ETP v1.1 for WITSML v1.4.1.1 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 domain standards.
Version	1.0
Abstract	This implementation specification • Explains how to transfer WITSML v1.4.1.1 data using ETP v1.1. • Must be used with the *ETP Specification v1.1_Docv1.1*.
Prepared by	Energistics and the WITSML SIG
Date published	16 October 2017
Document type	Implementation Specification
Keywords:	standards, energy, data, information, process, protocol

Acknowledgements

Energistics would like to thank members of the WITSML Special Interest Group who led the effort to produce this document. Team members of this effort include these companies: Baker Hughes, Halliburton, Kongsberg, PDS, and Petrolink.

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

The Energistics Transport Protocol (ETP) is a data-exchange specification that enables the efficient transfer of data between applications and systems. Among its capabilities, ETP 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—thereby reducing network traffic and latency.

ETP was initially developed as the API for WITSML v2.0+ to replace the SOAP API used through WITSML v1.4.1.1. Though ETP was not written for the prior generation of WITSML, its simplicity and performance gains are sufficiently compelling that the community asked to have the option to use ETP for the real-time portion of prior versions of WITSML, while continuing to use the legacy SOAP API for interacting with the store for objects which are not systematically growing. That is, the use of ETP can replace the WITSML 1.4.1.1 (and earlier) log object for real-time data transmission and the legacy store methods (WMLS_AddToStore, WMLS_GetFromStore, etc.) can continue to be used for all other data types.

This document:

- Provides the ML-specific implementation details for using ETP v1.1 to transfer WITSML v1.4.1.1 log data. (The goal is to develop an ETP implementation specification for each supported version of each Energistics domain standard.)
- Is a companion implementation specification to the *ETP Specification v1.1_Doc v1.1,* which defines the required behavior of all ETP protocols.
- Is normative for WITSML v1.4.1.1.

To use ETP v1.1 to send WITSML v1.4.1.1 data, you need:

- The ETP v1.1 (schemas) and the *ETP Specification v1.1 Doc v1.1*.
- This implementation specification. Sections in this document reference appropriate sections in the above-mentioned version of the *ETP Specification*.)
- An existing WITSML v1.4.1.1 server.

NOTE: New WITSML implementations should use WITSML v2.0. For more information, download the WITSML v2.0 standard and see the *ETP v1.1 for WITSML v2.0 Implementation Specification*.

1.1 Audience, Purpose and Scope

This document is intended for information technology professionals (e.g., software developers, programmers, etc.) who want to use the Energistics Transfer Protocol (ETP) v1.1 to stream log data defined by WITSML v1.4.1.1. This document:

- Explains how to map WITSML v1.4.1.1 log data to objects defined by the *ETP Specification v1.1* for streaming.
- Does not explain how to map ETP, in general, to a WITSML v1.4.1.1 log object.

Assumption: The audience has a good understanding of WITSML v1.4.1.1 and is familiar with the *ETP Specification* and related concepts and terminology.

1.1.1 ETP Protocols in Scope

To stream WITSML v1.4.1.1 log data, you must use these ETP protocols.

Protocol Number and Name	Description	Stability Index
Protocol 0: Core	Creates and manages ETP sessions. Observes standard ETP behavior. No ML-specific tailoring required.	**3 - Stable**
Protocol 1: ChannelStreaming **(Log data only in this version)**	Allows streaming WITSML 1.4.1.1 log data over the channel streaming protocol. For additional information required to stream WITSMLv1.4.1.1 log data (i.e., information not defined in the *ETP Specification*), see Chapter 3.	**3 - Stable**

NOTES:

1. For this version of this *Implementation Specification*, all other behaviors not described in this document (e.g., discovery, retrieving data from a store, etc.) operate as defined in the original *WITSML Store Application Program Interface (API) v1.4.1* specification. ETP has additional protocols (e.g., Discovery and Store protocols) which duplicate this legacy functionality and can also be implemented for a WITSML v1.4.1.1 server. Guidance on how to use these other protocols for WITSML v1.4.1.1 may be provided in a future version of this document.
2. While this document does not include specific details for implementing WITSML v1.3.1 on ETP, someone with a good understanding of v1.3.1 and the information in this guide can use it to implement v1.3.1 over ETP. This document is not normative for WITSML v1.3.1.
3. In the future, additional v1.4.1.1 data objects may be supported for transport with ETP. However, the Change Log and Message objects will not be supported because ETP has inherent capabilities that make these data objects obsolete.

1.2 Resource Set

The table below lists resources available for using ETP with WITSML v1.4.1.1. Resources are available from the standards download page on the Energistics website: www.energistics.org.

