ETP v1.1 for WITSML v2.0 Implementation Specification

ETP Overview	Energistics Transport Protocol (ETP) is a data-exchange specification that enables the efficient transfer of data between applications and systems, including real-time streaming of data. ETP is part of the Energistics Common Technical Architecture and can be used with all Energistics standards. ETP is the application program interface (API) for WITSML v2.0.
Version	1.0
Abstract	This document: • Provides information on how to implement ETP v1.1 to send and receive WITSML v2.0 data. • Must be used as a companion to the *ETP_Specification_v1.1_Doc_v1.1.pdf*.
Prepared by	Energistics and the WITSML SIG
Date published	16 October 2017
Document type	Implementation Specification
Keywords:	standards, energy, data, information, process, drilling, logging, ETP

Amendment History				
Std Version	**Doc Version**	**Date**	**Comment**	**By**
1.0	1.0	16 Oct 2017	First publication	Energistics and the WITSML SIG

Table of Contents

1 Introduction 6
1.1 Audience, Purpose, and Scope 6
1.1.1 Scope of this Document 6
1.1.2 ETP Protocols in Scope 7
1.2 Resource Set 8
1.2.1 Documentation Conventions 8
2 General Rules and Concepts 9
2.1 Example Implementation 9
2.2 ETP Specification is the Primary Source 9
2.3 All Applications Must Handle Optional Header in XML v1.0 9
2.4 Energistics Identifiers 9
2.4.1 WITSML Hierarchies 10
2.5 Mapping Common Tasks from WITSML v1.x to v2.0 10
2.5.1 General API Mapping 10
2.5.2 Determine Server Capabilities 11
2.5.3 Determine Which Wells a User has Access to 11
2.5.4 Determine Which Wellbores a User has Access to 11
2.5.5 Determine Which Wellbore Child Objects a User has Access to 11
2.5.6 Retrieve a Static Data Object from the Store 12
2.5.7 Retrieve a Trajectory with Stations or Mud Log from the Store 12
2.5.8 Retrieve a Log from the Store 12
2.6 Using Data Objects from Energistics *common* 12
3 Making a Connection (Protocol 0: Core) 14
3.1 Example Messages 14
4 Streaming Channel Data (Protocol 1: ChannelStreaming) 18
4.1 Overview of Growing Objects and How They Work 18
4.1.1 Getting Historical Channel (Log) Data 18
4.2 Example: Streaming Log Channel Data 18
4.2.1 Start Message 19
4.2.2 ChannelDescribe Message 19
4.2.3 ChannelMetadata Message 19
4.2.4 ChannelStreamingStart Message 25
4.2.5 ChannelData Message 25
4.2.6 ChannelStreamingStop Message 26
4.2.7 ChannelRangeRequest Message 26
4.3 Example: Streaming Trajectory Stations 28
4.3.1 Start Message 28
4.3.2 ChannelDescribe Message 28
4.3.3 ChannelMetadata Message 29
4.3.4 ChannelStreamingStart Message 31
4.3.5 ChannelData Message 32
4.3.6 ChannelStreamingStop Message 33
4.4 Examples: Streaming Alarms, Alerts, and Annotations 34
4.4.1 Start Message and ChannelDescribe Message 34
4.4.2 ChannelMetadata Message 34
4.4.3 Send Alarm Example ChannelData Message 37
4.4.4 Example Send Alert ChannelData Message 38
4.4.5 Send Annotation Example ChannelData Message 39

5 Discovering the Content of a Store (Protocol 3: Discovery) 41

5.1 Discovering “eml://” Must Return Supported URI Protocol Resources 41
5.2 Discovering “eml://witsml20” Must Return Folder Resources 42
5.3 Discovering a Data Object URI Must Return Folder Resources 42
5.4 Discovering a Child Data Object Type URI Must Return Data Object Resources 43
5.5 A Store May Limit the Maximum Number of Resources it Returns for Each GetResources Message 44

6 Interacting with a Store (Protocol 4: Store) 45

6.1 Overview of How it Works 45
6.2 PutObject 45
6.3 GetObject 46
6.3.1 Getting a Historical Log 47
6.4 DeleteObject 48

7 Subscribing to Change Notices (Protocol 5: StoreNotification) 49

7.1 RequestNotification 49
7.2 CancelNotification 51

8 Managing the Growing Part of an Object (Protocol 6: GrowingObject) 52

8.1 GrowingObjectPut 52
8.2 GrowingObjectGet 53
8.3 GrowingObjectGetRange 54
8.4 GrowingObjectDelete 55
8.5 GrowingObjectDeleteRange 55

9 Receiver-Immutable Fields 56

1 Introduction

The Energistics Transport Protocol (ETP) is a data-exchange specification that enables the efficient transfer of data between applications and systems. It defines a data-streaming mechanism so that data receivers do not have to poll for data and can receive new data as soon as they are available from a data provider.

ETP was initially developed as the API for WITSML v2.0+ (to replace the SOAP protocol API used up through WITSML v1.4.1). However, ETP is now also being developed for use with other Energistics domain standards (PRODML and RESQML), and is now part of the Energistics Common Technical Architecture (CTA).

This document:

- Provides the ML-specific implementation details for using ETP v1.1 to transfer WITSML v2.0 data. (Each version of each domain standard has/will have an ETP implementation specification.)
- Is a companion implementation specification to the *ETP Specification v1.1_Doc_v1.1,* which defines the behavior of all ETP protocols.
- Contains both normative and non-normative content. This Chapter 1 introduction is non-normative; all other chapters are normative. Consider content to be normative unless otherwise stated.

1.1 Audience, Purpose, and Scope

This document is intended for information technology professionals (e.g., software developers, programmers, etc.) who want to implement WITSML v2.0.

1.1.1 Scope of this Document

- The *Energistics Transfer Protocol* (*ETP) Specification* defines the basic protocol behavior for use with any of Energistics domain standards (WITSML, PRODML and RESQML).
- This implementation specification provides additional details needed to implement ETP (v1.1, doc v1.1) for use with the WITSML v2.0 data model.

NOTE: This implementation specification defines sender (i.e., a server, producer, or store) behavior only. User interface (UI) behavior has been considered in making decisions about sender behavior, but UI behavior is not defined.

1.1.2 ETP Protocols in Scope

The table below lists the ETP protocols that are in scope for this version of WITSML and lists references to the implementation details in this document.

ETP uses a stability index to indicate the maturity level of each protocol. This approach allows evolutionary development of the protocols, while allowing people to make informed decisions for an implementation plan (e.g., implement the stable protocols first).

More information is available in this implementation spec for the stable protocols. However, as SIG members continue to work on developing and implementing the protocols, learnings from that work will also be captured in this document.

Protocol Number and Name	Description	Stability Index
Protocol 0: Core	Creates and manages ETP sessions. Observes standard ETP behavior. No ML-specific tailoring required. For ETP message examples for WITSML v2.0, see Chapter 3.	**3 - Stable**
Protocol 1: ChannelStreaming	Defines a set of messages for exchanging channel-oriented data, where a channel is a time or depth series of individual data points. For WITSML v2.0 implementation details, see Chapter 4.	**3 - Stable**
Protocol 3: Discovery	Uses a RESTful (representational state transfer) approach to enable store customers to enumerate and understand the contents of store of data objects. For WITSML v2.0 implementation details, see Chapter 5.	**3 - Stable**
Protocol 4: Store	Used to perform CRUD operations (create, retrieve, update and delete) on data objects in a store. For WITSML v2.0 implementation details, see Chapter 6.	**3 - Stable**
Protocol 5: StoreNotification	Used to allow store customers to receive notification of changes to data objects in the store in an event-driven manner. For WITSML v2.0 implementation details, see Chapter 7.	**2 - Unstable**
Protocol 6: GrowingObject	Used to manage the growing parts of data objects that are index based (i.e., time and depth) but are not appropriate for Protocol 1.	**2 - Unstable**

1.2 Resource Set

The table below lists resources available for implementing WITSML v2.0 with the Energistics Transfer Protocol (ETP). Resources are available from the standards download page on the Energistics website: www.energistics.org.

Resource	Description
ETP_v1.1_for_WITSML_v2.0_Implementation_Specification_v1.0_Doc_v1.0.pdf	Specification that explains details for implementing ETP v1.1 for the WITSML v2.0 data model.
ETP v1.1	The Energistics Transfer Protocol (schemas, UML model, specification, etc.) published in November 2016, with the updated specification (document updates in Aug 2017). .
Known Issues Lists (KILs) for ETP Specification v1.1 (updated August 2017)	Testing and development of WITSML v2.0 on ETP v1.1 has revealed some issues which are document on this list.
WITSML v2.0	The WITSML v2.0 download (published in Oct 2016) includes schemas (XSD files) for data objects in the model; a UML model and related documentation.
ETP DevKit	The ETP DevKit is a .NET library providing a common foundation and the basic infrastructure needed to communicate via the Energistics Transfer Protocol (ETP).
Standards DevKit	The Standards DevKit wraps the WITSML, PRODML and RESQML schemas with Microsoft .NET objects. The developer can work directly with these objects rather than having to deal directly with the XML definitions.
Energistics_Identifier_Specification_v4.0_Doc_v1.1.pdf	Specification that states requirements for properly formatting Energistics identifiers which include UUIDs and URIs.

1.2.1 Documentation Conventions

- The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in IETF RFC 2119. (http://www.ietf.org/rfc/rfc2119.txt).
- Terminology: *sender* and *receiver*. For simplicity in this document, where needed for general cases, the word *sender* is used to collectively refer to servers, producers and stores, and the word *receiver* is used to collectively refer to clients, consumers, and customers. The specific terms (server, producer, etc.) are defined in the context of the protocols in which they are used in the *ETP Specification*.
- Protocol abbreviations. Each ETP protocol has both a name and protocol number as defined in the *ETP Specification*. For example the *Core* protocol is also *Protocol 0*. In some cases, this document may further abbreviate protocols, for example, *Protocol 0* may be *P0*.

2 General Rules and Concepts

This chapter lists rules and key concepts that apply across all protocols when using ETP v1.1 to transport WITSML v2.0 data.

2.1 Example Implementation

WITSML defines an interoperability specification about how the data is represented and a standard way of transporting the data electronically. It does not define the storage mechanism.

- **Representation.** WITSML data objects being exchanged between systems are represented as XML documents (NOTE: except for real-time streaming data, which isn't actually a WITSML data object so is not usually sent as an XML document). The data object schemas that comprise the WITSML standard define the formats of these data objects.
- **Transport.** Because XML documents are simply text files, they can be transported by various means, such as email or other file transfer method. A WITSML data object could even be in the form of a printed or faxed document.

  This document specifies a standardized way of electronically transporting WITSML data objects between systems using WebSocket-based protocols. However, companies can send and receive WITSML data streams without using the Energistics Transfer Protocol.
- **Storage.** While WITSML data objects transported between systems are represented as XML documents, the WITSML specification does not dictate how the WITSML data objects are to be stored within those systems, and the receiver application cannot assume any particular storage mechanism. The data objects might be stored as text files or in a database system.

2.2 ETP Specification is the Primary Source

This implementation specification is a companion to the *Energistics Transfer Protocol (ETP) Specification v1.1 Doc v1* and has been designed to be consistent with it. However, if you interpret any discrepancies between these documents, the *ETP Spec* is the primary source.

Please report any such discrepancies by posting a message in BaseCamp or if you do not have access to BaseCamp, send a message to info@Energistics.org.

2.3 All Applications Must Handle Optional Header in XML v1.0

Energistics uses XML version 1.0 and does not plan to update to a later version anytime soon. Section 2.8 of the XML specification (https://www.w3.org/TR/REC-xml/#sec-prolog-dtd) states:

*XML documents **should** begin with an XML declaration which specifies the version of XML being used*.

