J. Ritchie Carroll

Grid Protection Alliance

# Advanced Synchrophasor Protocol

## NASPI Fall Meeting

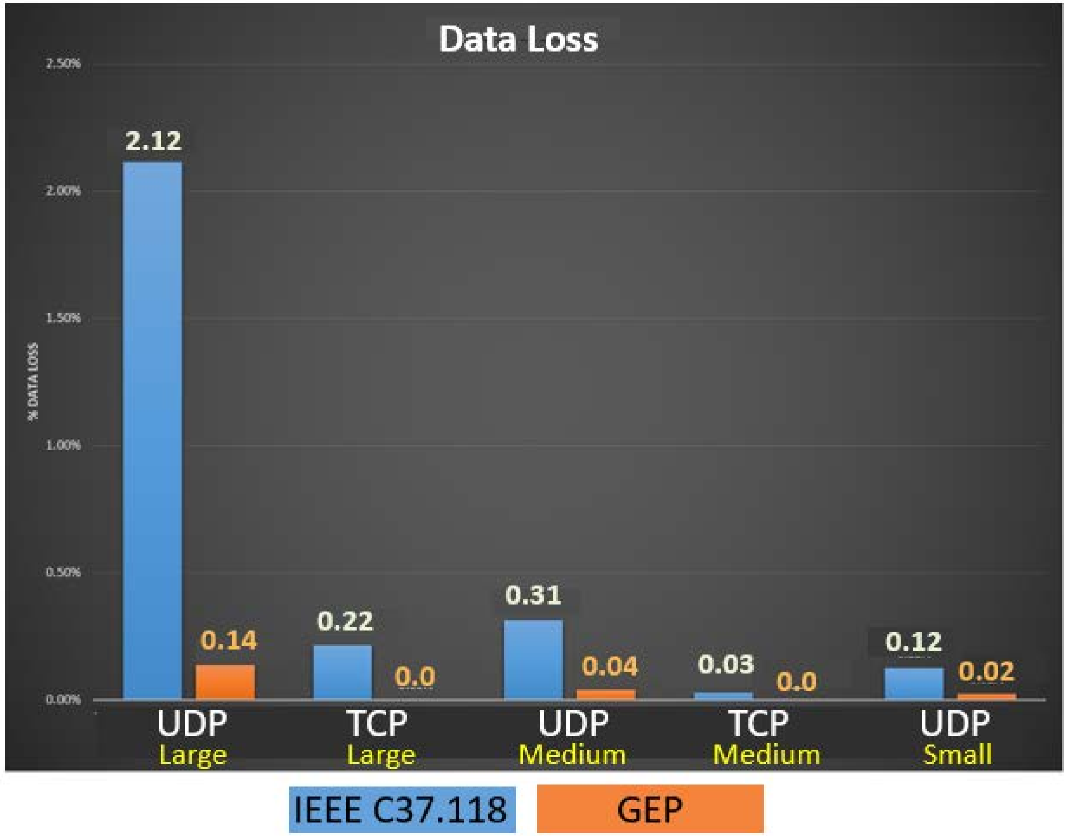
Springfield, MA

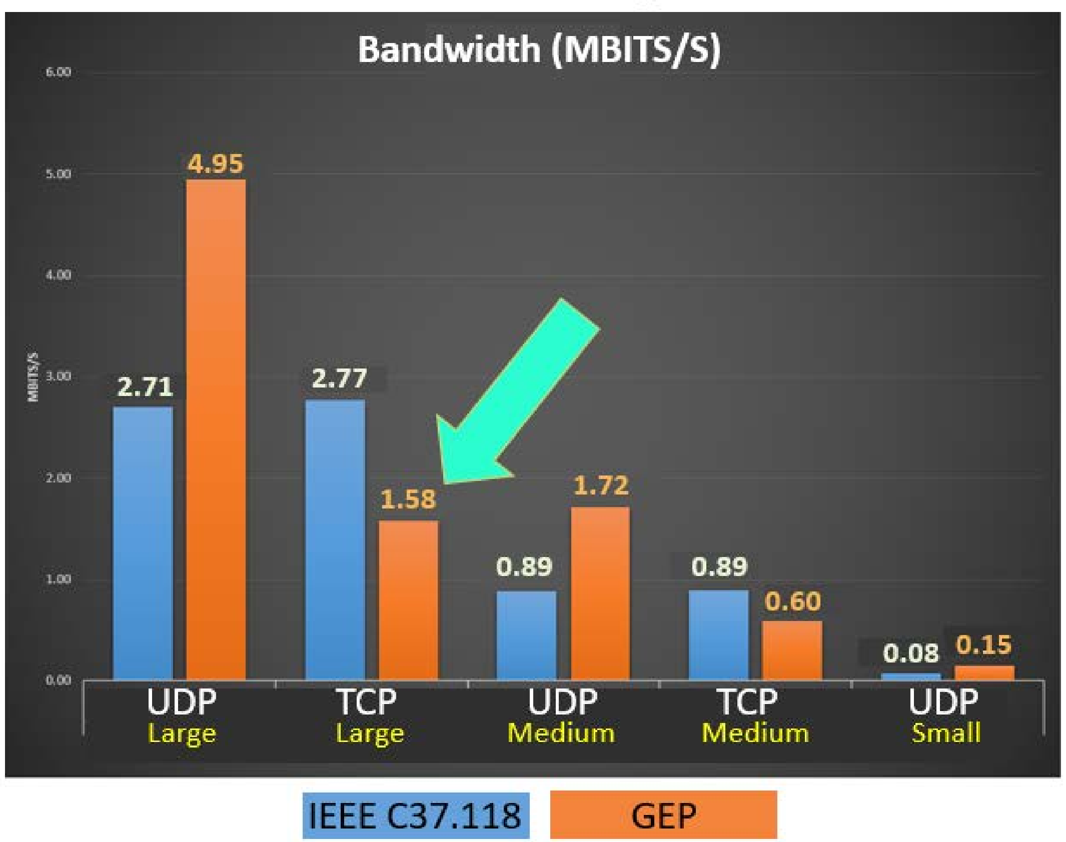
September 26, 2017

DOE FOA 1492

DE-OE0000859

# Industry Value – Less Data Loss / Lower Bandwidth



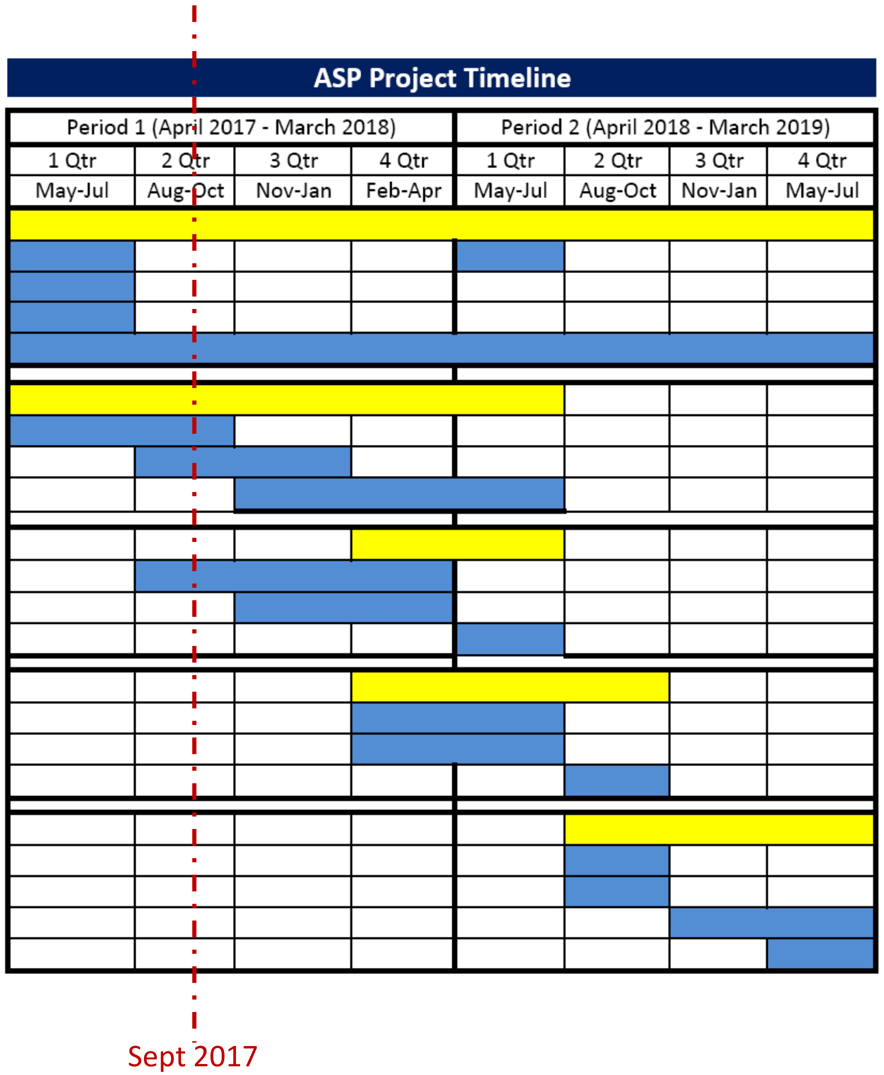


Data from testing at PeakRC

# Schedule and Deliverables

## Deliverables

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