Resource	Description
ETP_v1.1_for_WITSML_v1.4.1.1_Implementation_Specification_v1.0_Doc_v1.0.pdf	Specification that explains details for implementing ETP v1.1 (Protocols 0 and 1) for logs in the WITSML v1.4.1.1 data model.
ETP v1.1	The complete Energistics Transfer Protocol published in November 2016, with the updated specification.
Existing WITSML v1.4.1.1 server	As defined by the *WITSML Store Application Program Interface (API) v1.4.1.1* and related data object schemas.
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 RFC 2119. (http://www.ietf.org/rfc/rfc2119.txt).

1.3 Overview of Supported Use Cases

This specification supports the use cases listed below and detailed in Appendix A. While the content of the use cases is non-normative, the flow sequences provide useful information for using ETP with WITSML v1.4.1.1; it is recommended that these flows be supported.

The use cases support real-time streaming:

- From sensor to an aggregating server (Section A.1)
- From a wellsite (service company) to an aggregating server (Section A.2)
- From an aggregating server to a display client (Section A.3)

1.4 Acronyms and Terminology

All terminology is defined in the *ETP Specification* and the *Energistics Identifier Specification*. However, some key terms are repeated here for convenience.

Term	Description
ETP	Energistics Transfer Protocol
JSON	JavaScript Object Notation. A text-based, lightweight data interchange format. http://www.json.org/. ETP supports binary and JSON encoding of data for transport.
simple streamer	A data provider with limited capabilities; for example, a simple streamer does not store historical data and may not be able to provide well and wellbore information.
URI	Uniform resource identifier. A string of characters used to identify a resource, often at a location on the Internet.
UUID	Universally unique identifier as defined by RFC 4122 (https://tools.ietf.org/html/rfc4122). Internally represented by a 128-bit integer and usually represented as a 36 character sequence such as: 550e8400-e29b-41d4-a716-446655440000.

2 Rules and Concepts

This chapter lists rules and key concepts that apply across all protocols when using ETP v1.1 to transport WITSML v1.4.1.1 data.

2.1 ETP Specification is the Primary Source

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

2.2 Use and Construction of Identifiers

The identifiers used in ETP are defined in the *Energistics Identifier Specification*. Depending on the context, an identifier in ETP (and in WITSML v2.0) could be an Energistics UUID or an Energistics URI.

In WITSML v1.4.1.1, data object identifiers are compound natural identifiers based on the names of the objects and their mandatory parent well/wellbore hierarchies. For example, the identifier of a bhaRun is the combination of its nameWell, nameWellbore and name (i.e., the name assigned to the bhaRun). WITSML v1.4.1.1 also has optional uidWell, uidWellbore and uid fields which may or may not be populated, depending on the context. These name and uid fields are unique only within the scope of their parent objects.

In ETP and WITSML v2.0 a bhaRun is identified by a simple UUID (formatted like a Microsoft Registry Format GUID). This UUID is globally unique and does not depend on any hierarchy whatsoever.

Real-time data in WITSML v1.4.1.1 is carried in the log object as log curves. Log curves in WITSML v1.4.1.1 are not data objects in their own right—they exist only within the context of a log. So the compound natural identifier of a log curve is nameWell, nameWellbore, name of the log, and its mnemonic. Like other items in WITSML v1.4.1.1, it has an optional uid attribute.

In WITSML v2.0 a LogChannel is a top level object and has a UUID.

In an ETP real-time data stream, a channel is identified by a URI, which incorporates the UUID of the LogChannel data object. This URI could include any arbitrary (valid) hierarchy or no hierarchy at all, so long as the final portion contains an object type name followed by the UUID in parentheses.

However, for WITSML v1.4.1.1 with ETP, the object type must be logCurveInfo, even though that is not a true data object (global element), like this:

```
eml://witsml14/well(UUID)/wellbore(UUID)/log(UUID)/logCurveInfo(UUID)
```

For more information on the construction of an Energistics URI, see the *Energistics Identifier Specification.*

The next question is how to map the ETP UUIDs of the parent hierarchy and logCurveInfo into WITSML v1.4.1.1. For purposes of the use of ETP with WITSML v1.4.1.1 the following rules apply:

1. The optional uid attributes in WITSML v1.4.1.1 are mandatory for use as a UUID with ETP and each must be populated with a UID. This specifically includes uidWell, uidWelllbore and the uid attribute of the log object.
2. The already-mandatory mnemonic element within logCurveInfo is considered the UUID of the ETP channel, in spite of the fact that this is not an Energistics UUID. The uid field within logCurveInfo is NOT used as the UUID.

