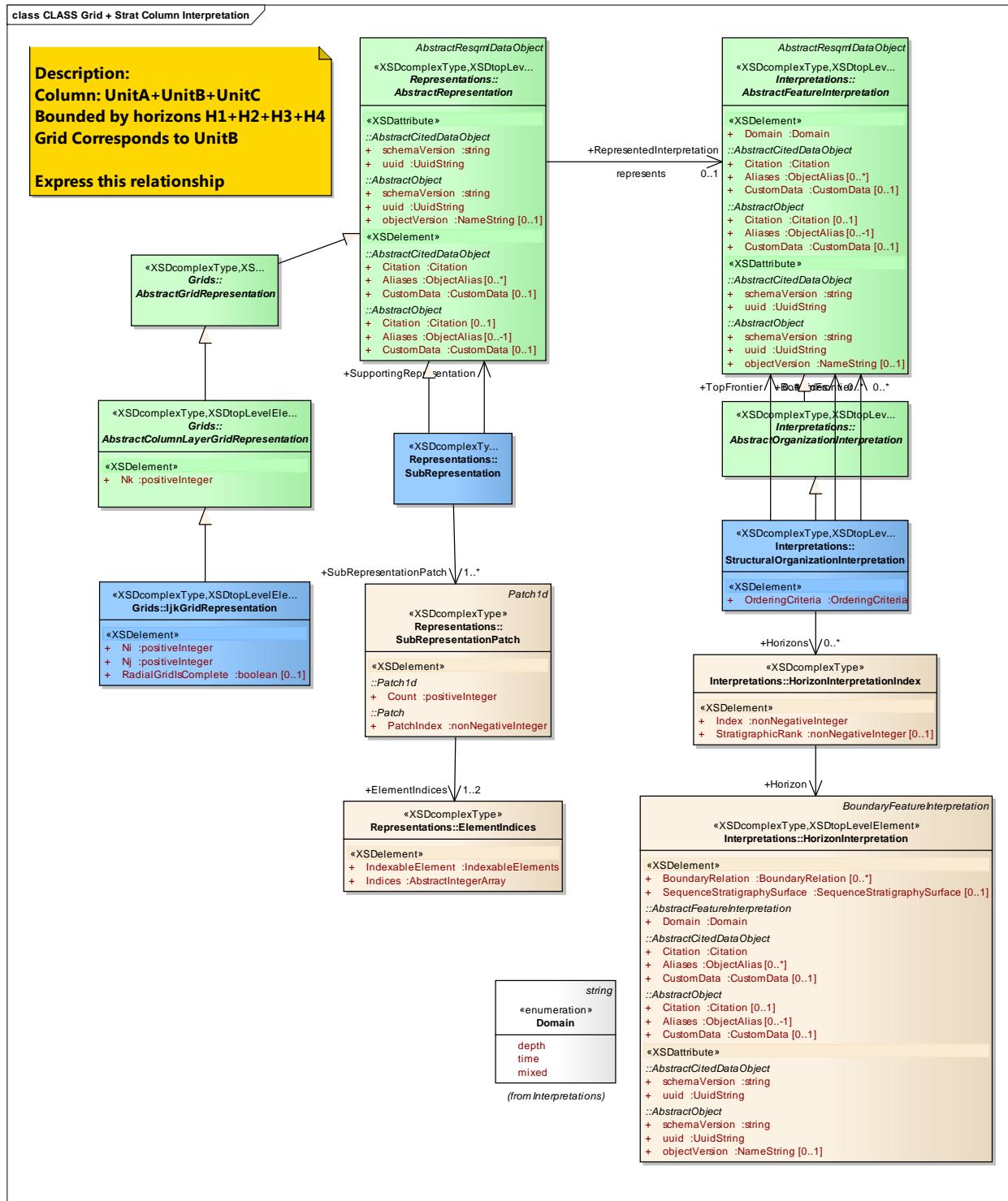


Figure 10-42. Class diagram for a grid, a subrepresentation and horizons

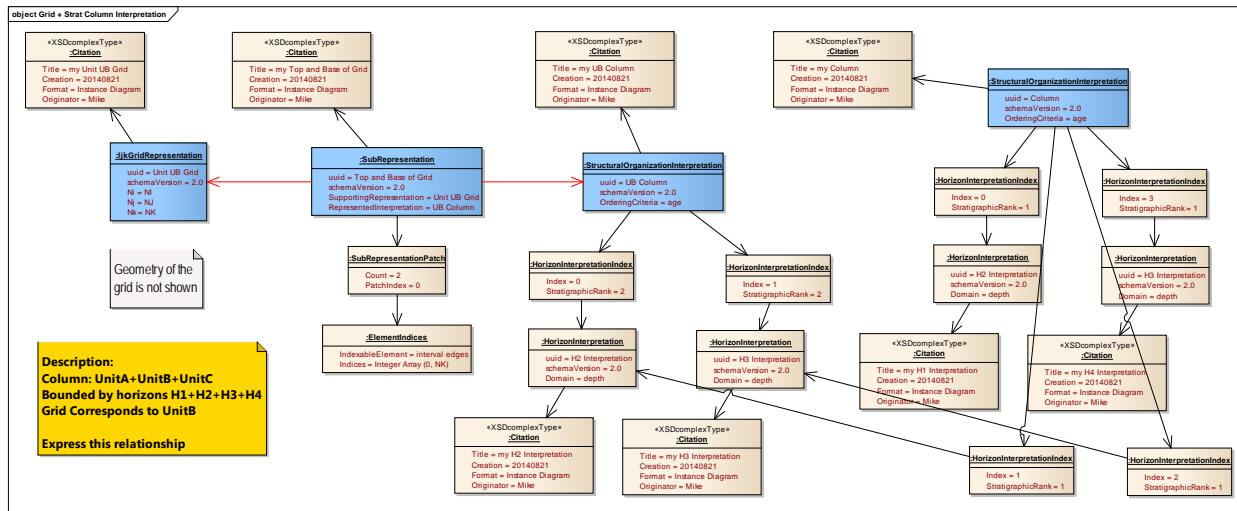


Figure 10-43. Instance diagram showing an IJK grid, a subrepresentation and two stratigraphic columns.

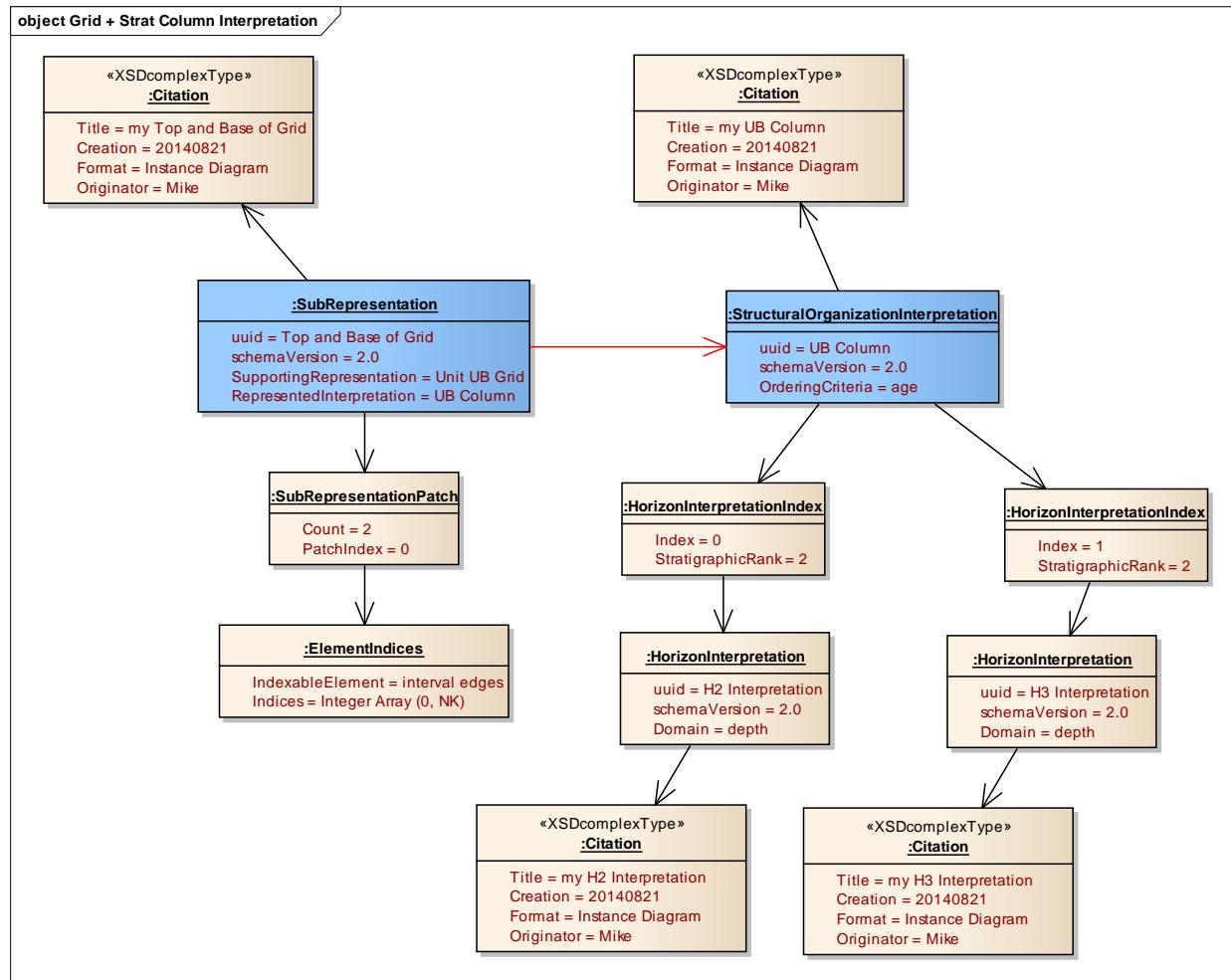


Figure 10-44. Instance diagram showing only the subrepresentation of the grid and the rank two stratigraphic column.

10.17.5 Association between IJK Grid and Stratigraphic Column

The following example (**Figure 10-45,Figure 10-46**) shows how to specify a relationship between the K layers of a grid and the units of a stratigraphic column. This specific example has two stratigraphic units: TopUnit and BaseUnit. The reservoir grid has been built so that the first two K layers correspond to TopUnit and the last three K layers correspond to BaseUnit.

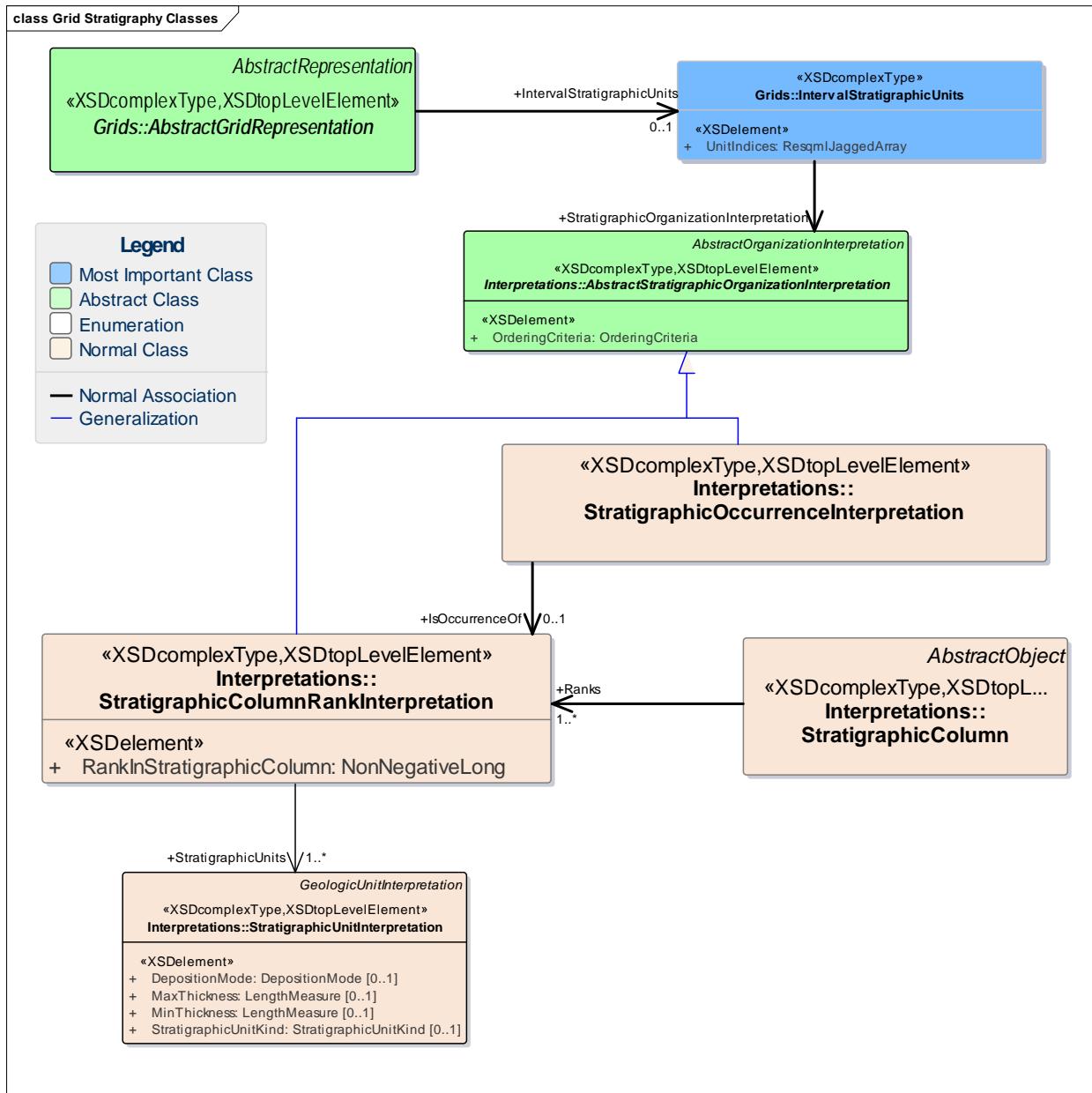


Figure 10-45. Class diagram showing only the grid and stratigraphic classes involved.

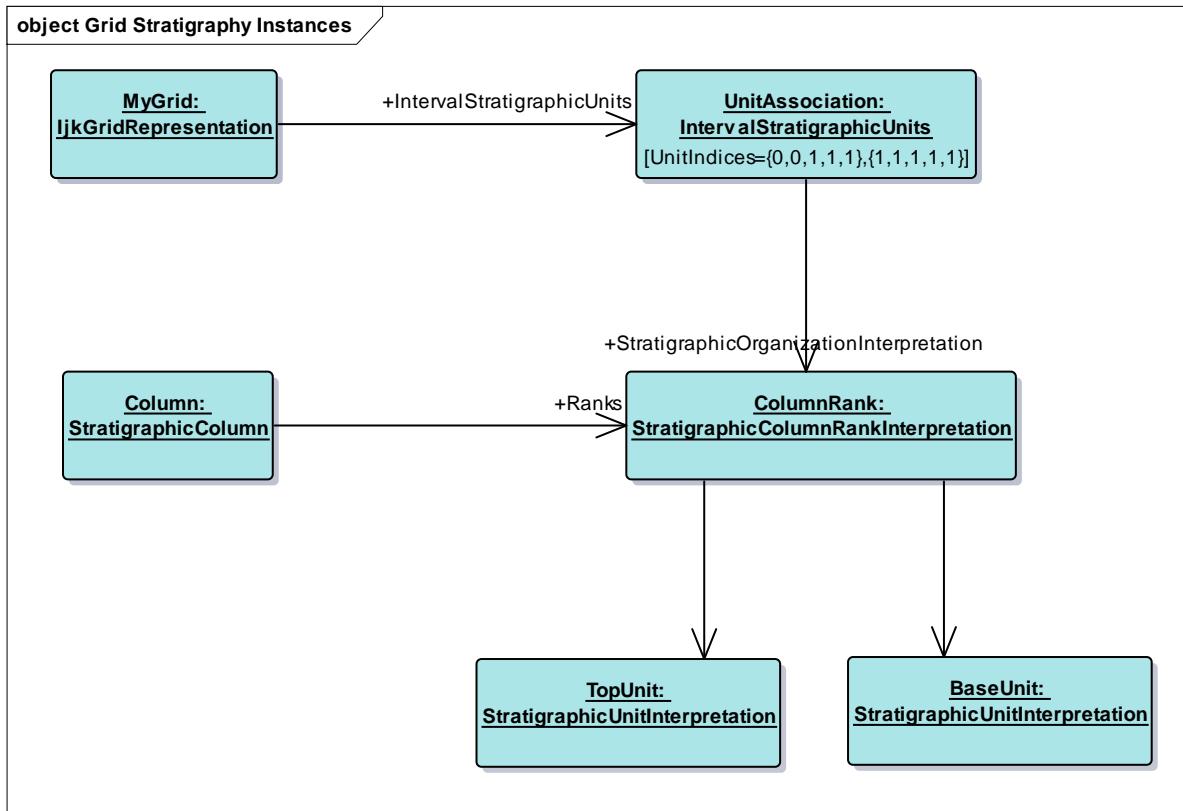


Figure 10-46. Instance diagram showing link between grid and stratigraphic column units.

Because each K layer is associated to a single unit, the cumulative length array of interval stratigraphic units contains only 1, while the elements contains, for each K layer, the index of stratigraphic unit in the stratigraphic rank interpretation.

10.17.6 Small IJK Grid with Nested Local Grid Refinement

Figure 10-47 and **Figure 10-48** is a grid example that shows the use of grid parentage and local grids. Grid parentage allows one grid to inherit geometry and properties from a parent grid, but has no immediate impact on the parent grid. Local grid sets are used to “activate” local grids and replace portions of the parent grid by the corresponding child grid or grids. (For more information on local grids, see Section 10.8 (page 151).) The RESQML construction is sufficiently general to allow both nested and irregularly shaped local grids. It is also sufficiently general to support refinement, coarsening, or combinations of the two.

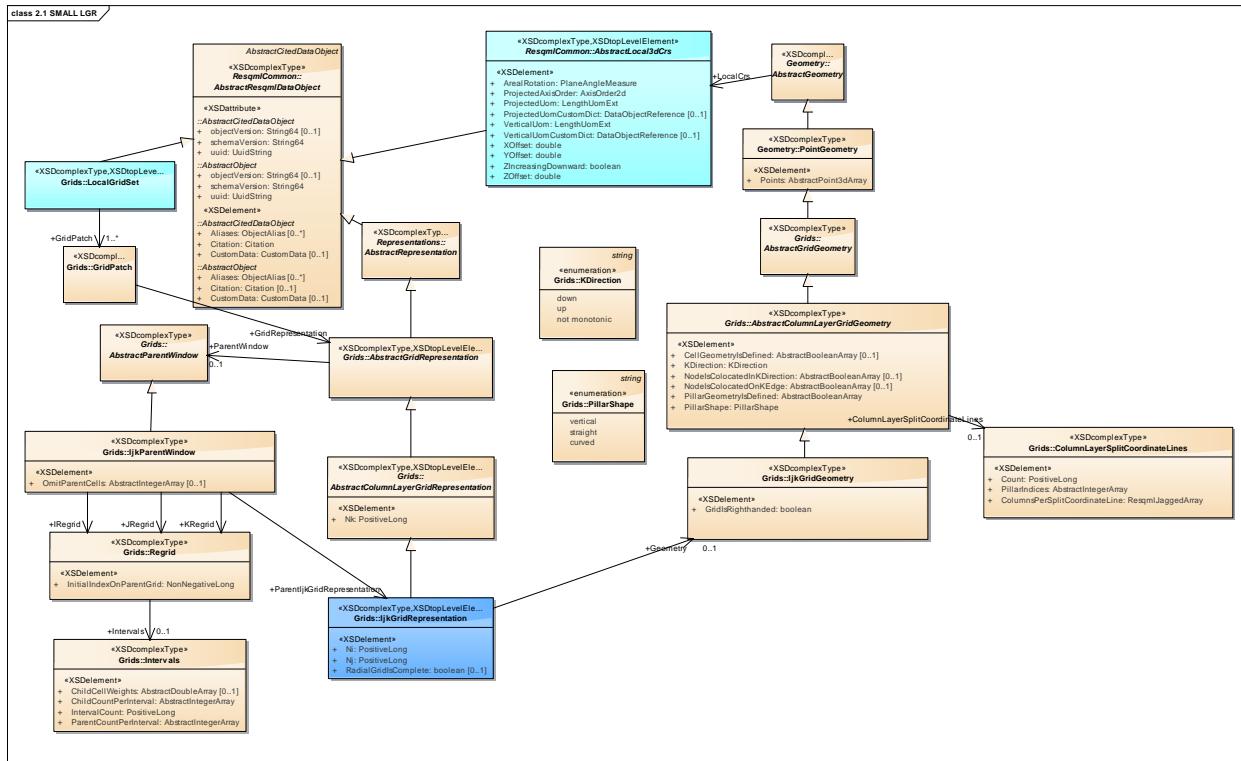


Figure 10-47. Class diagram for the nested LGR example.

Figure 10-47 is similar to the IJK grids examined previously, with the exception of the grid parentage construction. No properties are included within this example.