**ASP Project Timeline**

| | Period 1 (April 2017 - March 2018) | | | | Period 2 (April 2018 - March 2019) | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 Qtr | 2 Qtr | 3 Qtr | 4 Qtr | 1 Qtr | 2 Qtr | 3 Qtr | 4 Qtr |
| | May-Jul | Aug-Oct | Nov-Jan | Feb-Apr | May-Jul | Aug-Oct | Nov-Jan | May-Jul |
| **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) | | | | | | | | 🟦 |

# Project Partners

Advanced Synchrophasor Protocol Project

sttp

DOE FOA 1492
DE-OE0000859

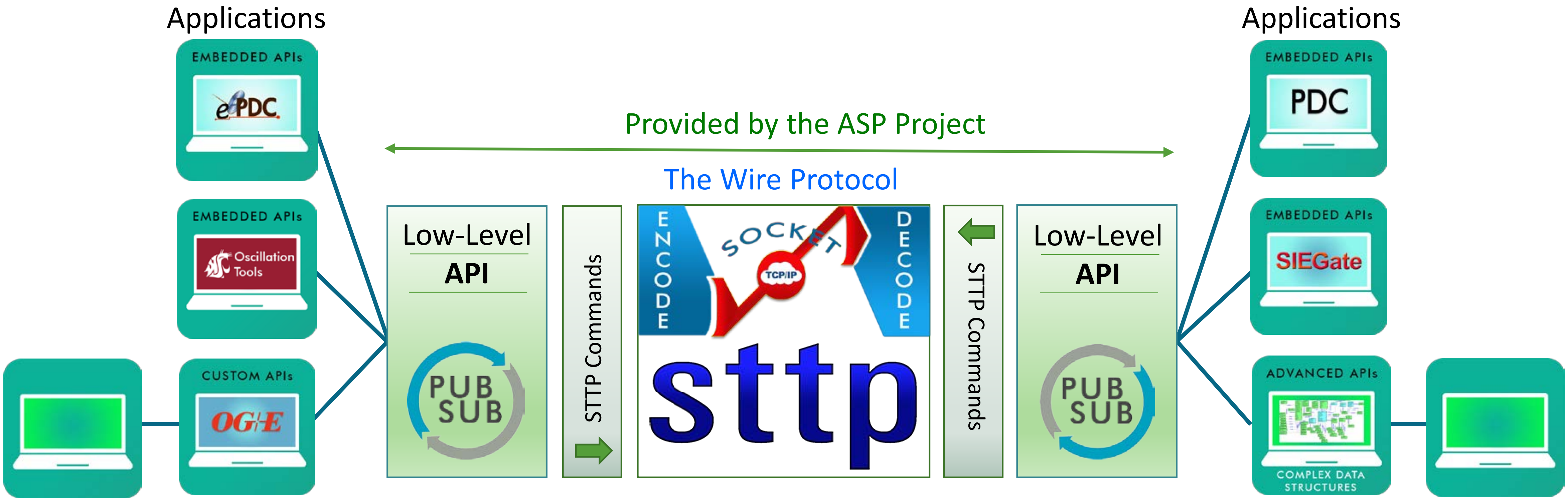
ASP

Streaming Telemetry Transport Protocol

GRID PROTECTION ALLIANCE
BONNEVILLE POWER ADMINISTRATION
WASHINGTON STATE UNIVERSITY
Electric Power Group
SEL SCHWEITZER ENGINEERING LABORATORIES
SPP Southwest Power Pool
OG&E
TVA
SDG&E
Dominion Energy
pjm
Utilicast
PEAKRELIABILITY assuring the wide area view
SOUTHERN CALIFORNIA EDISON
ercot
ISO new england
SOUTHERN COMPANY
EPRI ELECTRIC POWER RESEARCH INSTITUTE
USC University of Southern California
OSIsoft.
BRIDGE ENERGY GROUP
SPACE-TIME INSIGHT
PingThings
T&D Consulting Engineers
MEHTA TECH, INC.