A URI for WITSML v1.4.1.1 and ETP therefore looks like this:

```
eml://witsml14/well(uidWell)/wellbore(uidWellbore)/log(uidLog)/logCurveInfo(mnemonic)
```

To construct this ETP URI, use the WITSML v1.4.1.1 SOAP API to fetch object uids and mnemonics, and then use those to construct the URI.

2.2.1 Case Sensitivity of URIs

The UIDs contained in the URI follow the same rules as the URIs in the *WITSML Store API v1.4.1.1*.

Consistent with the *Energistics Identifier Specification*, a URI is case sensitive except for identifiers contained in parentheses, which should be treated as case-insensitive. This guidance is true for both identifiers that are UUIDs (GUID) or WITSML 1.x UIDs.

When composing a URI for a WITSML 1.x object, use the singular form and the case custom of the underlying ML. For example, in the URI above, *well* and *wellbore* each start with a lower-case letter but logCurveInfo is camel case.

2.3 Consistent Reference of Objects between WITSML and ETP

If a WITSML server and an ETP server are referring to same object, they MUST use the same key (UID).

2.4 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.

2.5 Content Types

The following table lists each WITSML v1.4.1.1 data object and its corresponding content type. NOTE: The type does NOT use the obj_, similar to the way the URI also does not include the obj_.

Data Object	Content Type
attachment	application/x-witsml+xml;version=1.4.1.1;type=attachment
bhaRun	application/x-witsml+xml;version=1.4.1.1;type=bhaRun
cementJob	application/x-witsml+xml;version=1.4.1.1;type=cementJob
convCore	application/x-witsml+xml;version=1.4.1.1;type=convCore
coordinateReferenceSystem	application/x-witsml+xml;version=1.4.1.1;type=coordinateReferenceSystem
drillReport	application/x-witsml+xml;version=1.4.1.1;type=drillReport
fluidsReport	application/x-witsml+xml;version=1.4.1.1;type=fluidsReport
formationMarker	application/x-witsml+xml;version=1.4.1.1;type=formationMarker
log	application/x-witsml+xml;version=1.4.1.1;type=log
logCurveInfo	application/x-witsml+xml;version=1.4.1.1;type=logCurveInfo
mudLog	application/x-witsml+xml;version=1.4.1.1;type=mudLog
objectGroup	application/x-witsml+xml;version=1.4.1.1;type=objectGroup
opsReport	application/x-witsml+xml;version=1.4.1.1;type=opsReport
rig	application/x-witsml+xml;version=1.4.1.1;type=rig
risk	application/x-witsml+xml;version=1.4.1.1;type=risk
sidewallCore	application/x-witsml+xml;version=1.4.1.1;type=sidewallCore
stimJob	application/x-witsml+xml;version=1.4.1.1;type=stimJob
surveyProgram	application/x-witsml+xml;version=1.4.1.1;type=surveyProgram
target	application/x-witsml+xml;version=1.4.1.1;type=target
toolErrorModel	application/x-witsml+xml;version=1.4.1.1;type=toolErrorModel
toolErrorTermSet	application/x-witsml+xml;version=1.4.1.1;type=toolErrorTermSet
trajectory	application/x-witsml+xml;version=1.4.1.1;type=trajectory
tubular	application/x-witsml+xml;version=1.4.1.1;type=tubular
wbGeometry (wellbore geometry)	application/x-witsml+xml;version=1.4.1.1;type=wbGeometry
well	application/x-witsml+xml;version=1.4.1.1;type=well
wellbore	application/x-witsml+xml;version=1.4.1.1;type=wellbore

3 Streaming Channel Data (Protocol 1: ChannelStreaming)

This chapter provides implementation details for using the ETP ChannelStreaming protocol (Protocol 1) to stream WITSML v1.4.1.1 log data.

3.1 Channel Identification

In ETP, WITSML log curves are represented by ETP channels. The channels are identified by a URI. This identifier is used in the channelUri field of the ChannelMetadataRecord message of ETP.

3.2 Describable Object Types

In WITSML v1.4.1.1:

- These objects must be supported in the ChannelDescribe message: well, wellbore, log and logCurveInfo URIs.
- The server should send all ChannelMetadataRecord objects in the hierarchy below the described object.
- The URI “eml://” or “eml://witsml14/” is not required to support the ChannelDescribe operation.

3.3 Mapping WITSML LogCurveInfo to the ETP Channel Metadata Record

For the structure of ChannelMetadataRecord, see the *ETP Specification,* Section 3.3.16.9.