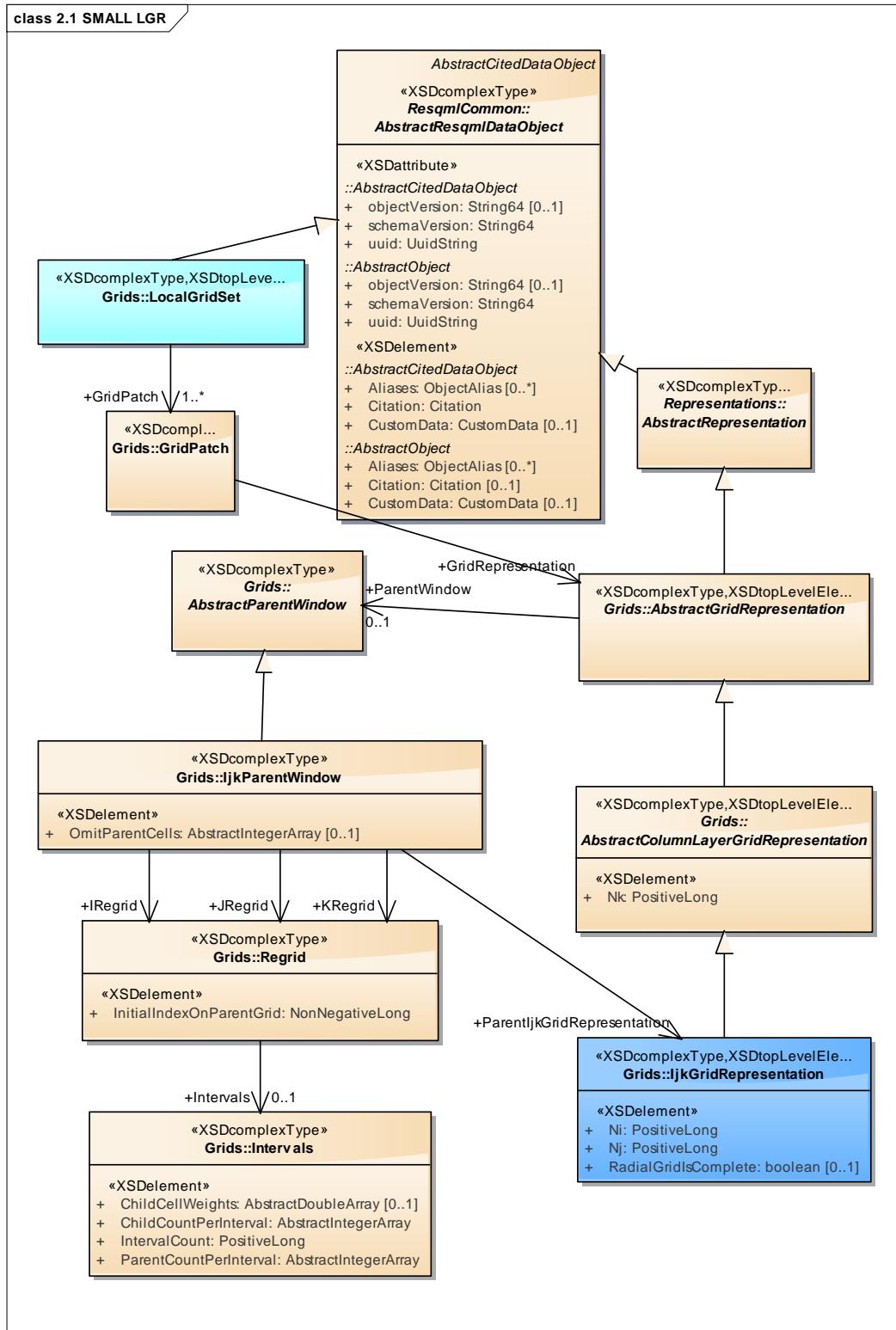
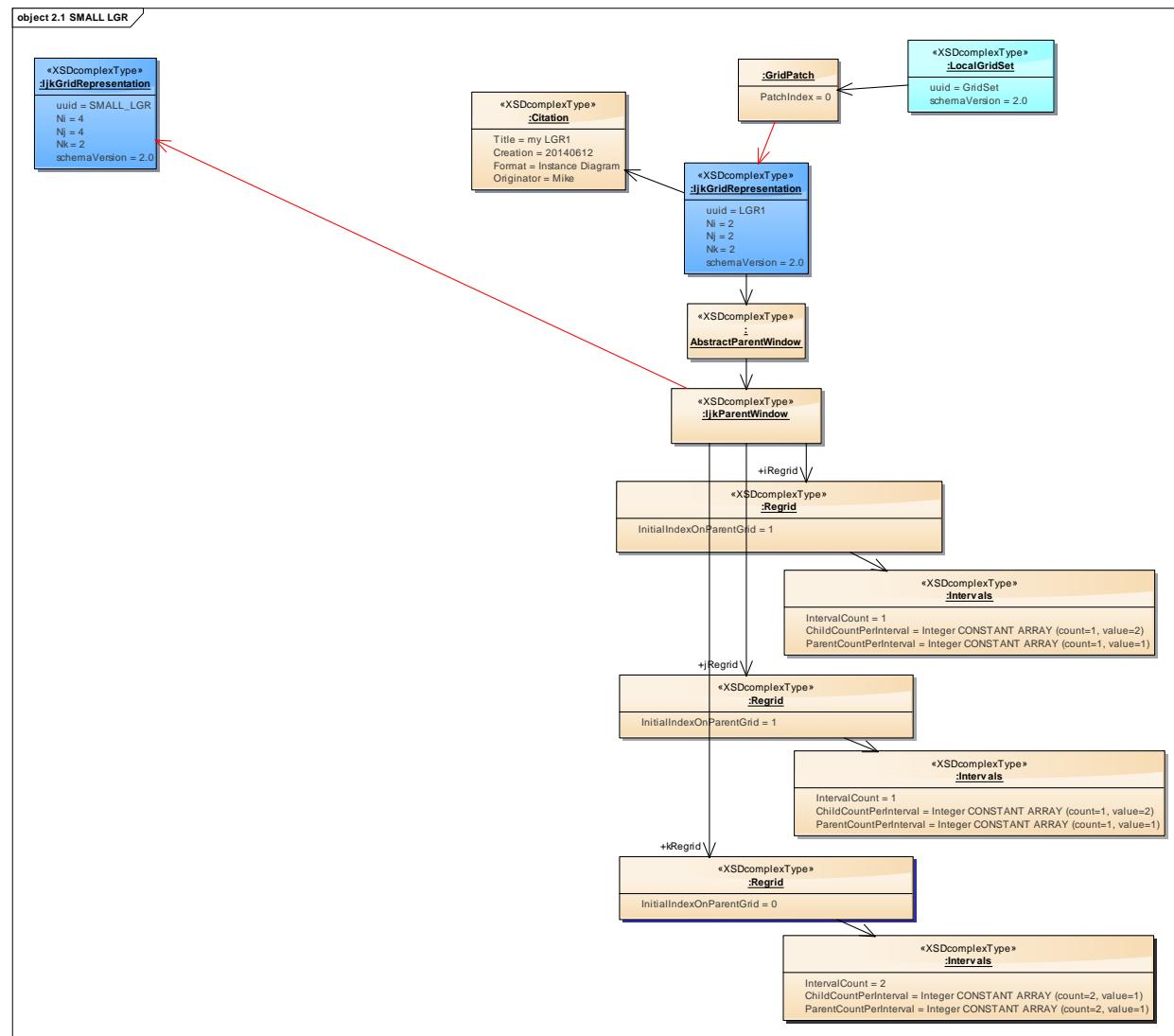
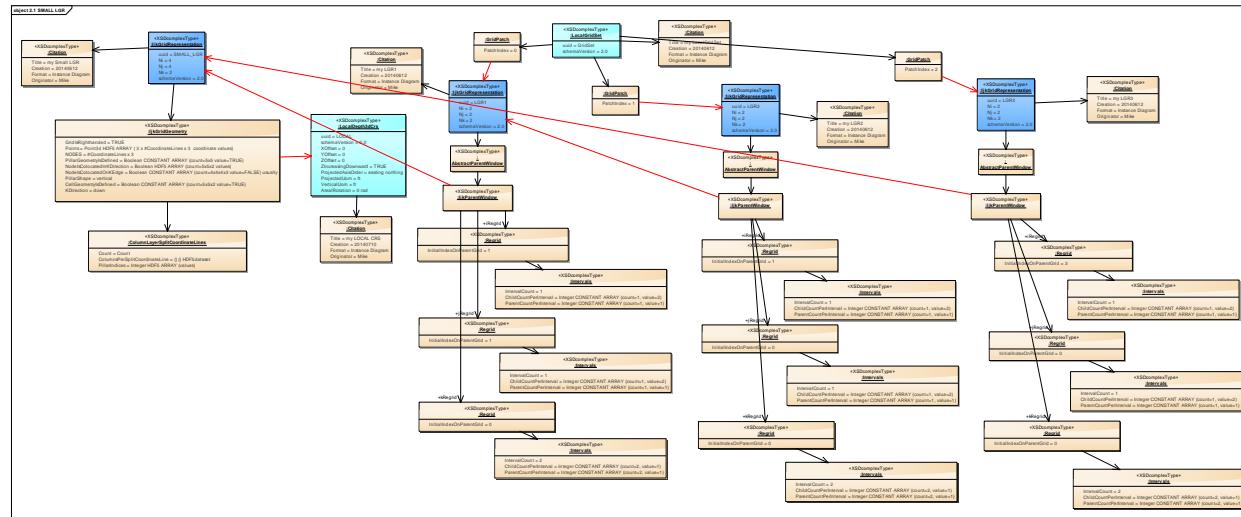


Figure 10-48. Class diagram showing only the parent-child grid specification.

Figure 10-48 shows the portion of the grid parentage construction based on an IJK grid parent. It supports I, J, and K regrid specifications.



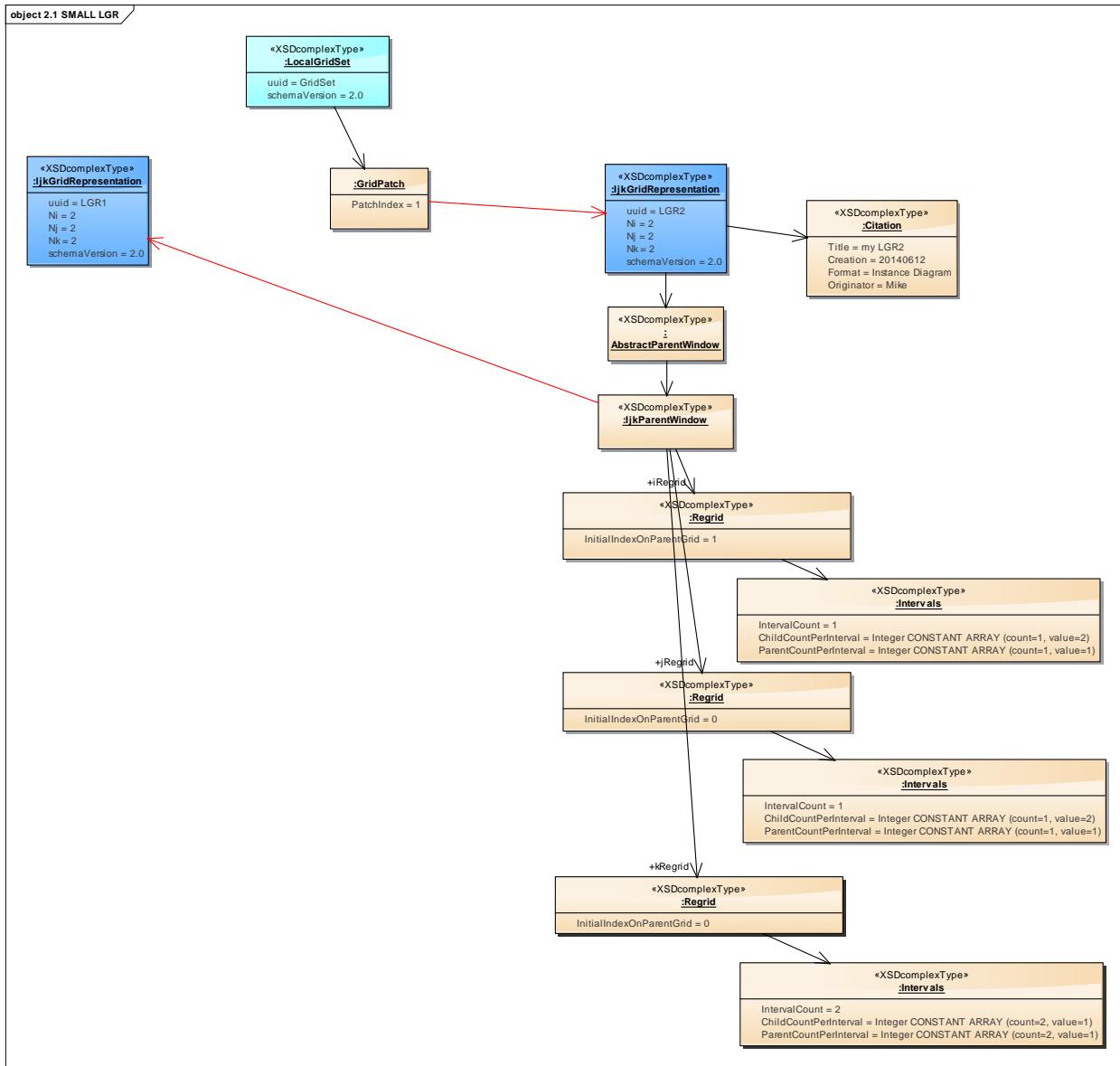


Figure 10-49. Instance diagram for the small nested LGR IJK grid example showing grid parentage and local grid set definitions. The constructions for LGR1 and the nested LGR (LGR2) are shown in more detail.

In **Figure 10-49**, the grid on the far left of the top diagram (SMALL_LGR) is a grid with explicit specification of its geometry. Local grid set 1 activates grids LGR1 and LGR3. Local grid set 2 activates grid LGR2. LGR1 and LGR3 have SMALL_LGR as their parent. LGR2 is nested in LGR1.

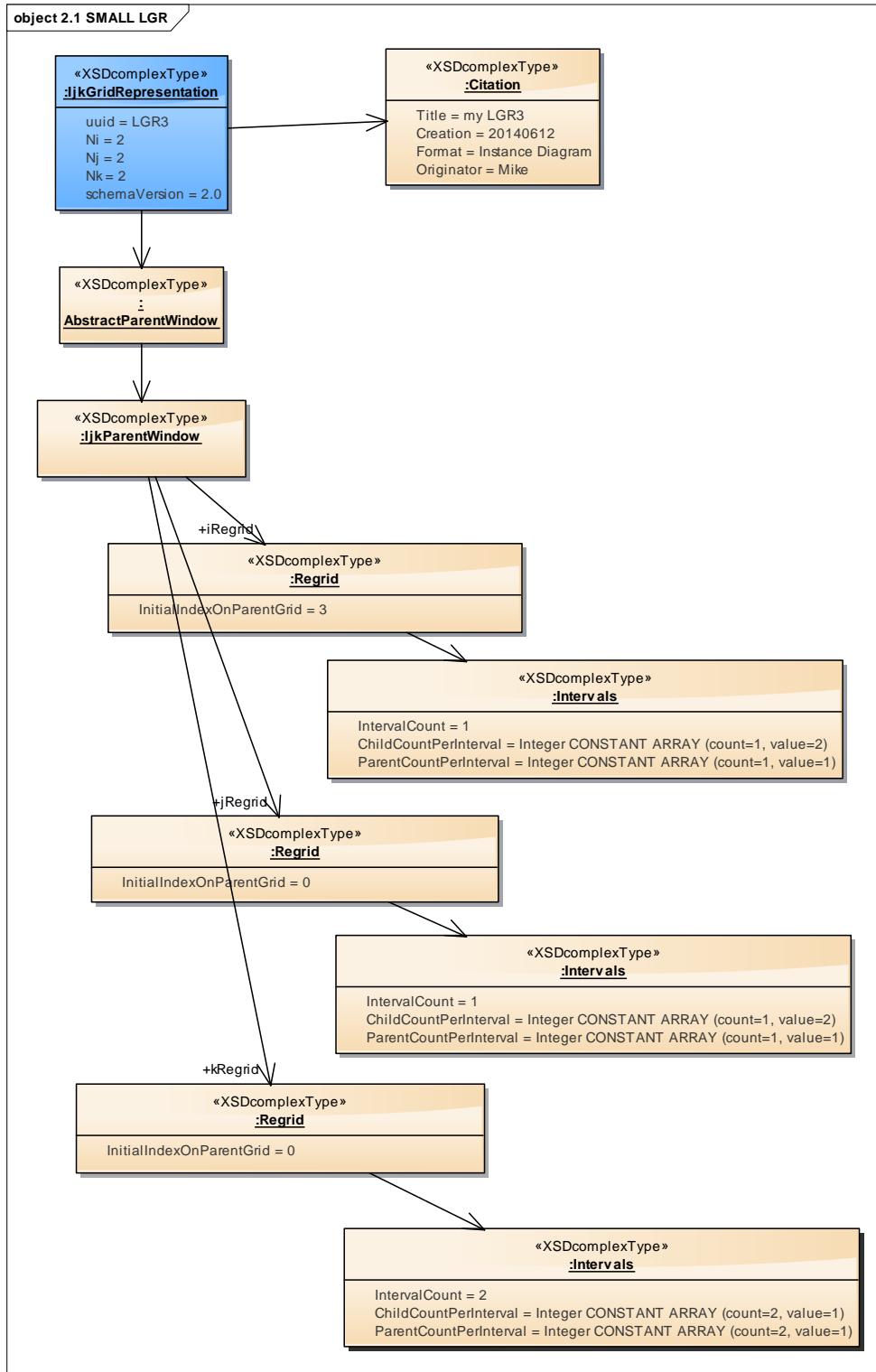


Figure 10-50. Regrid construction for grid LGR3.

In **Figure 10-50**, we examine the regrid construction for grid LGR3. The parent window is positioned starting at parent cell $(I,J,K)=(4,1,1)$. The I and J regrids are identical and represent a factor of 2 grid

refinement. The K regrid uses two intervals to indicate that K=1 is mapped to K=1 and that K=2 is mapped to K=2.

10.17.7 IJK Grid with Proportional Layering

In **Figure 10-51**, we can see how this example is built on the 4x3x2 faulted IJK grid, but now we take advantage of the parametric form of proportional layering, to convert the two layer model to a two zone model with 6 layers in the first zone and 18 layers in the second zone, for a total of 24 layers.

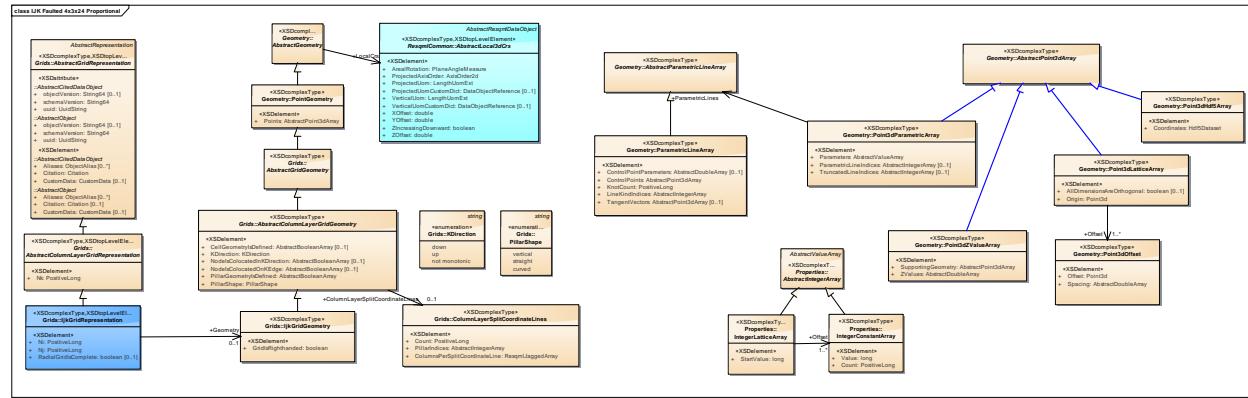


Figure 10-51. Class diagram for the 4 x 3 x 24 layer proportional IJK grid.

Specifically, as can be seen in **Figure 10-52**, the parameterization takes advantage of the interval edge index, which in this case takes on the values of 0...24. Interval edges 0..6 describe the top geologic zone and interval edges 6..24 describe the bottom. Along a single coordinate line it is not necessary to specify all 25 nodes explicitly. Instead nodes 0, 6, and 24 can be specified, and linear interpolation can otherwise be used to define layers within each zone. This is shown in the figure. The original points from the 4x3x2 grid now arise as the control points. There are now two sets of parametric lines. The first set consists of 24 straight lines, with three control points and three parametric values per line (one per coordinate line). The second set consists of 5x4 vertical lines, with one control point per line (one per pillar). To minimize any ambiguity in indexing, the second set of parametric lines now includes the explicit parametric line indices, although they are not strictly necessary.

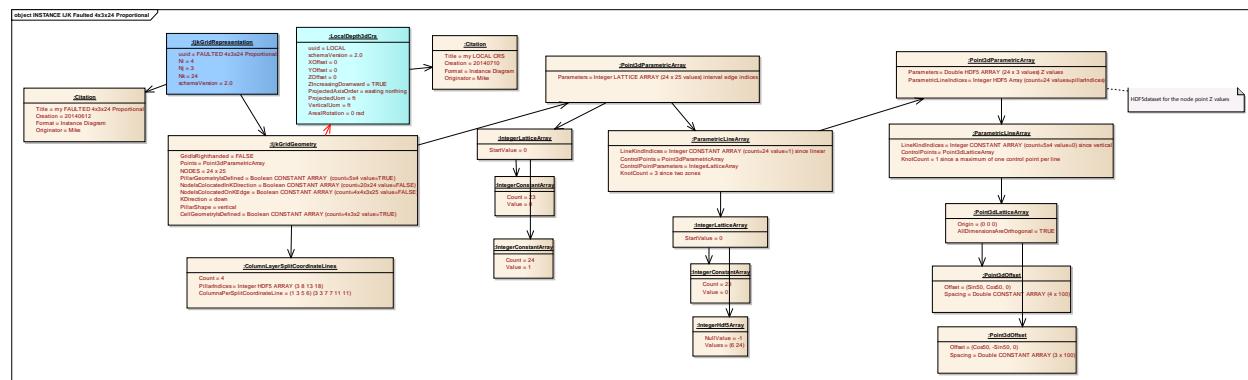


Figure 10-52. Instance diagram for the 4 x 3 x 24 layer proportional IJK grid.

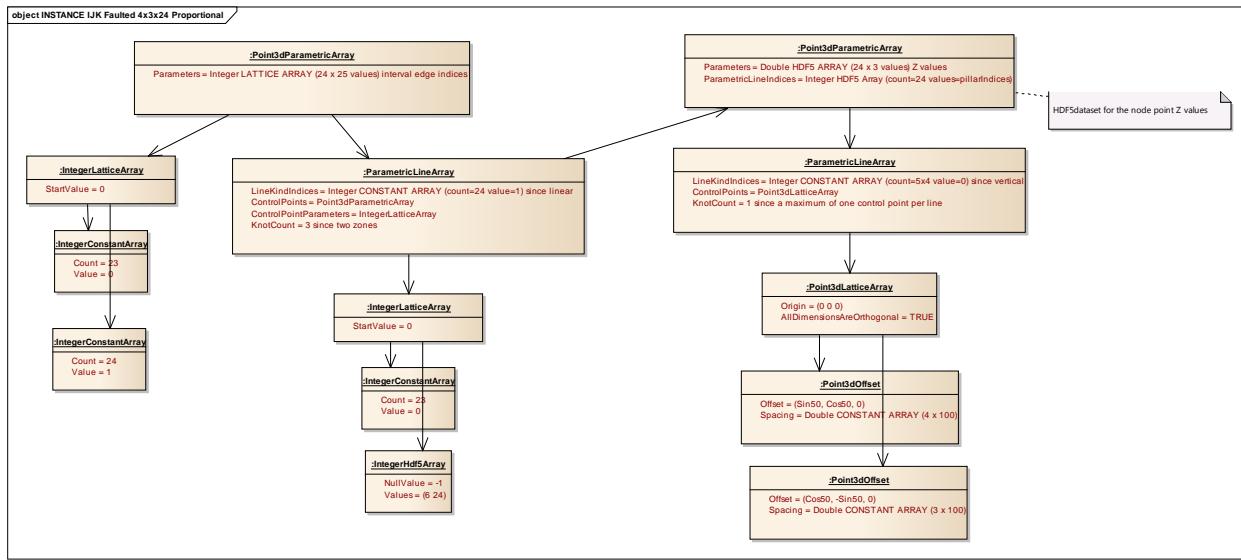


Figure 10-53. Instance diagram showing proportional layering (6+18 layers).

Notice in **Figure 10-53** that the integer lattice constructions, which have two dimensions, only have the second dimension (dimension=1) in the instance diagram. This is because the parameters for proportional layering depend only on the interval indices. In contrast, the generalization of this construction to eroded and truncated layering would, in general, require explicit parametric values that vary from line to line.

10.17.8 Higher Order Finite Element Prism24 Unstructured Column-layer Grid

This example demonstrates an unstructured column-layer grid representation and the use of higher order subnode geometry to create a distorted prism cell shape. For simplicity, this example is of an unfaulted grid.

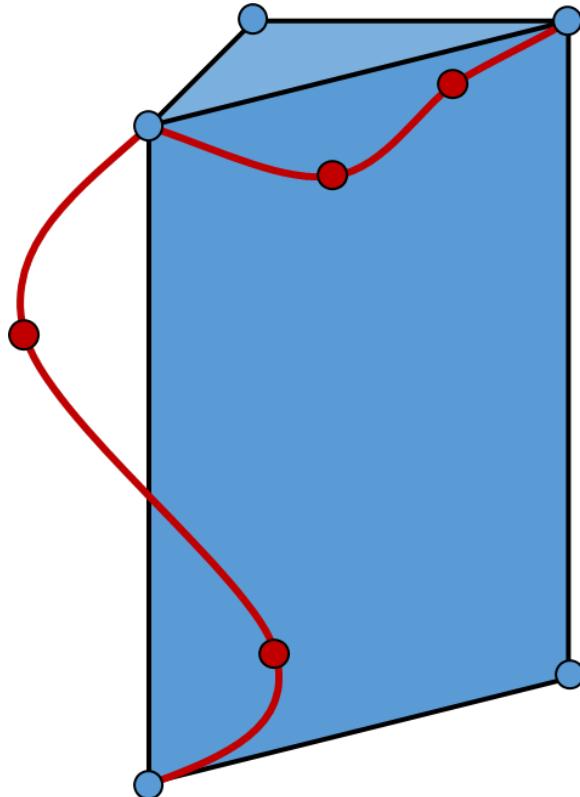


Figure 10-54. Description of a single lower order (blue prism) and higher (distorted prism) grid cell. All nine edges are distorted but not all are shown.

As shown in **Figure 10-54**, there are six blue nodes, defining the shape of the lower order triangular prism. There are two additional higher order nodes per edge, for each of the nine edges, which describe the distortion of each edge away from linearity. (Only two edges are shown.) The resulting distorted prism is described by a total of 24 nodes.

The description of the lower order aspects grid is similar to that of the IJK grid except that the grid uses an unstructured column-layer cell geometry (**Figure 10-55**). The unstructured column-layer cell geometry inherits from the column-layer cell geometry used for the IJK grid, but includes additional elements that explicitly describe the pillars and columns of the model. The higher order finite element grid is a simple kind, with two subnodes defined for each edge. The edges and the nodes per edge must be defined explicitly before they can be referenced by the subnode construction.

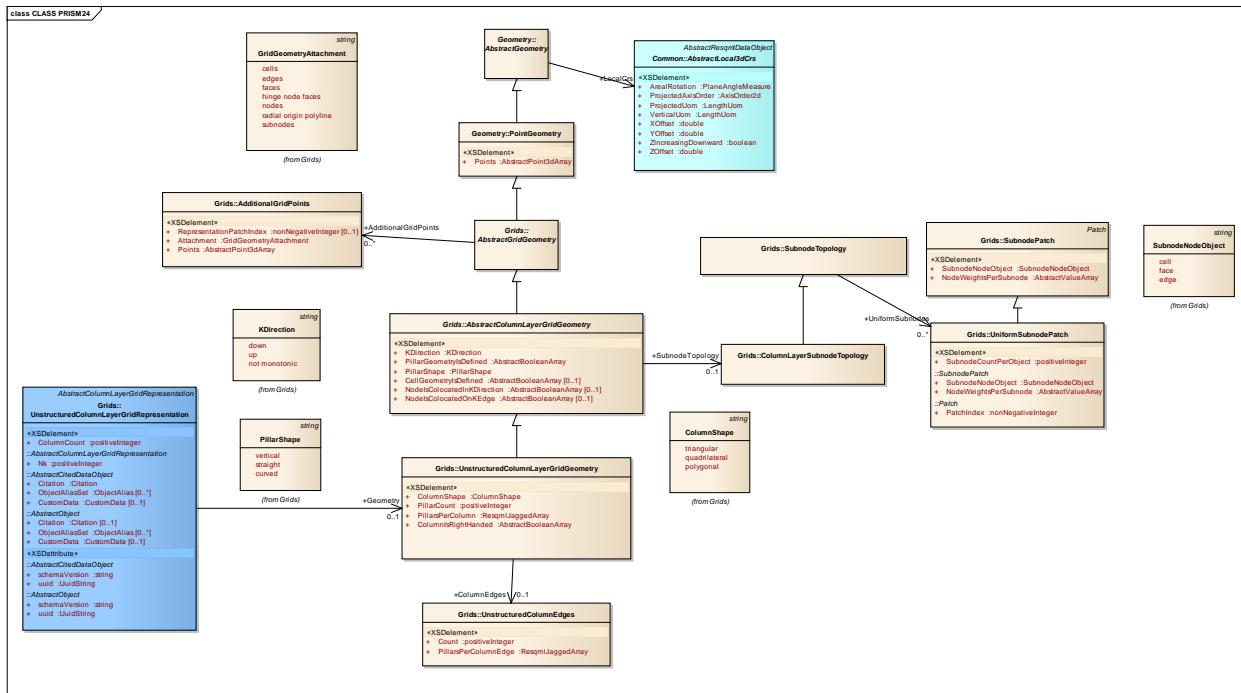


Figure 10-55. Class diagram for the Prism24 grid.

