sttp
Streaming Telemetry Transport Protocol

Version: 0.0.16 - June 26, 2017

Status: Initial Development

Abstract: This specification defines a publish-subscribe data transfer protocol that has been optimized for exchanging streaming time-series style data, such as synchrophasor data that is used in the electric power industry, over Internet Protocol (IP). The protocol supports transferring both real-time and historical time-series data at full or down-sampled resolutions. Protocol benefits are realized at scale when multiplexing very large numbers of time-series data points at high speed, such as, hundreds of times per second per data point.

Disclaimer

License

Table of Contents

Introduction

Use of synchrophasors by U.S. utilities continues to grow following the jump start provided by the Smart Grid Investment Grants. Even so, the dominant method to exchange synchrophasor data remains the IEEE C37.118-2005 [2] protocol that was designed for and continues to be the preferred solution for substation-to-control room communications. It achieves its advantages through use of an ordered set (a frame) of information that is associated with a specific measurement time. When IEEE C37.118 is used for PDC-to-PDC communication or for PDC-to-Application communication, large data frames are typically distributed to multiple systems. To address the challenges presented by these large frame sizes, many utilities implement purpose-built networks for synchrophasor data only. Even with these purpose-built networks, large frame sizes result in an increased probability of UDP frame loss, or in the case of TCP, increased communication latency. In addition, IEEE C37.118 has only prescriptive methods for the management of measurement metadata which is well-suited for substation-to-control-center use but which becomes difficult to manage as this metadata spans analytic solutions and is used by multiple configuration owners in a wide-area context.

To address these issues, the Advanced Synchrophasor Protocol (ASP) Project was proposed to DOE in response to FOA-1492. In this project proposal, the argument was made for a new protocol that overcomes the limitations of IEEE C37.118 for large-scale synchrophasor data system deployments. The new protocol proposed leveraged the successful design elements of the secure Gateway Exchange Protocol (GEP) that was originally developed by the Grid Protection Alliance (GPA) as part of the SIEGate project (DE-OE-536).

On May 1, 2017, a DOE grant (DE-OE-859) was awarded to GPA and the other 25 collaborators on ASP Project (see Contributors) to develop: (1) a detailed definition of new publish-subscribe protocol, now called the Streaming Time-series Transport Protocol (STTP) and (2) software to support it including production-grade implementations of STTP API's in multiple development platforms along with a collection of tools to test and validate the new protocol.

Scope of this Document

The purpose of this document is to define STTP and to include, as appendices, descriptions as to how to use its supporting software tools. This STTP specification is focused on effective "streaming data" delivery of which synchrophasor data is a very important use case.

In the Overview section of this specification, high-level features and the business value of STTP are presented. The balance of the sections of the specification provide the details of protocol design.

Appendix A - STTP API Reference provides instructions to enable software developers to integrate and use of STTP within other software systems.

Appendix B - IEEE C37.118 Mapping provides a detailed look at the process of transforming IEEE C37.118 into STTP as well as creating IEEE C37.118 streams from STTP.

While the format and structure of this document, established to facilitate collaboration, is different than that used by standards bodies, it is hoped that the content within this document can meet all the information requirements needed to enable repackaging of this specification into draft standard formats.

Definitions and Nomenclature

The styles used to show code, notes, etc.

> 🚧 To spice up the formatting of the spec, GitHub offers a library of emogi's. Some that we might want to play into nomenclature 🔍 💡 🖥 🔧 📁 📦 ✖️ 🆕 ➡️ ▶️ 🔄 #️⃣ 🔜 ➕ ⬛ 🔖 ⚠️ ℹ️ 📄 📊 🌎 🌐 From use of the atom editor, it looks like some are unique to GitHub and others are part of more standard collections. Or we could make some custom ones that would be included as images. Here's a link to full available set: https://gist.github.com/rxaviers/7360908

For example,

> ℹ️ This is an instructional note in the spec.

or for example,

> ⚠️ This is a very important note in the spec.