The table below maps the WITSML elements to the ETP attributes and calls out element examples from this example XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<logs xmlns="http://www.witsml.org/schemas/1series"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.4.1.1">
  <log uidWell="101e8e3a-5811-4b2e-b404-0367b360e4b6" uidWellbore="dd3406d4-
0d8d-4530-9b3a-337a03515a2c" uid="832TE2C54">
    <objectGrowing>true</objectGrowing>
    <serviceCompany>Baker Hughes</serviceCompany>
    <indexType>measured depth</indexType>
    <direction>increasing</direction>
    <indexCurve>Depth</indexCurve>
   <logCurveInfo uid="4389">
      <mnemonic>Depth</mnemonic>
      <unit>m</unit>
      <mnemAlias>Depth</mnemAlias>
      <minIndex uom="m">10</minIndex>
      <maxIndex uom="m">110</maxIndex>
      <curveDescription>The mnemonic of the index curve</curveDescription>
      <typeLogData>double</typeLogData>
    </logCurveInfo>
    <logCurveInfo uid="78964">
      <mnemonic>WOB</mnemonic>
      <classWitsml>wobAv</classWitsml>
      <unit>N</unit>
      <minIndex uom="m">10</minIndex>
      <maxIndex uom="m">110</maxIndex>
      <curveDescription>Weight On Bit</curveDescription>
      <typeLogData>double</typeLogData>
    </logCurveInfo>
```

```
  </log>
</logs>
```

ETP Attribute ChannelMetadata Record Property	Mapping to WITSML Elements	Example (from above XML)
channelUri	Construct a URI using the *Energistics Identifier Specification*. For WITSML v1.4.1.1, provide a fully qualified path to the **logCurveInfo**.	eml://witsml14/**well**(101e8e3a-5811-4b2e-b404-0367b360e4b6)**wellbore**(dd3406d4-0d8d-4530-9b3a-337a03515a2c)/**log**(832TE2C54)/**logCurveInfo**(WOB)
channelId	Provided by the channel data producer.	22
Indexes	For WITSML v1.4.1.1, only a single index is used.	See Section 3.4
channelName	The **mnemonic** tag in a logCurveInfo node.	WOB
dataType	The **typeLogData** tag in a logCurveInfo node.	double
uom	The **unit** tag in logCurveInfo.	N
startIndex	**Depth Log**: • For increasing logs, use the **minIndex** tag in logCurveInfo. • For decreasing logs, use **maxIndex**. **Time Log**: • For increasing logs, use the **minDateTimeIndex** tag in logCurveInfo. • For decreasing logs, use **maxDateTimeIndex**. **NOTE:** A simple streamer might return Null for **startIndex** because a sensor does not hold a history.	Depth 10000 With a scale of 3, this value is 10.0 (see the definition of *scale* in IndexMetadataRecord in the *ETP Specification*, Section 3.3.16.8). Time: 1448810886000000 would be 2015-11-29T15:28:06Z
endIndex	**Depth Log**: • For increasing logs, use the **maxIndex** tag in logCurveInfo. • For decreasing logs, use **minIndex**. **Time Log**: • For increasing logs, use the **maxDateTimeIndex** tag in logCurveInfo. • For decreasing logs, use **minDateTimeIndex**.	Depth 110000 With a scale of 3 (as in this example) the value is 110.0 (see the definition of *scale* in IndexMetadataRecord in the *ETP Specification*, Section 3.3.16.8). Time: 1448810886000000 would be 2015-11-29T15:28:06Z

ETP Attribute ChannelMetadata Record Property	Mapping to WITSML Elements	Example (from above XML)
	NOTE: A simple streamer might return Null for endIndex because sensor does not hold a history	
description	The **curveDescription** in the **logCurveInfo.**	Weight On Bit
status	Mapped from **objectGrowing** (= true) in the log object.	Active
contentType	The string is: **application/x-witsml+xml;version=1.4.1.1;type=**logCurveInfo	application/x-witsml+xml;version=1.4.1.1;type=logCurveInfo
source	The tag **serviceCompany** from the log object.	Baker Hughes
measureClass	Not possible to map from a standard WITSML v1.4.1.1 log object.	Empty string, i.e., “”
uuid	Server implementation specific. This field is not possible to map from a standard WITSML v1.4.1.1 log object.	null NOTE: If your v1.4.1.1 store is producing a proper UUID, then you would populate this field.
customData	If the **typeLogData** is **datetime**, then this field must be used. Then the datatype in the ChannelMetadataRecord must be long, and this field has a name-value pair, which is logicalType accompanied by timestamp-micros. For more information see the *ETP Specification* (Section 3.3.16.9). Time must be normalized to UTC per the *ETP Specification* (see Section 3.3.16.8) and the Avro Specification.	null
domainObject	If the logCurveInfo has an **axisDefinition** node, then the domainObject must be included. Optionally, if it has no axisDefinition node, this can hold the **logCurveInfo** type for this channel. Currently logCurveInfo does not have a stand-alone v1.4.1.1 schema, so this domain object cannot be schema validated. See the example schema below in Section 3.3.1.	null

3.3.1 DomainObject

For definitions of domain object and Record ChannelMetadataRecord, see the ETP Specification, Section 3.3.16.9.

For WITSML v1.4.1.1 log curves that have axisDefinition(s), place a DataObject on the ETP ChannelMetadataRecord.DomainObject which contains the logCurveInfo entry for that curve.

Example ChannelMetadataRecord:


```
    {
      "ChannelUri":
"eml://witsml14/well(uidWell)/wellbore(uidWellbore)/log(logwitharray)/logcurve
info(mnemonic)",
      "ChannelId": 2,
      "Indexes": [
        {
          "IndexType": 1,
          "Uom": "m",
          "DepthDatum": null,
          "Direction": 0,
          "Mnemonic": {
            "string": "DEPTH"
          },
          "Description": {
            "string": "Measured Depth"
          },
          "Uri": {
            "string":
"eml://witsml14/well(mdaq_well_etp_uid)/wellbore(mdaq_wellbore_etp_uid)/log(lo
gwitharray)/logcurveinfo(depth)"
          },
          "CustomData": {},
          "Scale": 5,
          "TimeDatum": null
        }
      ],
      "ChannelName": "ADIM_TOH_RT2",
      "DataType": "Energistics.Datatypes.ArrayOfDouble",
      "Uom": "g/cm3",
      "StartIndex": {
        "long": 100000000
      },
      "EndIndex": {
        "long": 101000000
      },
      "Description": "ADN top-of-hole oriented image",
      "Status": 0,
      "ContentType": null,
      "Source": "",
      "MeasureClass": "",
      "Uuid": null,
      "CustomData": {},
      "DomainObject": {
        "Energistics.Datatypes.Object.DataObject": {
          "Resource": {
```

```
            "Uri":
"eml://witsml14/well(mdaq_well_etp_uid)/wellbore(mdaq_wellbore_etp_uid)/log(lo
gwitharray)/logCurveInfo(adim_toh_rt2)",
            "ContentType": "application/x-
witsml+xml;version=1.4.1.1;type=logCurveInfo",
            "Name": "ADIM_TOH_RT2",
            "ChannelSubscribable": false,
            "CustomData": {},
            "ResourceType": "DataObject",
            "HasChildren": 0,
            "Uuid": {
              "string": "b106a1e6-2cf3-4c5f-84de-5dabf0854704"
            },
            "LastChanged": 1482324315058282,
            "ObjectNotifiable": true
          },
          "ContentEncoding": "gzip",
          "Data":
"H4sIAAAAAAAAEAIWSXWvCMBSG/0rovabVfcoxMNaLCbrBlLE7ydLTGmhySpPM7t8vtSI6L3Z38vI+7
/kgoNy2puo5tN+4sCWxztTWzTqn58nO+2bG+X6/H++nY2orPknTjH+ulmu1QyNH2jovrcLkRBX/Uwk
LOtqe8sVqu3l72b5vJkf+ktXemfrAuwPneOaw1egSAcaiIauVOE8BfpIhWO1FxZWZAj/UYLRd2AI7F
sjMk9InIkvTNDJHPTpkd+XIesdRF6D6K+XoVKsbr8nG9q/MUzOicrSjGhnF+azHgmkjKwR+BUAhvVx
TaBX2MPCzN/ifBpdU5VESBYWvOiacayA77XIsdVwohg13zOI5qC2wFRnwoQBFwXrRzz5UMKR9yDqgE
xmbsCm7Ybfsjt2zB/bIeueFBfhlqyj8+SfiF+oxRn45AgAA"
        }
      }
    },
```


Where Data is decrypted to be:

```
<logCurveInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" uid="ADIM_TOH_RT"
xmlns="http://www.witsml.org/schemas/1series">
  <mnemonic>ADIM_TOH_RT</mnemonic>
  <unit>g/cm3</unit>
  <minIndex uom="ft">1000</minIndex>
  <maxIndex uom="ft">1010</maxIndex>
  <curveDescription>ADN top-of-hole oriented image</curveDescription>
  <dataSource>ADN</dataSource>
  <typeLogData>double</typeLogData>
  <axisDefinition uid="1">
    <order>1</order>
    <count>16</count>
    <doubleValues>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</doubleValues>
  </axisDefinition>
</logCurveInfo>
```

3.4 Mapping WITSML LogCurveInfo to the ETP Index Metadata Record

The ETP IndexMetadataRecord is mainly constructed from the index curve of the WITSML v1.4.1.1 log. The table below maps the WITSML elements to the ETP attributes and calls out element examples from this example XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<logs xmlns="http://www.witsml.org/schemas/1series"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.4.1.1">
  <log uidWell="101e8e3a-5811-4b2e-b404-0367b360e4b6" uidWellbore="dd3406d4-
0d8d-4530-9b3a-337a03515a2c" uid="832TE2C54">
    <indexType>measured depth</indexType>
    <direction>increasing</direction>
    <indexCurve>Depth</indexCurve>
    <logCurveInfo uid="4389">
      <mnemonic>Depth</mnemonic>
      <unit>m</unit>
      <mnemAlias>Depth</mnemAlias>
      <curveDescription>The mnemonic of the index curve</curveDescription>
      <typeLogData>double</typeLogData>
    </logCurveInfo>
  </log>
</logs>
```

