Modernizing WITSML Data Exchange: Bridging SOAP and ETP Protocols for Migration Automation

Mr. Mukund Premchand Dasari

Backend Software Engineer

Abstract

With the energy sector's shift toward digital oilfield operations, the ability to efficiently migrate and utilize wellsite drilling data is critical. WITSML, the Wellsite Information Transfer Standard Markup Language, is a cornerstone standard for well data exchange but is often hampered in legacy environments by the use of the SOAP (Simple Object Access Protocol). ETP (Energistics Transfer Protocol) now serves as the modern, high-performance backbone for WITSML v2.0+. This paper details both the motivation and technical architecture for a SOAP to ETP conversion utility, comprehensively analyzes the protocol differences impacting performance, bandwidth, latency, and data integrity, and offers a practical approach to seamless WITSML data migration.

1. Introduction

The Wellsite Information Transfer Standard Markup Language (WITSML) is widely used to exchange drilling, completion, and production data in the upstream oil and gas sector. WITSML versions 1.3 and 1.4.1.1 are traditionally SOAP-based, relying on synchronous XML exchanges over HTTP. Although robust, these systems are inadequate for modern needs demanding real-time collaboration, high-frequency data ingestion, and low-latency streaming.

The Energistics Transfer Protocol (ETP), introduced as part of the Common Technical Architecture (CTA), is a WebSocket-based binary protocol engineered to handle real-time and asynchronous data transfer, supporting WITSML v2.0 and later. Bridging SOAP and ETP allows for seamless migration and hybrid infrastructure support, preserving historical datasets while enabling modern workflows.

What is WITSML?

WITSML (Wellsite Information Transfer Standard Markup Language) is an open industry XML data standard for wellsite and drilling data exchange, designed to enable seamless, real-time, and historical data flow between operators, service companies, and analytical applications within the oil and gas sector. Its data objects model a wide range of drilling, completion, and production activities.

What is SOAP Protocol?

SOAP (Simple Object Access Protocol) is a platform-independent, XML-based messaging protocol used primarily for exchanging structured information in web services. As applied in WITSML v1.4.1.1, SOAP enables data transfer via synchronous, stateless operations over HTTP(S), using verbose XML payloads

What is ETP Protocol?

ETP (Energistics Transfer Protocol) is a modern, binary, asynchronous protocol developed by Energistic s for real-time streaming, transfer, and management of energy data, replacing SOAP as the API for WITSML v2.0 and above. ETP leverages persistent WebSocket connections and Apache Avro binary serialization for highly efficient, scalable, and low-latency data movement.

2. Motivation and Industry Drivers

Key Challenges with SOAP-Based WITSML

- **High Latency**: SOAP operations often exceed 2s per object due to stateless HTTP interactions.
- **Bandwidth Overhead**: XML verbosity results in object sizes averaging ~10MB.
- **Synchronous Communication**: Inhibits streaming and real-time updates.
- **Security Overhead**: WS-Security per-request models lead to repeated authentication.
- **Complex Identifier Handling**: Compound UIDs complicate cross-platform mapping.

Opportunities Enabled by ETP

- **Sub-Second Latency**: Transfers as fast as 200ms/object.
- **Binary Efficiency**: Avro encoding reduces payloads to ~1MB/object.
- **Asynchronous Streaming**: Real-time log updates (e.g., 50Hz) using publish/subscribe.
- **Session-Level Security**: TLS-based persistent connections enhance both performance and security.
- **Global Identifiers**: Simplified object referencing via UUIDs.

3. Protocol Comparison: SOAP vs ETP

Feature	SOAP (Legacy)	ETP (Modern)
Transport	HTTP(S), XML	WebSocket, Avro (Binary)
Streaming Support	No	Yes (Real-time pub/sub)
Latency	~2s/object	~200ms/object
Bandwidth Usage	~10MB/object	~1MB/object
Security	WS-Security (per call)	TLS, persistent sessions
Identifier Format	Compound UIDs	UUIDs / URIs