Definition of key terms

The 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

Term	Definition
data-point	A measurement on a single member of a statistical population.
phasor	A complex equivalent of a simple cosine wave quantity such that the complex modulus is the cosine wave amplitude and the complex angle (in polar form) is the cosine wave phase angle.
publish/subscribe	A messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead characterize published messages into classes without knowledge of which subscribers, if any, there may be.
synchrophasor	A phasor calculated from data samples using a standard time signal as the reference for the measurement. Synchronized phasors from remote sites have a defined common phase relationship.
term	definition

Acronyms

Term	Definition
API	Application Program Interface
DOE	U.S. Department of Energy
GEP	Gateway Exchange Protocol
GPA	Grid Protection Alliance, Inc.
IP	Internet Protocol
PDC	Phasor Data Concentrator
PMU	Phasor Measurement Unit
STTP	Streaming Telemetry Transport Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTC	Universal Time Coordinated

Protocol Overview

> 🚧 Purpose of protocol, fundamentals of how it works (command and data) - include sub-section titles (4# items) as needed
>
> Describe what protocol is and is not - how it could operate with ZeroMQ, DDS, etc. as a transport layer - how it is different than thrift, protobuf, etc., i.e., atomized approach to native-type data vs serializing and deserializing data structures - why this has benefit at scale (i.e., very large structures)

Data transport requires the use of a command channel using TCP/IP for reliable delivery of important commands. Optionally a secondary data channel can be established using UDP/IP for the transport of data that can tolerate loss. When no secondary UDP/IP is used, both commands and data will share use of the TCP/IP channel for communications.

> 🍅 ❓ JRC: *The question has been raised if a UDP only transport should be allowed? In this mode, any critical commands and responses would basically be sent over UDP. Thought would need to be given to commands and/or responses that never arrive and the consequences thereof.*

more

> ℹ️ Although not precluded from use over other data transports, the design of this protocol is targeted and optimized for use over Internet Protocol, specifically TCP/IP and UDP/IP. Even so, since the command/response implementation and data packet distribution of the STTP protocol is fairly simple, it is expected that commonly available middleware data transport layers, such as ZeroMQ or Data Distribution Service (DDS), could easily support and transmit data using the STTP protocol should any of the messaging distribution and management benefits of these transport layers be useful to a particular deployment environment. However, these types of deployments are outside the scope of this documentation. If needed, STTP integrations with middleware layers should be added as reference implementation repositories to the STTP organizational site.

Protocol Feature Summary

> 🚧 This is the protocol promotional section that includes a bulleted list of the "value points" for the protocol

- Perform at high volume / large scale
- Minimize data losses (e.g., over UDP)
- Lower bandwidth requirements (e.g., over TCP)
- Optimized for the performant delivery of individual data-points
- Automated exchange of metadata (no centralized registry required)
- Detect and expose communication issues
- Security and availability features that enable use on critical systems to support critical operations
- Publish/subscribe at data-point level
- API implemented in multiple languages on multiple platforms

Design Philosophies

- Minimize external libraries and dependencies for reference implementations
- Keep portability in mind with all protocol design work
- Target smallest possible API functionality â€"specialized use cases will be handled by example
- Set design mantra to be â€œkeep it simpleâ€ *as possible*

Data-point Structure

- Contents:
    - Identification - maps to 128-bit Guid, transport mapping should be small
    - Timestamp (required? could simply be a auto-incrementing counter)
    - Value - multiple native types supports
    - Flags - standardize minimal set of simple flags, complex state can be new data-point

Commands and Responses

> 🚧 Purpose of command/response structure, fundamentals of how it works, why it is needed

Commands

All commands must be sent over the command channel.

Code	Command	Source	Description
0x00	Set Operational Modes	Subscriber	Defines desired set of operational modes.
0x01	Metadata Refresh	Subscriber	Requests publisher send updated metadata.
0x02	Subscribe	Subscriber	Defines desired set of data-points to begin receiving.
0x03	Unsubscribe	Subscriber	Requests publisher terminate current subscription.
0x0n	etc.		
0xFF	NoOp	Any	Periodic message to allow validation of connectivity.