Example: <?xml version="1.0" encoding="UTF-8" standalone="" ?>

- In XML v1.0, this declaration header is optional. (FYI: In XML version 1.1, this same header was changed to be required.) WITSML-enabled applications must be programmed to work whether or not the optional header is present. Receiver applications must be prepared for either case, and sender applications should follow the W3C guidance and provide it. In the future, the WITSML specification could make it mandatory.
- The standalone parameter is optional.

2.4 Energistics Identifiers

To identify specific instances of a data object, Energistics standards use identifiers as defined in the *Energistics Identifier Specification,* which may be*:*

- A universally unique identifier (UUID), which identifies a *single instance* of a data object.

- NOTE: By definition, each Energistics top-level object (TLO) inherits from AbstractObject (in Energistics *common*), which means it must have a UUID and required metadata. For more information about TLOs, see the *CTA Overview Guide*.

- An Energistics Uniform Resource Identifier (URI), which identifies *one or more instances* of a data object.

NOTE: Identification in previous (pre-v2.0) versions of WITSML was unique only within a server and depended on defined hierarchies required to begin with well/wellbore. Now, with the use of UUIDs, top-level data objects must be universally unique.

2.4.1 WITSML Hierarchies

In pre-v2.0 versions of WITSML, all data objects (except for well) were required to be a child of a wellbore, thereby requiring a fixed hierarchy of well/wellbore/*dataobject* (where *dataobject* is an object such as log, trajectory, report, etc.) to identify an object.

In the v2.0+ WITSML design, use of Energistics identifiers means all domain data objects are now uniquely addressable in WITSML-enabled software. For more information, on hierarchies, URIs, and how they are constructed, see Chapter 5, Discovering the Content of a Store (Protocol 3: Discovery).

Types of Hierarchy	Support Requirements
• Ones composed with the "top" of the hierarchy as any TLO (including the traditional well/wellbore hierarchy). For example: – eml://witsml20/Well(uuid)/Wellbore(uuid) – eml://witsml20/Log(uuid) - eml://witsml20/Rig(uuid)	**Required.** If WITSML-enabled software supports a TLO, then the software must also support related hierarchies with the object at the "top" (as shown in the left cell in this table row).

2.5 Mapping Common Tasks from WITSML v1.x to v2.0

This section provides examples of common tasks done in the context of WITSML workflows and explains how these tasks were done in previous versions of WITSML (i.e., v1.3.1 and v1.4.1.1) and explains how to execute those same tasks in WITSML 2.0 using ETP.

2.5.1 General API Mapping

This table maps SOAP API functions to the ETP protocol and message used to perform the same or similar task.

SOAP API Function (WITSML 1.X)	ETP Protocol (WITSML 2.0)	ETP Message
GetVersion	Core	RequestSession
GetCap	Core	RequestSession
GetBaseMsg	(none)	(none)
GetFromStore (child objects)	Discovery	GetResources
GetFromStore (single object)	Store	GetObject
AddToStore	Store	PutObject
UpdateInStore	Store	PutObject
DeleteFromStore	Store	DeleteObject

NOTE: Streaming of channel data and data management of growing object parts are not covered by the above mapping.

2.5.2 Determine Server Capabilities

This table describes how a client determines a server's capabilities (what objects it supports and functions (protocols) it supports, etc.)

Version	Description of how to do this task
WITSML 1.x	Send a GetVersion request to retrieve a list of WITSML versions supported by the store. Send a GetCap request to retrieve a list of supported WITSML API functions, data objects and other server capability settings.
WITSML 2.0	Use the Core protocol to send a RequestSession message to request support for specific versions of certain protocols along with which MLs, versions, and data objects are supported. The supported protocol capabilities are also returned in the OpenSession message. For more information, see the ETP Specification, Section 2.1.1 Requirements (Protocol Negotiation).

2.5.3 Determine Which Wells a User has Access to

Version	Description of how to do this task
WITSML 1.x	Send a GetFromStore request for wells with a template containing uid and name.
WITSML 2.0	Use Discovery protocol to send a GetResources message with a URI of eml://witsml20/Well. For more information, see Chapter 5.

2.5.4 Determine Which Wellbores a User has Access to

Version	Description of how to do this task
WITSML 1.x	Send a GetFromStore request for wellbores with a template containing uid, uidWell, name and nameWell.
WITSML 2.0	Use Discovery protocol to send a GetResources message with a URI of eml://witsml20/Wellbore. For more information, see Chapter 5.

2.5.5 Determine Which Wellbore Child Objects a User has Access to

Version	Description of how to do this task
WITSML 1.x	Send a GetFromStore request for each child object type, e.g., bhaRun, log, mudLog, etc., with a template containing uid, uidWell, uidWellbore, name, nameWell and nameWellbore.
WITSML 2.0	To recursively discover the supported data object types below a wellbore: use the Discovery protocol, send a GetResources message with the URI of a specific wellbore, e.g. eml://witsml20/Wellbore(uuid).

Version	Description of how to do this task
	When a customer sends a GetResources message with a specific data object's URI, the store returns folder resources for each supported data object type at that level of the object hierarchy. For more information, see Chapter 5.

2.5.6 Retrieve a Static Data Object from the Store

Version	Description of how to do this task
WITSML 1.x	Send a GetFromStore request with a template of the data object to return, specifying the elements to include in the response and including the identifiers.
WITSML 2.0	Use the Store protocol to send a GetObject message with a URI of the data object to return. For more information, see Chapter 6.

2.5.7 Retrieve a Trajectory with Stations or Mud Log from the Store

In WITSML v2.0, mud log is now named "wellbore geology" with intervals.

Version	Description of how to do this task
WITSML 1.x	Send a GetFromStore request with a template of the data object to return, specifying the elements to include in the response and including the identifiers and growing elements, e.g. trajectoryStation or geologyInterval. Trajectory stations and geology intervals are not sent or received in real time, so polling is required.
WITSML 2.0	To retrieve a growing object's header, use the Store protocol to send a GetObject message with a URI of the data object to be returned. Historical growing parts can be retrieved using the GrowingObject protocol by sending a GrowingObjectGet or GrowingObjectGetRange message. To send or receive trajectory stations or geology intervals in real-time, see Section 4.2.

2.5.8 Retrieve a Log from the Store

Version	Description of how to do this task
WITSML 1.x	Send a GetFromStore request with a template of the log to return, specifying the elements to include in the response and including the identifiers and logData element. Log curve data is not sent or received in real time, so polling is required.
WITSML 2.0	To retrieve a log header, use the Store protocol to send a GetObject message with a URI of the log to be returned. To send or receive log channel data, use the Channel Streaming protocol. For more information, see Section 4.1.

2.6 Using Data Objects from Energistics *common*

Beginning with v2.0, WITSML is now implemented with data objects in Energistics *common,* which is part of the Energistics Technical Architecture (CTA). (For more information about the Energistics CTA, see the *CTA Overview Guide.)*

Energistics *common* contains data objects that support vital underpinnings of virtually all upstream technical workflows and functionality that must work consistently across all domains: for example, coordinate reference systems and units of measure.

In addition, Energistics *common* includes a category of objects informally referred to by the Energistics community as 'decorator' objects' because they provide additional, specialized information about another domain data object. Logically, decorator objects have no meaning by themselves; they must reference another top-level domain object (for example, a well, wellbore or survey). So-called decorator objects include:

- **Activity**. A set of objects that provides information about activities and actors (software and/or people) that modify a referenced data object. Its main data objects are:
  - Activity
  - Activity Template
- **Data Assurance Record**. An object that declares conformance with a pre-defined data assurance policy of the referenced data object.
- **Graphical Information.** Set of objects that describe graphical information (line type, weight, color, etc.) of a data object.

This implementation spec provides information about how to handle these objects, as required for various protocols.

3 Making a Connection (Protocol 0: Core)

Protocol Stability Index: 3 – Stable (for more information on stability indexes, see Section 1.1.2).

WITSML v2.0 requires no specific tailoring of ETP; for information on implementing Protocol 0 (P0), see the *ETP Specification*.

Energistics has ETP DevKits available for .NET, Java and JavaScript. For more information, see http://www.energistics.org/standards-portfolio/etp-devkit.

3.1 Example Messages

This section includes some example messages for Protocol 0.

Example RequestSession message:

```
// Opening web socket connection...
// [] Socket opened.
// [] Message sent at 2017-06-16 17:15:34.3673
{"protocol":0,"messageType":1,"correlationId":0,"messageId":1,"messageFlags":0}
{
  "applicationName": "Application.Name",
  "applicationVersion": "1.0.0.0",
  "requestedProtocols": [
    {
      "protocol": 1,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "producer",
      "protocolCapabilities": {}
    },
    {
      "protocol": 3,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    },
    {
      "protocol": 4,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    },
    {
      "protocol": 5,
      "protocolVersion": {
        "major": 1,
```

```
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    },
    {
      "protocol": 6,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    }
  ],
  "supportedObjects": []
}
```

Example OpenSession message in response:

```
// [1eb614ee-ee09-46d2-9c86-c8ee0be0a8bd] Message received at 2017-06-16
17:15:35.0693
{"protocol":0,"messageType":2,"correlationId":1,"messageId":1,"messageFlags":0}
{
  "applicationName": "Application.Name",
  "applicationVersion": "1.0.0.0",
  "sessionId": "1eb614ee-ee09-46d2-9c86-c8ee0be0a8bd",
  "supportedProtocols": [
    {
      "protocol": 1,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "producer",
      "protocolCapabilities": {}
    },
    {
      "protocol": 3,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {
        "maxGetResourcesResponse": {
          "item": 1000.0
        }
      }
```

```
    },
    {
      "protocol": 4,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    },
    {
      "protocol": 5,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    },
    {
      "protocol": 6,
      "protocolVersion": {
        "major": 1,
        "minor": 1,
        "revision": 0,
        "patch": 0
      },
      "role": "store",
      "protocolCapabilities": {}
    }
  ],
  "supportedObjects": [
    "application/x-eml+xml;version=2.1;type=Activity",
    "application/x-eml+xml;version=2.1;type=ActivityTemplate",
    "application/x-eml+xml;version=2.1;type=DataAssuranceRecord",
    "application/x-witsml+xml;version=1.3.1.1;type=log",
    "application/x-witsml+xml;version=1.3.1.1;type=message",
    "application/x-witsml+xml;version=1.3.1.1;type=mudLog",
    "application/x-witsml+xml;version=1.3.1.1;type=rig",
    "application/x-witsml+xml;version=1.3.1.1;type=trajectory",
    "application/x-witsml+xml;version=1.3.1.1;type=wbGeometry",
    "application/x-witsml+xml;version=1.3.1.1;type=well",
    "application/x-witsml+xml;version=1.3.1.1;type=wellbore",
    "application/x-witsml+xml;version=1.4.1.1;type=attachment",
    "application/x-witsml+xml;version=1.4.1.1;type=log",
    "application/x-witsml+xml;version=1.4.1.1;type=message",
    "application/x-witsml+xml;version=1.4.1.1;type=mudLog",
    "application/x-witsml+xml;version=1.4.1.1;type=rig",
    "application/x-witsml+xml;version=1.4.1.1;type=trajectory",
    "application/x-witsml+xml;version=1.4.1.1;type=wbGeometry",
    "application/x-witsml+xml;version=1.4.1.1;type=well",
    "application/x-witsml+xml;version=1.4.1.1;type=wellbore",
    "application/x-witsml+xml;version=2.0;type=Attachment",
    "application/x-witsml+xml;version=2.0;type=Channel",
    "application/x-witsml+xml;version=2.0;type=ChannelSet",
    "application/x-witsml+xml;version=2.0;type=Log",
    "application/x-witsml+xml;version=2.0;type=Rig",
```

```
    "application/x-witsml+xml;version=2.0;type=RigUtilization",
    "application/x-witsml+xml;version=2.0;type=Trajectory",
    "application/x-witsml+xml;version=2.0;type=Well",
    "application/x-witsml+xml;version=2.0;type=Wellbore",
    "application/x-witsml+xml;version=2.0;type=WellCompletion"
  ]
}
```

Example CloseSession message:

```
// [1eb614ee-ee09-46d2-9c86-c8ee0be0a8bd] Message sent at 2017-06-16 17:17:22.9219
{"protocol"0"messageType"5"correlationId"0"messageId"4"messageFlags"0}
{
    "reason": "Session closed"
}
// [1eb614ee-ee09-46d2-9c86-c8ee0be0a8bd] Socket closed.
```

4 Streaming Channel Data (Protocol 1: ChannelStreaming)

Protocol Stability Index: 3 – Stable (for more information on stability indexes, see Section 1.1.2).