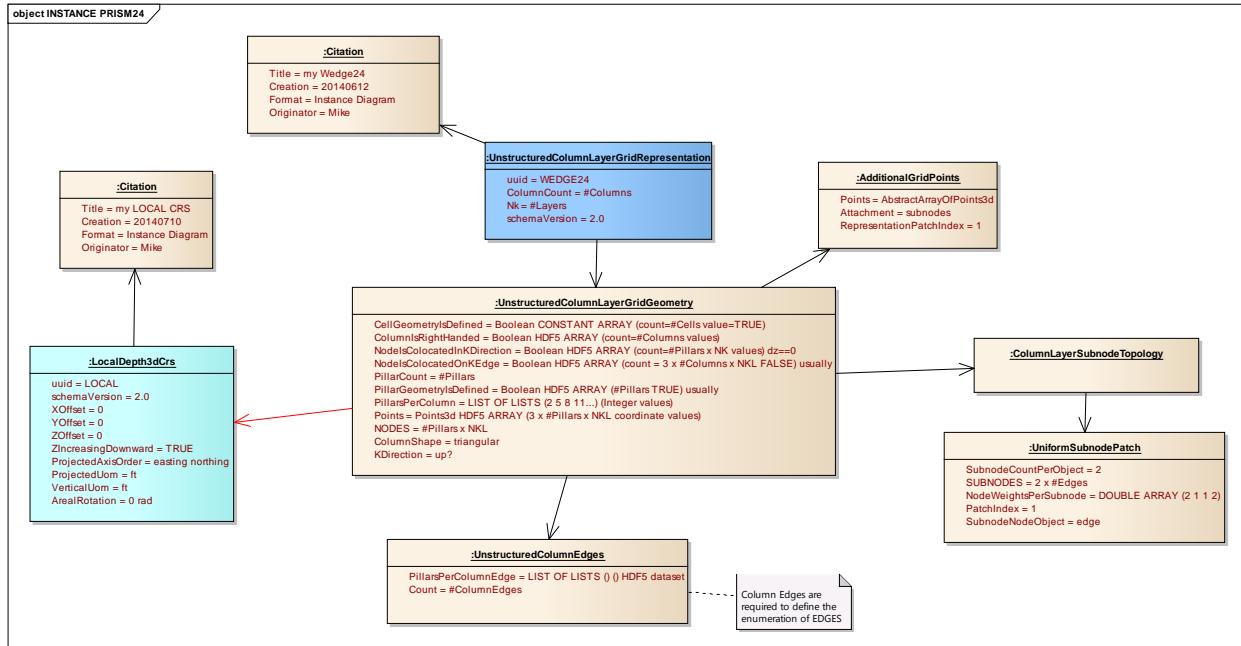


Figure 10-56. Instance diagram for the Prism24 grid.

Section 10.13.3 (page 172) explains the edge indexing for an unstructured column-layer grid is given by “ColumnEdgeCount x NKL + CoordinateLineCount x NK + SplitEdgeCount”. In **Figure 10-56**, the column edges are defined explicitly using a jagged array construction. Because the grid is unfaulted, the coordinate line count is identical to the pillar count. This model has no split edges, hence the edge indexing is known. There are two subnodes per edge, defined using the uniform subnode patch. Each subnode is defined parametrically by the two nodes per edge: the first with weights of (2 1) and the second with weights of (1 2). In other words, subnode1 = (2/3)*edgenode1 + (1/3)*edgenode2 while

$\text{subnode2} = (1/3)*\text{edgenode1} + (2/3)*\text{edgenode2}$. Once defined, these subnodes may be used to provide topological support for higher order properties or geometry. In this example, the positions of the subnodes are modified when higher order geometry points are attached, giving the distorted shape. As a reminder, the schema itself does not describe the method of geometric interpolation between the subnodes on an edge or between the subnodes on a face, although natural interpolation algorithms do arise. For instance, in this figure, edge interpolation follows a natural cubic spline and face interpolation follows the iso-parametric mapping of the lowest order object.

10.17.9 Block-Centered General Purpose Grid

Figure 10-57 (class diagram) is a very simple example, which is included to show the use of grid patches in the general purpose grid. **Figure 10-58** (instance diagram) consists of four grid patches (1, 2, 3, and 6). Because this is a block-centered grid with no explicit geometry, the local 3D CRS does not appear as part of the representation. Instead, it may appear as part of the geometry valued properties attached to this grid.

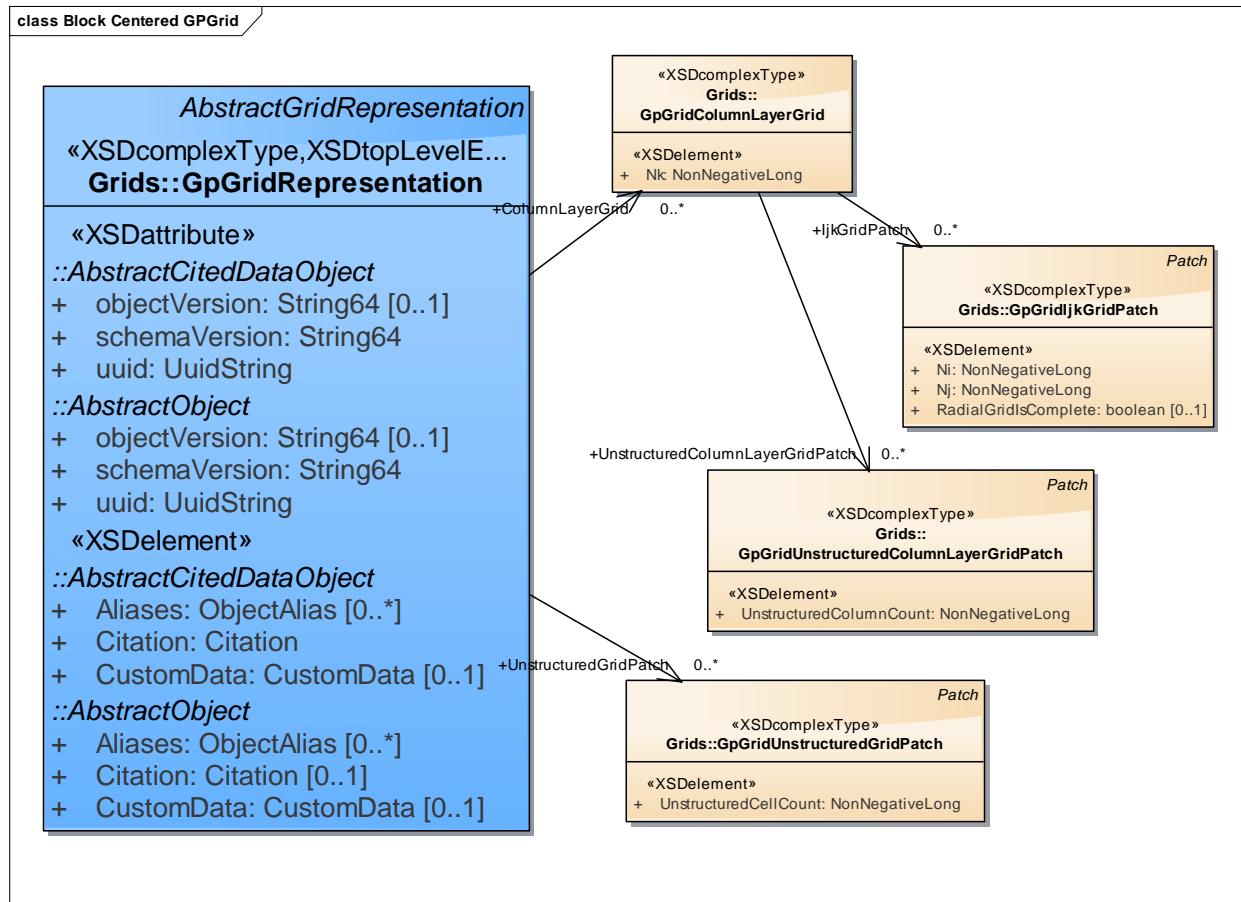


Figure 10-57. Block-centered general purpose grid class diagram.

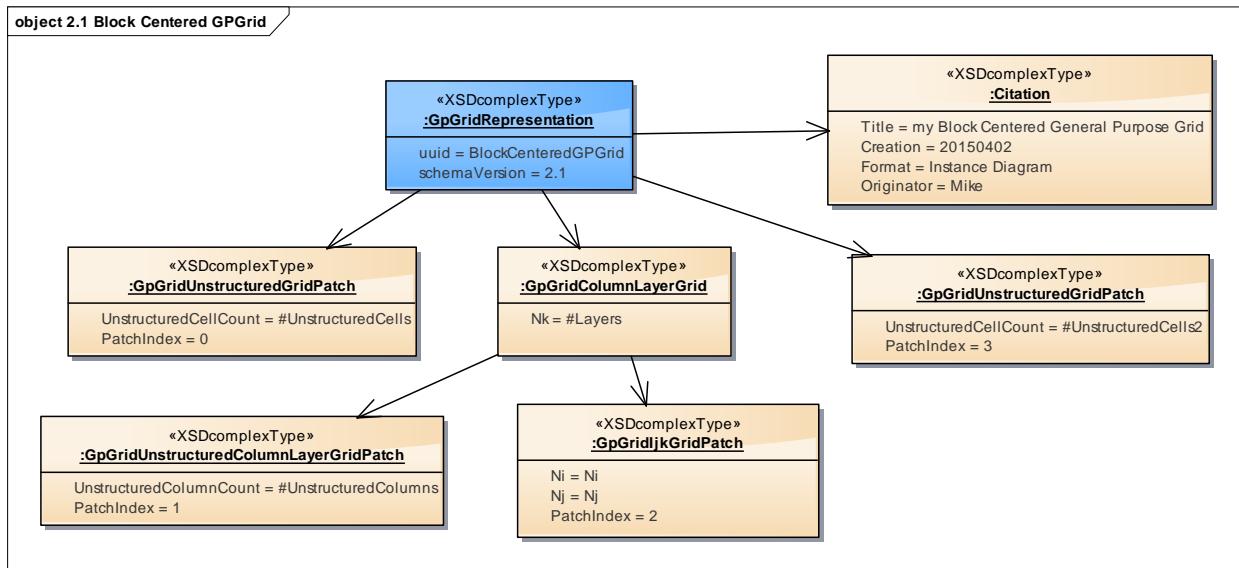


Figure 10-58. Block-centered general purpose grid instance diagram.

10.17.10 Unstructured Grid with Discontinuous Properties

This example shows an unstructured grid representation, and the use of a subrepresentation to support a cell-node property which is discontinuous from cell to cell, despite the underlying grid being unfaulted.

Figure 10-59 shows a very simple grid; **Figure 10-60**, **Figure 10-61**, **Figure 10-62** are the class diagrams and **Figure 10-64**, **Figure 10-65**, and **Figure 10-66** are the instance diagrams showing the data.

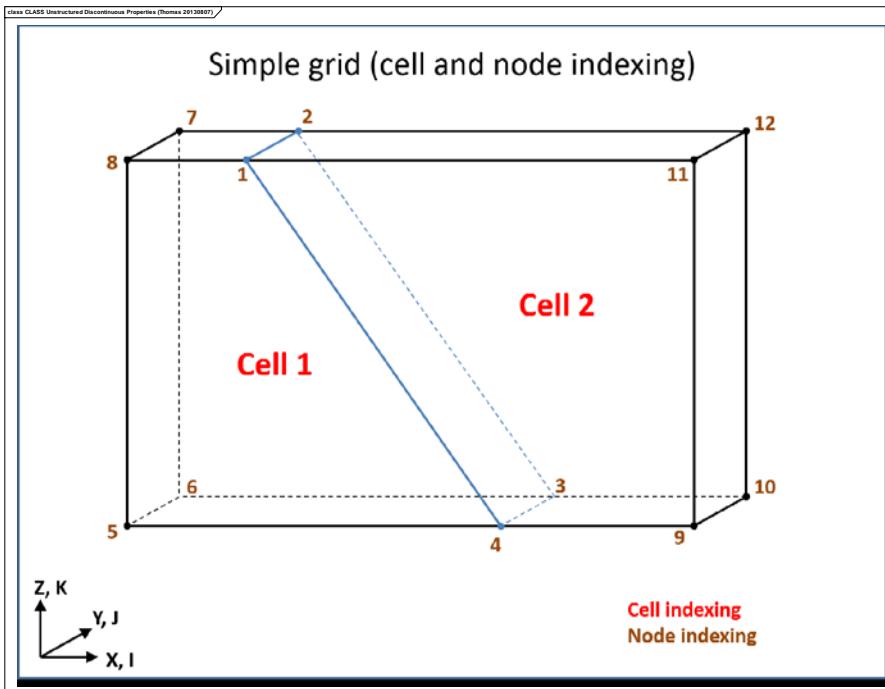


Figure 10-59. Unfaulted unstructured grid, with discontinuous properties.

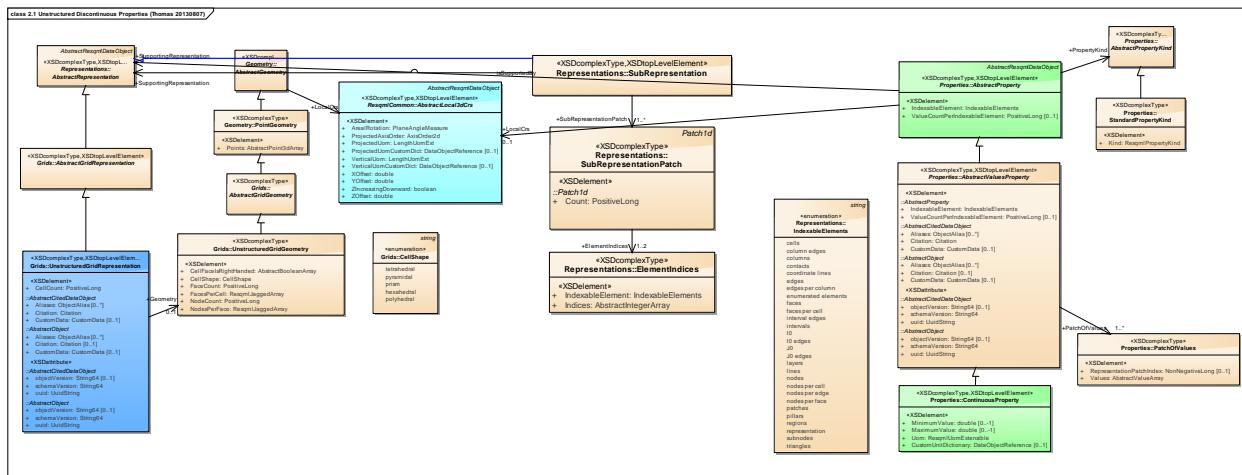


Figure 10-60. Class diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (1 of 4).

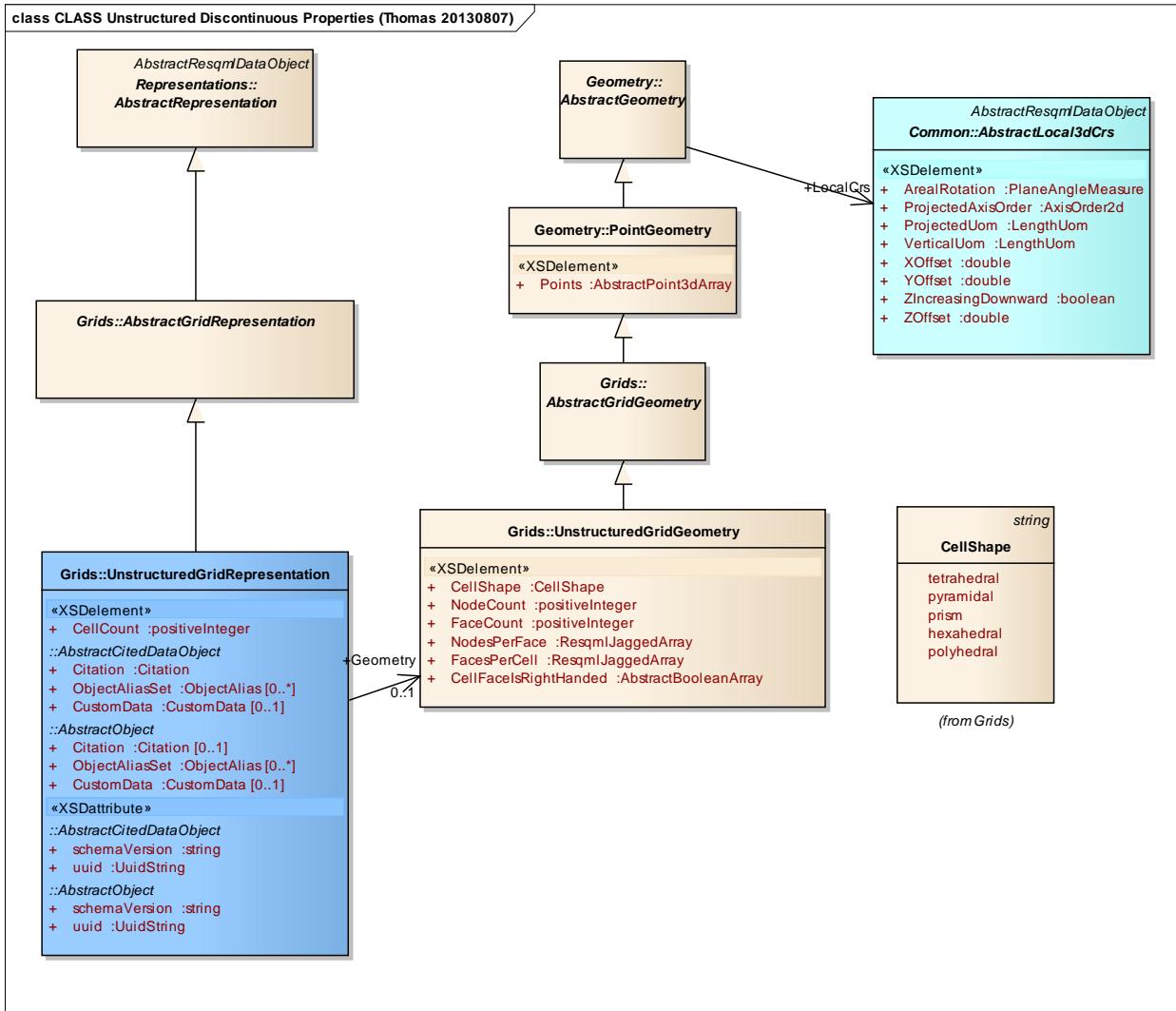


Figure 10-61. Class diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (2 of 4).

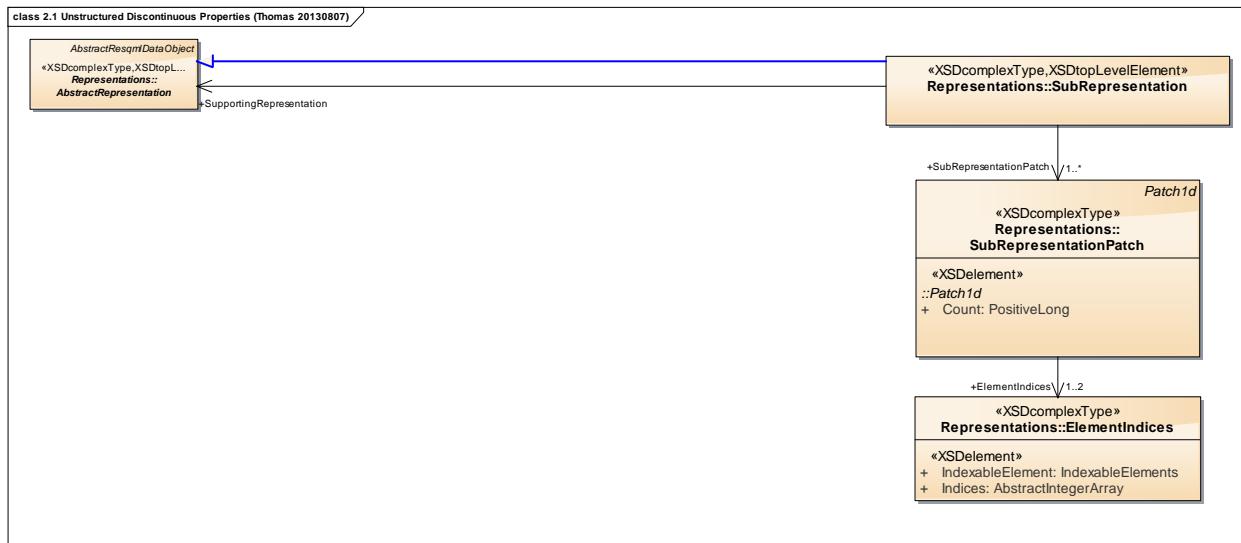


Figure 10-62. Class diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (3 of 4).

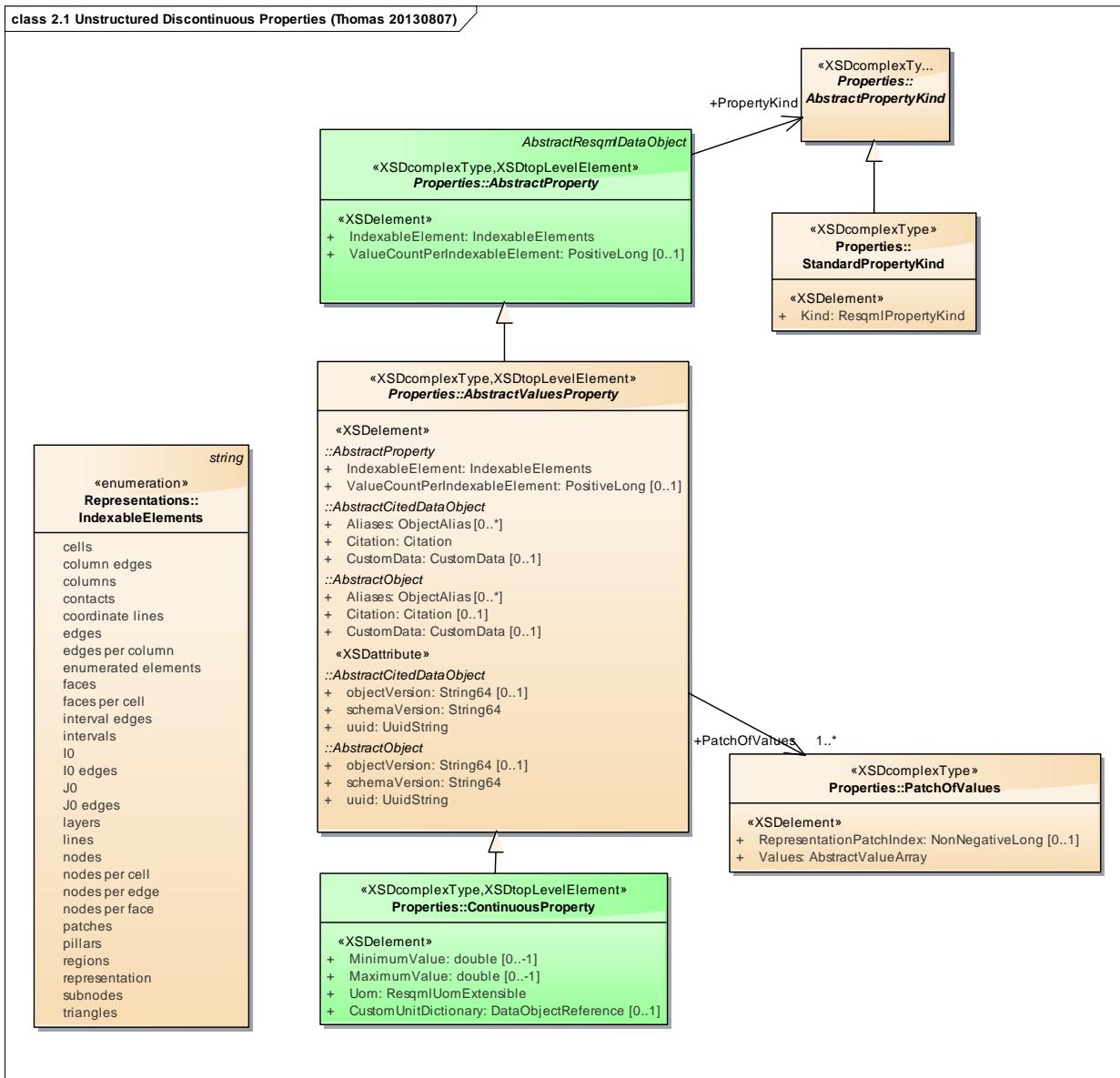


Figure 10-63. Class diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (4 of 4).