ETP Attribute IndexMetadata Record Properties	Mapping to WITSML Element	Example
indexType	The **indexType** in the log object.	`Depth`
uom	The unit in the **logCurveInfo**.	`m`
depthDatum	URI of a well datum.	eml://witsml14/well(10e8e3a-5811-4b2e-b404-0367b360e4b6)/wellDatum(KB)
direction	The direction in the log object.	Increasing
mnemonic	The **mnemonic** tag in the **logCurveInfo** node.	`Depth`
description	The **curveDescription** in the **logCurveInfo.**	The mnemonic of the index curve.
uri	Construct a URI according to the *Energistics Identifier Specification*. For WITSML v1.4.1.1, provide a fully qualified path to the **logCurveInfo .**	eml://witsml14/well(101e8e3a-5811-4b2e-b404-0367b360e4b6)wellbore(dd3406d4-0d8d-4530-9b3a-337a03515a2c)/log(832TE2C54)/logCurveInfo(Depth)
customData	Not possible to map from a standard WITSML v1.4.1.1 log object.	null
scale	See ETP Specification, Section 3.3.16.8.	3
timeDatum	Not possible to map from a standard WITSML v1.4.1.1 log object.	null

3.5 Creating the Underlying Log Object From the ChannelMetadata

- If no well and wellbore exists, channel streaming must fail.
- If the well and wellbore exists, and:
  - If the log exists: you must add the curve to the existing log.
  - If the log does not exist: you must create the log and add the curve to it.

Appendix A. Use Cases

This appendix provides use cases to help you understand how to use ETP v1.1 to stream WITSML v1.4.1.1 log data. The displayed flow sequences are based on the *ETP Specification*, with appropriate tailoring and details specific to the use case.

Though this appendix is non-normative, it is recommended that these flows be supported.

NOTE: In the flow sequence tables below, all **bold text** are message terms from the *ETP Specification*.

A.1 Use Case: Real-Time Streaming from a Sensor to an Aggregating Server

This section outlines three possible real-time streaming options for this use case:

- Simple Streamer: Aggregator initiates the connection
- Simple Streamer: Sensor initiates the connection
- Basic Streaming (i.e., not a simple streamer)

A.1.1 Actors

Producer and consumer are roles defined in the *ETP Specification* (see Section 3.4.2). For the three streaming options described above, these are the actors:

- **Producer**: A sensor providing a stream of real-time data
- **Consumer:** An ETP v1.1/WITSML v1.4.1.1 aggregator

A.1.2 Simple Streamer: Aggregator Initiates the Connection

In ETP, a “simple streamer” is a data provider with limited capabilities, for example, it does not store historical data and may not be able to provide well and wellbore information. Because of these limited capabilities, the ETP message sequence is also simple.

A.1.2.1 Preconditions

The aggregator is responsible for mapping the channel data that it receives into its WITSML hierarchy of well/wellbore/log/logCurveInfo. This mapping requires prior communication between the producer and consumer companies to define the meaning and mapping of the data channels, so that the equipment can be configured (e.g., determine the ETP WebSocket server at a known URL.)

A.1.2.2 Flow Sequence

When the consumer/aggregator initiates the connection, the recommended sequence of activities to stream data using ETP is described in the table below.

	Producer (Sensor) / ETP Server	Consumer (Aggregator) / ETP Client
1	Provide an ETP WebSocket server at a known URL (see Preconditions above).	
2		**Protocol 0:** Open WebSocket via HTTP call to WebSocket URL.
3		Send **RequestSession** message and role as a “consumer”.
4	Send **OpenSession**. Include ProtocolCapabilities of SimpleStreamer = true.	
5		**Protocol 1:** Send **Start** message, which may contain throttling parameters.