Protocol 1 (P1) allows “true” real-time streaming of channel data (i.e., no polling required).

A channel is the measurement of one physical property versus a range of some other dimension, usually time, depth or both. Because data (a new index and its corresponding property) continue to be added, WITSML refers to these as *growing objects*. WITSML v2.0 has several growing objects, including a log (which grows systematically), and trajectories and cuttings intervals (which grow randomly). For more information on the organization of these domain objects, see the WITSML Technical Usage Guide v2.0.

Also, notifications—like alarms, alerts and annotations—which are not explicitly modeled in XML, can be streamed in real time as channels; this functionality replaces the Message object in previous versions of WITSML.

This chapter provides examples describing the basic steps for streaming these objects.

4.1 Overview of Growing Objects and How They Work

Growing objects in WITSML v2.0 consist of a constant part—similar to a header—and a growing part (or a master/detail, if that pattern is more familiar). In WITSML v1411, each growing part is typically not addressable on its own; it exists only in an array of its parent. For example, in the v1411 XML schema, a Trajectory has an unbounded array called TrajectoryStation, whose type is also called TrajectoryStation. There is no way in XML to address a single TrajectoryStation because there is no global element for it.

To overcome this historical limitation of WITSML, in v2.0 the growing parts have their own global elements. These are streamed in real time on a channel as they are produced, eliminating the need for a consumer application to poll a producer application repeatedly to see if any new data is available. Streaming the growing portion is done by subscribing to the main object using Protocol 1 (P1). The growing part objects are transferred as channel data.

NOTE: Any object that is “channel subscribable” (that is, its Resource record has a channelSubcribable flag set to “true” as described in Section 3.3.17.3 in the *ETP Specification*) can be described in Protocol 1.

The header (or master) portion of the growing object MUST be sent as part of the channel metadata, because it applies to the entire set of data, not just a portion. This WITSML XML document is attached as a domainObject to the ChannelMetadata message.

It is not necessary to send a new ChannelMetadata record each time any attribute of the header domain data object is changed in the store. Even though a ChannelDescribe is a subscription to changes, the only changes that should cause the sending of a new ChannelMetadata record is the addition or deletion of a Channel. If the consumer wants an update to the metadata, then the consumer must issue a new ChannelDescribe message.

4.1.1 Getting Historical Channel (Log) Data

To retrieve historical channel data, use ChannelRangeRequest (see Section 4.2.7). When Protocol 2 is available, you can use it to retrieve historical channel data.

4.2 Example: Streaming Log Channel Data

Sections 4.2.1 through 4.2.6 are an example of how to stream a log; the heading names refer to the name of the message used. Section 4.2.7 is an example of how to do a channel change request.

4.2.1 Start Message

To begin a channel streaming session, the consumer must send a Start message to the producer as shown in the example message below, which also uses these example parameters:

Property Name	Example
maxDataItems	50
maxMessageRate [ms]	2000

```
// [91e2682e-5f0a-44bc-86fb-2ba708e78f7f] Message sent at 2017-08-12 10:47:26.8676
{"protocol":1,"messageType":0,"correlationId":0,"messageId":8,"messageFlags":0}
{
  "maxMessageRate": 2000,
  "maxDataItems": 50
}
```

NOTE: After receiving a Start message, a simple-streamer producer immediately sends channel metadata and begins streaming channel data messages to the consumer.

A non-simple producer must wait until it receives ChannelDescribe message(s) to respond with channel metadata and then needs a ChannelStreamingStart message before it can send channel data, as explained in the next sections.

4.2.2 ChannelDescribe Message

To retrieve channel metadata from a non-simple producer, the consumer must send a ChannelDescribe message. Below is an example ChannelDescribe messages for this example URI:

Property Name	Example
uris	[eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-b112e3042339)]

```
// [91e2682e-5f0a-44bc-86fb-2ba708e78f7f] Message sent at 2017-08-12 10:47:33.7457
{"protocol":1,"messageType":1,"correlationId":0,"messageId":9,"messageFlags":0}
{
  "uris": [
    "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-b112e3042339)"
  ]
}
```

4.2.3 ChannelMetadata Message

The producer replies to ChannelDescribe message with the following ChannelMetatdata message; below, see the tables that show mappings for channel and index metadata and the domain object XML. NOTE: As shown in the example, a channel must be unique within the context of a server, so the channel URI requires a UUID. The channel index refers to the type of index for the channel (either time, depth or both) so the channel index URI does not require a UUID.

```
// [91e2682e-5f0a-44bc-86fb-2ba708e78f7f] Message received at 2017-08-12
10:47:59.2218
{"protocol":1,"messageType":2,"correlationId":9,"messageId":25,"messageFlags":3}
{
  "channels":
    {
      "channelUri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/Channel(06dd54c2-1e9d-442c-80da-7b15768f31ce)",
      "channelId": 0,
```

```
      "indexes": [
        {
          "indexType": "Depth",
          "uom": "m",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "MD",
          "description": "MD",
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/ChannelIndex(MD)",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        },
        {
          "indexType": "Time",
          "uom": "s",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "TIME",
          "description": "TIME",
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/ChannelIndex(TIME)",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        }
      ],
      "channelName": "ROP",
      "dataType": "double",
      "uom": "m/h",
      "startIndex": 900,
      "endIndex": 8800,
      "description": "ROP",
      "status": "Active",
      "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
      "source": "MWD",
      "measureClass": "Velocity",
      "uuid": "06dd54c2-1e9d-442c-80da-7b15768f31ce",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/Channel(06dd54c2-1e9d-442c-80da-7b15768f31ce)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
          "name": "ROP",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "06dd54c2-1e9d-442c-80da-7b15768f31ce",
          "lastChanged": 0,
          "objectNotifiable": true
        },
        "contentEncoding": "gzip",
        "data": [ binary data truncated for readability ]
      }
    },
    {
      "channelUri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/Channel(ef87b732-a9c6-44ba-bc72-9f40386fecea)",
```

```
      "channelId": 1,
      "indexes": [
        {
          "indexType": "Depth",
          "uom": "m",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "MD",
          "description": "MD",
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/ChannelIndex(MD)",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        },
        {
          "indexType": "Time",
          "uom": "s",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "TIME",
          "description": "TIME",
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/ChannelIndex(TIME)",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        }
      ],
      "channelName": "HKLD",
      "dataType": "double",
      "uom": "klbf",
      "startIndex": 900,
      "endIndex": 8800,
      "description": "HKLD",
      "status": "Active",
      "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
      "source": "MWD",
      "measureClass": "Force",
      "uuid": "ef87b732-a9c6-44ba-bc72-9f40386fecea",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/Channel(ef87b732-a9c6-44ba-bc72-9f40386fecea)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
          "name": "HKLD",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "ef87b732-a9c6-44ba-bc72-9f40386fecea",
          "lastChanged": 0,
          "objectNotifiable": true
        },
        "contentEncoding": "gzip",
        "data": [ binary data truncated for readability ]
      }
    },
    {
      "channelUri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
```

```
b112e3042339)/Channel(ff9615df-1fb5-45fe-9157-1a7f4ef4e9ae)",
      "channelId": 2,
      "indexes": [
        {
          "indexType": "Depth",
          "uom": "m",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "MD",
          "description": "MD",
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/ChannelIndex(MD)",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        },
        {
          "indexType": "Time",
          "uom": "s",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "TIME",
          "description": "TIME",
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/ChannelIndex(TIME)",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        }
      ],
      "channelName": "GR",
      "dataType": "double",
      "uom": "gAPI",
      "startIndex": 900,
      "endIndex": 8800,
      "description": "GR",
      "status": "Active",
      "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
      "source": "MWD",
      "measureClass": "Gamma_Ray",
      "uuid": "ff9615df-1fb5-45fe-9157-1a7f4ef4e9ae",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/ChannelSet(21b0c388-5a5d-4153-9759-
b112e3042339)/Channel(ff9615df-1fb5-45fe-9157-1a7f4ef4e9ae)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
          "name": "GR",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "ff9615df-1fb5-45fe-9157-1a7f4ef4e9ae",
          "lastChanged": 0,
          "objectNotifiable": true
        },
        "contentEncoding": "gzip",
        "data": [ binary data truncated for readability ]
      }
    }
```

```
  ]
}
```

Channel Metadata Record:

Metadata Property	Channel Property	Example
channelUri	Channel URI	eml://witsml20/Channel(4a403536-ebf2-490f-a2c2-ae3a38713aa6)
channelId	Unique channel identifier, per session	0
channelName	Channel.Citation.Mnemonic	GR
dataType	Channel.DataType	double
uom	Channel.Uom	gAPI
startIndex	Channel.StartIndex.Depth	0.0
endIndex	Channel.EndIndex.Depth	0.0
description	Channel.Citation.Description or Channel.Citation.Title	GR1AX
status	Channel.GrowingStatus	active
contentType	Channel Content Type	application/x-witsml+xml;version=2.0;type=Channel
source	Channel.Citation.Source	ACME Drilling Company
measureClass	Channel.ChannelClass.Title	Gamma_Ray
uuid	Channel.Uuid	4a403536-ebf2-490f-a2c2-ae3a38713aa6
indexes	See Index Metadata Record below	[ ]
domainObject	DataObject with Resource and Data. While this property is optional in ETP, for WITSML v2.0 you must send the domain object. The Channel Metadata message does not have elements for all necessary WITSML elements (for example, AxisDefinitions); therefore, you must always send the domain object.	{ }
customData	n/a	{ }

Index Metadata Record:

Metadata Property	ChannelIndex Property	Example
indexType	Channel.Index.IndexType	ChannelIndexTypes.Depth
uom	Channel.Index.Uom	ft
depthDatum	Channel.Index.DatumReference	KB
direction	Channel.Index.Direction	IndexDirections.Increasing
mnemonic	Channel.Index.Mnemonic	MD

Metadata Property	ChannelIndex Property	Example
description	Channel.Index.Mnemonic	measured depth
scale	n/a	3
uri	n/a	eml://witsml20/Channel(4a403536-ebf2-490f-a2c2-ae3a38713aa6)/MD