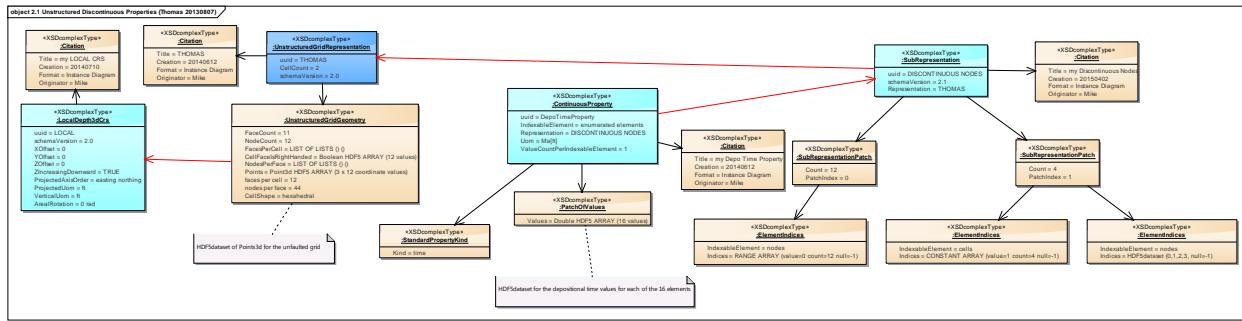


Figure 10-64. Instance diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (1 of 3).

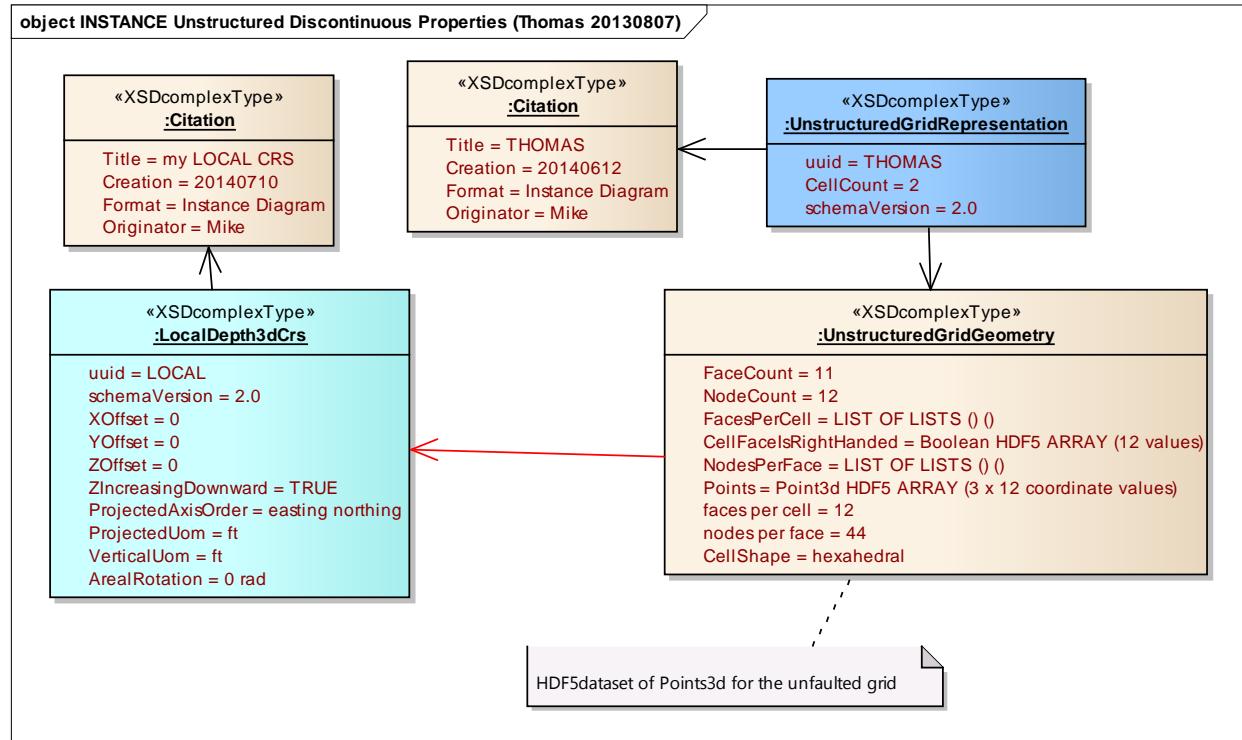


Figure 10-65. Instance diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (2 of 3).

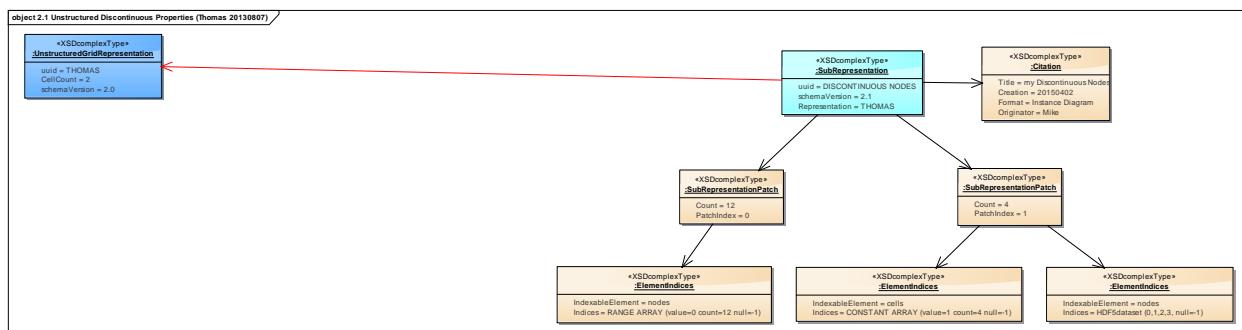


Figure 10-66. Instance diagrams of an unfaulted unstructured grid, with a subrepresentation to support discontinuous properties (3 of 3).

10.18 References

Jenny, P., Lee, S. H., & Durlofsky, L. J. (2001). Modeling Flow in Geometrically Complex Reservoirs Using Hexahedral Multi-Block Grids. Society of Petroleum Engineers. doi:

<http://dx.doi.org/10.2118/66357-MS>. Copyright 2001, Society of Petroleum Engineers. Further reproduction prohibited without permission.

Lasseter, T.J. (2004). Improving integrated interpretation accuracy and efficiency using a single consistent reservoir model from seismic to simulation. *The Leading Edge* (2004), 23(11):1118.

<http://dx.doi.org/10.1190/1.1825932>. Used with permission from SEG/standard re-use policy.

Martin, K., Macdonald, C., The Schiehallion field: applying a geobody modelling approach to piece together a complex turbidite field, presented at the DEVEX conference held in Aberdeen, Scotland, 12-13 May, 2010. Permission for reuse given by personal communication from author.

Perrin, M., Rainaud, J-F, et al. 2013. *Shared Earth Modeling: Knowledge driven solutions for building and managing subsurface 3D geological models*. p123. Paris: Editions Technip ISBN 98-207108-1002-5

Zhang, Y., King, M. J., & Datta-Gupta, A. (2011). Robust Streamline Tracing Using Intercell Fluxes in Locally Refined and Unstructured Grids. Society of Petroleum Engineers. doi:

<http://dx.doi.org/10.2118/140695-MS>. Copyright 2011, Society of Petroleum Engineers. Further reproduction prohibited without permission.

11 Streamlines

In a reservoir engineering context, streamlines are a way to visualize and represent fluid flow. They are especially useful for understanding the volumetric sweep relationships between wells or within a reservoir. They have many applications and, for example, have been used as a basis for fluid flow simulation, sweep management, well rate optimization, and infill well placement (Datta-Gupta and King 2007).

In a more general sense, streamlines are geometric lines that are everywhere tangential to a vector field, and as a consequence they never cross in space (**Figure 11-1**). Also as a consequence, streamlines are static objects. If the underlying vector field changes with time, then the streamlines must be re-drawn. This is in contrast to a “streakline,” which describes the physical trajectory of a particle in a time-varying velocity field. Streamlines and streaklines are only identical for steady velocity fields.

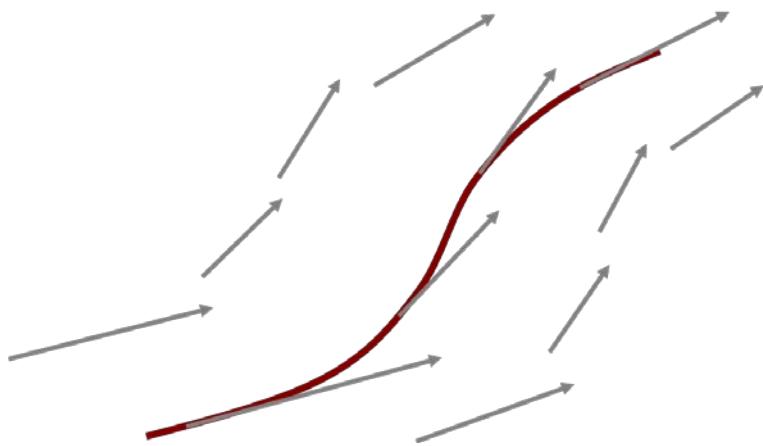


Figure 11-1. Streamlines are the geometric lines that are everywhere tangential to a vector field.

In RESQML, the streamlines data object is used to describe collections of streamlines (**Figure 11-2**).

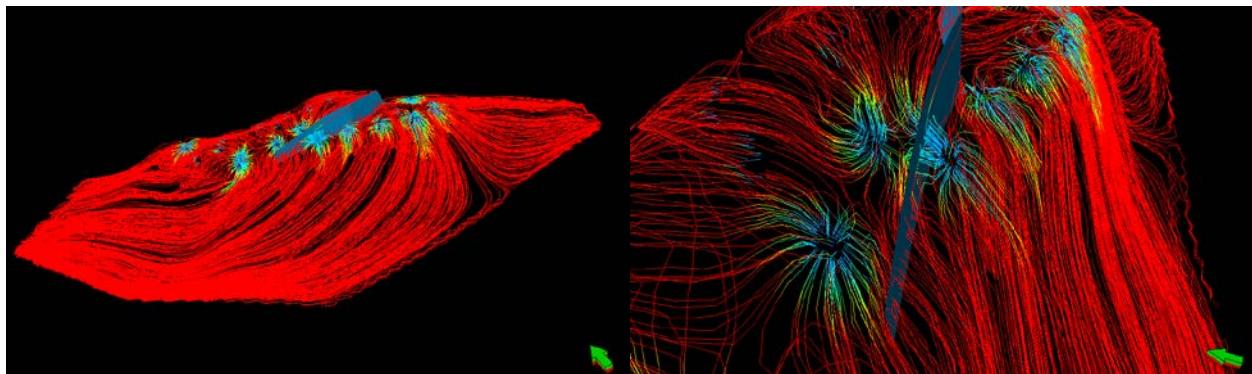


Figure 11-2. Streamlines within a 3D reservoir model (a) Full field, (b) Detailed view near a fault surface.

The streamlines feature describes the nature of the vector field, while the streamlines representation describes the geometry and topology of the lines. All streamlines representations must have both an interpretation and a streamlines feature. As with all other representations in RESQML, properties may be attached to these representations. For example, in **Figure 11-2**, the streamlines property visualized is the “time of flight to the producer,” which indicates how quickly fluids drain from the reservoir. Although the streamlines appear to cross in the two dimensional field of view, they do not cross in three dimensions.

11.1 Streamlines and the RESQML Knowledge Hierarchy

The RESQML description of streamlines is based upon a streamlines feature, the generic feature interpretation, and a streamlines representation. They are related through the abstract feature and the abstract feature interpretation (Figure 11-3). Streamlines are an example of a technical feature because they do not carry a geologic interpretation. However, information about geologic interpretations may be inferred from the underlying computational grid, as described below.

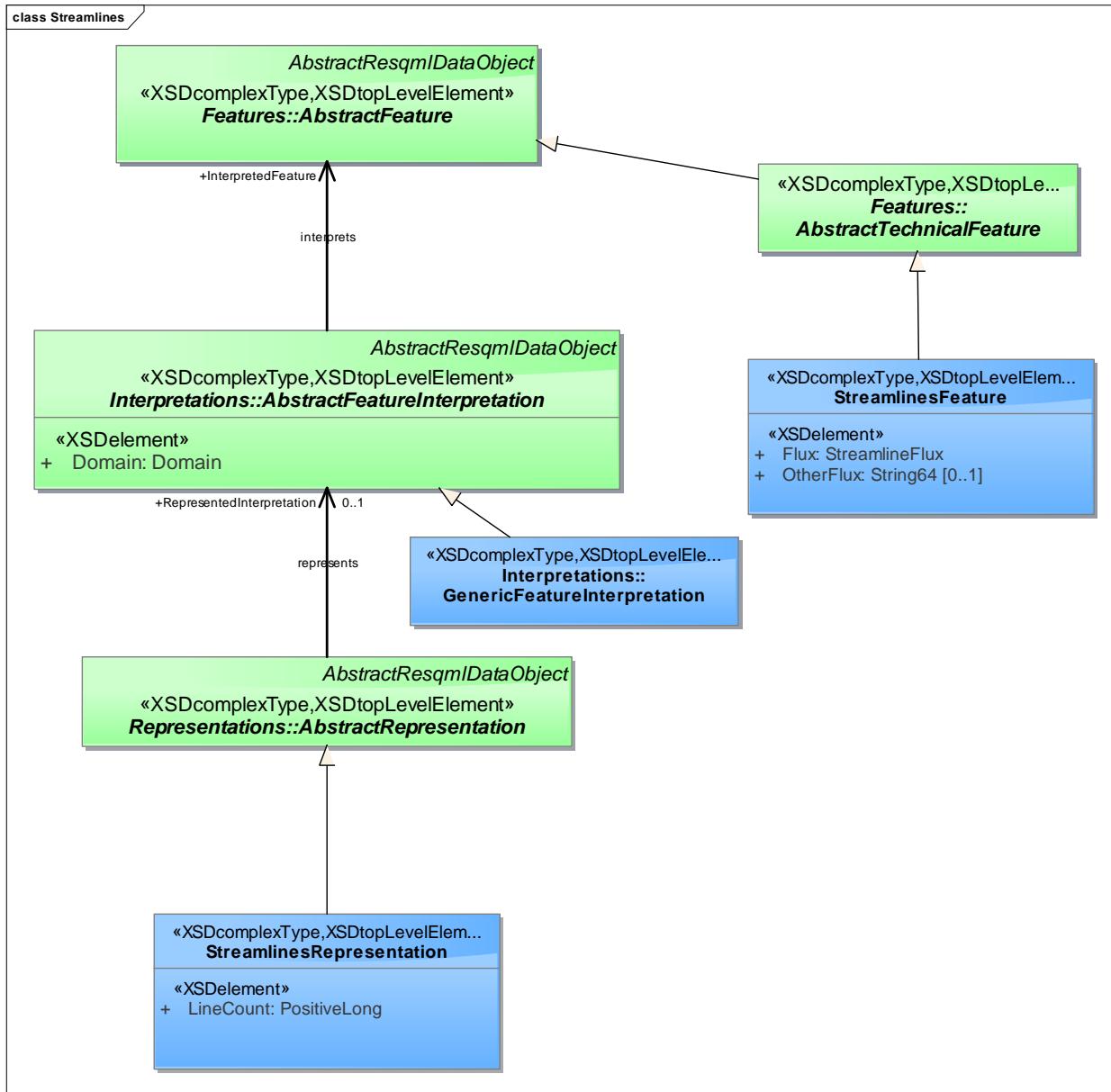


Figure 11-3. Streamlines feature, interpretation, and representation.

Although RESQML allows the relationship between representation and feature interpretation to be optional, for the streamlines representation, all data sets must include a generic feature interpretation and must include the streamlines feature.

11.2 Streamlines: Feature

The streamlines feature is used to describe the underlying vector field from which the streamlines are constructed (**Figure 11-4**). In a finite difference calculation, the vector field is projected and integrated onto the cross-sectional area of the grid cell faces to obtain a flux on that face. The streamlines feature describes that flux. In a streamline tracing algorithm, the vector field is reconstructed from the flux and interpolated within the cells of the grid. In most instances, the resulting trajectory geometry may then be obtained by analytic integration within each cell.

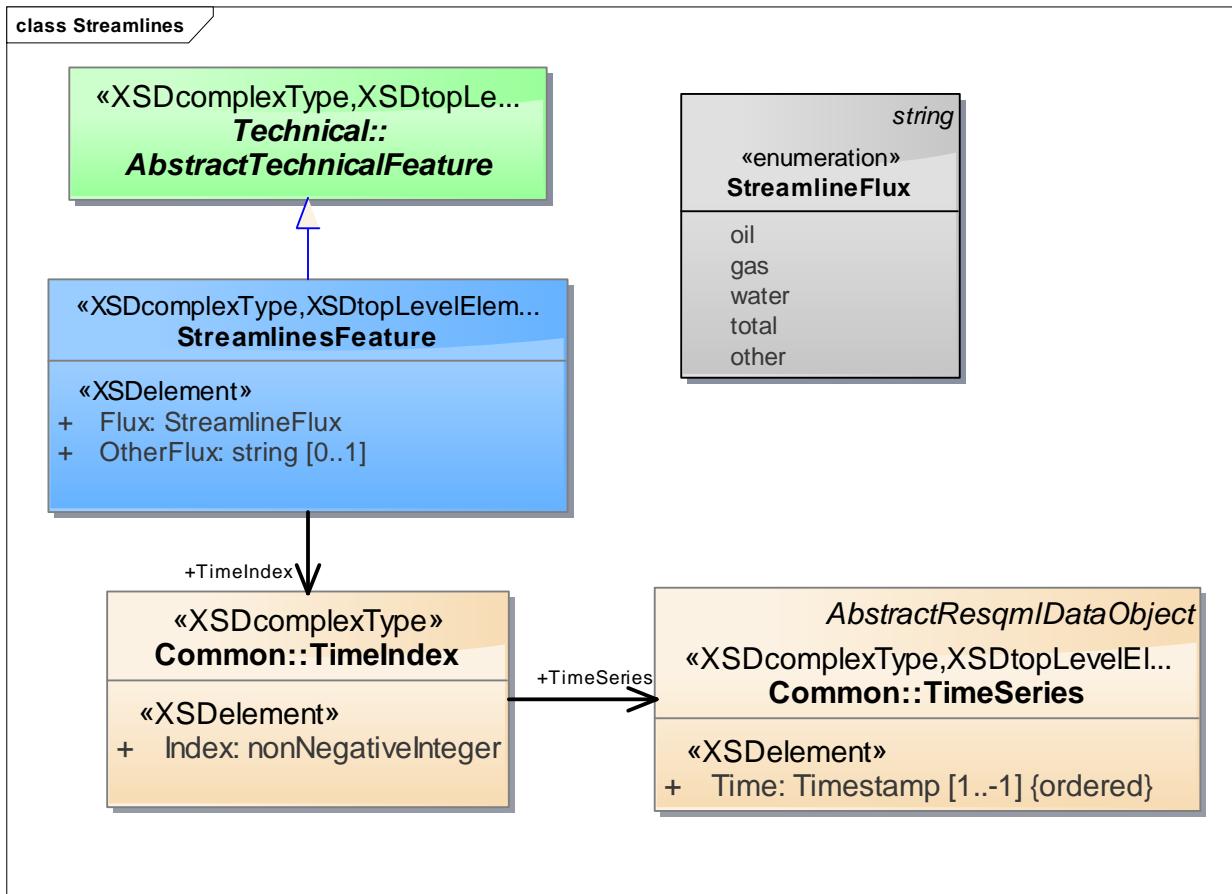


Figure 11-4. Streamlines feature.

11.2.1 Flux

The choice of flux type can be quite general. It is most common to sum the volumetric flux for all phases (streamline flux = total). However, we may also choose to generate streamlines for individual phases (streamline flux = oil, gas, or water) to visualize the flow of each phase. We may also choose to visualize the flow of individual components in a compositional simulation, or other fluxes, for instance, thermal. In these cases, the enumerated streamline flux should have the value of “other” and the optional “OtherFlux” element should be used to describe the flux. Other flux must only be used when the streamline flux is other. As the flux is time varying in reservoir simulation, the time index into a time series is mandatory. It is used to specify the time at which the flux is calculated and for which the streamlines are traced.

11.3 Streamlines: Interpretation

Streamlines use the RESQML feature/interpretation/representation knowledge hierarchy (**Figure 11-3**). We specifically reserve the use of “interpretation” to classify the methodology used in the tracing of streamlines. We have already described how streamlines are based on the flux through cell faces.

However, once we move beyond simple rectilinear cells in unfaulted reservoir models, there are differing algorithms on how the velocity should best be interpolated within the cells, and whether the streamline trajectories that follow the velocities should be computed analytically or numerically. Additional algorithmic choices also arise in the treatment of non-neighbor connection fluxes at faults within a reservoir model.

A single feature-interpretation may have multiple representations. These representations may, for example, differ in the density of streamlines, or the spatial resolution of nodes along a line. A common example of a change of density is where we may choose to construct the streamlines for a particular producer, e.g., locally at high density, and then again create a full field view of the streamlines in the reservoir with fairly uniform density. Although it is common to store node locations where streamlines enter and leave cells, denser node representations may also be used to delineate the shape of the lines within each cell. A shared feature indicates that there is a common underlying velocity field while a shared interpretation indicates that the methodology to interpolate velocity within cells and the methodology used to trace the corresponding streamlines are shared.

11.4 Streamlines: Representation

The streamlines representation is constructed from a number of optional objects which together support a variety of workflows (**Figure 11-5**).

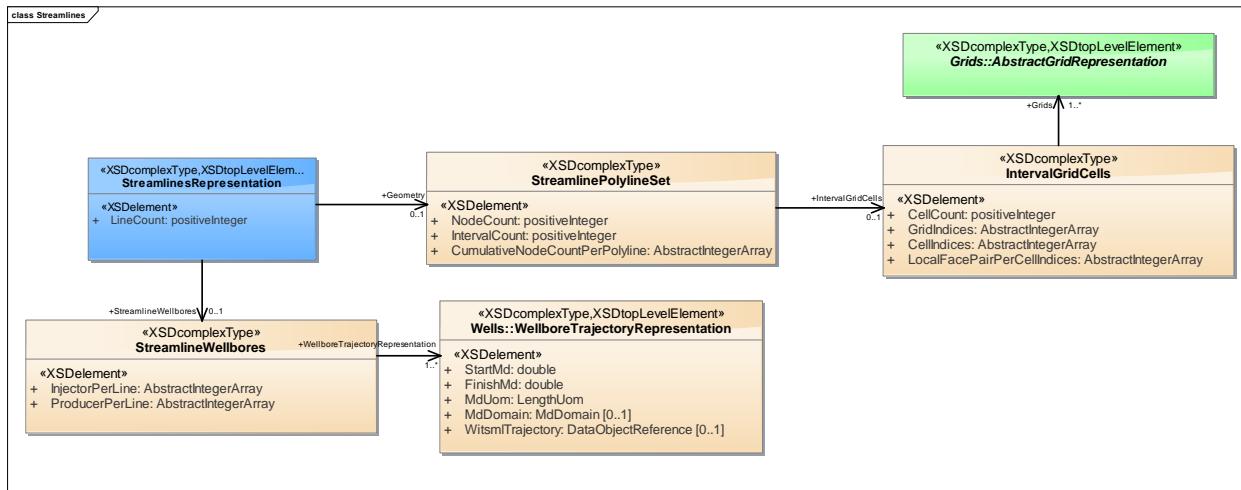


Figure 11-5. Streamlines representation.

11.4.1 Lines

The most basic representation has only a single mandatory element, which is simply a count of the number of streamlines. Streamlines use “lines” as their indexable element. The line index may be used to represent properties whose values do not vary along a streamline. For example, in **Figure 11-6**, each streamline takes on a different color based on the producer. These streamline bundles show the drainage volumes of each producer. Other examples of properties represented in this way are the flux injected or produced per streamline or the total time of flight between an injector and a producer.