	Producer (Sensor) / ETP Server	Consumer (Aggregator) / ETP Client
6	Send **ChannelMetadata** message(s), which contains local identifiers for the channel(s).	
7		Decode **ChannelMetadata**, assigning channels to a well/wellbore/log/logCurveInfo hierarchy. This decoding and assigning of channels must be completed before any channel data can be received.
8	Begin a sending loop, sending **ChannelData** messages for all channel data as they are available.	
9		Decode each **ChannelData** message and append it to the appropriate WITSML log object and/or makes it available as ETP channel data to other consumers.
10		**Protocol 0:** To terminate data transfer, send **CloseSession**.

To terminate the session, the consumer typically sends the CloseSession message, but the producer can also send a CloseSession message if, for some reason, it had to stop sending data.

A.1.3 Simple Streamer: Sensor Initiates the Connection

When the sensor/producer initiates the connection, the first steps in the flow sequence change as shown in the table below. (The gray cells are the same as defined in Section A.1.2.2 above).

	Producer (Sensor) / ETP Client	Consumer (Aggregator) / ETP Server
1		Provide an ETP WebSocket server at a known URL.
2	**Protocol 0:** Open WebSocket via HTTP call to WebSocket URL	
3	Send **RequestSession** message, requesting Protocol 1 and requesting the server role as a consumer. Include ProtocolCapabilities of SimpleStreamer = true.	
4		Send **OpenSession.**
5		**Protocol 1:** Send **Start** message, which may contain throttling parameters.
6	Send **ChannelMetadata** message(s), which contains local identifiers for the channel(s).	
7		Decode **ChannelMetadata**, assigning channels to a well/wellbore/log/logCurveInfo hierarchy. This decoding and assigning of channels must be completed before any channel data can be received.
8	Begin a sending loop, sending **ChannelData** messages for all channel data as they are available.	

	Producer (Sensor) / ETP Client	Consumer (Aggregator) / ETP Server
9		Decode each **ChannelData** message and append it to the appropriate WITSML log object and/or makes it available as ETP channel data to other consumers.
10		**Protocol 0:** To terminate data transfer, send **CloseSession**.

A.1.4 Basic Streaming (i.e., not “simple streaming”)

Alternatively, the producer may implement “basic” streaming. In this case, the ProtocolCapabilities of SimpleStreamer is omitted, and the consumer must send **ChannelDescribe** and **ChannelStreamingStart** messages.

	Producer (Sensor) / ETP Server	Consumer (Aggregator) / ETP Client
1	Provide an ETP WebSocket server at a known URL.	
2		**Protocol 0:** Open WebSocket via HTTP call to WebSocket URL.
3		**Protocol 0:** Send **RequestSession** message and role as a “consumer”.
4	Send **OpenSession** message which defines the supported protocols.	
5		**Protocol 1**: Send **Start** message, which may contain throttling parameters.
6		Send **ChannelDescribe** message, which lists the channels of interest as an Energistics URI, for example: well(idwell)/wellbore(idwellbore)/log(idlog)/logCurveInfo(mnemonic)
7	• If the producer cannot provide the channel data in the **ChannelDescribe** message, it sends a **ProtocolException** message with an appropriate error code and a description of the error (see the *ETP Specification*, Appendix A). • If the requested channels are recognized, it sends **ChannelMetadata** messages, which each contains a channel URI. for example: eml://witsml14/well(uidWell)/wellbore(uidWellbore)/log(uidLog)/logCurveInfo(mnemonic)	
8		To specify the required channels and their starting points, send **ChannelStreamingStart**.
9	Begin a sending loop, sending the requested **ChannelData** messages.	
10		Update the displays with data from the received **ChannelData** messages.
11	(Optional) While sending data, send channel change messages, as appropriate: **ChannelDataChange, ChannelStatusChange, ChannelRemove.**	
12		(Optional) While receiving data, send data flow control messages, as appropriate: **ChannelStreamingStop, ChannelStreamingStart.**
13		To terminate data transfer, send **CloseSession**.

A.2 Use Case: Real-Time Streaming from Wellsite to an Aggregating Server

In this use case, a service company is at the wellsite, e.g., a mud-logging or LWD data provider (producer role) and is sending data to an ETPv1.1/WITSML v1.4.1.1 aggregator (consumer).

In this case, the producer (service company) initiates the session.

A.2.1 Actors

Producer: service company

Consumer: An ETP v1.1/WITSML v1.4.1.1 aggregator

A.2.2 Preconditions

N/A

A.2.3 Post Conditions

The aggregator must store the received **ChannelMetadata** and **ChannelData** and make it available over WITSML v1.4.1.1 API as a WITSML v1.4.1.1 Log object. It should also be available for ETP consumers connecting later (after the data is received and stored).