Domain Object XML:

```
<Channel schemaVersion="2.0" uuid="4a403536-ebf2-490f-a2c2-ae3a38713aa6"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.energistics.org/energyml/data/witsmlv2">
  <Citation xmlns="http://www.energistics.org/energyml/data/commonv2">
    <Title>GR1AX - 170115-141020-039</Title>
    <Originator>DataGenerator</Originator>
    <Creation>2017-01-15T20:10:20.039Z</Creation>
    <Format>PDS.Witsml, Version=2016.1.0.0, Culture=neutral,
PublicKeyToken=null</Format>
  </Citation>
  <Mnemonic>GR</Mnemonic>
  <DataType>double</DataType>
  <Uom>gAPI</Uom>
  <GrowingStatus>active</GrowingStatus>
  <Source>MWD</Source>
  <TimeDepth>depth</TimeDepth>
  <ChannelClass>
    <ContentType
xmlns="http://www.energistics.org/energyml/data/commonv2">application/x-
eml+xml;version=2.1;type=PropertyKind</ContentType>
    <Title
xmlns="http://www.energistics.org/energyml/data/commonv2">Gamma_Ray</Title>
    <Uuid xmlns="http://www.energistics.org/energyml/data/commonv2">b0512912-439f-
447d-a780-158cc2e46406</Uuid>
  </ChannelClass>
  <StartIndex xsi:type="DepthIndexValue">
    <Depth>0.9</Depth>
  </StartIndex>
  <EndIndex xsi:type="DepthIndexValue">
    <Depth>8.8</Depth>
  </EndIndex>
  <LoggingMethod>MWD</LoggingMethod>
  <Index>
    <IndexType>measured depth</IndexType>
    <Uom>m</Uom>
    <Direction>increasing</Direction>
    <Mnemonic>MD</Mnemonic>
    <DatumReference>MSL</DatumReference>
  </Index>
  <Index>
    <IndexType>date time</IndexType>
    <Uom>s</Uom>
    <Direction>increasing</Direction>
    <Mnemonic>TIME</Mnemonic>
    <DatumReference>MSL</DatumReference>
  </Index>
</Channel>
```

4.2.4 ChannelStreamingStart Message

To begin receiving channel data from a non-simple producer, the consumer must send the ChannelStreamingStart message, with one ChannelStreamingInfo per channel. Below is an example messages based on this example data:

Property Name	Comment	Example
channelId	Unique channel identifier, per session	0
startIndex	null, int or double	0.0
receiveChangeNotification	Subscribe to change notifications	false

```
// [91e2682e-5f0a-44bc-86fb-2ba708e78f7f] Message sent at 2017-08-12 10:57:57.0696
{"protocol":1,"messageType":4,"correlationId":0,"messageId":10,"messageFlags":0}
{
  "channels": [
    {
      "channelId": 0,
      "startIndex": {
        "item": null
      },
      "receiveChangeNotification": false
    },
    {
      "channelId": 1,
      "startIndex": {
        "item": null
      },
      "receiveChangeNotification": false
    },
    {
      "channelId": 2,
      "startIndex": {
        "item": null
      },
      "receiveChangeNotification": false
    }
  ]
}
```

4.2.5 ChannelData Message

The producer then sends channel data as shown in this example ChannelData message:

```
// [91e2682e-5f0a-44bc-86fb-2ba708e78f7f] Message received at 2017-08-12
10:58:17.8405
{"protocol":1,"messageType":3,"correlationId":0,"messageId":26,"messageFlags":1}
{
  "data": [
    {
      "indexes": [
        8800,
        1426614650000000
      ],
      "channelId": 0,
      "value": {
        "item": 0.393856590797127
      },
      "valueAttributes": []
```

```
    },
    {
      "indexes": [
        8800,
        1426614650000000
      ],
      "channelId": 1,
      "value": {
        "item": 0.517663621584728
      },
      "valueAttributes": []
    },
    {
      "indexes": [
        8800,
        1426614650000000
      ],
      "channelId": 2,
      "value": {
        "item": 0.285254846459839
      },
      "valueAttributes": []
    }
  ]
}
```


4.2.6 ChannelStreamingStop Message

To stop/pause streaming for the specified channel IDs, the consumer sends the ChannelStreamingStop message.

Property Name	Example
channels	[0]


```
// [91e2682e-5f0a-44bc-86fb-2ba708e78f7f] Message sent at 2017-08-12 10:58:39.4898
{"protocol":1,"messageType":5,"correlationId":0,"messageId":11,"messageFlags":0}
{
  "channels": [
    0,
    1,
    2
  ]
}
```


4.2.7 ChannelRangeRequest Message

This section contains an example of a ChannelRangeRequest message and a ChannelData response using this example data:

Property Name	Example
channelRanges	An array of ChannelRangeInfo
channelId	[0, 1, 2]
startIndex	0
endIndex	10000

ChannelRangeRequest message:

```
// [3b197bda-c85d-4fc9-ab86-b9c4263407d1] Message sent at 2017-08-12 14:07:00.2222
{"protocol":1,"messageType":9,"correlationId":0,"messageId":7,"messageFlags":0}
{
  "channelRanges": [
    {
      "channelId": [
        0,
        1,
        2
      ],
      "startIndex": 0,
      "endIndex": 10000
    }
  ]
}
```

ChannelData response:

```
// [3b197bda-c85d-4fc9-ab86-b9c4263407d1] Message received at 2017-08-12
14:07:01.7736
{"protocol":1,"messageType":3,"correlationId":7,"messageId":6,"messageFlags":1}
{
  "data": [
    {
      "indexes": [
        900,
        1426614604000000
      ],
      "channelId": 0,
      "value": {
        "item": 0.738779145171297
      },
      "valueAttributes": [
        {
          "attributeId": 0,
          "attributeValue": {
            "item": true
          }
        }
      ]
    },
    {
      "indexes": [
        900,
        1426614604000000
      ],
      "channelId": 1,
      "value": {
        "item": 0.63008946768478
      },
      "valueAttributes": []
    },
    {
      "indexes": [
        900,
        1426614604000000
      ],
```

```
      "channelId": 2,
      "value": {
        "item": 0.189139311755607
      },
      "valueAttributes": []
    }
  ]
}
```

4.3 Example: Streaming Trajectory Stations

This section contains an example of how to stream trajectory stations (part of a trajectory, a randomly growing data object per Section 4.1 above). The heading names refer to the name of the message used. Use this example as a guide for how to stream cuttings intervals, another randomly growing object.

4.3.1 Start Message

To begin a channel streaming session, the consumer sends a Start message to the producer as shown in the example message below, which also uses these example parameters:

Property Name	Example
maxDataItems	50
maxMessageRate [ms]	2000

```
// [df2f6bb0-ce6d-422c-904f-2bac206de113] Message sent at 2017-04-19 00:41:07.7577
{"protocol":1,"messageType":0,"correlationId":0,"messageId":3,"messageFlags":0}
{
  "maxMessageRate": 2000,
  "maxDataItems": 50
}
```

NOTE: After receiving a Start message, a simple-streamer producer immediately sends channel metadata and begins streaming channel data messages to the consumer.

A non-simple producer must wait until it receives ChannelDescribe message(s) to respond with channel metadata and then needs a ChannelStreamingStart message before it can send channel data, as explained in the next sections.

4.3.2 ChannelDescribe Message

To retrieve channel metadata from the non-simple producer, the consumer must send a ChannelDescribe message. Below is an example ChannelDescribe message for this example URI:

Property Name	Example
uris	[eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)]

```
// [460f515b-0491-4c89-b4a2-51de16df9bef] Message sent at 2017-04-19 00:10:39.9393
{"protocol":1,"messageType":1,"correlationId":0,"messageId":4,"messageFlags":0}
{
  "uris": [
    "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)"
```

```
  ]
}
```

4.3.3 ChannelMetadata Message

The producer replies to ChannelDescribe message with the following ChannelMetatdata message; below, see the tables that show mappings for channel and index metadata and the domain object XML.

```
// [460f515b-0491-4c89-b4a2-51de16df9bef] Message received at 2017-04-19
00:10:39.9483
{"protocol":1,"messageType":2,"correlationId":4,"messageId":6,"messageFlags":3}
{
  "channels": [
    {
      "channelUri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-
2a897db562ec)",
      "channelId": 0,
      "indexes": [
        {
          "indexType": "Depth",
          "uom": "ft",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": "MD",
          "description": "measured depth",
          "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-
2a897db562ec)/MD",
          "customData": {},
          "scale": 3,
          "timeDatum": null
        }
      ],
      "channelName": "Plan #2",
      "dataType": "bytes",
      "uom": "ft",
      "startIndex": 0,
      "endIndex": 0,
      "description": "Plan #2",
      "status": "Inactive",
      "contentType": "application/x-
witsml+xml;version=2.0;type=part_TrajectoryStation",
      "source": "",
      "measureClass": "",
      "uuid": "0655e8cd-b590-4f89-9b30-2a897db562ec",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Trajectory",
          "name": "Plan #2",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "0655e8cd-b590-4f89-9b30-2a897db562ec",
          "lastChanged": 1492571586409000,
          "objectNotifiable": true
        },
        "contentEncoding": "gzip",
```

```
        "data": [ binary data truncated for readability ]
      }
    }
  ]
}
```

Channel Metadata Record

Metadata Property	Trajectory Property	Example
channelUri	Trajectory URI	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
channelId	Unique channel identifier, per session	0
channelName	Trajectory.Citation.Title	Plan #2
dataType	The type of the channel data value	bytes
uom	Trajectory.MDMin.Uom	ft
startIndex	Trajectory.MDMin.Value	0.0
endIndex	Trajectory.MDMax.Value	123.456
description	Trajectory.Citation.Description or Trajectory.Citation.Title	Plan #2
status	Trajectory.GrowingStatus	inactive
contentType	Trajectory Station Content Type	application/x-witsml+xml;version=2.0;type=part_TrajectoryStation
source	The tag serviceCompany from the trajectory object.	ACME Drilling Company
measureClass	NA	
uuid	Trajectory.Uuid	0655e8cd-b590-4f89-9b30-2a897db562ec
indexes	See Index Metadata Record below	[ ]
domainObject	DataObject with Resource and Data	{ }
customData	n/a	{ }

Index Metadata Record

Metadata Property	Domain Object Property	Example
indexType	ChannelIndexTypes.Depth	Depth
uom	Trajectory.MdMn.uom	ft
depthDatum	Trajectory.MdMn.datum	KB
direction	IndexDirections.Increasing	Increasing
mnemonic	n/a	MD
description	n/a	measured depth
uri	n/a	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)/MD

Domain Object XML

```
<Trajectory xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/terms/"
schemaVersion="2.0" uuid="0655e8cd-b590-4f89-9b30-2a897db562ec"
xmlns="http://www.energistics.org/energyml/data/witsmlv2">
  <Citation xmlns="http://www.energistics.org/energyml/data/commonv2">
    <Title>Plan #2</Title>
    <Originator>PDS</Originator>
    <Creation>2017-04-19T03:06:37.823Z</Creation>
    <Format>PDS.WITSMLstudio.Desktop, Version=2017.1.0.0 </Format>
    <LastUpdate>2017-04-19T03:13:06.409Z</LastUpdate>
  </Citation>
  <GrowingStatus>inactive</GrowingStatus>
  <DTimTrajStart>2001-10-31T08:15:00.0000000+00:00</DTimTrajStart>
  <DTimTrajEnd>2001-11-03T16:30:00.0000000+00:00</DTimTrajEnd>
  <MdMn uom="ft" datum=”KB”>0</MdMn>
  <MdMx uom="ft" datum=”KB”>0</MdMx>
  <ServiceCompany>Anadrill</ServiceCompany>
  <MagDeclUsed uom="dega">-4.038</MagDeclUsed>
  <AziVertSect uom="dega">82.7</AziVertSect>
  <DispNsVertSectOrig uom="ft">0</DispNsVertSectOrig>
  <DispEwVertSectOrig uom="ft">0</DispEwVertSectOrig>
  <Definitive>true</Definitive>
  <Memory>true</Memory>
  <FinalTraj>true</FinalTraj>
  <AziRef>grid north</AziRef>
  <Wellbore>
    <ContentType
xmlns="http://www.energistics.org/energyml/data/commonv2">application/x-
witsml+xml;version=2.0;type=Wellbore</ContentType>
    <Title xmlns="http://www.energistics.org/energyml/data/commonv2">A-42</Title>
    <Uuid xmlns="http://www.energistics.org/energyml/data/commonv2">d7cb2b2b-b0c1-
48d6-9d8b-5253f4507beb</Uuid>
    <UuidAuthority xmlns="http://www.energistics.org/energyml/data/commonv2">B-
01</UuidAuthority>
  </Wellbore>
</Trajectory>
```