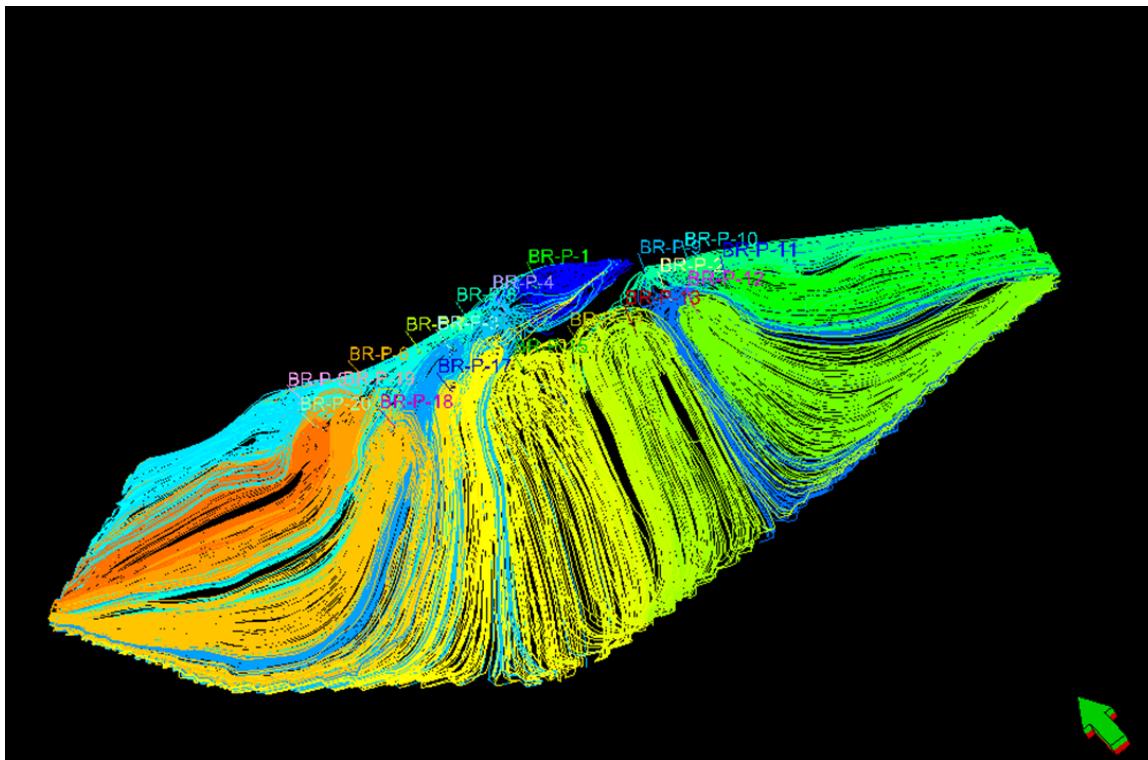


Figure 11-6. Streamline visualization of the drainage volumes of the producing wells in a reservoir.

11.4.2 Streamline Sources and Sinks

The streamline wellbores object is used to reference the injectors and producers. Streamlines may originate at an injector, along the boundary of a model, or within the reservoir due to volume expansion. The injector per line element indicates the injector for each streamline, or takes on the null value. Similarly, the producer per line element indicates whether streamlines terminate at a producer, along the boundary of a model or within the reservoir near a stagnation area. 0-based wellbore indices are defined based upon the order in which a reference to the wellbore trajectory representation appears.

11.4.3 Streamline Geometry

The geometry of the streamlines is described using a polyline set. The indexable element type “nodes” is used to attach point geometry to the nodes of the polylines, with an explicit total node count. The polyline set usually consists of multiple polylines, and the node count per polyline array is used to differentiate each polyline within the set. The node indexable elements are used to store properties. As an example, the time of flight to producer displayed in **Figure 11-2**.

The indexable element type “intervals” is used to attach properties or otherwise reference the intervals within the polyline set. If N is the number of nodes on a polyline ($N > 1$), then that line has $N - 1$ intervals. The interval count in the polyline set is the total sum of intervals summed over the polylines.

11.4.4 Polyline Interval Grid Cells

The polyline set is used to describe the geometry of lines in space. For streamlines specifically, these lines have been developed based on an underlying computational grid. The relationship from line to grid is described using the interval grid cells object. The grid indices element is the most important, because it indicates which grid, if any, corresponds to each interval of the streamline polyline set. Nulls are used to indicate intervals that do not lie on any grid. Non-null grid values indicate that that interval is a cell.

The indexable element type “cells” is used to attach properties or otherwise reference those intervals with non-null grid indices. The cell count is specified explicitly and must be equal to the total count of intervals

with non-null grid indices. The remaining elements (cell indices array and local face pair per cell indices array) are used to further specify the spatial support of the streamlines on the cells of the grid(s).

11.5 References

Datta-Gupta, Akhil and King, Michael J. (2007). *Streamline Simulation: Theory and Practice*. SPE Textbook Series, Vol. 11.

Klausen, R.A., Rasmussen, A.F. & Stephansen, A.F. Comput Geosci (2012). Velocity interpolation and streamline tracing on irregular geometries 16: 261. doi:10.1007/s10596-011-9256-0

Zhang, Y., M.J. King, and A. Datta-Gupta (2012), Robust streamline tracing using inter-cell fluxes in locally refined and unstructured grids, Water Resources Research, 48, W06521, doi:10.1029/2011WR011396.

12 Wells

One of the long-term goals for RESQML is to leverage other Energistics standards, such as WITSML and PRODML, to link the well-related information with the reservoir. Beginning with v2.0 of WITSML and PRODML, all of the Energistics MLs use the same conventions (UUID, etc.) for data object identification, making it easier to reference objects across MLs.

Unlike WITSML, which is used to represent actual drilled wells, RESQML must be able to represent wellbores and associated information during the planning phase, before any administrative information (such as legal names, precise locations, etc.) about the well is available.

To fill these gaps, RESQML has adopted its own well-related data objects. These data objects have enough information to meet RESQML needs and can also be used as proxies for existing WITSML data objects.

These data objects are also compatible with the “What is a Well” approach as defined by PPDM (<https://www.ppdm.org/ppdm-standards/what-is-a-well-definitions>).

12.1 Special Requirements for Using WITSML 1.4.1 Data Objects with RESQML (in an EPC file)

Because it was developed several years earlier, the current most-used version of WITSML, 1.4.1.1, has different requirements for identifying and storing data objects. (For rules on WITSML data object identification, see Section 2.2 of the *WITSML STORE Application Programming Interface* specification.) Therefore, the following additional rules must be observed when using WITSML data objects in an Energistics package:

- Each file created in WITSML must contain only one business object, e.g., a single well, wellbore, log, etc. This rule is so that these individual business objects can each be correctly referenced in an Energistics package.
- WITSML UIDs are mandatory for each data object. In an Energistics package, the WITSML UID serves the role of a UUID as specified in the *Energistics Packaging Conventions Specification* (for a link to this document, see Section 1.3.2 (page 13)). Currently use of UUIDs in WITSML is recommended but not required. For Energistics packages, use of UUIDs is strongly recommended.

12.2 Wellbore Organization Overview

In WITSML, data objects are organized into a single hierarchy, with the well data object at the top. The well data object is the parent of one or more wellbore data objects (child data object); a wellbore data object can have one or more children, such as logs and trajectories.

In RESQML, the wellbore is the core data object. Because the well data object information is mainly administrative, no reservoir-specific requirements are needed. Thus, for this administrative information, RESQML relies on the WITSML link between a wellbore and its well.

Figure 12-1 shows an overview of the wellbore organization, which is based on the RESQML knowledge hierarchy (see Chapter 4 (page 39). The sections below the figure explain it.

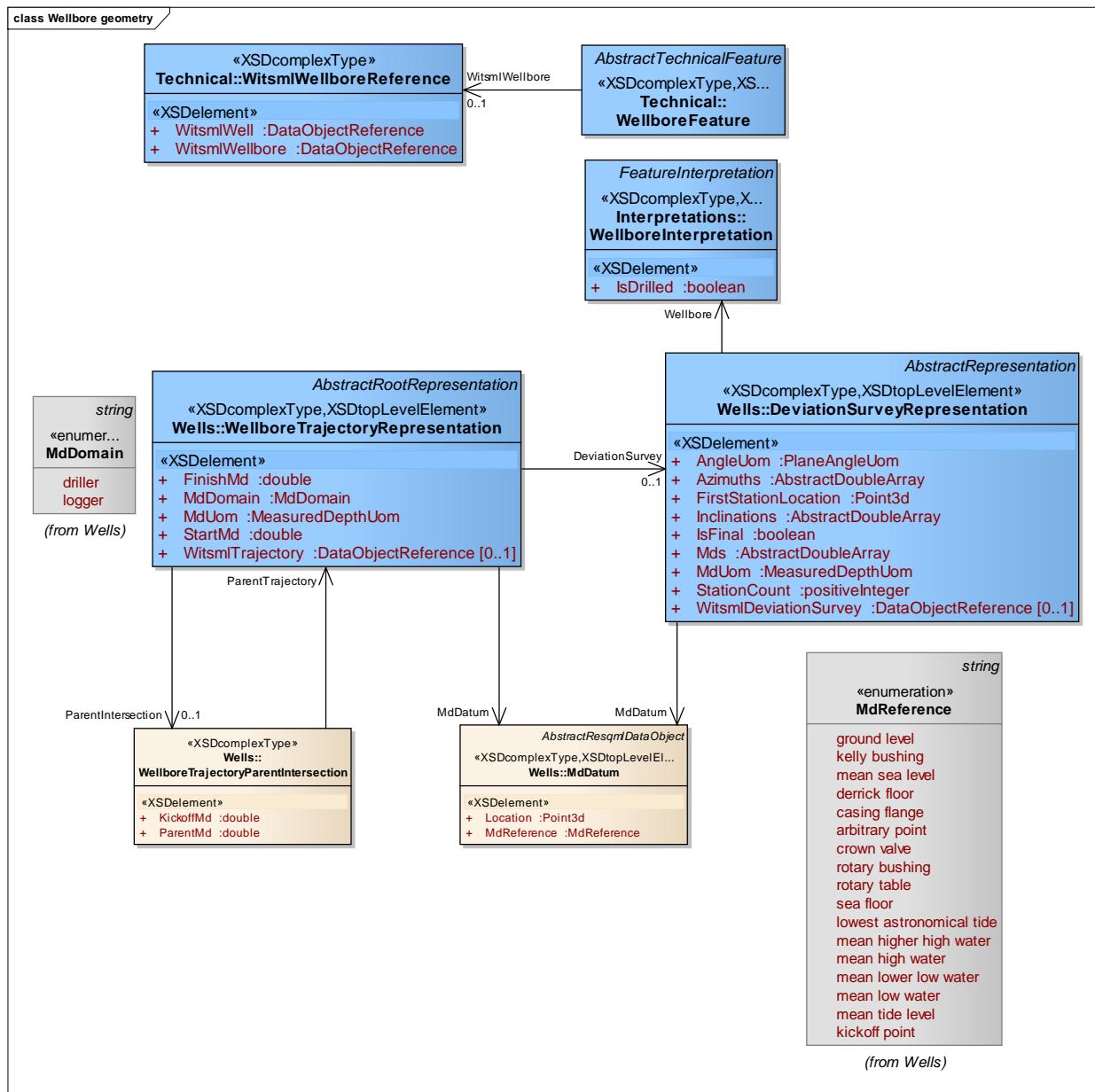


Figure 12-1. Main wellbore data object organization based on the RESQML "FIRP" knowledge hierarchy.

12.2.1 Wellbore Feature

Because the location of a wellbore can continually evolve between planning and drilling, it is important to be able to link all of these evolutions. RESQML provides the ability to do this using a wellbore technical feature. The wellbore technical feature provides a consistent identification of a wellbore through all of its different phases and provides the reference to a potential WITSML representation of the same wellbore.

12.2.2 Wellbore Interpretation

A wellbore interpretation is an opinion about a wellbore trajectory. It typically represents an actual drilled wellbore or any version of a planned wellbore trajectory. This “drilled” information is part of the data object.

The wellbore interpretation is used as an umbrella for all the representations of a specific trajectory, such as:

- Different types of representations: directional survey versus point discretization
- Different sampling used during logging or marker interpretation
- Different computation mode or refinement

12.2.3 Wellbore Location

A wellbore location is defined using a wellbore trajectory representation and MD datum, which are explained in this section.

12.2.3.1 Wellbore Trajectory Representation

The wellbore trajectory representation has these main uses:

- **Defines the geometry of the trajectory inside a RESQML coordinate system**, so that geometrical interactions between its wellbore and other elements of a model are unambiguous. A wellbore trajectory representation can be associated with a WITSMIL trajectory, but it also must be able to stand by itself, so that it can represent wellbores during planning. The wellbore trajectory representation object includes sufficient information to uniquely define the location of a vertical well. However, all other wells should contain geometry described as a parametric line. (For more information, see Section 6.3 (page 59).)
- **Serves as the reference for all the measured depth information captured along the wellbore**. It references one MD datum (explained below), which provides information about the location of the MD datum and its type.
- **Indicates the unit of measure and the range of measured depths covered by the trajectory**. In the simple case, the range goes from zero to total depth. But for multi-lateral wells, the range may begin at kickoff points. A driller or logger domain is also included to indicate which measured depth reference was used.
- If a trajectory is based on a deviation survey, it can indicate the survey that it originated from.

For multi-lateral wells, wellbores can have additional parentage information to indicate:

- Which wellbore trajectory it is kicked off from.
- The kickoff measured depth, both in the current wellbore trajectory and its parent wellbore trajectory. Each value is expressed in the measured depth unit of measure of its trajectory.

12.2.3.2 MD Datum

Because the RESQML data model does not extend beyond the wellbore (see Section 12.1, the MD datum is a mandatory component of the wellbore. It includes an MD reference enumeration that is identical to that of WITSMIL, with one exception: it also has an “arbitrary point” to support reservoir modeling workflows for well planning and field development.

The MD datum is a top-level data object so that it may be shared among multiple wellbores (e.g., in multi-lateral wells) or several versions of trajectories. The MD datum exists independently of the wellbore trajectory representation and needs to have a uniquely defined local 3D coordinate reference system (CRS), though the datum and the trajectory must refer to the same local coordinate system.

In practice, the local 3D CRS used by the MD datum is often the composite of a projected CRS and a vertical CRS, without any translation or rotation. The local 3D CRS is also a top-level data object, which means that it can be shared between the MD datum and the geometry of the wellbore trajectory representation, although there is no requirement that they do so. For example, the CRS of the MD datum may be shared among multiple wellbores from the same wellhead, while the CRS of the wellbore trajectory representation may be chosen to be the same as that of a reservoir modeling grid, to facilitate intersection calculations.

12.2.3.3 Deviation Survey

Before a trajectory can be computed in a model coordinate system, it is usually represented by a deviation survey. It is useful to transfer this survey when a trajectory is not yet computed or for quality control.

The survey references MD datum pointing to a measured depth reference, indicates the location of the first survey station in the model, and then, for each station, specifies a measured depth, azimuth, and inclination. To distinguish it from other intermediate versions of survey, an "is final" tag is used to indicate if this is the currently accepted version of the survey. The deviation survey can be associated with a WITSML survey.

12.3 Well Logs

From a RESQML perspective, well log curves are simply property values assigned on a sampling (a list of measured depths) defined along the trajectory (**Figure 12-2**). The list of measured depths is held in the wellbore frame representation, which corresponds to a WITSML well log.

The wellbore frame represents the wellbore interpretation, references the trajectory on which it is based, and defines a sampling where each node has a given measured depth. The sampling can be defined using either of these methods:

- Explicit list of values, which indicates aperiodic sampling.
- Lattice of values, which indicates periodic sampling.

The property values can be attached to either the nodes or the intervals of the sampling. For more information about properties, see Chapter 7.

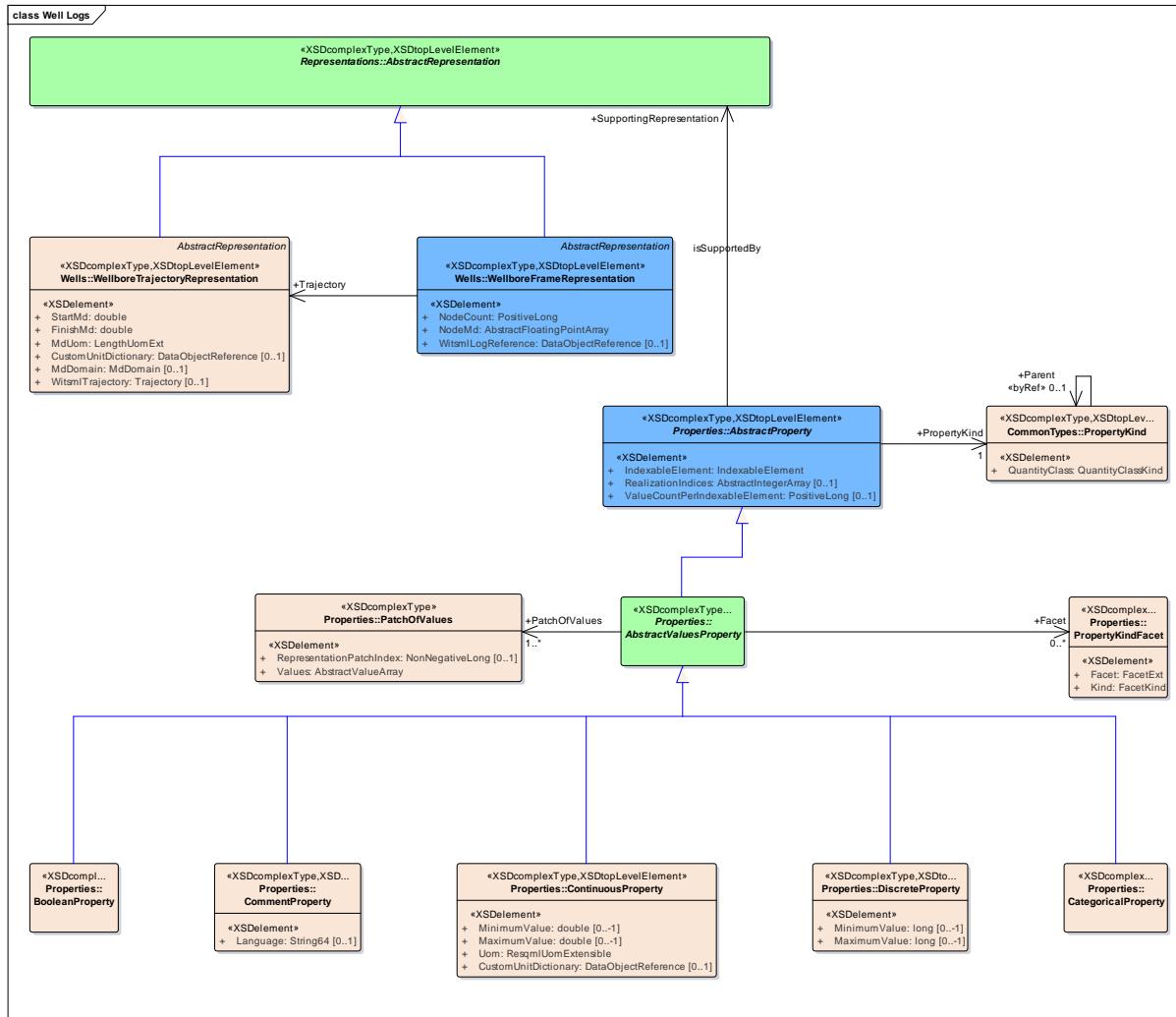


Figure 12-2. RESQML treats well log curves as properties, which are attached to a wellbore trajectory using a wellbore frame.

12.4 Markers and Links to Geological Boundaries

A specialized wellbore marker frame is used to carry marker interpretations along the well (Figure 12-3, blue boxes). Each measured depth node can be associated with a marker. Each marker has full traceability through a citation element. Markers can also be associated with individual geological boundary elements through optional references to a geologic, fluid, or contact feature. When a wellbore marker defines a fluid contact, fluid marker, or geologic boundary, it is highly recommended that you provide a related boundary feature interpretation. If you don't have a real boundary feature interpretation, then just create an empty one to house the UUID.

Markers can also be associated with a WITSML formation marker.

12.5 Interpretations and Links to Organizations

To provide more detailed interpretation information than just a marker list, it is possible to associate a stratigraphic and fluid organization interpretation to an entire marker frame (**Figure 12-3**). Interval stratigraphic units and cell fluid phase units are used to indicate which stratigraphic or fluid organization is represented and which element of the organization interpretation is associated with each interval of the wellbore frame. When an interval is not represented in the organization, -1 is used. For more information about organizations in RESQML, see Chapter 9.3 (page 127).

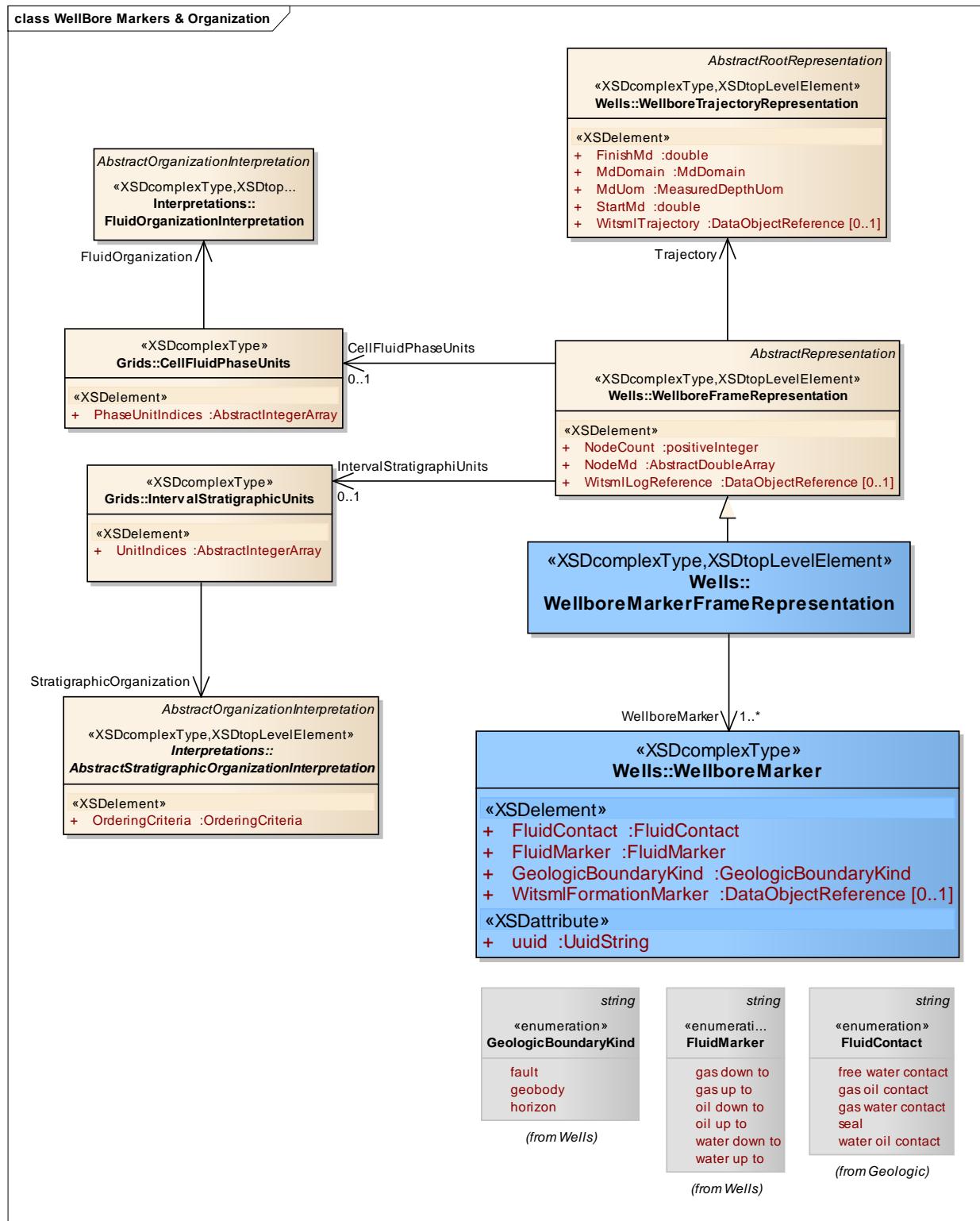


Figure 12-3. A specialized wellbore marker frame representation (blue box) is used to carry marker interpretations along a well. It is also used to link to various types of geologic boundaries (gray boxes) or RESQML stratigraphic and fluid organizations (tan boxes).

12.6 Blocked Wells

Reservoir simulation often requires discretization of a well based on one or several reservoir grids. The blocked wellbore representation (**Figure 12-4**) is the intersection of the grids and a trajectory. It is a well marker frame where each node typically represents a limit between grid cells. It has these characteristics:

- Contains the list of grids on which it has been discretized and the number of grid cells that are intersected.
- For each frame interval corresponding to a grid cell, the corresponding grid index in the list is stored.
- For the intervals between nodes that may not correspond to actual grid cells—for example if the well is outside of any grid or grid geometries have not yet been defined—a value of -1 is used.
- For each actual grid cell, the blocked well contains the cell index and the cell face indices where the trajectory enters and exits. So each cell has a pair of local face indices, and when the trajectory begins or ends within the cell, the entry or exit value is -1.

