Advanced Synchrophasor Protocol

J. Ritchie Carroll
Grid Protection Alliance

NASPI Fall Meeting
Springfield, MA
September 26, 2017
Industry Value – Less Data Loss / Lower Bandwidth

Data from testing at PeakRC
## Deliverables

- Update PMP
- Release ASP Spec
- Develop α Toolkit
- Develop Demo Plan
- Publish Demo Results
- Publish API

### ASP Project Timeline

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Otr May-Jul</td>
<td>1 Otr May-Jul</td>
</tr>
<tr>
<td>2 Otr Aug-Oct</td>
<td>2 Otr Aug-Oct</td>
</tr>
<tr>
<td>3 Otr Nov-Jan</td>
<td>3 Otr Nov-Jan</td>
</tr>
<tr>
<td>4 Otr Feb-Apr</td>
<td>4 Otr Feb-Apr</td>
</tr>
</tbody>
</table>

#### 1.0 Project Governance
- 1.1 Update PMP (D1)
- 1.2 Update Data Management Plan
- 1.3 Establish Contracts
- 1.4 Manage Project and Submit Reports

#### 2.0 Protocol Specification
- 2.1 Define Requirements
- 2.2 Create Initial Design (M1)
- 2.3 Release ASP Specification (D2 - M2)

#### 3.0 Alpha Software Development
- 3.1 Develop Alpha APIs
- 3.2 Develop Alpha Tool Kit (D3)
- 3.3 Release Alpha Versions (M3)

#### 4.0 Incorporate the ASP APIs into Tool Suites
- 4.1 Incorporate Alpha ASP into EPG Tools
- 4.2 Incorporate Alpha ASP in WSU Tools
- 4.3 Bench Test EPG and WSU Tools

#### 5.0 Demonstrations and Final ASP Specification
- 5.1 Develop EPG Tool Demo Plan (D4)
- 5.2 Develop WSU Too Demo Plan
- 5.3 Conduct Demo & Publish Results (D5-M4)
- 5.4 Publish Ver 1.0 API with Documentation (D6 - M5)

*Sept 2017*
# Project Partners

## Advanced Synchrophasor Protocol Project

**Streaming Telemetry Transport Protocol**

### Project Collaborators

<table>
<thead>
<tr>
<th>Project Collaborator</th>
<th>Project Partner</th>
<th>Vendor</th>
<th>Utility</th>
<th>Demonstration Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bonneville Power Administration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bridge Energy Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dominion Energy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electric Power Group</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Electric Power Research Institute</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERCOT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid Protection Alliance (Prime)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO New England</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MehtaTech</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oklahoma Gas &amp; Electric</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>WSU</td>
</tr>
<tr>
<td>OSiSoft</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peak Reliability</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PingThings</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PJM Interconnection</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Southern California Edison</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>San Diego Gas &amp; Electric</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schweitzer Engineering Laboratories</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Southern Company Services</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Southwest Power Pool</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space-Time Insight</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trudnowski &amp; Donnelly Consulting Engineers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Utilicast</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tennessee Valley Authority</td>
<td></td>
<td>✓</td>
<td></td>
<td>WSU</td>
</tr>
<tr>
<td>University of Southern California</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V&amp;R Energy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Washington State University</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Counts:
- 26 Total Collaborators
- 11 Vendor Collaborators
- 11 Utility Collaborators
- 12 Demonstration Host Collaborators
To make STTP easy to use, an API will be provided.
The Core – What’s Moving On the Wire

The Wire Protocol

sttp
Base Interface Implemented with Commands

The Wire Protocol

STTP Commands

ENCODER

DECODER

STTP

STTP Commands
API Hides the Wire Level Details

Provided by the ASP Project

The Wire Protocol

Low-Level API

STTP Commands

sttp

STTP Commands

Low-Level API

PUB SUB

ENCODE

DECODE

PUB SUB

ENCODE

DECODE

PUB SUB

NASPI Work Group Meeting - Springfield, MA - September 26, 2017
Application Layer Enables More Complex Integrations

Key STTP Requirements:
- Performant Data Exchange at Scale
- Extensible Metadata
- Access Control and Security
- Bidirectional Connectivity