4.3.4 ChannelStreamingStart Message

To begin receiving channel data from a non-simple producer, the consumer must send the ChannelStreamingStart message, with one ChannelStreamingInfo per channel. Below is an example ChannelStreamingStart messages based on this example data:

Property Name	Comment	Example
channelId	Unique channel identifier, per session	0
startIndex	null, int or double	0.0
receiveChangeNotification	Subscribe to change notifications	false

ETP ChannelStreamingStart message:

```
// [df2f6bb0-ce6d-422c-904f-2bac206de113] Message sent at 2017-04-19 00:46:29.3673
{"protocol":1,"messageType":4,"correlationId":0,"messageId":5,"messageFlags":0}
{
  "channels": [
    {
```

```
      "channelId": 0,
      "startIndex": {
        "item": {
          "long": 0
        }
      },
      "receiveChangeNotification": false
    }
  ]
}
```

4.3.5 ChannelData Message

The producer then sends channel data as shown in this example ChannelData message:

```
// [df2f6bb0-ce6d-422c-904f-2bac206de113] Message received at 2017-04-19
00:46:29.8031
{"protocol":1,"messageType":3,"correlationId":0,"messageId":7,"messageFlags":1}
{
  "data": [
    {
      "indexes": [
        123.456
      ],
      "channelId": 0,
      "value": {
        "item": [ binary data truncated for readability ]
      },
      "valueAttributes": []
    }
  ]
}
```

Decoded Data Object XML

```
<part_TrajectoryStation xmlns="http://www.energistics.org/energyml/data/witsmlv2"
xmlns:eml="http://www.energistics.org/energyml/data/commonv2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" uid="34ht5">
    <DTimStn>2001-10-21T08:15:00Z</DTimStn>
      <TypeTrajStation>tie in point</TypeTrajStation>
      <Md uom="ft">123.456</Md>
      <Tvd uom="ft">123.456</Tvd>
      <Incl uom="dega">0</Incl>
      <Azi uom="dega">47.3</Azi>
      <Mtf uom="dega">47.3</Mtf>
      <Gtf uom="dega">0</Gtf>
      <DispNs uom="ft">0</DispNs>
      <DispEw uom="ft">0</DispEw>
      <VertSect uom="ft">0</VertSect>
      <Dls uom="dega/ft">0</Dls>
      <RateTurn uom="dega/ft">0</RateTurn>
      <RateBuild uom="dega/ft">0</RateBuild>
      <MdDelta uom="ft">0</MdDelta>
      <TvdDelta uom="ft">0</TvdDelta>
      <GravTotalUncert uom="m/s2">0</GravTotalUncert>
      <DipAngleUncert uom="dega">0</DipAngleUncert>
      <MagTotalUncert uom="nT">0</MagTotalUncert>
      <GravAccelCorUsed>false</GravAccelCorUsed>
      <MagXAxialCorUsed>false</MagXAxialCorUsed>
      <SagCorUsed>false</SagCorUsed>
      <MagDrlstrCorUsed>false</MagDrlstrCorUsed>
```

```
      <StatusTrajStation>position</StatusTrajStation>
      <Valid>
         <MagTotalFieldCalc uom="nT">51.19</MagTotalFieldCalc>
         <MagDipAngleCalc uom="dega">41.5</MagDipAngleCalc>
         <GravTotalFieldCalc uom="ft/s2">0.999</GravTotalFieldCalc>
      </Valid>
      <MatrixCov>
         <VarianceNN uom="ft2">0.005236</VarianceNN>
         <VarianceNE uom="ft2">0.005236</VarianceNE>
         <VarianceNVert uom="ft2">2.356194</VarianceNVert>
         <VarianceEE uom="ft2">0.005236</VarianceEE>
         <VarianceEVert uom="ft2">0.005236</VarianceEVert>
         <VarianceVertVert uom="ft2">0.785398</VarianceVertVert>
         <BiasN uom="ft">0</BiasN>
         <BiasE uom="ft">0</BiasE>
         <BiasVert uom="ft">0</BiasVert>
      </MatrixCov>
      <Location xsi:type="GeodeticWellLocation" uid="loc-1">
         <Latitude uom="dega">59.7553</Latitude>
         <Longitude uom="dega">1.71347417</Longitude>
      </Location>
      <RawData>
         <GravAxialRaw uom="ft/s2">0.116</GravAxialRaw>
         <GravTran1Raw uom="ft/s2">-0.168</GravTran1Raw>
         <GravTran2Raw uom="ft/s2">-1654</GravTran2Raw>
         <MagAxialRaw uom="nT">22.77</MagAxialRaw>
         <MagTran1Raw uom="nT">22.5</MagTran1Raw>
         <MagTran2Raw uom="nT">27.05</MagTran2Raw>
      </RawData>
      <CorUsed>
         <GravAxialAccelCor uom="ft/s2">0.11</GravAxialAccelCor>
         <GravTran1AccelCor uom="ft/s2">0.14</GravTran1AccelCor>
         <GravTran2AccelCor uom="ft/s2">0.13</GravTran2AccelCor>
         <MagAxialDrlstrCor uom="nT">0.17</MagAxialDrlstrCor>
         <MagTran1DrlstrCor uom="nT">0.16</MagTran1DrlstrCor>
         <MagTran2DrlstrCor uom="nT">0.24</MagTran2DrlstrCor>
         <SagIncCor uom="dega">0</SagIncCor>
         <SagAziCor uom="dega">0</SagAziCor>
         <StnMagDeclUsed uom="dega">-4.038</StnMagDeclUsed>
         <DirSensorOffset uom="ft">48.3</DirSensorOffset>
      </CorUsed>
   </part_TrajectoryStation>
```

4.3.6 ChannelStreamingStop Message

To stop/pause streaming for the specified channel IDs, the consumer sends the ChannelStreamingStop message.

Property Name	Example
channels	[0]

```
// [df2f6bb0-ce6d-422c-904f-2bac206de113] Message sent at 2017-04-19 01:07:09.1561
{"protocol":1,"messageType":5,"correlationId":0,"messageId":8,"messageFlags":0}
{
  "channels": [
    0
  ]
}
```

4.4 Examples: Streaming Alarms, Alerts, and Annotations

P1 can also be used to stream notifications about operations in separate channels. Data consumers subscribe to alarms, alerts, and annotations in essentially the same way as they do to streaming any other data object as described in the *ETP Specification*.

We define these three categories of information:

- **Alarm:** notification that an incident requiring immediate attention is happening now.
- **Alert:** notification that an incident could happen or is imminent (e.g., an alert of an impending storm).
- **Annotation:** additional notation or description that may be interesting or helpful.

To implement these notifications:

- The alarm and alert channels use the DataItem.ValueAttributes dictionary to include additional metadata to further describe those events.
  - Each example shows the required data items, though additional ones can be used.
- Currently, no pre-defined alarm or alert content types or enumerations are specified, so they cannot be auto-discovered.
- Alarm and alert channels must be configured and communicated during data mapping discussions between service providers and data aggregators.
- This capability is a direct replacement for the Message object from previous versions of WITSML.

The examples use a time-based index and string data values containing the descriptions for the alarms, alerts, and annotations. You can specify additional indexes, such as measured depth, as needed.

4.4.1 Start Message and ChannelDescribe Message

As in previous examples above in this chapter, the consumer sends a Start message and ChannelDescribe message to the non-simple producer.

4.4.2 ChannelMetadata Message

For this particular example, the producer replies to the ChannelDescribe message with the following ChannelMetatdata message; below, also see the domain object XML.

```
// [e5245366-2d61-4769-89a1-e36475439f28] Message received at 2017-04-14
14:07:50.1001
{"protocol":1,"messageType":2,"correlationId":3,"messageId":2,"messageFlags":3}
{
  "channels": [
    {
      "channelUri": " eml://witsml20/Channel(cb224581-c2be-4fc6-be5f-91da7904fe7f)",
      "channelId": 1,
      "indexes": [
        {
          "indexType": "Time",
          "uom": "us",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": null,
          "description": "Time sent",
          "uri": null,
          "customData": {},
          "scale": 0,
          "timeDatum": null
        }
      ],
      "channelName": "ALARM",
```

```
      "dataType": "string",
      "uom": "none",
      "startIndex": null,
      "endIndex": null,
      "description": "Alarm",
      "status": "Active",
      "contentType": null,
      "source": "Console",
      "measureClass": "Unitless",
      "uuid": "cb224581-c2be-4fc6-be5f-91da7904fe7f",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/Channel(cb224581-c2be-4fc6-be5f-91da7904fe7f)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
          "name": "ALARM",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "cb224581-c2be-4fc6-be5f-91da7904fe7f",
          "lastChanged": 1492196225411050,
          "objectNotifiable": true
        },
        "contentEncoding": "",
        "data": [ binary data truncated for readability ]
      }
    },
    {
      "channelUri": " eml://witsml20/Channel(503be6e8-47c7-45de-a2e9-af7850301cef)",
      "channelId": 2,
      "indexes": [
        {
          "indexType": "Time",
          "uom": "us",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": null,
          "description": "Time sent",
          "uri": null,
          "customData": {},
          "scale": 0,
          "timeDatum": null
        }
      ],
      "channelName": "ALERT",
      "dataType": "string",
      "uom": "none",
      "startIndex": null,
      "endIndex": null,
      "description": "Alert",
      "status": "Active",
      "contentType": null,
      "source": "Console",
      "measureClass": "Unitless",
      "uuid": "503be6e8-47c7-45de-a2e9-af7850301cef",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/Channel(503be6e8-47c7-45de-a2e9-af7850301cef)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
```

```
          "name": "ALERT",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "503be6e8-47c7-45de-a2e9-af7850301cef",
          "lastChanged": 1492196225431674,
          "objectNotifiable": true
        },
        "contentEncoding": "",
        "data": [ binary data truncated for readability]
      }
    },
    {
      "channelUri": " eml://witsml20/Channel(5579b4a1-994a-415d-b511-39b7b4fe0ee6)",
      "channelId": 3,
      "indexes": [
        {
          "indexType": "Time",
          "uom": "us",
          "depthDatum": null,
          "direction": "Increasing",
          "mnemonic": null,
          "description": "Time sent",
          "uri": null,
          "customData": {},
          "scale": 0,
          "timeDatum": null
        }
      ],
      "channelName": "ANNOT",
      "dataType": "string",
      "uom": "none",
      "startIndex": null,
      "endIndex": null,
      "description": "Annotation",
      "status": "Active",
      "contentType": null,
      "source": "Console",
      "measureClass": "Unitless",
      "uuid": "5579b4a1-994a-415d-b511-39b7b4fe0ee6",
      "customData": {},
      "domainObject": {
        "resource": {
          "uri": "eml://witsml20/Channel(5579b4a1-994a-415d-b511-39b7b4fe0ee6)",
          "contentType": "application/x-witsml+xml;version=2.0;type=Channel",
          "name": "ANNOT",
          "channelSubscribable": true,
          "customData": {},
          "resourceType": "DataObject",
          "hasChildren": 0,
          "uuid": "5579b4a1-994a-415d-b511-39b7b4fe0ee6",
          "lastChanged": 1492196225448718,
          "objectNotifiable": true
        },
        "contentEncoding": "",
        "data": [ binary data truncated for readability ]
      }
    }
  ]
}
```