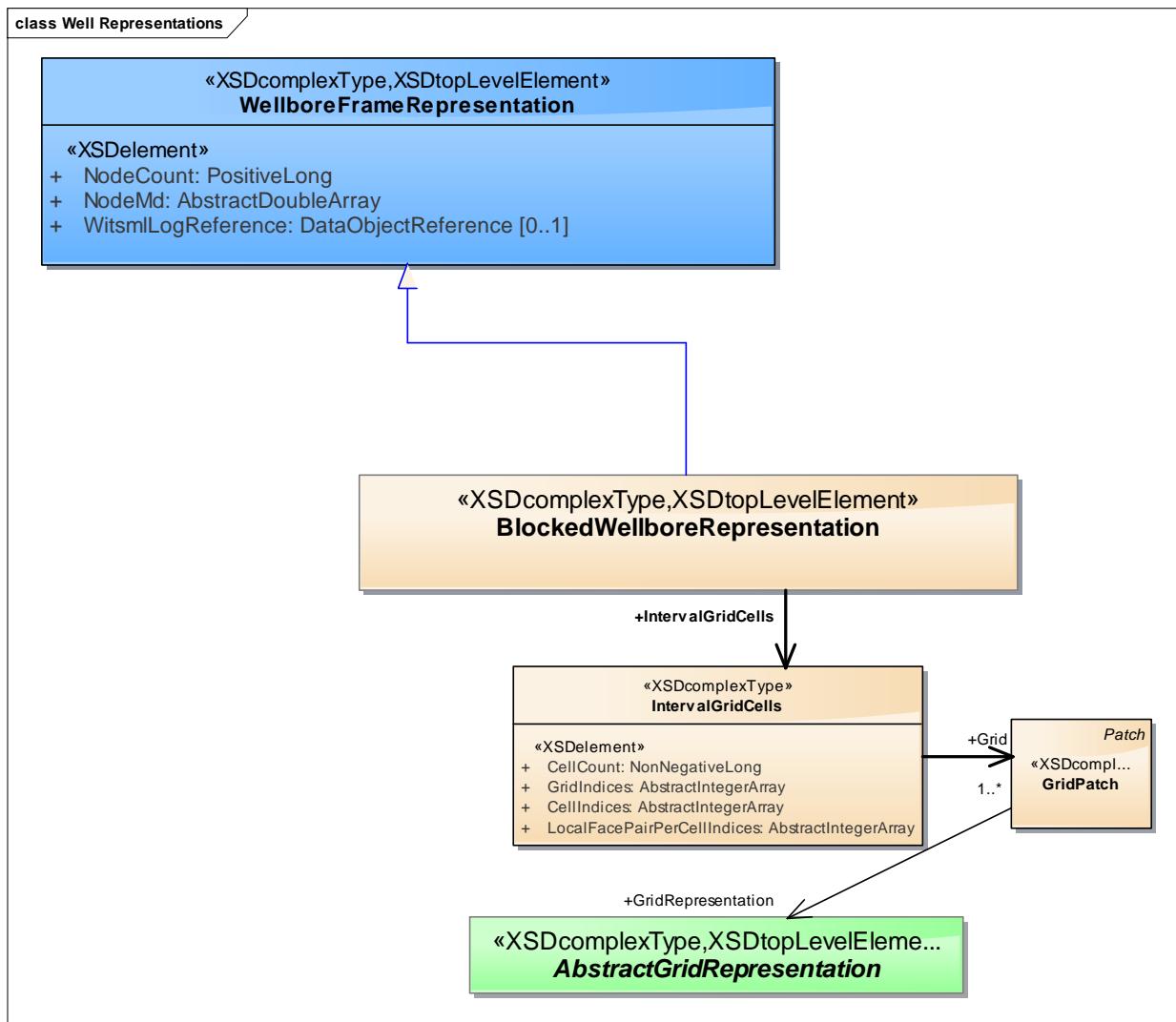


Figure 12-4. The blocked wellbore representation is the intersection of the grids and a trajectory. It is a well marker frame where each node typically represents a limit between grid cells.

12.7 Example: Two Vertical Wells

To show how RESQML works, here's a simple example of two vertical wells:

- **Figure 12-5** shows the well class diagram, which is expanded into four parts (so you can read it) in **Figure 12-6** through **Figure 12-9**.
- **Figure 12-10** (page 232) shows the instance diagram of this two-well example, which is expanded into the two parts of **Figure 12-11** and **Figure 12-12**.

12.7.1 Class Diagram

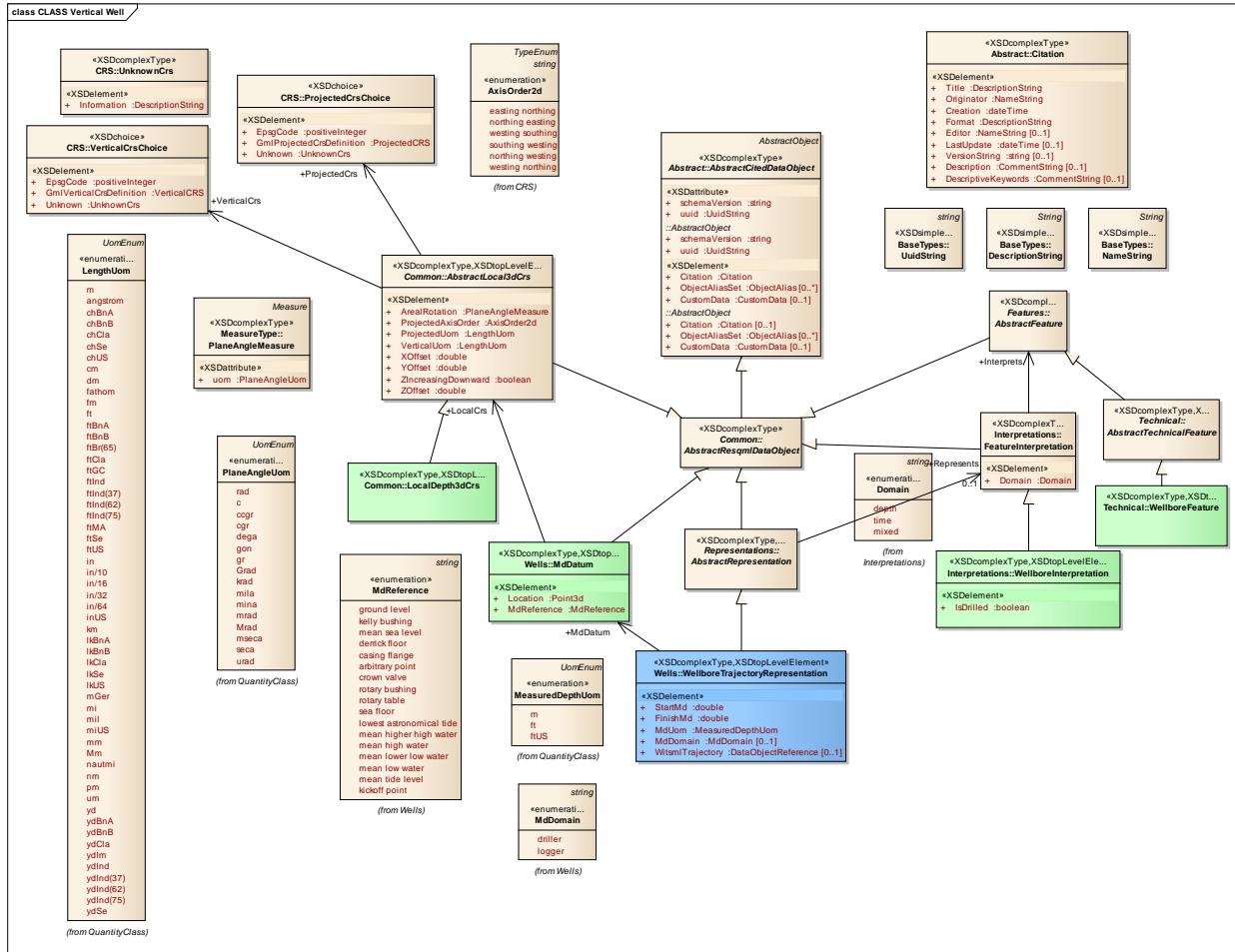


Figure 12-5. Overview of the class diagram for a wellbore trajectory representation. Portions of this figure are expanded in Figure 13-6 through Figure 13-9.

The single major mandatory component of the wellbore trajectory representation is the MD datum (**Figure 12-7**), which corresponds to the wellhead. The MD datum exists independently of the wellbore trajectory representation and needs to have a uniquely defined local 3D coordinate reference system (CRS) (**Figure 12-6**).

The vertical well geometry (highlighted cells and **Figure 12-8**) is known from the location of the MD Datum and the measured depth range (start and finish MD values) of the wellbore trajectory representation. In this specific case, there is no need to specify a parametric line to describe its geometry.

Finally, the use of wellbore interpretations and features (**Figure 12-9**) allows us to document relationships between wellbore trajectory representations. For example, multiple representations with a shared

interpretation may arise, if we have three ways of representing nominally the same geometry: a minimum-curvature trajectory, a piecewise linear trajectory at a low resolution corresponding to the intersections with a coarse reservoir simulation grid, and a piecewise linear trajectory at high resolution corresponding to the intersections with a detailed 3D geologic model. Wellbores that share the same feature but different interpretations may correspond to a single nominal target well and the many well plans for that well.

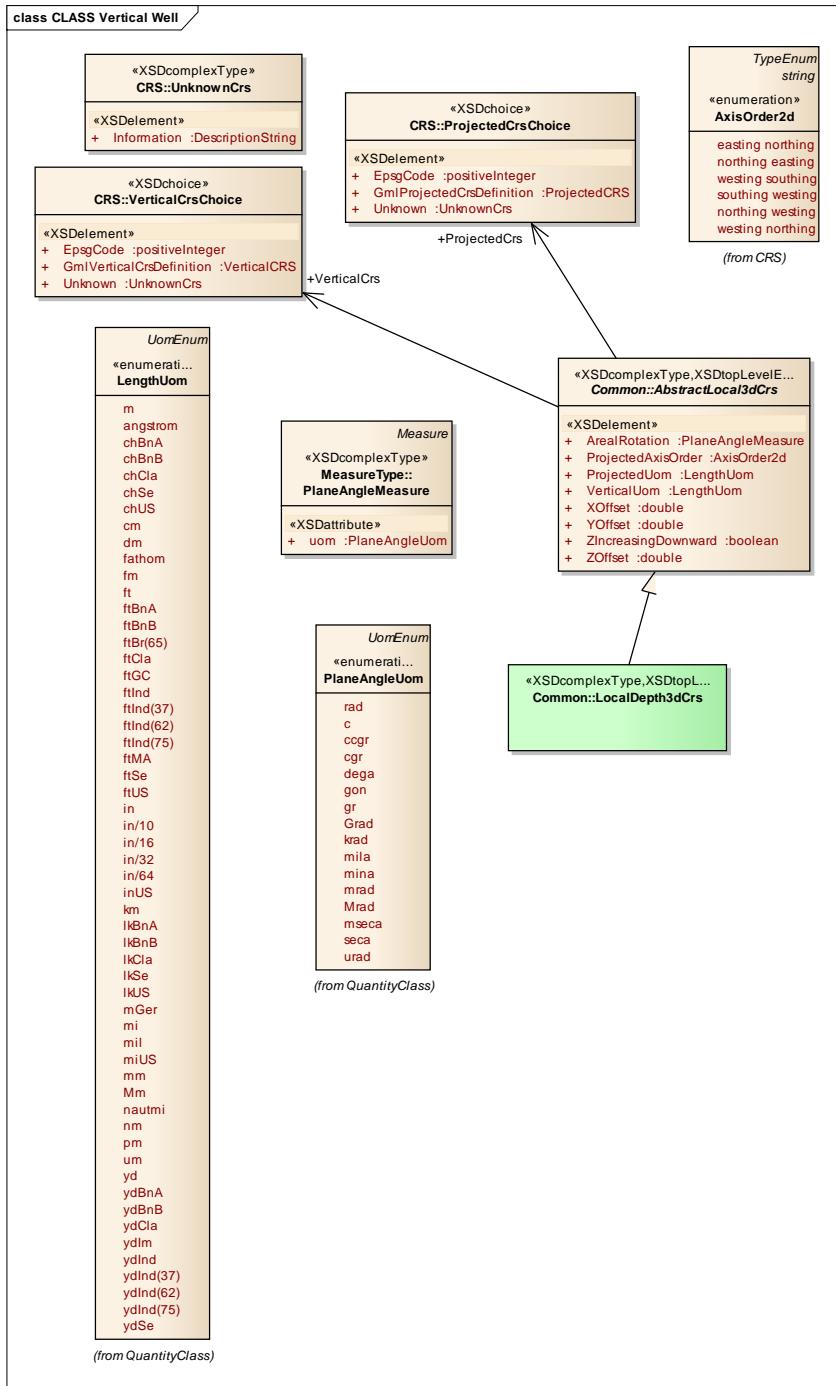


Figure 12-6. (Part 1 of 4) Components of the class diagram for a wellbore trajectory representation, including the description of the local depth 3D CRS, which is used by the MD datum object.

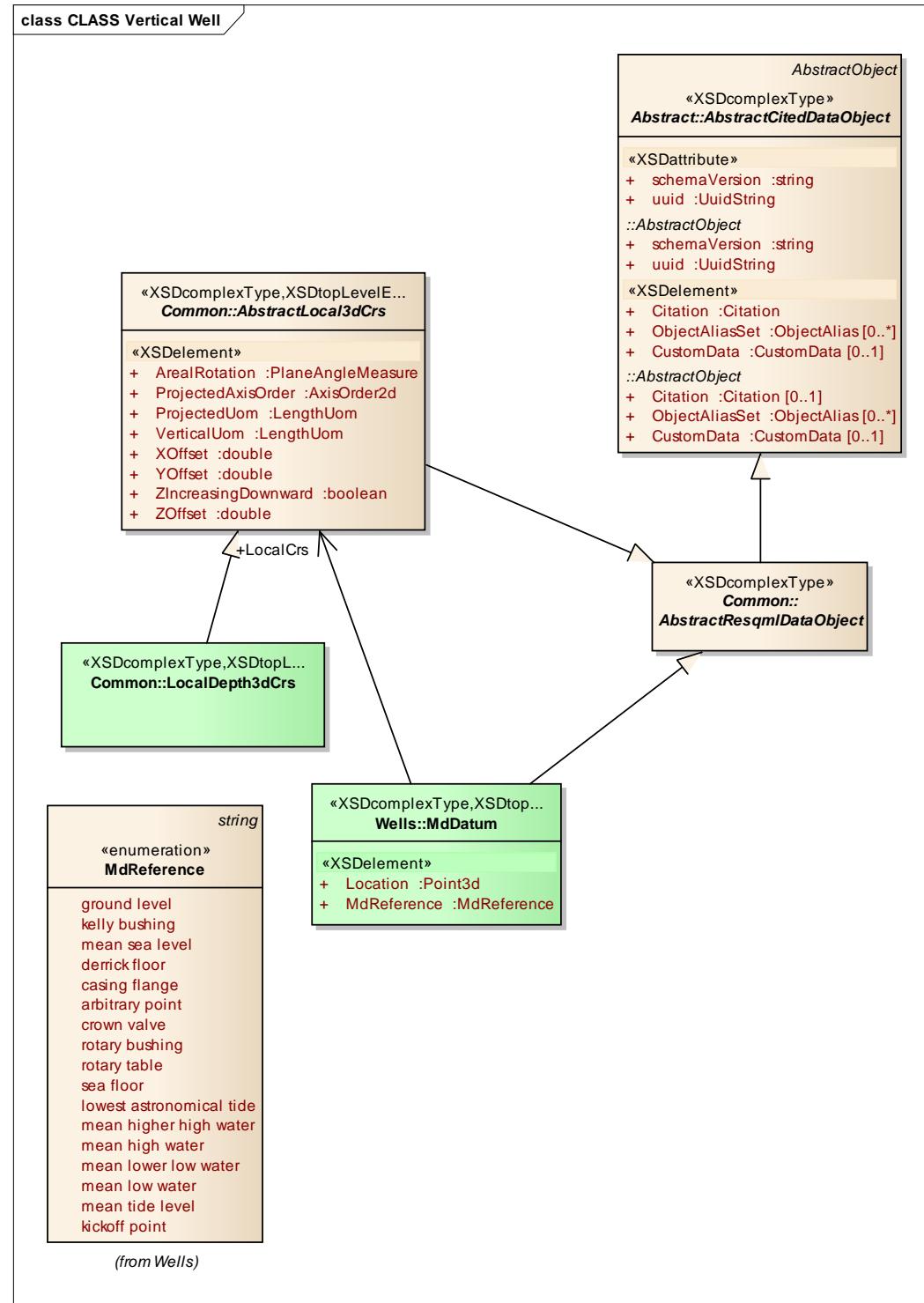


Figure 12-7. (Part 2 of 4) Components of the class diagram for a wellbore trajectory representation: Description of the MD datum data object, which is a mandatory component of the wellbore trajectory representation.

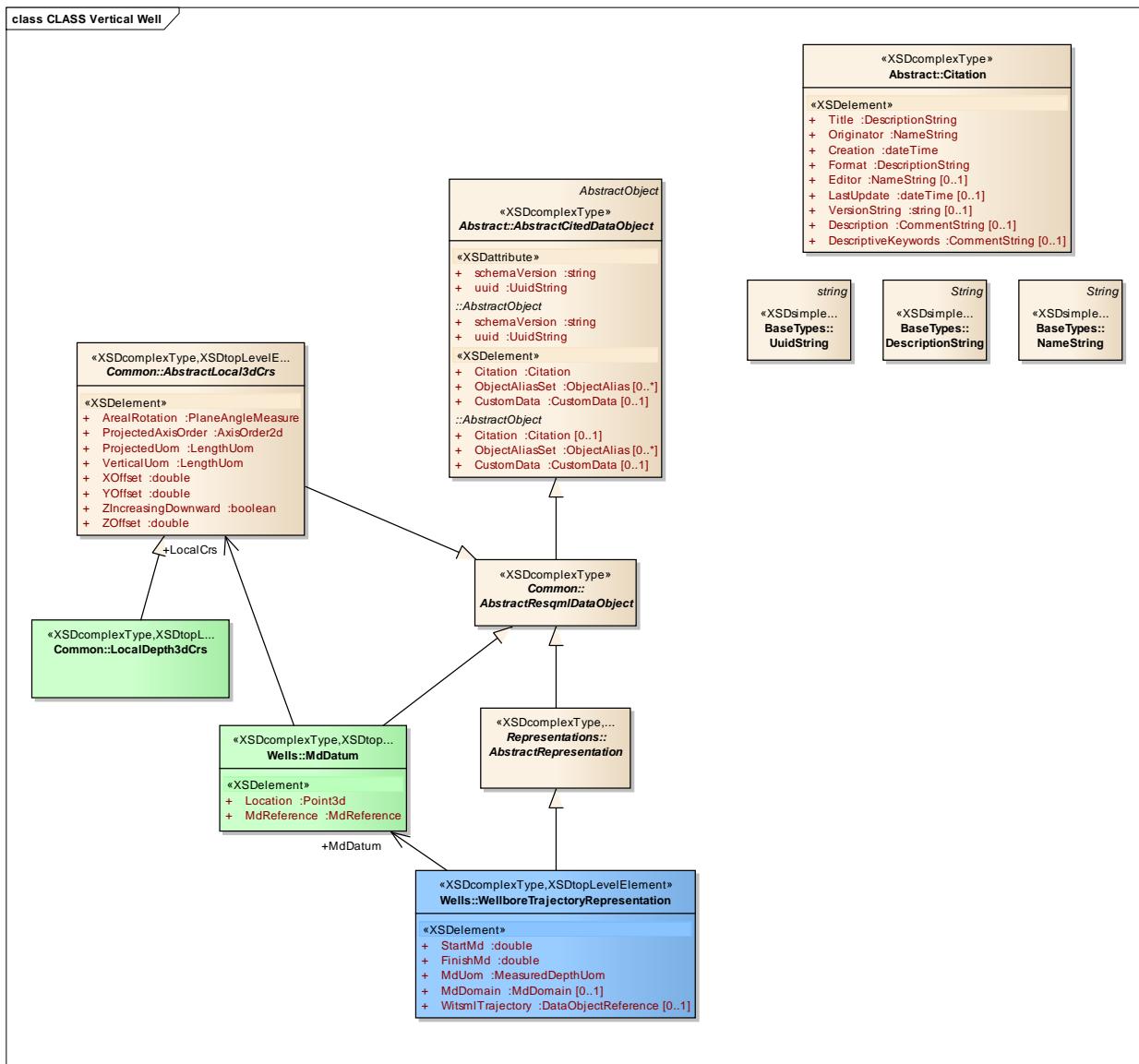


Figure 12-8. (Part 3 of 4) Components of the class diagram for a wellbore trajectory representation:
Description of the wellbore trajectory representation includes its implicit geometry.

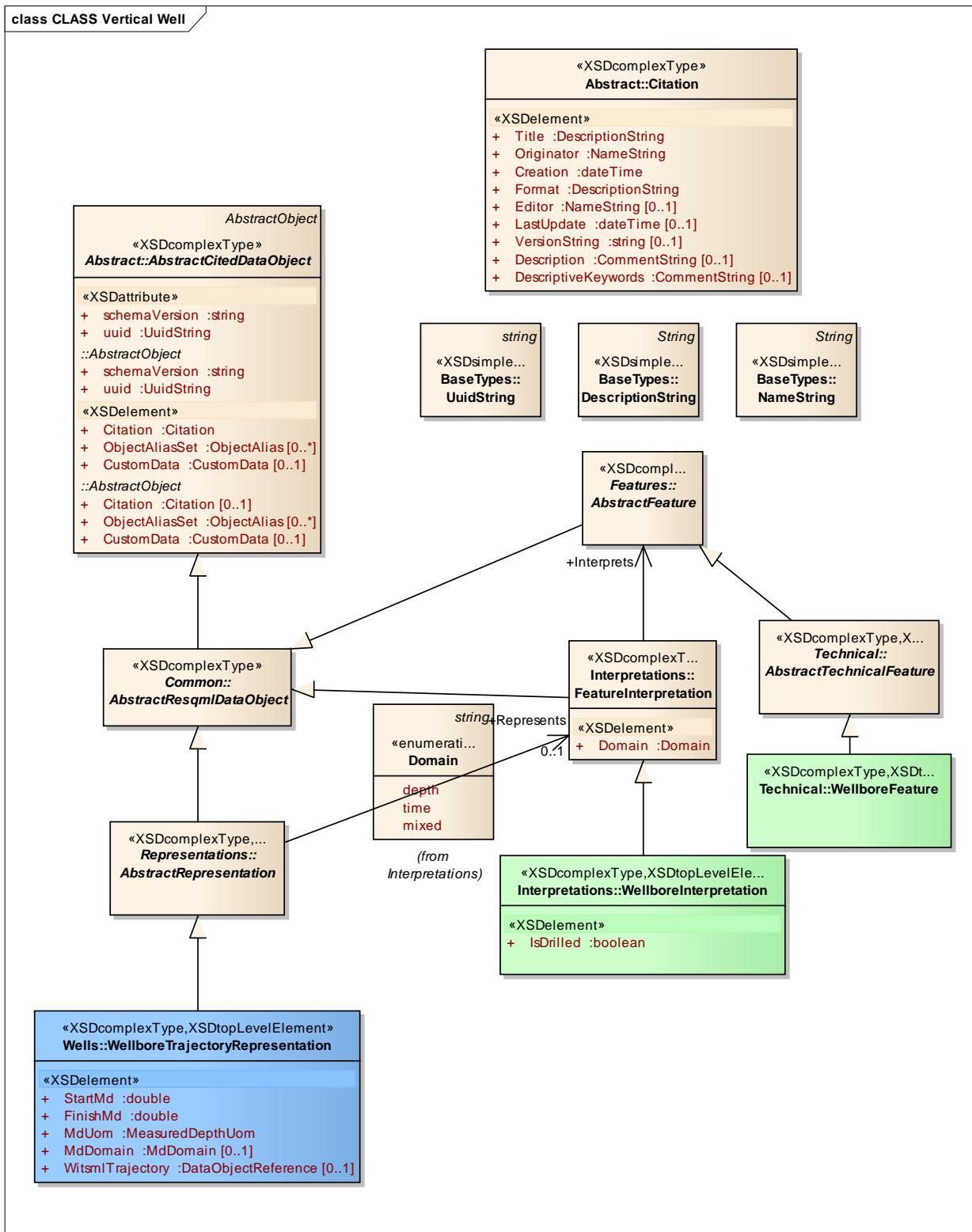


Figure 12-9. (Part 4 of 4) Components of the class diagram for a wellbore trajectory representation: Description of the relationship between the wellbore trajectory representation, the wellbore feature interpretation, and the wellbore feature.

12.7.2 Instance Diagram

Figure 12-10 shows the instance diagram of this two-well example, which is expanded into the two parts of **Figure 12-11** and **Figure 12-12**.