Provided by the ASP Project

The Wire Protocol
Applications & Advanced APIs

Initial Common Use Case

- Complex Structure Encoding (e.g., IEEE C37.118)
  - Includes, as needed, data concentration at final consumer

Diagram:
- IEEE C37.118 In
- STTP
- IEEE C37.118
- Publisher
- Subscriber
- Existing Tool/Analytic that only Speaks IEEE C37.118
Advanced Data Logic
- Variable distribution of redundantly measured values
- Blue-sky state data reduction (for apps that desire this)

Gateway transmission of other protocol data
- ICCP, DNP3, Modbus, OPC, OpenFMB

Dynamic Data Volume
- Adjust data publication volume based on system conditions, e.g., sending more information when an event has been detected for increased monitoring and detail (where desired)
The STTP API

Publisher

- **Methods**
  - Connect
  - DefineMetadata
  - Disconnect
  - DisconnectSubscriber
  - SendData

- **Callbacks / Events**
  - SubscriberConnected
  - SubscriberSessionEstablished
  - SubscriberDisconnected

Subscriber

- **Methods**
  - Connect
  - Disconnect
  - RequestMetadataTables
  - RequestMetadata
  - Subscribe
  - Unsubscribe
  - SecureDataChannel

- **Callbacks / Events**
  - ReceivedMetadataTables
  - ReceivedMetadata
  - ReceivedDataPoints
STTP API Provides Access to Metadata

- **Core DataPoint Metadata**
  - Point ID (guid)
  - Device ID (guid)
  - Tag (string)
  - AlternateTag (string)
  - Description (string)
  - Enabled (bool)
  - Created (date-time)
  - Updated (date-time)

- **Device Metadata**
  - Device ID (guid)
  - Name (string)
  - etc.

- **Synchrophasor Metadata**
  - Point ID (guid)
  - SignalReference (string)
  - Protocol (string)
  - SignalType (string)
  - EngineeringUnits (string)
  - PhasorType (string)
  - Phase (string)
  - DataRate (float)
  - etc.
Commands & Responses

 Commands

- NegotiateSession: Establishes connection and encoding rules
- MetadataRefresh: Requests publisher send requested metadata
- Subscribe: Requests publisher start sending requested data
- Unsubscribe: Requests publisher stop sending data
- SecureDataChannel: Establishes security for UDP channel, if used
- RuntimeIDMapping: Defines runtime ID mappings for data points
- DataPointPacket: Defines set of published data points
- NoOp: Used to validate connectivity

 Responses

- Succeeded: Response for succeeded command
- Failed: Response for failed command
Data Point Packet Command

Command:
- uint8 commandCode
- uint16 length
- uint8[] payload

Payload for DataPointPacket Command (commandCode = 0x06):

- **A.Element1**
  - ID
  - Value
  - Time
  - Quality

- **A.Element2**
  - ID
  - Value
  - Time
  - Quality

- **B.Element1**
  - ID
  - Value
  - Time
  - Quality

- **B.Element2**
  - ID
  - Value
  - Time
  - Quality

Pass Payload to Compression Algorithm

Publish Command Bytes to IP Layer
Data Packet Payload: Data Points

Data Packet Command has a target size, e.g., MTU of 1,500 bytes minus 40-byte TCP header = 1,460 bytes:

Command Code = 0x06

STTP Commands

1. Data Packet Command
   - struct {
     uint32 id;
     uint8[] value;
     uint8[] state;
   } DataPoint;

2. Data Point
   - struct {
     uint32 id;
     uint8[] value;
     uint8[] state;
   } DataPoint;

3. Data Point
   - struct {
     uint32 id;
     uint8[] value;
     uint8[] state;
   } DataPoint;

The total number of Data Points per Data Packet Command payload is variable and depends on the size of each Data Point.
Routing Data to a Subscriber

- A-Data (Priority 1)
- B-Data (Priority 2)
- Metadata

**Routing Processor**

- Only data intended for subscriber will be published.
- The STTP data is interleaved.