4. System Architecture

Pipeline Components

Component	Description
SOAP Client	Connects to WITSML 1.4.1.1 server via WS-Security and SOAP APIs
XML Parser	Parses SOAP XML and normalizes data into intermediary format
Data Mapper	Resolves units, identifiers, schema versions
ETP Encoder	Translates normalized objects into Avro-compliant binary payloads
ETP Publisher	Establishes WebSocket session and transmits data to WITSML 2.0 server
Logger/Monitor	Tracks metrics, exceptions, ACKs, and retry attempts

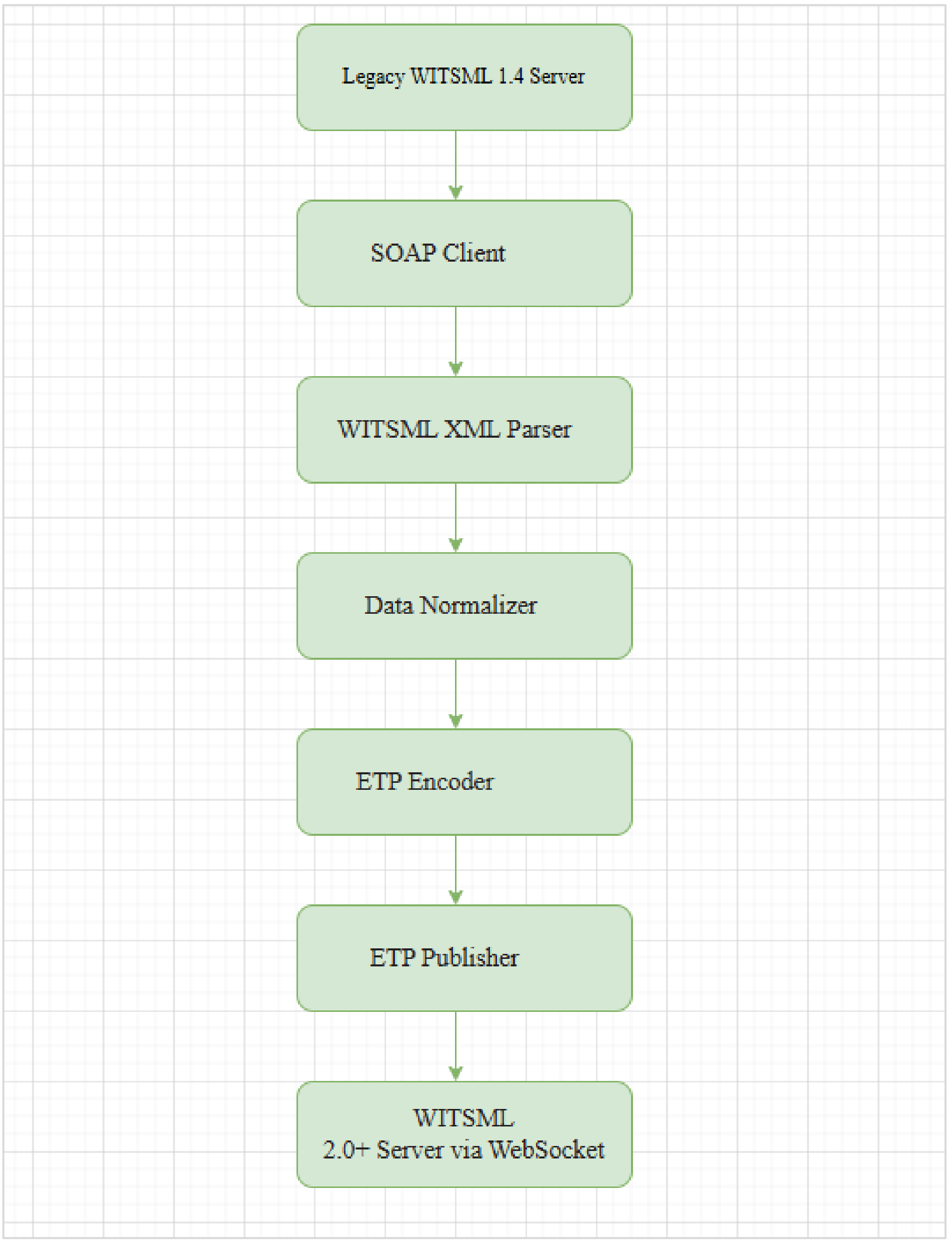

Figure 1: Architecture Diagram

5. Migration Workflow

Steps:

1. **Connection & Authentication**
   - SOAP client authenticates using Web Service -Security or token-based auth.
2. **Data Acquisition**
   - WMLS_GetFromStore retrieves data like log, trajectory, and mudLog.
3. **Parsing & Normalization**
   - XML payloads validated against WITSML schema; units resolved via Energistics UOM dictionary.
4. **Mapping & Encoding**
   - Compound UIDs mapped to UUIDs; ChannelMetadataRecord and IndexMetadataRecord constructed.