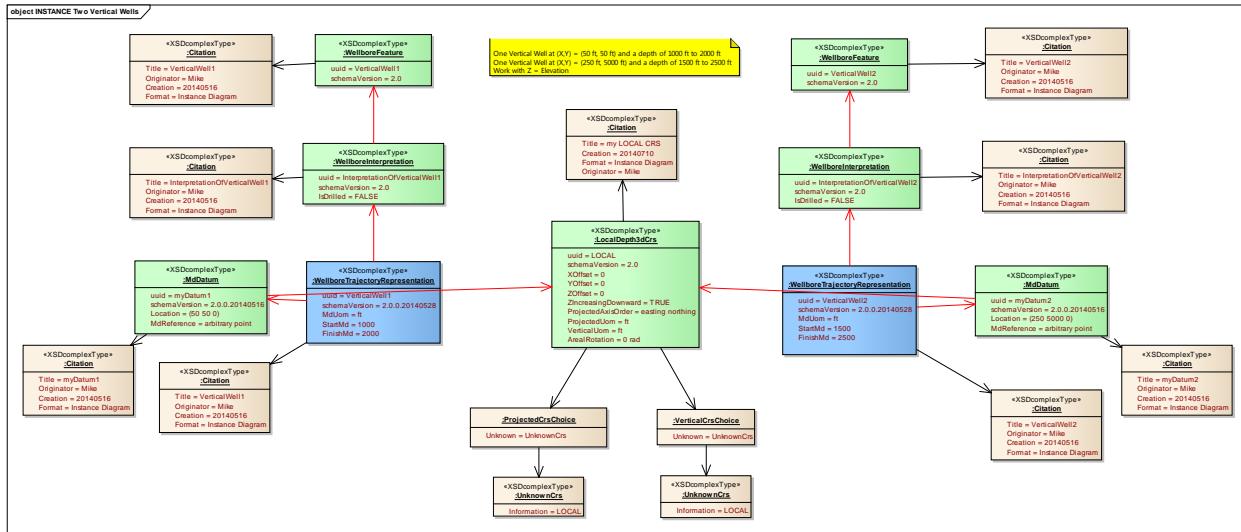


Figure 12-10. Overview of the instance diagram for a wellbore trajectory representation. Portions of this figure are expanded in **Figure 13-11** and **Figure 13-12**.

In **Figure 12-11**, we can see that one well is located at a surface position of (50 ft, 50 ft) in the local coordinate reference system, and extends from a depth of 1000-2000 ft. The other is at a surface position of (250 ft, 5000 ft) and extends from a depth of 1500-2500 ft.

For this specific example (**Figure 12-12**), the local 3D CRS references an unknown projected CRS and an unknown vertical CRS as their spatial reference. This would be unusual in practice for a drilled well, but may arise in a reservoir simulation well planning workflow in which the simulation applications may not be CRS aware.

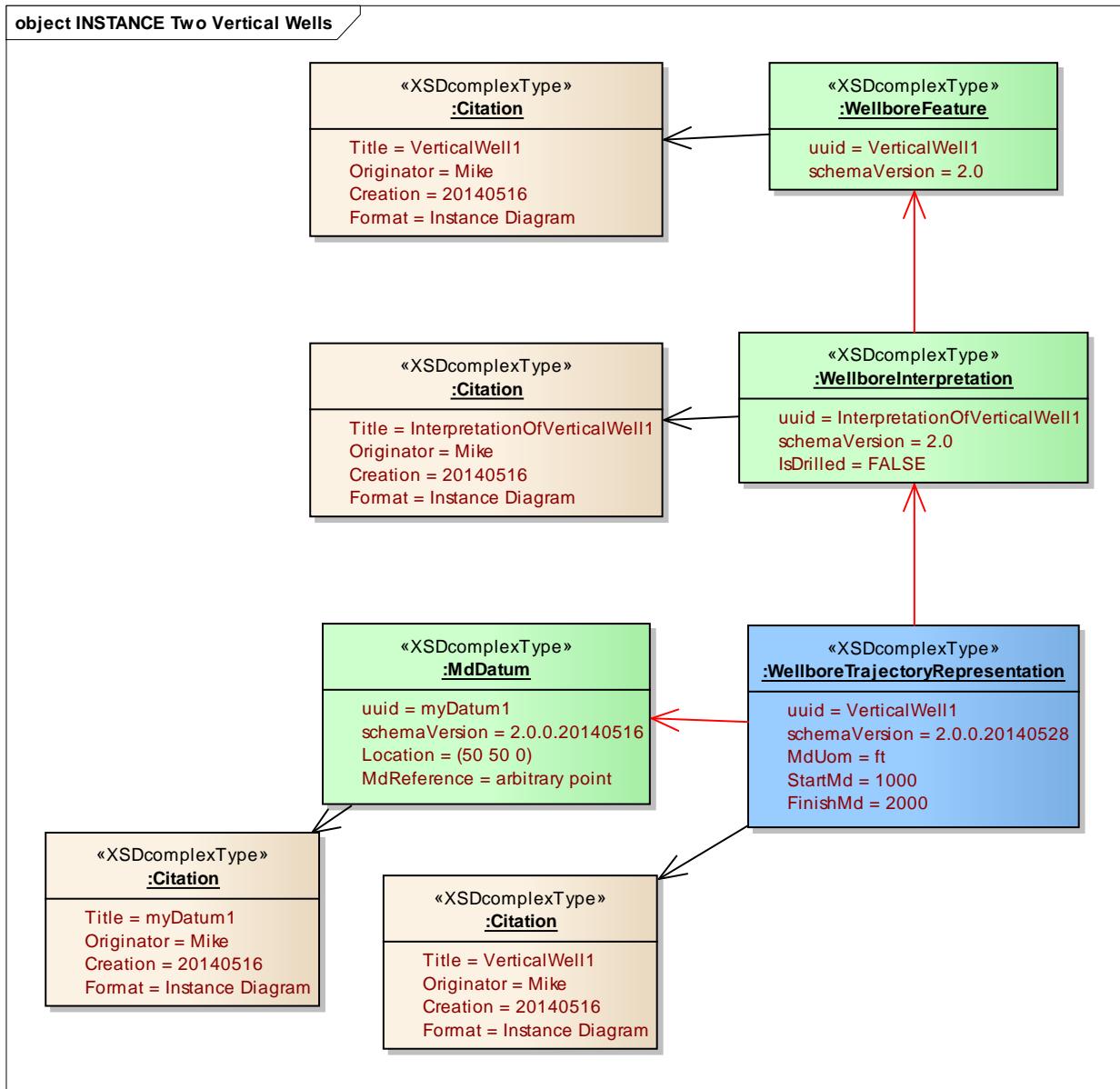


Figure 12-11. (1 of 2) Components of the instance diagram for a wellbore trajectory representation: Instance of the wellbore trajectory representation for the first vertical well.

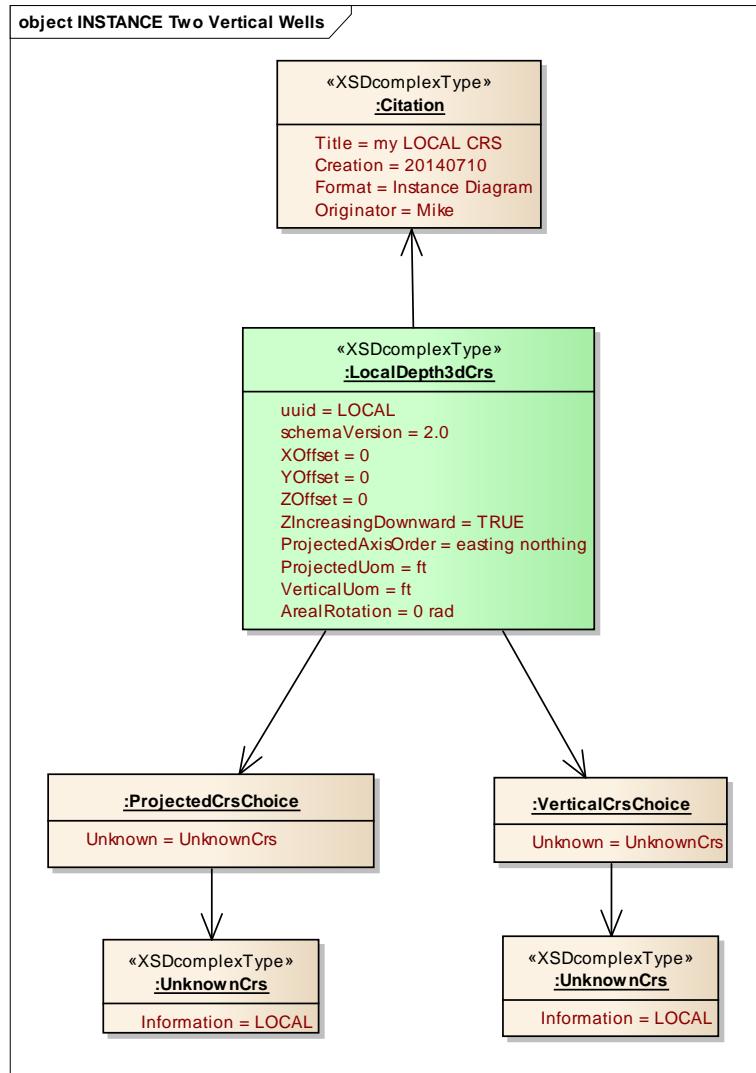


Figure 12-12. (2 of 2) Components of the instance diagram for a wellbore trajectory representation: Instance of the local depth 3D CRS, which is referenced by the MD datum data object.

13 Seismic

A seismic survey is an organization of seismic lines. In the context of RESQML, a seismic survey feature does not refer to any vertical dimension information, but only areally at shot point locations or common midpoint gathers. However, vertical coordinates may be transferred as part of the seismic coordinate geometry.

The seismic traces, if needed by reservoir models, are transferred in an industry standard format such as SEGY. The SEGY format contains information about the number of samples in the seismic traces and whether the vertical domain is in time or depth. This section only discusses the areal aspects of seismic surveys.

RESQML supports two basic kinds of seismic surveys:

- Seismic lattice (organization of the traces for the 3D acquisition and processing phases).
- Seismic line (organization of the traces for the 2D acquisition and processing phases).

Additionally, to transport several 3D seismic surveys or seismic lines of one or more 2D seismic surveys together:

- 3D Seismic lattices can be aggregated into a seismic lattice set (3D survey set), if needed.
- 2D Seismic lines can be aggregated into a seismic line set (2D seismic survey or surveys) in the same way.

Thus there are four seismic survey features that are represented in RESQML as follows:

- A seismic lattice is generally represented using a grid 2D representation.
- A seismic lattice set is then represented by several grid 2D representations.
- A seismic line is generally represented using a polyline representation.
- The seismic line set is then represented by several polyline representations.

For more information on representations, see Chapter 5.

In RESQML seismic surveys are technical features that do not have multiple interpretations. Although it is possible to re-interpret a seismic survey for improved physical properties or for positioning, these types of relationships are not included within the RESQML knowledge hierarchy. (For more information on the knowledge hierarchy, see Chapter 4.)

13.1 Seismic Lattice

A seismic lattice defines the seismic trace organization during a 3D acquisition. It is represented by a regular IJ 2D grid (**Figure 13-1**) where each node corresponds to one final common midpoint (CMP) gather.

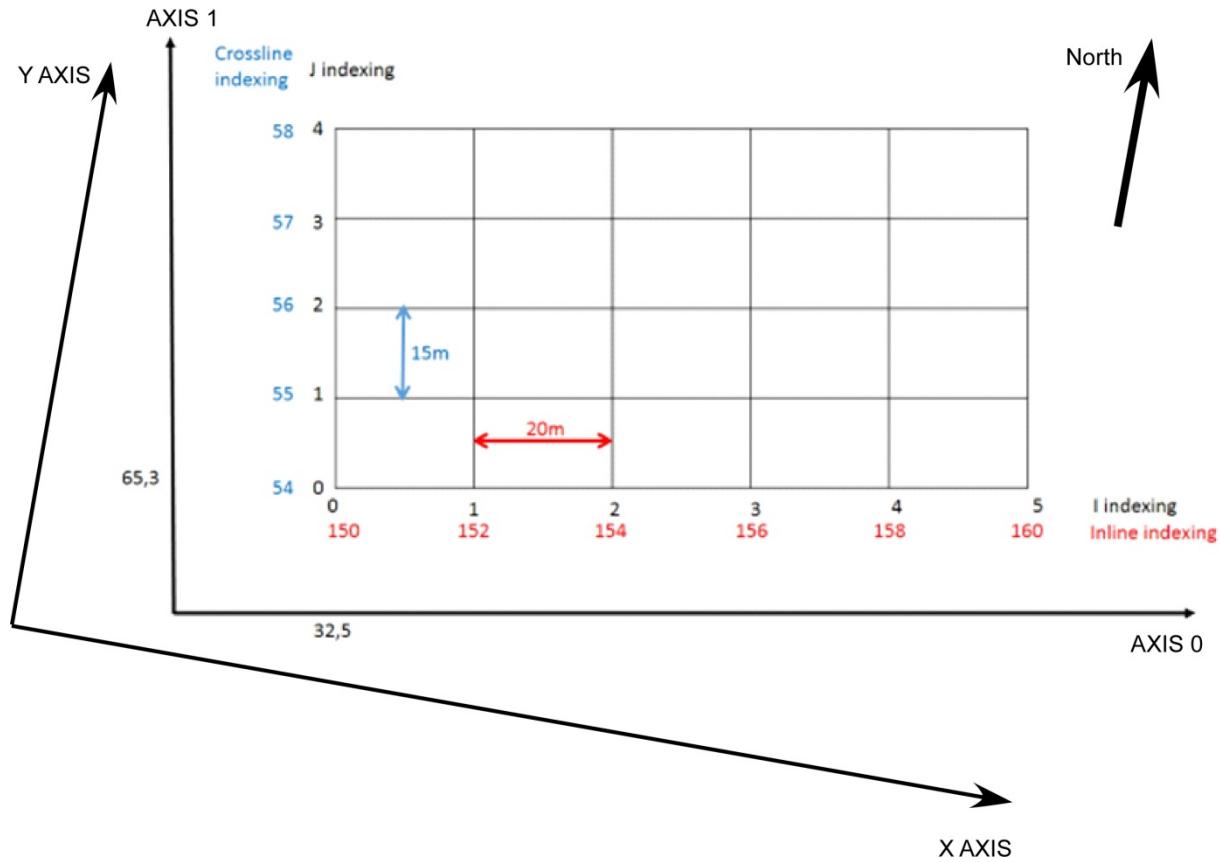


Figure 13-1. A seismic lattice.

At the feature level, the dimensions of seismic lattices are usually oriented along an inline axis and a crossline axis. The indexing of these two dimensions must be regularly incremented along each axis. This is the only constraint; for example, there is no constraint that the origin indices must begin at 0,0.

13.2 Seismic Lattice Representation

As a feature, there is no requirement to define an equivalent 1D index from the inline and crossline indices. However, there is a requirement to do so as a representation. (For more information in grids, see Chapter 10.)

Two types of representations may be used to provide the actual geometry of surveys:

- The 2D representation is a grid 2D representation where the geometry is provided by a 2D lattice. When using a grid 2D representation, inline/crossline corresponds to the fastest/slowest indices, respectively.
- The 3D representation is an IJK grid representation where the geometry is provided by a 3D lattice, which corresponds to the geometry of a 3D volume. Because the fast axis (I) in a seismic volume corresponds to vertical traces, inline/crossline typically corresponds to the J/K indices, respectively, although the details will vary depending upon the selection of columns or pillars, as discussed in more detail below.

At the representation level, the dimensions of the seismic lattices are typically regularly spaced. The indexing of these two dimensions is very constrained; the origin indices are always 0,0 and the increment between two node indices is always equal to 1.

Because the lattice geometry is based on axes defined by vectors, the direction of these vectors provides the angular information usually associated with a survey, such as the rotation relative to the local CRS.

Figure 13-1 shows how we use an array lattice of points 3D (ArrayLatticeOfPoints3d) to define the geometry of the representation of the survey. **Figure 13-2** is an example instance diagram of a seismic lattice.

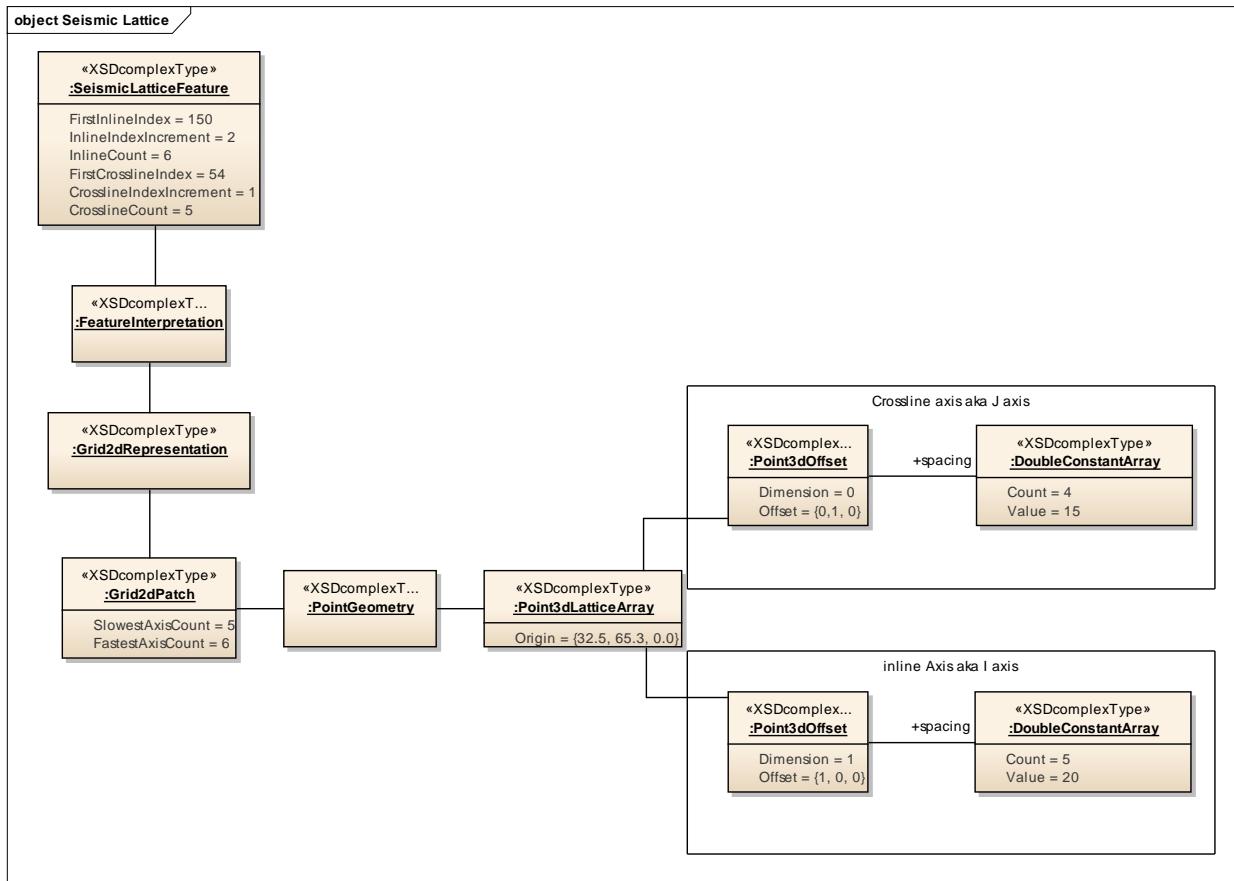


Figure 13-2. Instance diagram of a seismic lattice.

13.3 Seismic Line

A seismic line defines the seismic common midpoint (CMP) organization for the 2D survey acquisition and processing. It is represented by a polyline (**Figure 13-3**) where each node corresponds to one seismic gather location.

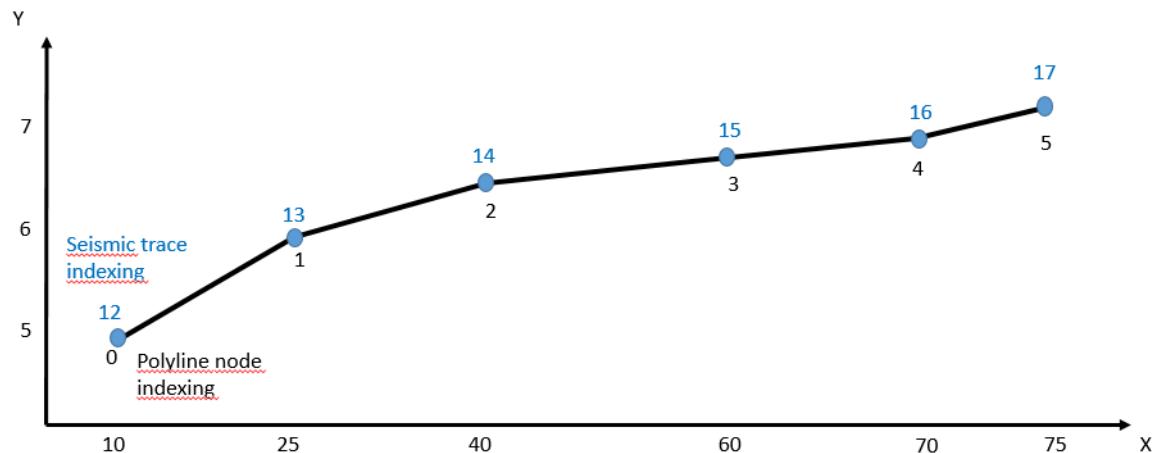


Figure 13-3. A seismic line.

At the feature level, the dimension of a seismic line is the dimension of the seismic traces. The indexing of this dimension must be regularly incremented, but this is the only constraint.

At the representation level, the dimension of a seismic line is geometrically oriented and corresponds to the indexing of the polyline nodes. The indexing of this dimension is very constrained: the origin index is always 0, and the index increment between two nodes is always equal to 1.

Figure 13-4 shows how RESQML uses an array of explicit points 3D (ArrayOfExplicitPoints3d) to define the geometry of the representation of the survey.

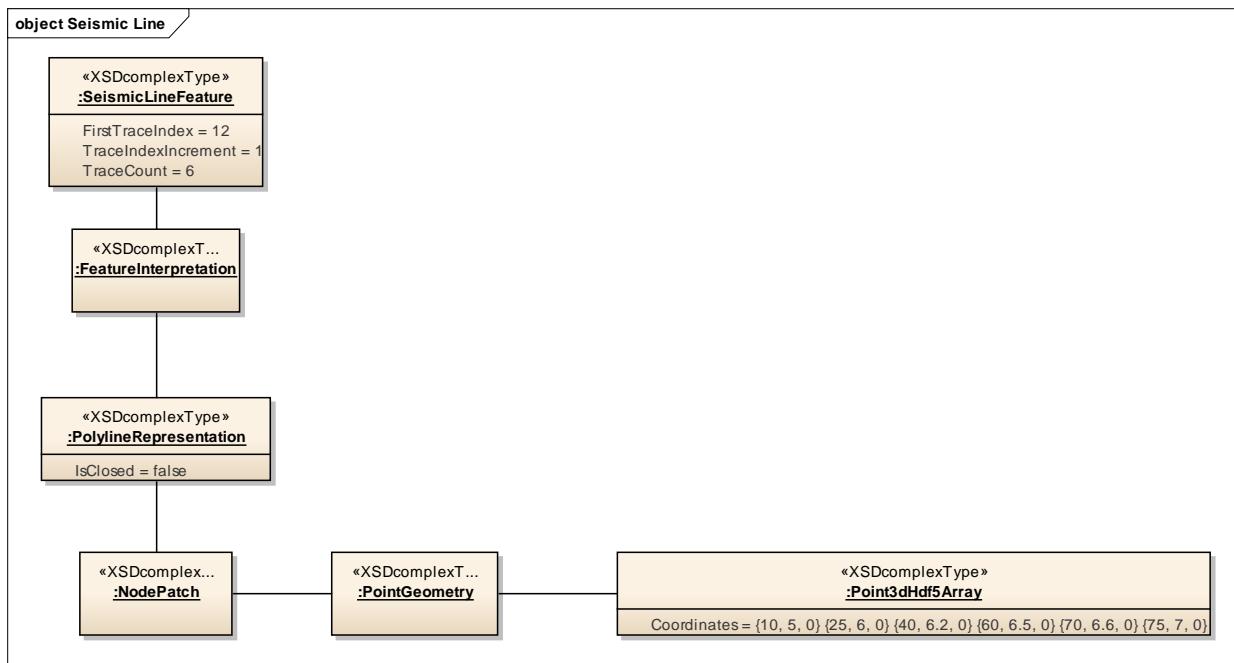


Figure 13-4. Geometry of the representation of a seismic survey.

13.4 Seismic Survey as a Part of the Geometry of the Business Object

Any business object, such as a horizon, can define its geometry by means of Z values. The Z values are then combined with the geometry of a seismic survey to give XYZ points.

This is done using the class point 3D Z value array (Point3dZValueArray) (**Figure 13-5**), which can define Z values on a subset or on the whole of an existing representation indicated in the attribute supporting geometry.

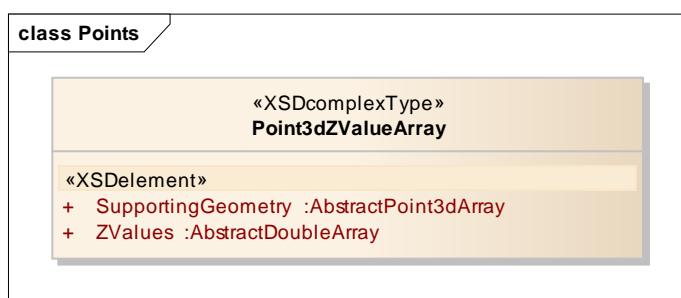


Figure 13-5. Point3dZValueArray is used to define a Z point for a business object; when it's combined with a seismic survey it gives the XYZ points.