Example Channel data object decoded as XML:

```
<Channel xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:dc="http://purl.org/dc/terms/" xmlns:gml="http://www.opengis.net/gml/3.2"
schemaVersion="2.0" uuid="cb224581-c2be-4fc6-be5f-91da7904fe7f"
xmlns="http://www.energistics.org/energyml/data/witsmlv2">
  <Citation xmlns="http://www.energistics.org/energyml/data/commonv2">
    <Title>ALARM</Title>
    <Originator>PDS</Originator>
    <Creation>2017-04-14T00:00:00Z</Creation>
    <Format>MessageConsole</Format>
    <LastUpdate>2017-04-14T23:05:06.233893Z</LastUpdate>
    <Description>Alarm</Description>
  </Citation>
  <Mnemonic>ALARM</Mnemonic>
  <DataType>double</DataType>
  <Uom xsi:nil="true" />
  <GrowingStatus>active</GrowingStatus>
  <Wellbore>
    <ContentType
xmlns="http://www.energistics.org/energyml/data/commonv2">application/x-
witsml+xml;version=2.0;type=Wellbore</ContentType>
    <Title xmlns="http://www.energistics.org/energyml/data/commonv2">WB-A1</Title>
    <Uuid xmlns="http://www.energistics.org/energyml/data/commonv2">2cf95836-8d9c-
4b56-a6a0-803d21e12a24</Uuid>
  </Wellbore>
  <TimeDepth>Time</TimeDepth>
  <ChannelClass>
    <ContentType
xmlns="http://www.energistics.org/energyml/data/commonv2">application/x-
eml+xml;version=2.1;type=PropertyKind</ContentType>
    <Title xmlns="http://www.energistics.org/energyml/data/commonv2">none</Title>
    <Uuid xmlns="http://www.energistics.org/energyml/data/commonv2">a818e404-dbc9-
41e0-adc8-455eafc5b8d1</Uuid>
  </ChannelClass>
  <LoggingCompanyName>PDS</LoggingCompanyName>
  <Index>
    <IndexType>date time</IndexType>
    <Uom>us</Uom>
    <Direction>increasing</Direction>
  </Index>
</Channel>
```

4.4.3 Send Alarm Example ChannelData Message

The example here shows the message received by a channel-streaming consumer (in response to the ChannelMetadata message in Section 4.4.2 above) for the alarm “BOP pressure dropped” with the value attributes described in the table below.

These metadata attributes must be included; optionally, additional metadata attributes and values can be specified.

Attribute ID	Attribute Name	Attribute Value
0	Target URI	eml://witsml20/RigUtilization(f781dea4-0255-43d4-a3df-b0be17c25ca8)
1	Severity	5

Example ChannelData message or an alarm:

```
// [e5245366-2d61-4769-89a1-e36475439f28] Message received at 2017-04-14
17:15:02.1977
{"protocol":1,"messageType":3,"correlationId":0,"messageId":4,"messageFlags":1}
{
  "data": [
    {
      "indexes": [
        1492208102193729
      ],
      "channelId": 1,
      "value": {
        "item": "BOP pressure dropped"
      },
      "valueAttributes": [
        {
          "attributeId": 0,
          "attributeValue": {
            "item": "eml://witsml20/RigUtilization(f781dea4-0255-43d4-a3df-
b0be17c25ca8)"
          }
        },
        {
          "attributeId": 1,
          "attributeValue": {
            "item": 5.0
          }
        }
      ]
    }
  ]
}
```

4.4.4 Example Send Alert ChannelData Message

The example here shows the message received by a channel-streaming consumer (in response to the ChannelMetadata message in Section 4.4.2 above) for the alert “Tropical storm warning” with the value Attributes shown in the table.

These metadata attributes must be included; optionally, additional metadata attributes and values can be specified.

Attribute ID	Attribute Name	Attribute Value
0	Target URI	eml://witsml20/RigUtilization(f781dea4-0255-43d4-a3df-b0be17c25ca8)
1	Type	Weather

Example ChannelData message for an alert:

```
// [e5245366-2d61-4769-89a1-e36475439f28] Message received at 2017-04-14
17:21:03.7521
{"protocol":1,"messageType":3,"correlationId":0,"messageId":5,"messageFlags":1}
{
  "data": [
    {
      "indexes": [
```

```
        1492208463740115
      ],
      "channelId": 2,
      "value": {
        "item": "Tropical storm warning"
      },
      "valueAttributes": [
        {
          "attributeId": 0,
          "attributeValue": {
            "item": "eml://witsml20/RigUtilization(f781dea4-0255-43d4-a3df-
b0be17c25ca8)"
          }
        },
        {
          "attributeId": 1,
          "attributeValue": {
            "item": "Weather"
          }
        }
      ]
    }
  ]
}
```

4.4.5 Send Annotation Example ChannelData Message

The example here shows the message received by a channel-streaming consumer ((in response to the ChannelMetadata message in Section 4.4.2 above) for the annotation “Starting run 200” with the value attributes shown in the table.

These metadata attributes must be included; optionally, additional metadata attributes and values can be specified.

Attribute ID	Attribute Name	Attribute Value
0	Target URI	eml://witsml20/Log(987af9a4-0362-8361-a3df-b0be17ab7f0e)

Example ChannelData message for an annotation:

```
// [e5245366-2d61-4769-89a1-e36475439f28] Message received at 2017-04-14
17:25:14.1222
{"protocol":1,"messageType":3,"correlationId":0,"messageId":6,"messageFlags":1}
{
  "data": [
    {
      "indexes": [
        1492208714117192
      ],
      "channelId": 3,
      "value": {
        "item": "Starting run 200"
      },
      "valueAttributes": [
        {
          "attributeId": 0,
          "attributeValue": {
            "item": "eml://witsml20/Log(987af9a4-0362-8361-a3df-b0be17ab7f0e)"
          }
```

```
        }
      ]
    }
  ]
}
```

5 Discovering the Content of a Store (Protocol 3: Discovery)

Protocol Stability Index: 3– Stable (for more information on stability indexes, see Section 1.1.2).

In previous versions of WITSML, each data object (except for well) was required to be a child of a wellbore. In the v2.0+ Energistics CTA, most data objects are now top-level objects (TLO). So objects that are not children of wellbore (e.g., rig) can be accessed directly. For more information, see Section 2.4.1.

This chapter describes Protocol 3 (P3), discovery behavior for stores. Sections 5.1 through 5.4 are replicated from the *ETP Specification* (ETP_Specification_v1.1_Doc_v1.1.pdf, Section 2.3.4)

5.1 Discovering "eml://" Must Return Supported URI Protocol Resources

See an example in Figure 5-1.

- The store must return a folder for each supported version of an ML and each supported version of Energistics *common*. For example:
  - WITSML Store (1.3.1.1)
  - WITSML Store (1.4.1.1)
  - WITSML Store (2.0)
  - Energistics common (2.0)
  - Energistics common (2.1)
- The URI must specify the ML family and its 2-digit version number.
  - eml://witsml20
- The ContentType must not be null. Some examples formats are shown here:
  - For a WITSML v2.0 data object: application/x-witsml+xml;version=2.0
  - For objects in Energistics common: application/x-eml+xml; version=2.1
  - For more information about ContentType and its format, see the *Energistics Identifier Specification*.
- "/" is deprecated; you must use "eml://"
- If the store is aware of data residing in multiple backend data sources, then the store must first return dataspaces:
  - eml://ow_proj_01/witsml20
  - A dataspace can be followed by none or another or a series of dataspaces. To navigate through dataspaces, continue to issue ChannelDescribe messages until you encounter one of the Energistics standards names (i.e., witsml, prodml, resqml or eml) (as in the first item in this list above).
  - You know you've encountered a data model when the store returns one of the Energistics MLs ((i.e., witsml, prodml, resqml or eml).
- The full regular expression is:

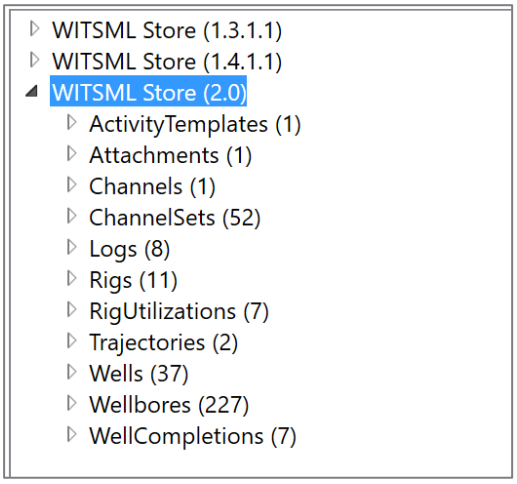
/^eml:\/\/(.*\/)*((witsml|resqml|prodml|eml)([0-9]+))(\/((obj_|part_)?(\w+))(\(([\-\w]+)\))?)?(\?[^#]*)?(#.*)?$/i

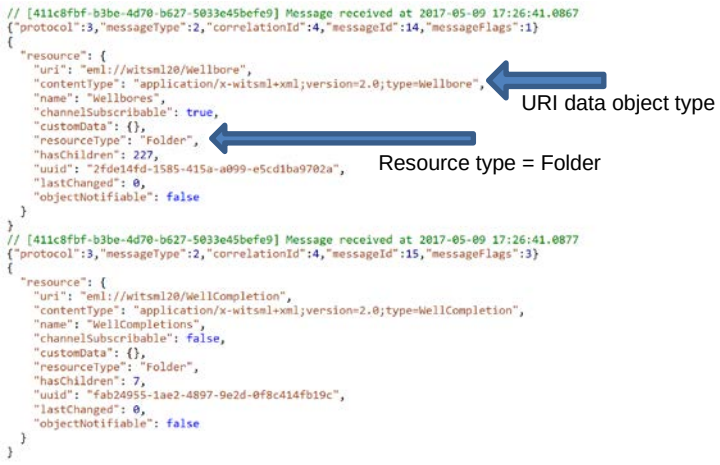
NOTE: This expression includes parameters that may be needed for future ETP capabilities.

Figure 5-1 "eml://" must return a folder for each supported version of an ML and Energistics common.

5.2 Discovering "eml://witsml20" Must Return Folder Resources

- The store must return a folder for each supported top-level data object (TLO) type, for example:

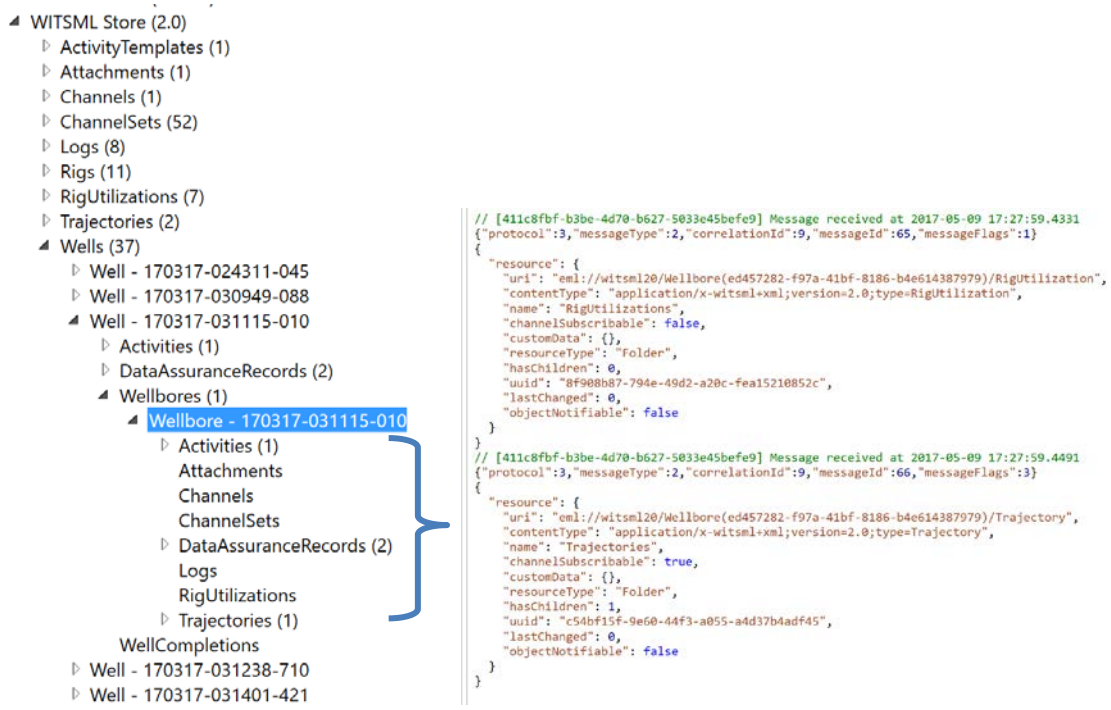




- The URI must specify the data object type but not a UUID.
  - eml://witsml20/Wellbore
- The ContentType must specify the types of data objects that will be returned when discovering the URI.
- Empty folders must be returned.