Set Operational Modes Command

This must be the first command sent after a successful connection - the command must be sent before any other commands or responses are exchanged so that the "ground-rules" for the communications session can be established. The rule for this operational mode negotiation is that once these modes have been established, they will not change for the lifetime of the connection.

The subscriber must send the command and the publisher must await its reception. If the publisher does not receive the command in a timely fashion (time interval controlled by configuration), it will disconnect the subscriber.

> ℹ️ In modes of operations where the publisher is initiating the connection, the publisher will still be waiting for subscriber to initiate communications with a `Set Operational Modes` command.

- Wire Format: Binary
- Requested operational mode negotiations
  - String encoding
  - Compression modes
  - UDP data channel usage / port

Metadata Refresh Command

- Wire Format: Binary
  - Includes current metadata version number

Subscribe Command

- Wire Format: Binary
  - Includes metadata expression and/or individual Guids for desired data-points

Unsubscribe Command

- Wire Format: Binary

NoOp Command

No operation keep-alive ping. It is possible for the command channel to remain quiet for some time if most data is being transmitted over the data channel, this command allows a periodic test of client connectivity.

- Wire Format: Binary

Responses

Responses are sent over a designated channel based on the nature of the response.

Code	Response	Source	Channel	Description
0x80	Succeeded	Publisher	Command	Command request succeeded. Response details follow.
0x81	Failed	Publisher	Command	Command request failed. Response error details follow.
0x82	Data-point Packet	Any	Data	Response contains data-points.
0x83	Signal Mapping	Any	Command	Response contains data-point Guid to run-time ID mappings.
0x8n	etc.			

> ℹ️ For the response table above, when a response is destined for the data channel, it should be understood that a connection can be established where both the command and data channel use the same TCP connection.

Succeeded Response

- Wire Format: Binary (header)
  - Base wire format includes *in-response-to* command code
  - Can include response that is specific to source command:

Succeeded Response for Metadata Refresh

- Wire Format: String + Binary
  - Includes response message with stats like size, number of tables etc.
  - Includes temporal data-point ID for "chunked" metadata responses
  - Includes number of metadata data-points to be expected

Succeeded Response for Subscribe

Subscriber will need to wait for

- Wire Format: String + Binary
  - Includes response message with stats like number of actual points subscribed, count may not match requested points due to rights or points may no longer exist, etc.
  - Includes temporal data-point ID for "chunked" signal mapping responses
  - Includes number of signal mapping data-points to be expected

Succeeded Response for Unsubscribe

- Wire Format: String
  - Includes message as to successful unsubscribe with stats like connection time

Failed Response

- Wire Format: String + Binary (header)
  - Base wire format includes *in-response-to* command code
  - Includes error message as why command request failed
  - Can include response that is specific to source command:

Failed Response for Set Operational Modes

Failed responses to operational modes usually indicate lack of support by publisher. Failure response should include, per failed operational mode option, what options the publisher supports so that the operational modes can be re-negotiated by resending operational modes with a set of *supported* options.

- Wire Format: Binary
  - Includes operational mode that failed followed by available operational mode options

Data-point Packet Response

- Wire Format: Binary
  - Includes a byte flag indicating content, e.g.:
  - Data compression mode, if any
  - Total data-points in packet
  - Includes serialized data-points

ℹ️ The data-point packet is technically classified as a response to a `subscribe` command. However, unlike most responses that operate as a sole response to a parent command, data-packet responses will continue to flow for available measurements until an `unsubscribe` command is issued.