5. **Publishing**
   - Persistent WebSocket session established with the ETP server; objects pushed asynchronously.
6. **Monitoring & Validation**
   - SHA256 checksums, retries, and ACKs ensure integrity.

6. Real-Time Streaming Capabilities

ETP supports the following modes:

- **Simple Streamer**: Periodic data updates pushed automatically.
- **Client-Initiated Streaming**: Clients request data windows dynamically.
- **Hybrid Streaming**: Combines on-demand and push-based streaming.

Use cases:

- Real-time rig monitoring
- Log curve visualization
- Collaborative vendor dashboards

7. Security and Reliability

Aspect	SOAP	ETP
Encryption	WS-Security (SOAP Headers)	TLS
Session Handling	Stateless	Persistent WebSocket
Integrity	XML schema validation	Checksums, ACK/NACK, retries
Authentication	Per-request token	Handshake-based session auth

ETP's persistent session model reduces redundancy, increases reliability, and enables high-availability streaming.

8. Performance Benchmarks

Metric	SOAP (Legacy)	ETP (Converted)	Gain
Transfer Speed	~500 KB/s	~5 MB/s	10×
Bandwidth/Object	~10 MB	~1 MB	10× smaller
Latency/Object	~2 sec	~200 ms	10× faster

9. Business Use Cases

- **Legacy Data Modernization**
  - Makes historical data accessible for modern analytics and AI pipelines.
- **Hybrid Cloud Enablement**
  - Supports phased cloud migration and parallel SOAP/ETP infrastructure.
- **Regulatory & Compliance**
  - Full audit trail and object traceability during data transfers.
- **AI/ML Readiness**
  - ETP-ready data enables predictive modelling, anomaly detection, and optimization.

10. Extensibility & Customization

- **ETP-to-SOAP reverse conversion** for interoperability testing.
- **Plug-in support** for vendor-specific WITSML extensions.

- **Visual Config UI** for transformation templates and connection profiles.

11. Limitations and Challenges

- **Schema Drift**: Variability between vendor implementations requires plug-in mapping layers.
- **Version Compatibility**: Ensuring fidelity when mapping from v1.4 to v2.0 can be non-trivial.
- **Network Constraints**: ETP's streaming benefits require stable, long-lived network sessions.

12. Future Roadmap

- Support for **ETP 1.2** event-driven features
- **UI-driven transformation builders**
- Integration with **data lakes**, **object storage**, and **real-time dashboards**
- Support for **offline/nearline** batching modes

13. Conclusion

This SOAP to ETP conversion utility bridges a significant technological divide in upstream data systems. It not only modernizes access to legacy WITSML assets but enables next-generation cloud workflows, real-time streaming, and AI integration. With optimized latency, bandwidth, and reliability, the utility paves the way for digital oilfield operations that are smarter, faster, and more connected.

Appendix A: Sample Conversion

SOAP Input (Trajectory)

xml
CopyEdit

```xml
<trajectory uid="traj-001">
<name>Well_A_Trajectory</name>
<mdMn uom="m">1000</mdMn>
<mdMx uom="m">2500</mdMx>
</trajectory>
```

ETP (JSON Abstract)

json
CopyEdit

```json
{
"uuid": "traj-001",
"name": "Well_A_Trajectory",
"startDepth": 1000,
"endDepth": 2500,
"unit": "m"
}
```

Appendix B: References

1. https://energistics.org/energistics-transfer-protocol-etp
2. https://energistics.org/witsml-developers-users
3. https://docs.energistics.org/CTA/CTA_TOPICS/CTA-000-021-0-C-sv2100.html
4. https://energistics.org/sites/default/files/2022-10/WO0118_Statoil_ETP_Test-1.pdf