5.3 Discovering a Data Object URI Must Return Folder Resources

- The store must return a folder for each supported child data object type.
  - Well → Wellbore, WellCompletion
  - Wellbore → MudLog, RigUtilization, Trajectory, etc.



- The store must return a folder for each supported decorator object type, e.g.:
  - Activity
  - GraphicalInformation
  - DataAssuranceRecord
- A URI must specify the child data object type but no UUID.
  - eml://witsml20/Wellbore(uuid)/Log
- The ContentType must specify the type of child data objects that will be returned when discovering the URI.
  - application/x-witsml+xml;version=2.0;type=Well
  - application/x-eml+xml;version=2.1;type=DataAssuranceRecord
- Empty folders must be returned.
- Decorator object type URIs must reference the ML family and version of the object they are associated with.
  - eml://witsml20/Well(uuid_well)/Activity
  - eml://witsml20/Well(uuid_well)/DataAssuranceRecord

5.4 Discovering a Child Data Object Type URI Must Return Data Object Resources

- A data object resource URI must retain the hierarchy used in discovery.
  - eml://witsml20/Well(uuid_well)/Wellbore(uuid_wellbore)/Log(uuid_log)
  - eml://witsml20/Log(uuid_log)/ChannelSet(uuid_channelSet)
- A decorator object URI must reference the ML family and version of the data object it is associated with.
  - eml://witsml20/Well(uuid_well)/Activity(uuid_activity)

  - eml://witsml20/Well(uuid_well)/DataAssuranceRecord(uuid_data_assurance)
- Decorator objects may also be referenced directly.
  - eml://eml21/DataAssuranceRecord(uuid)

5.5 A Store May Limit the Maximum Number of Resources it Returns for Each GetResources Message

- To protect itself, a store may limit the max number of resources it returns for each GetResources message. The Discovery protocol capabilities must include an entry indicating the limit.
- When a store reaches the limit for the number of GetResourcesResponse for a single GetResources message, then a store must send ProtocolException with an error code of EPERMISSION_DENIED (6).

6 Interacting with a Store (Protocol 4: Store)

Protocol Stability Index: 3 – Stable (for more information on stability indexes, see Section 1.1.2).

The Store protocol (P4) is used to perform typical data management operations (i.e., CRUD) and replaces the core functions from previous versions (v1.x) of the WITSML API.

This chapter provides example messages and data object XML.

6.1 Overview of How it Works

In WITSML v1.x, client software was responsible for knowing whether a data object existed in the store, before executing an AddToStore or UpdateInStore request. With ETP, P4 provides a PutObject message, which is treated as an "upsert" operation; this means: if the data object does not exist, it will be created in the store, otherwise it will be replaced.

Partial updates are not supported; therefore, PutObject messages must contain complete data object definitions. Upsert operations perform a full replace of the static object or header portion of a growing object. All required elements must be specified. Server-managed elements are ignored. Receiver-immutable element behavior is honored; for a list of the elements considered to be immutable, see Chapter 9.

6.2 PutObject

To add/update a single data object in a store, send a PutObject message.

The Resource object's content type and UUID properties are critical in identifying the format and type of data object specified in the data property. Do not use the URI to determine any relationships or data object hierarchy. Any data object references must be contained within the data object itself.

For more information about properties and URIs, see:

- *ETP Specification* (Store Protocol (P4))
- *Energistics Identifier Specification* (Chapter 5 and Chapter 6)

The examples below use the following properties.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
uuid	0655e8cd-b590-4f89-9b30-2a897db562ec
name	Plan #2
contentType	application/x-witsml+xml;version=2.0;type=Trajectory

Data object XML:

```
<Trajectory xmlns="http://www.energistics.org/energyml/data/witsmlv2"
            xmlns:eml="http://www.energistics.org/energyml/data/commonv2"
            uuid="0655e8cd-b590-4f89-9b30-2a897db562ec" schemaVersion="2.0">
  <eml:Citation>
    <eml:Title>Plan #2</eml:Title>
    <eml:Originator>PDS</eml:Originator>
    <eml:Format>PDS.WITSMLstudio.Desktop, Version=2017.1.0.0</eml:Format>
  </eml:Citation>
  <DTimTrajStart>2001-10-31T08:15:00.0000000+00:00</DTimTrajStart>
  <DTimTrajEnd>2001-11-03T16:30:00.0000000+00:00</DTimTrajEnd>
  <ServiceCompany>Anadrill</ServiceCompany>
  <MagDeclUsed uom="dega">-4.038</MagDeclUsed>
  <AziVertSect uom="dega">82.7</AziVertSect>
```

```
  <DispNsVertSectOrig uom="ft">0</DispNsVertSectOrig>
  <DispEwVertSectOrig uom="ft">0</DispEwVertSectOrig>
  <Definitive>true</Definitive>
  <Memory>true</Memory>
  <FinalTraj>true</FinalTraj>
  <AziRef>grid north</AziRef>
  <Wellbore>
    <eml:ContentType>application/x-
witsml+xml;version=2.0;type=Wellbore</eml:ContentType>
    <eml:Title>A-42</eml:Title>
    <eml:Uuid>d7cb2b2b-b0c1-48d6-9d8b-5253f4507beb</eml:Uuid>
    <eml:UuidAuthority>B-01</eml:UuidAuthority>
  </Wellbore>
</Trajectory>
```

Example PutObject message followed by the Acknowledge message reply

```
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message sent at 2017-04-18 20:32:21.5683
{"protocol":4,"messageType":2,"correlationId":0,"messageId":3,"messageFlags":0}
{
  "dataObject": {
    "resource": {
      "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
      "contentType": "application/x-witsml+xml;version=2.0;type=Trajectory",
      "name": "Plan #2",
      "channelSubscribable": false,
      "customData": {},
      "resourceType": "DataObject",
      "hasChildren": -1,
      "uuid": "0655e8cd-b590-4f89-9b30-2a897db562ec",
      "lastChanged": 0,
      "objectNotifiable": false
    },
    "contentEncoding": "gzip",
    "data": [ binary data truncated for readability ]
  }
}
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message received at 2017-04-18
20:32:22.4184
{"protocol":4,"messageType":1001,"correlationId":3,"messageId":6,"messageFlags":0}
{}
```

6.3 GetObject

To retrieve a single data object from a store, send a GetObject message.

Example GetObject message with the Object message reply:

```
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message sent at 2017-04-18 20:37:15.9653
{"protocol":4,"messageType":1,"correlationId":0,"messageId":4,"messageFlags":3}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)"
}
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message received at 2017-04-18
20:37:16.0225
{"protocol":4,"messageType":4,"correlationId":4,"messageId":7,"messageFlags":3}
{
  "dataObject": {
    "resource": {
      "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
```

```
      "contentType": "application/x-witsml+xml;version=2.0;type=Trajectory",
      "name": "Plan #2",
      "channelSubscribable": true,
      "customData": {},
      "resourceType": "DataObject",
      "hasChildren": -1,
      "uuid": "0655e8cd-b590-4f89-9b30-2a897db562ec",
      "lastChanged": 1492565542132000,
      "objectNotifiable": true
    },
    "contentEncoding": "gzip",
    "data": [ binary data truncated for readability ]
  }
}
```


Decoded data object XML:


```
<Trajectory xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:dc="http://purl.org/dc/terms/"
schemaVersion="2.0" uuid="0655e8cd-b590-4f89-9b30-2a897db562ec"
xmlns="http://www.energistics.org/energyml/data/witsmlv2">
  <Citation xmlns="http://www.energistics.org/energyml/data/commonv2">
    <Title>Plan #2</Title>
    <Originator>PDS</Originator>
    <Creation>2017-04-19T01:32:22.131Z</Creation>
    <Format>PDS.WITSMLstudio.Desktop, Version=2017.1.0.0 </Format>
    <LastUpdate>2017-04-19T01:32:22.132Z</LastUpdate>
  </Citation>
  <GrowingStatus>inactive</GrowingStatus>
  <DTimTrajStart>2001-10-31T08:15:00.0000000+00:00</DTimTrajStart>
  <DTimTrajEnd>2001-11-03T16:30:00.0000000+00:00</DTimTrajEnd>
  <ServiceCompany>Anadrill</ServiceCompany>
  <MagDeclUsed uom="dega">-4.038</MagDeclUsed>
  <AziVertSect uom="dega">82.7</AziVertSect>
  <DispNsVertSectOrig uom="ft">0</DispNsVertSectOrig>
  <DispEwVertSectOrig uom="ft">0</DispEwVertSectOrig>
  <Definitive>true</Definitive>
  <Memory>true</Memory>
  <FinalTraj>true</FinalTraj>
  <AziRef>grid north</AziRef>
  <Wellbore>
    <ContentType
xmlns="http://www.energistics.org/energyml/data/commonv2">application/x-
witsml+xml;version=2.0;type=Wellbore</ContentType>
    <Title xmlns="http://www.energistics.org/energyml/data/commonv2">A-42</Title>
    <Uuid xmlns="http://www.energistics.org/energyml/data/commonv2">d7cb2b2b-b0c1-
48d6-9d8b-5253f4507beb</Uuid>
    <UuidAuthority xmlns="http://www.energistics.org/energyml/data/commonv2">B-
01</UuidAuthority>
  </Wellbore>
</Trajectory>
```


6.3.1 Getting a Historical Log

To get a historical log, use Protocol 4 to retrieve the metadata for logs. Based on the returned channel uuids and index ranges of that operation, use ChannelRangeRequest in Protocol 1 (see Section 4.2.7) to get the historical data for each channel.

6.4 DeleteObject

To delete a single data object from a store, send a DeleteObject message. The store must return either an Acknowledge message or an Exception.

Example DeleteObject message and the Acknowledge message reply:

```
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message sent at 2017-04-18 20:42:10.5635
{"protocol":4,"messageType":3,"correlationId":0,"messageId":5,"messageFlags":0}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)"
}
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message received at 2017-04-18
20:42:10.6722
{"protocol":4,"messageType":1001,"correlationId":5,"messageId":8,"messageFlags":0}
{}
```

7 Subscribing to Change Notices (Protocol 5: StoreNotification)

Protocol Stability Index: 2 – Unstable (for more information on stability indexes, see Section 1.1.2).

StoreNotification (P5) is used to notify customers when event-driven changes occur to objects in the store. Customers subscribe and have the option to receive notification only or notifications and the changed object (by checking the includeObjectData flag).

Customer applications subscribe to receive change notifications on specific data objects.

P5 replaces and extends the functionality provided by the changeLog object in previous versions of WITSML.

NOTE: Channel data and growing object parts are not included in change notifications. Subscriptions to those objects are made with the Protocol 1 ChannelDescribe message. For more information, see Section 4.3.2.