Signal Mapping Response

- Wire Format: Binary
  - Includes a mapping of data-point Guids to run-time signal IDs
  - Includes per data-point ownership state, rights and delivery characteristic details

Data-point Characteristics

- Priority (e.g., control over data delivery priority)
- Reliability (e.g., must be sent over TCP channel)
- Verification (e.g., notification of successful transport)
- Exception (e.g., delivery on change)
- Resolution (e.g., down-sampling)

Metadata

- Wire Format: Tabular XML format (XML) - highly compressible
- Primary data-point identifier is Guid (define)
- Extensibility
- Rights based content restriction

Dataset Contents

- Minimum required dataset for STTP operation
- Industry specific dataset extensions (outside scope of this doc)

Dataset Filtering

- Format of expressions that work against metadata
  - SQL style expressions
  - Regex style expressions
- Application of expressions
  - Metadata reduction
  - Data-point access security

Dataset Versioning

- Versioned
- Difference based publication

Dataset Serialization

- Serialization for transport
  - Packet based publication using temporal data-point
  - Publisher reduction by access rights and diff-version
  - Subscriber reduction by filter expression
- Serialization to local repository
  - Merging considerations
  - Conflict resolution
  - Ownership control

Compression

- Types of compression
  - Stateful data compression (TCP)
  - Per-packet data compression (UDP)
  - Metadata compression (GZip)
- Compression algorithm extensibility
  - Negotiating desired compression algorithm

Security

- Access control list (ACL) security is always on

Encrypted Communications

- Transport layer security (TLS) over TCP command channel
- UDP data channel traffic secured via AES keys exchanged over TCL command channel

Strong Identity Validation

- X.509 certificates
- Self-signed certificates

Publisher Initiated Security Considerations

How does publisher initiated connection, to cross security zones in desired direction, affect identity validation and TLS?

Access Control Lists

- Allow/deny for specific points (data-point explicit)
- Allow/deny for group with specific points (group explicit)
- Allow/deny for filter expression (filter implicit)
- Allow/deny for group with filter expression (group implicit)

Expression based Access Control

- Expressions can be used to define filters and groups
- How do filters work against extensible metadata, missing columns?

Access Control Precedence

- (1) Data Point Explicit
- (2) Group Explicit
- (3) Filter Implicit
- (4) Group Implicit

References and Notes

1. The MIT Open Source Software License

2. IEEE Standard C37.118â„¢, Standard for Synchrophasors for Power Systems

3. RFC 2119, Current Best Practice Scott Bradner, Harvard University, 1997

4. STTP repository on GitHub

5. …

Contributors

The following individuals actively participated in the development of this standard.

- J. Ritchie Carroll, GPA
- F. Russell Robertson, GPA
- Stephen C. Wills, GPA

ASP Project Participants

Project Collaborators	Project Financial Partner	Vendor	Utility	Demonstration Host
Bonneville Power Administration	◆		◆	
Bridge Energy Group				
Dominion Energy	◆		◆	EPG
Electric Power Group	◆	◆		
Electric Power Research Institute				
ERCOT			◆	
Grid Protection Alliance (Prime)	◆	◆		
ISO New England			◆	
MehtaTech		◆		
Oklahoma Gas & Electric	◆		◆	WSU
OSIsoft		◆		
Peak Reliability			◆	
PingThings		◆		
PJM Interconnection			◆	EPG
Southern California Edison			◆	
San Diego Gas & Electric	◆		◆	WSU
Schweitzer Engineering Laboratories	◆	◆		
Southern Company Services			◆	
Southwest Power Pool	◆		◆	WSU
Space-Time Insight		◆		
Trudnowski & Donnelly Consulting Engineers		◆		
Utilicast	◆	◆		
Tennessee Valley Authority	◆		◆	WSU
University of Southern California				
V&R Energy		◆		
Washington State University	◆	◆		
26	11	11	12	6

Specification Copyright Statement

Major Version History

Version	Date	Notes
0.1	TBD, 2017	Initial draft for validation of use of markdown
0.0	June 15, 2017	Specification template

Appendix A - STTP API Reference

appendix body

appendix body

> 💡 Links to language specific auto-generated XML code comment based API documentation would be useful.

Appendix B - IEEE C37.118 Mapping

appendix body

appendix body

Specification Development To-Do List

- ☑ Determine the location for posting images (June 19, 2017)
- ☐ Sample item 2 (date)
- ☐ Sample item 3 (date)