**Publisher**

- Individual Data Point

**Subscriber**
Structured data payloads are encoded at a binary level and transmitted over the “wire” using the Internet Protocol (IP)

IP based connections use TCP for commands and optionally UDP for data transmission:
- TCP provides reliable communications allowing for high-yield stateful compression
- UDP can be used for data transmission with the potential for UDP data loss and with less compression than TCP*

* Methods to implement STTP in Unicast/Multicast only configurations will be documented for use cases where a “no command” based STTP may represent a preferred option over Unicast/Multicast IEEE C37.118.
Wire Protocol Security

- Security at Socket Layer (over TCP)
  - Primary security is added at the socket using industry standard Transport Layer Security (TLS or SSL)
  - X.509 certificates are used to authenticate connections and provide encryption through public key infrastructure

- UDP Security
  - When existing command channel is secured with TLS, UDP uses AES symmetric encryption with keys exchanged over the TLS secure channel
Two Types of Connections Supported

**Forward**
- *Subscriber connects to Publisher* – typical operation where a listening server-based publisher with connecting client-based subscribers

**Reverse**
- *Publisher connects to Subscriber* – operation where client-based publisher connects to listening server-based subscriber; used to cross security zones in desired direction

Bidirectional Communications Allowed
- Once connection is established, publisher/subscriber functions can operate in either direction over the single connection
Publisher and Subscriber operations are “functions” in STTP – not “objects”

As such, a publisher “sends” data and a subscriber “receives” data – always

Crossing Security Zone

The Wire Protocol

sttp

Secure Environment

Data Source / Publisher

Data Sync / Subscriber

Corporate Network

Client-Style Connection

Server-Style Connection

Firewall allows connections out but not in

Connection

Data Flow
Wire Level Structure and Payload Examples

The Wire Protocol

```
struct {
    uint8 commandCode;
    uint16 length;
    uint8[] payload;
} Command;

struct {
    uint8 responseCode;
    uint8 commandCode;
    uint16 length;
    uint8[] payload;
} Response;
```

```
enum {
    Null = 0,       // 0-bytes
    SByte = 1,      // 1-byte
    Int8 = 2,       // 2-bytes
    Int16 = 3,      // 4-bytes
    Int32 = 4,      // 8-bytes
    Byte = 5,       // 1-byte
    UInt8 = 6,      // 2-bytes
    UInt16 = 7,     // 4-bytes
    UInt32 = 8,     // 8-bytes
    Decima = 9,     // 16-bytes
    Double = 10,    // 8-bytes
    Single = 11,    // 4-bytes
    Ticks = 12,     // 8-bytes
    Bool = 13,      // 1-byte
    Guid = 14,      // 16-bytes
    String = 15,    // 64-bytes, max
    Buffer = 16     // 64-bytes, max
} ValueType;     // sizeof(uint8), 1-byte
```

```
enum {
    Normal = 0,     // Defines normal state
    BadTime = 1 << 0, // Defines bad time state
    BadValue = 1 << 1, // Defines bad value state
    UnreasonableValue = 1 << 2, // Defines unreasonable value state
    CalculatedValue = 1 << 3, // Defines calculated value state
    ReservedFlag1 = 1 << 4, // Defines reserved flag 1
    ReservedFlag2 = 1 << 5, // Defines reserved flag 1
    UserDefinedFlag1 = 1 << 6, // Defines user defined flag 1
    UserDefinedFlag2 = 1 << 7, // Defines user defined flag 1
} QualityFlags;  // sizeof(uint8), 1-byte
```

```
struct {
    StringEncodingFlags encodings;
    uint16 udpPort;
    NamedVersions stateful;
    NamedVersions stateless;
} OperationalModes;
```
The STTP Specification

- Specification development is open on GitHub:
- First draft release (Version 0.8) – November 1
- Daily builds of specification are available in PDF, HTML and GitHub markdown formats

- Topics include:
  - Protocol Overview
  - Establishing Connections
  - Commands and Responses
  - Compression
  - Security
  - among others

- Anyone can propose an edit with a pull-request
  - See “How to Contribute” on spec site for details

https://github.com/stttp
Some Current Specification Issues

- Should the specification be targeted for “general industrial process data exchange” rather than specifically for the electric industry?
- Should the protocol be able to support non-IP protocols communications?
- What is the mandatory minimum set of metadata?
- How best to support a unidirectional data feed (UDP only)?
- What is the minimum set of target languages? (More than C, C# and Java?)
Demonstrations

- **WSU Tools at:**
  - TVA, SDG&E, SPP, OG&E

- **EPG Tools at:**
  - PJM, Dominion