NOTE: Any object that is “object notifiable” (that is, its Resource record has an objectNotifiable flag set to “true” as described in Section 3.3.17.3 in the *ETP Specification*) can be described in Protocol 5.

7.1 RequestNotification

To subscribe to change notifications, send the RequestNotification message.

The examples below use the following properties and example data.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
uuid	0655e8cd-b590-4f89-9b30-2a897db562ec
startTime	1492566496661602
includeObjectData	true When this flag is true, the store sends both the notification and the updated data object.
objectTypes	(none)

Example NotificationRequest message:

```
// [5c7c2f28-4332-4c0f-acf5-a14aad678f3a] Message sent at 2017-04-18 20:48:21.0475
{"protocol":5,"messageType":1,"correlationId":0,"messageId":6,"messageFlags":0}
{
  "request": {
    "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
    "uuid": "23045b53-ffc5-44b6-bc69-5b9a628cb874",
    "includeObjectData": true,
    "startTime": 1492566496661602,
    "objectTypes": []
  }
}
```

When a data object is changed (e.g., upsert or delete), the subscriber receives notification messages (and the updated data object, if the includeObjectData flag is true).

Example upsert ChangeNotification message:

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message received at 2017-04-18
20:58:50.4440
{"protocol":5,"messageType":2,"correlationId":3,"messageId":7,"messageFlags":0}
{
  "change": {
    "changeType": "Upsert",
    "changeTime": 1492567130384390,
    "dataObject": {
      "resource": {
        "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
        "contentType": "application/x-witsml+xml;version=2.0;type=Trajectory",
        "name": "Plan #2",
        "channelSubscribable": true,
        "customData": {},
        "resourceType": "DataObject",
        "hasChildren": -1,
        "uuid": "0655e8cd-b590-4f89-9b30-2a897db562ec",
        "lastChanged": 1492567128810000,
        "objectNotifiable": true
      },
      "contentEncoding": "gzip",
      "data": [ binary data truncated for readability ]
    }
  }
}
```

Example DeleteNotification message:

NOTE: Per Sect 3.4.6.3 of the *ETP Specification*, for deleted objects, the object data is not present irrespective of the value of the includeObjectData field in the original request.

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message received at 2017-04-18
20:59:53.6191
{"protocol":5,"messageType":3,"correlationId":3,"messageId":9,"messageFlags":0}
{
  "delete": {
    "changeType": "Delete",
    "changeTime": 1492567193611638,
    "dataObject": {
      "resource": {
        "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
        "contentType": "application/x-witsml+xml;version=2.0;type=Trajectory",
        "name": "Plan #2",
        "channelSubscribable": true,
        "customData": {},
        "resourceType": "DataObject",
        "hasChildren": -1,
        "uuid": "0655e8cd-b590-4f89-9b30-2a897db562ec",
        "lastChanged": 1492567128810000,
        "objectNotifiable": true
      }    }
  }
}
```

7.2 CancelNotification

To stop receiving change notifications for a data object, send a CancelNotification message.

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message sent at 2017-04-18 21:02:17.3312
{"protocol":5,"messageType":4,"correlationId":0,"messageId":6,"messageFlags":0}
{
  "requestUuid": "23045b53-ffc5-44b6-bc69-5b9a628cb874"
}
```

8 Managing the Growing Part of an Object (Protocol 6: GrowingObject)

Protocol Stability Index: 2 – Unstable (for more information on stability indexes, see Section 1.1.2).

The Growing Object protocol (P6) is used to request the historical portion of a growing object, with the exception of WITSML Log, ChannelSet or Channel data objects. Single growing parts or a range of growing parts can be requested or deleted, but only one growing part at a time can be added to a store. Unlike the Store protocol (P4)—where content encoding can be either an empty string or gzip—the data object parts transferred using P6 are not compressed and the content encoding must be text/xml.

8.1 GrowingObjectPut

To add/update a single part for a growing object in a store, send GrowingObjectPut message.

The examples below use the following properties and example data.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
contentType	application/x-witsml+xml;version=2.0;type=part_TrajectoryStation
contentEncoding	text/xml

Data object XML:

```
<part_TrajectoryStation xmlns="http://www.energistics.org/energyml/data/witsmlv2"
                        xmlns:eml="http://www.energistics.org/energyml/data/commonv2"
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        uid="34ht5">
    <DTimStn>2001-10-21T08:15:00Z</DTimStn>
      <TypeTrajStation>tie in point</TypeTrajStation>
      <Md uom="ft">0</Md>
      <Tvd uom="ft">0</Tvd>
      <Incl uom="dega">0</Incl>
      <Azi uom="dega">47.3</Azi>
      <Mtf uom="dega">47.3</Mtf>
      <Gtf uom="dega">0</Gtf>
      <DispNs uom="ft">0</DispNs>
      <DispEw uom="ft">0</DispEw>
      <VertSect uom="ft">0</VertSect>
      <Dls uom="dega/ft">0</Dls>
      <RateTurn uom="dega/ft">0</RateTurn>
      <RateBuild uom="dega/ft">0</RateBuild>
      <MdDelta uom="ft">0</MdDelta>
      <TvdDelta uom="ft">0</TvdDelta>
      <GravTotalUncert uom="m/s2">0</GravTotalUncert>
      <DipAngleUncert uom="dega">0</DipAngleUncert>
      <MagTotalUncert uom="nT">0</MagTotalUncert>
      <GravAccelCorUsed>false</GravAccelCorUsed>
      <MagXAxialCorUsed>false</MagXAxialCorUsed>
      <SagCorUsed>false</SagCorUsed>
      <MagDrlstrCorUsed>false</MagDrlstrCorUsed>
      <StatusTrajStation>position</StatusTrajStation>
      <Valid>
         <MagTotalFieldCalc uom="nT">51.19</MagTotalFieldCalc>
         <MagDipAngleCalc uom="dega">41.5</MagDipAngleCalc>
         <GravTotalFieldCalc uom="ft/s2">0.999</GravTotalFieldCalc>
```

```
      </Valid>
      <MatrixCov>
         <VarianceNN uom="ft2">0.005236</VarianceNN>
         <VarianceNE uom="ft2">0.005236</VarianceNE>
         <VarianceNVert uom="ft2">2.356194</VarianceNVert>
         <VarianceEE uom="ft2">0.005236</VarianceEE>
         <VarianceEVert uom="ft2">0.005236</VarianceEVert>
         <VarianceVertVert uom="ft2">0.785398</VarianceVertVert>
         <BiasN uom="ft">0</BiasN>
         <BiasE uom="ft">0</BiasE>
         <BiasVert uom="ft">0</BiasVert>
      </MatrixCov>
      <Location xsi:type="GeodeticWellLocation" uid="loc-1">
         <Latitude uom="dega">59.7553</Latitude>
         <Longitude uom="dega">1.71347417</Longitude>
      </Location>
      <RawData>
         <GravAxialRaw uom="ft/s2">0.116</GravAxialRaw>
         <GravTran1Raw uom="ft/s2">-0.168</GravTran1Raw>
         <GravTran2Raw uom="ft/s2">-1654</GravTran2Raw>
         <MagAxialRaw uom="nT">22.77</MagAxialRaw>
         <MagTran1Raw uom="nT">22.5</MagTran1Raw>
         <MagTran2Raw uom="nT">27.05</MagTran2Raw>
      </RawData>
      <CorUsed>
         <GravAxialAccelCor uom="ft/s2">0.11</GravAxialAccelCor>
         <GravTran1AccelCor uom="ft/s2">0.14</GravTran1AccelCor>
         <GravTran2AccelCor uom="ft/s2">0.13</GravTran2AccelCor>
         <MagAxialDrlstrCor uom="nT">0.17</MagAxialDrlstrCor>
         <MagTran1DrlstrCor uom="nT">0.16</MagTran1DrlstrCor>
         <MagTran2DrlstrCor uom="nT">0.24</MagTran2DrlstrCor>
         <SagIncCor uom="dega">0</SagIncCor>
         <SagAziCor uom="dega">0</SagAziCor>
         <StnMagDeclUsed uom="dega">-4.038</StnMagDeclUsed>
         <DirSensorOffset uom="ft">48.3</DirSensorOffset>
      </CorUsed>
   </part_TrajectoryStation>
```

Example GrowingObjectPut message:

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message sent at 2017-04-18 22:13:05.8780
{"protocol":6,"messageType":5,"correlationId":0,"messageId":8,"messageFlags":0}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
  "contentType": "application/x-witsml+xml;version=2.0;type=part_TrajectoryStation",
  "contentEncoding": "text/xml",
  "data": [ binary data truncated for readability ]
}
```

8.2 GrowingObjectGet

To retrieve a single part of a growing object in a store, send a GrowingObjectGet message. The example below use the following example data.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
uid	34ht5

Example GrowingObjectGet message:

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message sent at 2017-04-18 22:19:38.1476
{"protocol":6,"messageType":3,"correlationId":0,"messageId":9,"messageFlags":0}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
  "uid": "34ht5"
}
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message received at 2017-04-18
22:19:38.1777
{"protocol":6,"messageType":6,"correlationId":9,"messageId":11,"messageFlags":3}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
  "contentType": "application/x-witsml+xml;version=2.0;type=part_TrajectoryStation",
  "contentEncoding": "text/xml",
  "data": [ binary data truncated for readability ]
}
```

8.3 GrowingObjectGetRange

To retrieve a range of parts for a growing object in a store, send GrowingObjectGetRange. The example below uses the following properties and example data. The reply from the store isn't shown.

The elevation reference for the range request must match the elevation reference of the object in the store. If not, the error ENOTSUPPORTED error code 7 must be raised.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
startIndex	0.0
endIndex	1000.0
uom	ft
depthDatum	KB

Example GrowingObjectGetRange message:

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message sent at 2017-04-18 22:20:52.1109
{"protocol":6,"messageType":4,"correlationId":0,"messageId":10,"messageFlags":0}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
  "startIndex": {
    "item": 0.0
  },
  "endIndex": {
    "item": 1000.0
  },
  "uom": "ft",
  "depthDatum": "KB"
}
```

8.4 GrowingObjectDelete

To delete a single part for a growing object in a store, a customer must send a GrowingObjectDelete message. The example below use the following example data.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
uid	34ht5

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message sent at 2017-04-18 22:23:43.9395
{"protocol":6,"messageType":1,"correlationId":0,"messageId":11,"messageFlags":0}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
  "uid": "34ht5"
}
```

8.5 GrowingObjectDeleteRange

To delete a range of parts from a growing object in a store, a customer must send a GrowingObjectDeleteRange message. The example below use the following properties and example data.

Property Name	Example
uri	eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)
startIndex	0.0
endIndex	1000.0
uom	ft
depthDatum	KB

```
// [848b3bb0-76b4-4707-b38d-868dd0b4dadc] Message sent at 2017-04-18 22:20:52.1109
{"protocol":6,"messageType":2,"correlationId":0,"messageId":12,"messageFlags":0}
{
  "uri": "eml://witsml20/Trajectory(0655e8cd-b590-4f89-9b30-2a897db562ec)",
  "startIndex": {
    "item": 0.0
  },
  "endIndex": {
    "item": 1000.0
  },
  "uom": "ft",
  "depthDatum": "KB"
}
```

9 Receiver-Immutable Fields

A number of elements/attributes must be immutable, i.e., after the object is created the fields cannot be changed. If a receiver (i.e., client, consumer, or customer) tries to change an immutable item, the sender (i.e., server, producer or store) must send an EPERMISSION_DENIED error.

This section list these immutable fields.

For all top-level objects:

- uuid

For Citation:

- creation
- lastUpdate
- originator

For Channel:

- mnemonic
- dataType
- uom
- index

For ChannelIndex:

- indexType
- uom
- direction
- mnemonic