A.2.4 Flow Sequence

#	Producer (Service Company) / ETP Client	Consumer (Aggregator) / ETP Server
1		Provide a WITSML v1.4.1.1 server and a WebSocket server.
2	Using a WITSML store API: • Query the server to determine well/wellbore/log hierarchy. • Create new log objects, if necessary, so it can receive the streaming data. NOTE: You can also perform these tasks using ETP protocols (Discovery and Store); additional guidance on how to use these ETP protocols will be published in a future version of this document.	
3	**Protocol 0:** Open WebSocket using HTTP call to WebSocket URL.	
4	Send **RequestSession** message, requesting Protocol 1 and requesting the server has the role of consumer.	
5		Send **OpenSession** message.
6		**Protocol 1:** Send **Start** message, which may contain throttling parameters.
7		Send **ChannelDescribe** message, which identifies the context of data to send.
8	Send **ChannelMetadata** message, which contains Energistics URI for the channel(s), for example: eml://witsml14/well(idwell)/wellbore(idwellbore)/log(idlog)/logCurveInfo(mnemnoic).	

#	Producer (Service Company) / ETP Client	Consumer (Aggregator) / ETP Server
9		Internally associate the channel indexes to the appropriate log/curve locations.
10		Send **ChannelStreamingStart** message to specify an array of the required channels.
11	Begin sending loop, sending **ChannelData** messages as requested.	
12		Decode **ChannelData** messages and append the data to the appropriate WITSML log objects and/or makes it available as ETP channel data to other consumers.
13	(Optional) While sending data, send channel change messages as appropriate: **ChannelDataChange, ChannelStatusChange, ChannelRemove.**	
14.		(Optional) While receiving data, send data flow control messages as appropriate: **ChannelStreamingStop,ChannelStreamingStart**
15.	**Protocol 0:** To terminate data transfer, send **CloseSession**	

A.3 Use Case: Real-Time Streaming from an Aggregating Server to a Display Client

In this use case, a real-time display application that understands WITSML v1.4.1.1 (consumer) wants to get real-time ETP data from an aggregator (producer) to drive displays and data plots. The client is capable of navigating the WITSML data hierarchy using the WITSML v1.4.1.1 Store API.

An ETP consumer populates the required channel URIs into a ChannelDescribe message that it sends to an ETP producer to request a transfer of real-time data. A consumer without any additional information can use a URI of “eml://witsml14” which allows the producer the freedom to send any WITSML v1.4.1.1 information that it desires. The following table provides URIs for key WITSML data objects.

To send information for a specific:	It sends this URI:
well	eml://witsml14/well(<well-uid>)
wellbore	eml://witsml14/well(uidWell)/wellbore(uidWellbore)
log	eml://witsml14/well(uidWell)/wellbore(uidWellbore)/log(uid)
log curve	eml://witsml14/well(uidWell)/wellbore(uidWellbore)/log(uid)/logCurveInfo(mnemonic)

A.3.1 Preconditions

N/A

A.3.2 Actors

Producer: An ETP v1.1/WITSML v1.4.1.1 aggregator

Consumer: A real-time display application.

A.3.3 Flow Sequence

	Producer (WITSML Server) / ETP Server	Consumer (Real-Time Display) / ETP Client
1		Query the server using WITSML v1.4.1.1 AP to determine well/wellbore/log/curve hierarchy.
2		Open WebSocket using HTTP call to WebSocket URL.
3		**Protocol 0:** Send **RequestSession** message, requesting Protocol 1 and requesting the server role as a producer.
4	Send **OpenSession** message which defines the supported protocols.	
5		**Protocol 1**: Send **Start** message, which may contain throttling parameters.
6		Send **ChannelDescribe** message, which lists the channels of interest as an Energistics URI, for example: well(idwell)/wellbore(idwellbore)/log(idlog)/logCurveInfo(mnemonic)
7	• If the producer cannot provide the channel data in the **ChannelDescribe** message, it sends a **ProtocolException** message with an appropriate error code and a description of the error (see the *ETP Specification*, Appendix A). • If the requested channels are recognized, it sends **ChannelMetadata** messages, which each contains a channel URI. for example: eml://witsml14/well(uidWell)/wellbore(uidWellbore)/log(uidLog)/logCurveInfo(mnemonic)	
8		To specify the required channels and their starting points, send **ChannelStreamingStart**.
9	Begin a sending loop, sending the requested **ChannelData** messages.	
10		Update the displays with data from the received **ChannelData** messages.
11	(Optional) While sending data, send channel change messages, as appropriate: **ChannelDataChange, ChannelStatusChange, ChannelRemove.**	
12.		(Optional) While receiving data, send data flow control messages, as appropriate: **ChannelStreamingStop, ChannelStreamingStart.**
13.		To terminate data transfer, send **CloseSession**.