The subset or the whole of an existing geometry (attribute supporting geometry above) is generally handled by the class Point3dFromRepresentationLatticeArray, where we use an integer lattice array to select the nodes where the Z values are applied (**Figure 13-6**).

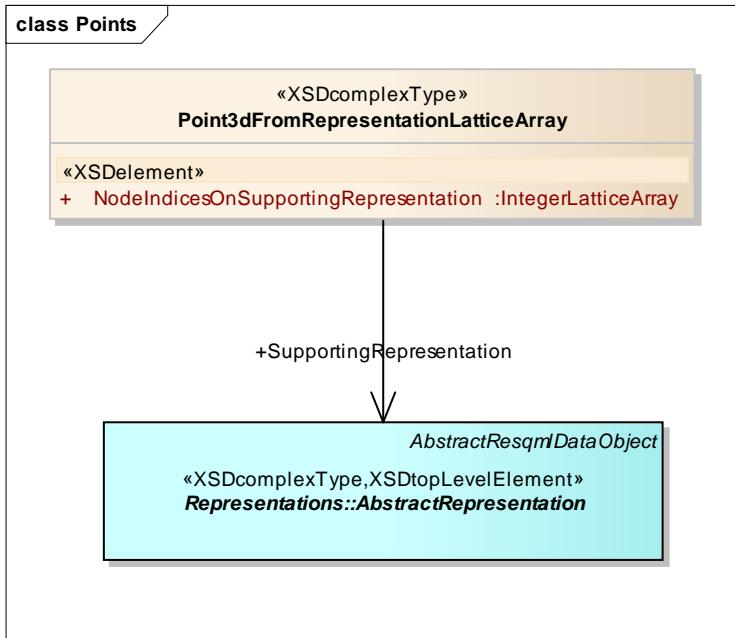


Figure 13-6. Point3d from representation using a lattice array construction.

The subset of nodes is given by means of (**Figure 13-7**):

- A start value (origin): linearized index of the origin node of the subset.
- For each lattice dimension defined by its order in the XML instance, the list of the offset between selected nodes.

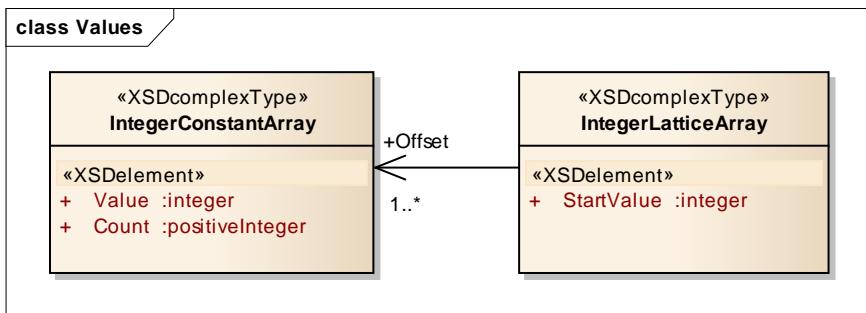


Figure 13-7. Integer lattice array construction.

13.4.1 Example

Figure 14-8 shows an example survey. Figure 13-9 shows how the green and red 2D grid patched are defined.

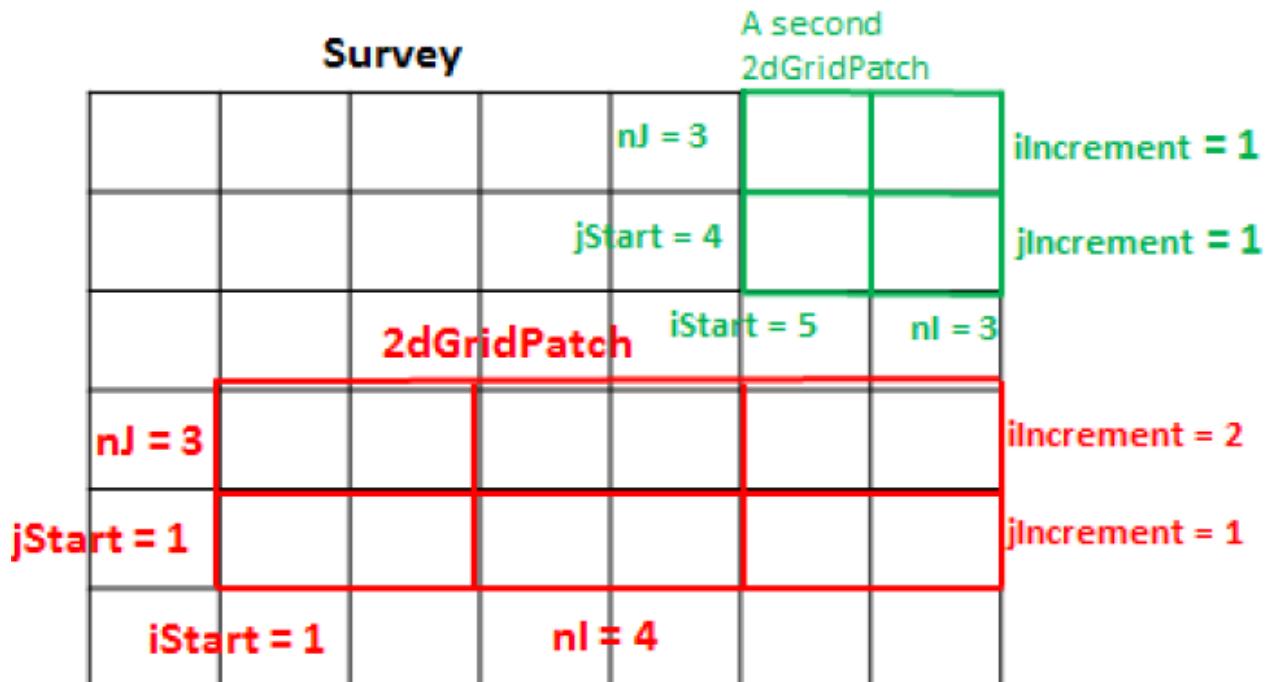


Figure 13-8. An example survey.

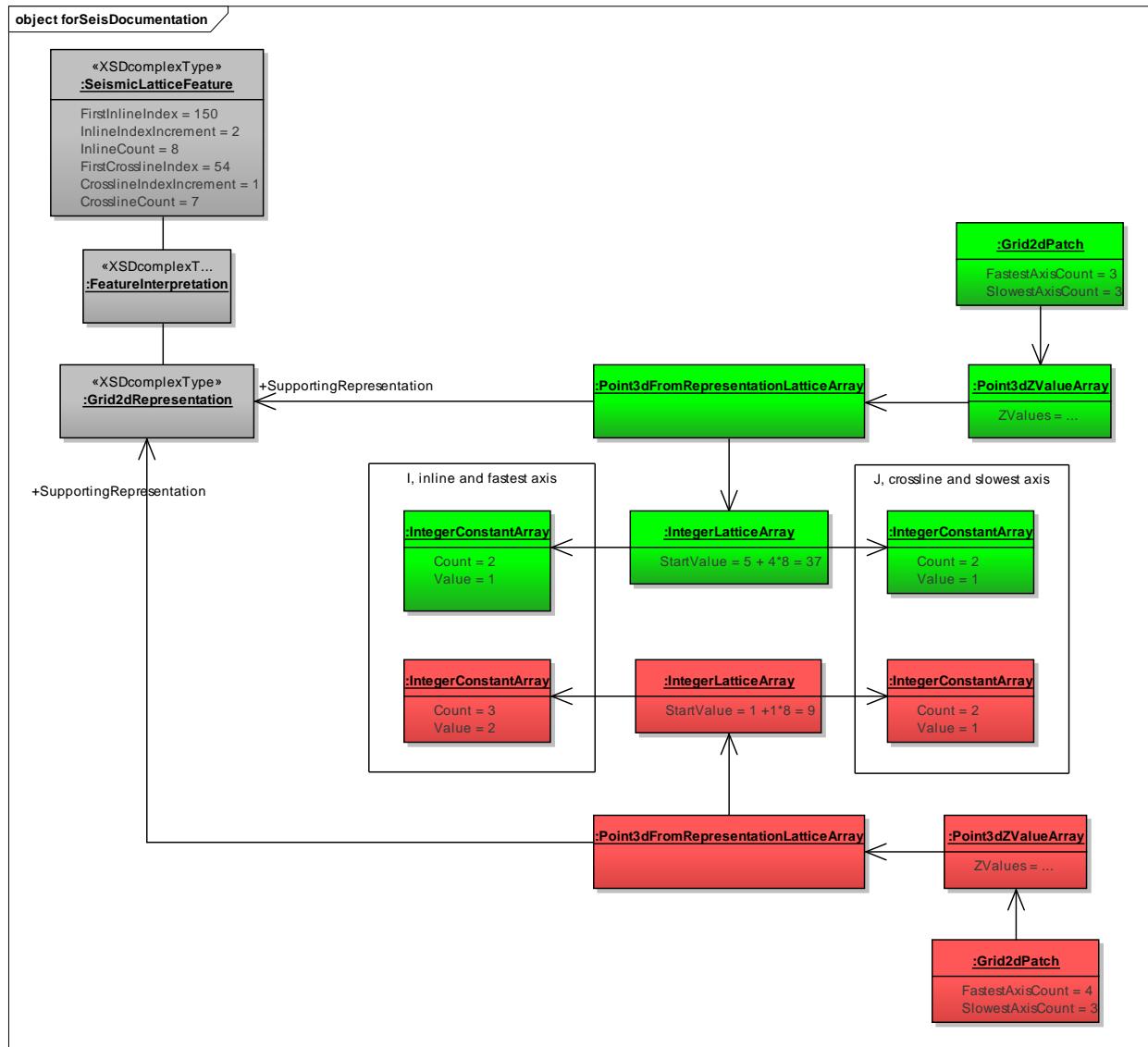


Figure 13-9. The green grid 2D patch and the red grid 2D patch both have a geometry, which is supported by the representation of the seismic lattice.

13.5 Seismic Survey as Extra Information on the Geometry of a Representation

Any representation based on point geometry may have seismic coordinates associated with the points. (**Figure 13-10**). (For more information on point geometry, see Section 7.3.)

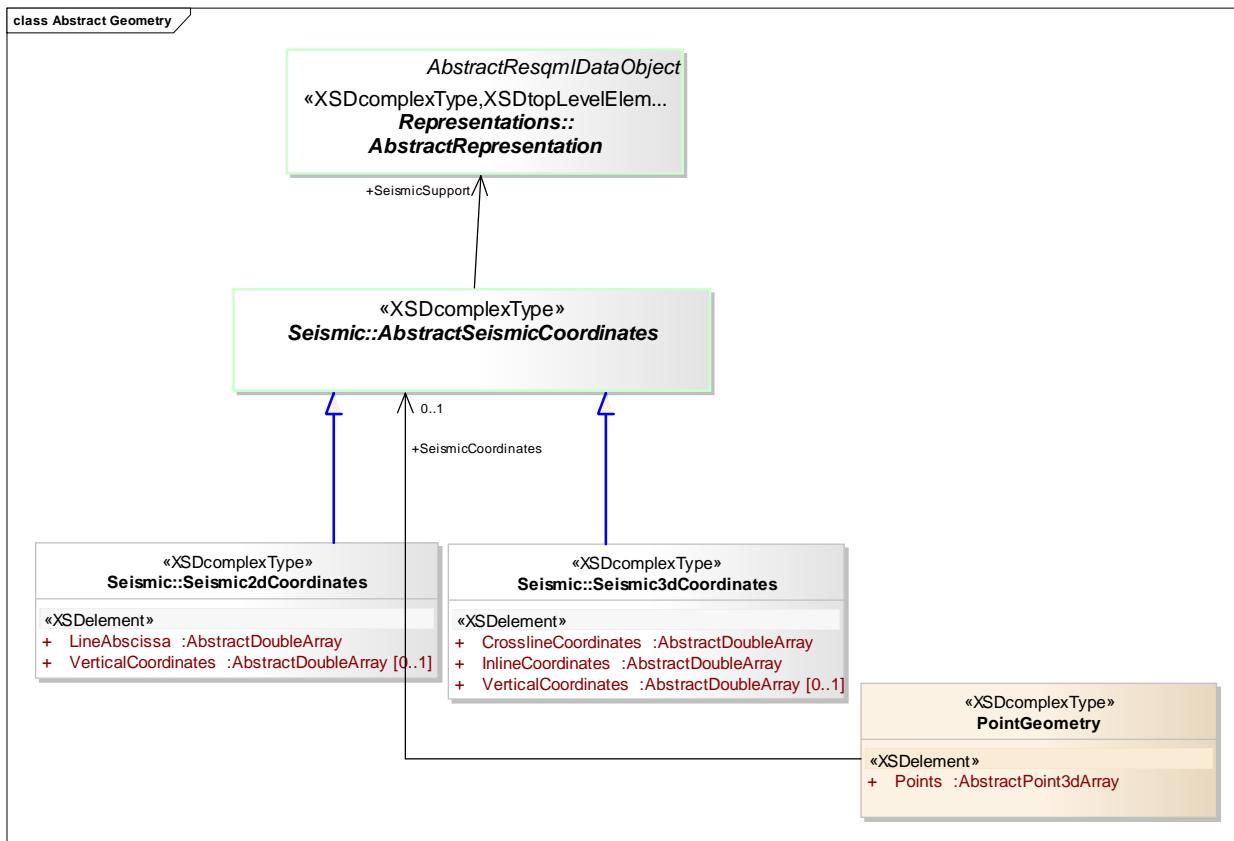


Figure 13-10. Seismic coordinates may be associated with the point geometry of any representation.

Seismic coordinates are based on a supporting representation, which must be a representation of a seismic survey.

- If the seismic survey is a seismic line, then you must give seismic coordinates using the seismic 2D coordinates.
- If the seismic survey is a seismic lattice, then you must give seismic coordinates using the seismic 3D coordinates.

The seismic coordinates must be ordered exactly as the points are ordered in the representation of the business object. The 3D seismic coordinates consist of three arrays of (InlineCoordinate, CrosslineCoordinate, VerticalCoordinate) values. The 2D seismic coordinates consist of two arrays of (LineAbscissa, VerticalCoordinates) values. These coordinates are not constrained to be integers and so can represent positions between the nodes of a seismic survey.

13.6 Seismic Survey Representations on a Grid

There is a reservoir modeling practice in which an IJK grid used for geologic modeling or flow simulation is aligned with the inline/crossline lattice of a seismic survey. There are two natural ways to express this relationship in RESQML, both using subrepresentations.

- If the intent is to align the lattice with the edges of the cells, then the indexable element kind is “pillars” and the NIL x NJL indexing would correspond to inline and crossline, respectively.
- Similarly, if the intent is to align the lattice with the cells themselves, then the indexable element kind is “columns” and the NI x NJ indexing corresponds to inline and crossline, respectively.

The explicit choice of indexable element in the subrepresentation removes the ambiguity in alignment (column or pillars) that often makes this workflow confusing.

Appendix A. Release Notes

This appendix contains summary descriptions of changes per published version of changes to this document and related data objects.

A.1 Version 2.1

Published 11 November 2016

Description of Change	Primary Affected Documentation
Documentation pertaining to Energistics CTA was moved to the <i>Energistics CTA Overview Guide</i> .	Much of chapter 2 content now in CTA Guide; chapters 2 and 3 re-organized. Chapters 4-13 same basic order.
Objects moved from RESQML to Energistics common. <ul style="list-style-type: none"> • (ResqmlCommon) Jagged Array • (ResqmlCommon) Property-related objects (PropertyKind, PropertyKindDictionary, GeologicTime, TimeIndex, TimeIndices, TimeSeries, and TimeSeriesParentage) • Activity Model (add-on object for RESQML v2.0.1). 	Various
The name of the RESQML common folder was changed from <i>common</i> to ResqmlCommon; edits made throughout the document. Figures updated as necessary. Objects in ResqmlCommon eliminated due to changes to Energistics common: <ul style="list-style-type: none"> • AbstractResqmlDataObject 	Throughout
Clarification of definitions for <i>feature</i> and <i>interpretation</i> . Changes for feature are based, in part, on reorganization of the UML model structure (nested feature sub-packages of technical and geological have been eliminated).	Sections 3.1.3 and 4.1
Edits to explain and reflect structural changes of the RESQML UML model. Previous hierarchies or nested packages were “flattened” to address issues and improve performance for schema generation process.	Chapter 4 and 5
Simplified treatment of patch indexing	Section 5.3
Properties: These changes were made and many of the related data objects were moved to Energistics common (see above): <ul style="list-style-type: none"> • Added Boolean property • Edited text to explain the addition of a property kind dictionary • Explanation of how users can extend the property kind list • Added a new facet: orientation • Added text to explain how to map simulator key words to RESQML property aliases • Clarification of how units of measure work with properties (quantity class, property kind and property value) • Added a bullet to explain how to handle results of a simulation run for property values representing the same property over time 	Chapter 7
Earth Model <ul style="list-style-type: none"> • Updated the documentation to better explain how to group together 	Chapter 8

Description of Change	Primary Affected Documentation
<p>faults in an interpretation set and introduce a fault or a fault network in a stratigraphic occurrence.</p> <ul style="list-style-type: none"> Clarified documentation about the stratigraphic rank of a horizon in a structural organization interpretation. Clarified contact interpretation parts. Removed contact relationships and clarified verbs for binary contact interpretation parts. Geologic unit interpretation now references lithology kind enumerations (originally from WITSML but now moved to Energistics common) 	
Documented how to associate stratigraphic units with IJK grid cells in K expansion context by creating a local property kind called "geologic k".	Section 10.9
Added information about alternate indexing.	Section 10.10. 2.1

Appendix B. HDF5 Basics

HDF5 is a set of open file formats and libraries designed to store and organize large amounts of numerical data, and improve speed and efficiency of data processing. Specifically, it provides: machine/architecture-independent "binary" format (supported on Windows, Linux, etc. APIs available in C++, Java, .NET, etc); hyper-slabbing of array data so that IJK sub-arrays may be extracted without reading the entire data file; and built-in data compression.

This appendix contains a high-level of how to implement HDF5 in RESQML in the C programming language.

Beginning with RESQML v2.0, use of HDF5 for large data sets (greater than 2GB) is mandatory.

For more information on HDF5, available tools and tutorials, or to download the libraries, see the HDF Group website at: <http://www.hdfgroup.org/HDF5/>.

B.1 Overview of RESQML HDF5

For HDF5 conventions and formats used with RESQML, see 5.2.2 (page 50).

The "heavy" data associated with a RESQML object may be stored in an HDF file in one or more "data sets." Datasets are collected hierarchically in the file in *groups*.

B.2 Adding HDF5 to your Code

HDF5 is provided as dynamic link libraries with a C-like application interface. You may add references to these DLLs directly using the C headers provided with the libraries. You may similarly link to these libraries in a .NET application using P/Invoke, or in a Java application using JNI.

A number of language-specific implementation layers are available on the HDF Group website. These implementations "wrap" the C interface, and they may provide a more intuitive way of accessing HDF5. These implementations vary in terms of the amount of the HDF5 API that is covered, and they may not be appropriate for advanced use of HDF5. However, RESQML requires only a subset of the HDF5 API, so you may find one of these implementations suitable for your development needs.

The rest of this chapter illustrates a number of typical RESQML HDF5 scenarios using the C API.

B.3 HDF5 Handles

The HDF5 API uses a handle-based pattern. This means that the typical programming scenario is as follows:

- Create or open an HDF5 element, such as a file or Dataset. HDF5 returns a handle of type *hid_t*.
- Interact with the HDF5 element using the handle.
- Close the handle using the appropriate "close" method. For example, you can close a file handle using the *H5Fclose* function.

In the examples below, the closing of handles is omitted for clarity.

B.4 Reading from a RESQML HDF5 File

Reading data from a RESQML HDF5 file involves a number of steps. This brief overview shows the key steps in this process, omitting details, such as error handling and handle closing, for clarity.

B.4.1 Reading from a RESQML HDF5 File

Reading data from an HDF5 file involves these steps:

- (i) Open the file
- (ii) Open the dataset and read the data

(iii) Close the file

- **Open a RESQML HDF5 File**

To open a RESQML HDF5 file, use the *H5Fopen* function from the H5F family of functions. Many options are available, but this example shows how to open a file for reading only.

```
hid_t FileID = H5Fopen(_FileNameH5F_ACC_RDONLY, H5P_DEFAULT);
```

- **Open a Dataset and Read Data**

Reading data from a Dataset involves a number of steps: (i) open the hierarchical groups containing the Dataset, (ii) open the Dataset, (iii) get the Dataspace information to determine the rank and dimensions of the data in the Dataset, (iv) read the data, and (v) close all the handles when done.

The path to the Dataset—including the groups it is contained in—is provided in the XML file. This path may be broken into group components, and each of these groups may be opened in turn. Alternatively, the complete path information may be passed to the dataset-opening function. This sample opens the dataset directly using the complete path information.

```
hid_t DatasetID = H5Dopen2(FileID, DatasetName, H5P_DEFAULT);
```

After the Dataset is open, you may query the Dataspace to get information on the rank and dimensions of the data. This information is usually also contained in the RESQML XML file, but you may still want to read it from the HDF5 file to verify the integrity of the file.

```
hid_t DataspaceID = H5Dget_space(DatasetID);
int ndims = H5Sget_simple_extent_ndims(DataspaceID);
if (ndims == ExpectedDimensionsRank)
    ndims = H5Sget_simple_extent_dims(DataspaceID, Dimensions, NULL);
```

Reading the data may be done all at once or in pieces called *hyperslabs*. Reading using hyperslabs can be more efficient in terms of memory usage, but for clarity the sample below reads all data into a memory buffer at once.

```
herr_t e = H5Dread(DatasetID, TypeID, H5S_ALL, H5S_ALL, H5P_DEFAULT, Buffer);
```

B.5 Writing Data to a RESQML HDF5 File

Writing data to a RESQML HDF5 file is similar to reading data from a file. You must create the appropriate types as before, and all handles must be closed when they are no longer required.

B.5.1 Enabling Compression

RESQML is designed to store large quantities of data, and HDF5 helps with this by compressing the data as it is saved to a file.

HDF5's implementation of compression provides the writer with a trade-off between compression efficiency and access speed for readers. It does this by splitting the data into fixed-sized "chunks" and compressing each chunk individually.

The larger a chunk is, the more efficient the compression will be. However, this efficient compression is at the cost of reading efficiency, because HDF5 needs to read and decompress into memory larger parts of the file to get at a particular part of the data.

Smaller chunks improve the reading efficiency of HDF5, because less data must be read and decompressed into memory to get at a particular part of the data. The downside to this is that the compression is less efficient and resulting file sizes will be larger.

When determining a chunk size to use, consider how the data will be accessed. For example, a likely use case for applications reading explicit grids is to extract only particular layers out of the data using hyperslabbing. A chunking setup that supports this scenario is to make a chunk for each layer. This approach makes accessing the data a layer at a time maximally efficient, while still giving HDF5 a large enough block of data to compress.

Enabling compression involves creating an HDF5 Parameter that is used when creating a Dataset. This parameter contains the size of the chunk and the compression properties.

```
hid_t DatasetParameterID = H5Pcreate(H5P_DATASET_CREATE);
if (DatasetParameterID > 0)
{
    herr_t e = H5Pset_chunk(DatasetParameterID, ChunkRank, ChunkDimensions);
    e = H5Pset_deflate(DatasetParameterID, 1);
}
```

B.5.2 Creating the Groups that Contain the Dataset

Groups are handled slightly differently when writing to a file: HDF5 does not create groups automatically when a Dataset path is supplied. Your application must do one of the following:

- If a group doesn't yet exist, explicitly create each group.
- If a group does exist, open it before creating the dataset.

With the current HDF5 format, only two groups are required:

- The top-level RESQML group.
- A group for the object being stored, which has the object's GUID as its name.

The top-level group is created using the FileID:

```
hid_t RESQMLGroup = H5Gcreate2(FileID, "RESQML", H5P_DEFAULT, H5P_DEFAULT,
H5P_DEFAULT);
```

The object group is created using the RESQMLGroupID:

```
hid_t ObjectGroupID = H5Gcreate2(RESQMLGroup, ObjectGuidString, H5P_DEFAULT,
H5P_DEFAULT, H5P_DEFAULT);
```

Remember to close these IDs when you are finished with them.

B.5.3 Creating the Dataspace

The Dataspace tells HDF5 how much data is to be stored and how it is configured in terms of rank and dimensions.

```
hsize_t Dimensions[Rank];
Dimensions[0] = Length0;
//Etc.
hid_t DataspaceID = H5Screate_simple(Rank, Dimensions, NULL);
```

B.5.4 Creating the Dataset

The Dataset is created using the ObjectGroupID, the DataspaceID, the DatasetCompressionParameter, and the Dataset name.

```
hid_t DatasetID = H5Dcreate2(ObjectGroupID, DatasetName, TypeID, DataspaceID,
H5P_DEFAULT, DatasetParameterID, H5P_DEFAULT);
```

B.5.5 Writing the Data

Writing of the data to the Dataset is done using the DatasetID the TypeID of the data being stored, and a pointer to the data buffer. Hyperslabbing may be used to write the data a portion at a time.

```
herr_t e = H5Dwrite(DatasetID, TypeID, H5S_ALL, H5S_ALL, H5P_DEFAULT, Buffer);
```