| 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 |

# To make STTP easy to use, an API will be provided



Applications

Provided by the ASP Project

The Wire Protocol

Low-Level **API**

STTP Commands

STTP Commands

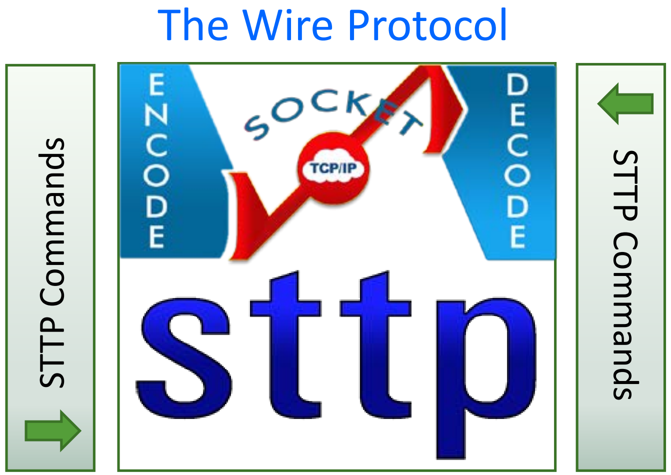
Low-Level **API**

Applications

# The Core – What's Moving On the Wire

The Wire Protocol

# Base Interface Implemented with Commands

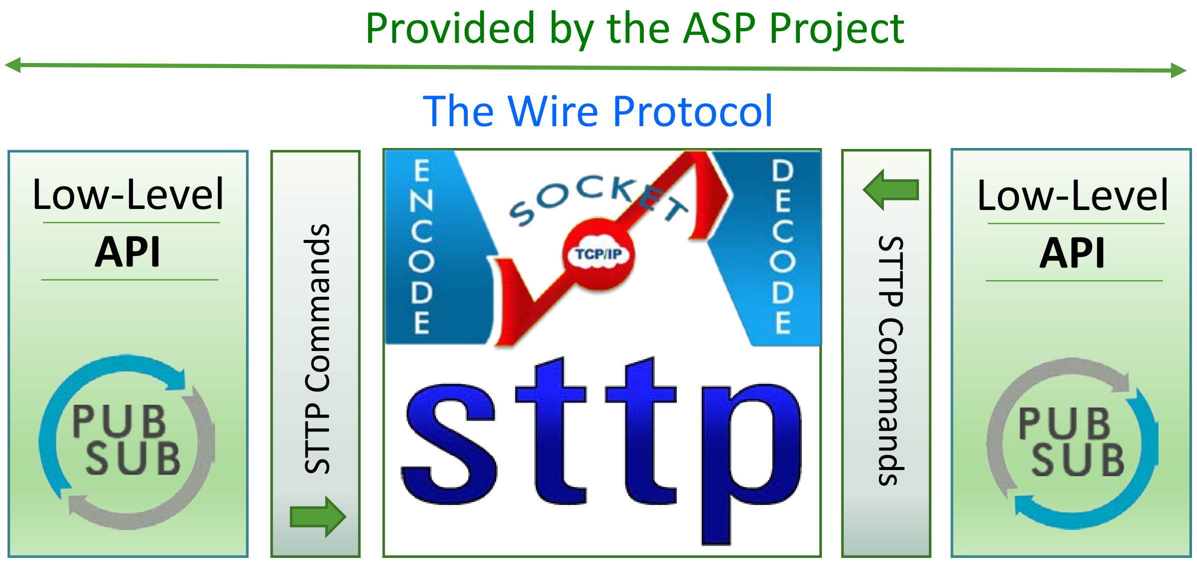


The Wire Protocol

STTP Commands

STTP Commands

# API Hides the Wire Level Details



Provided by the ASP Project

The Wire Protocol

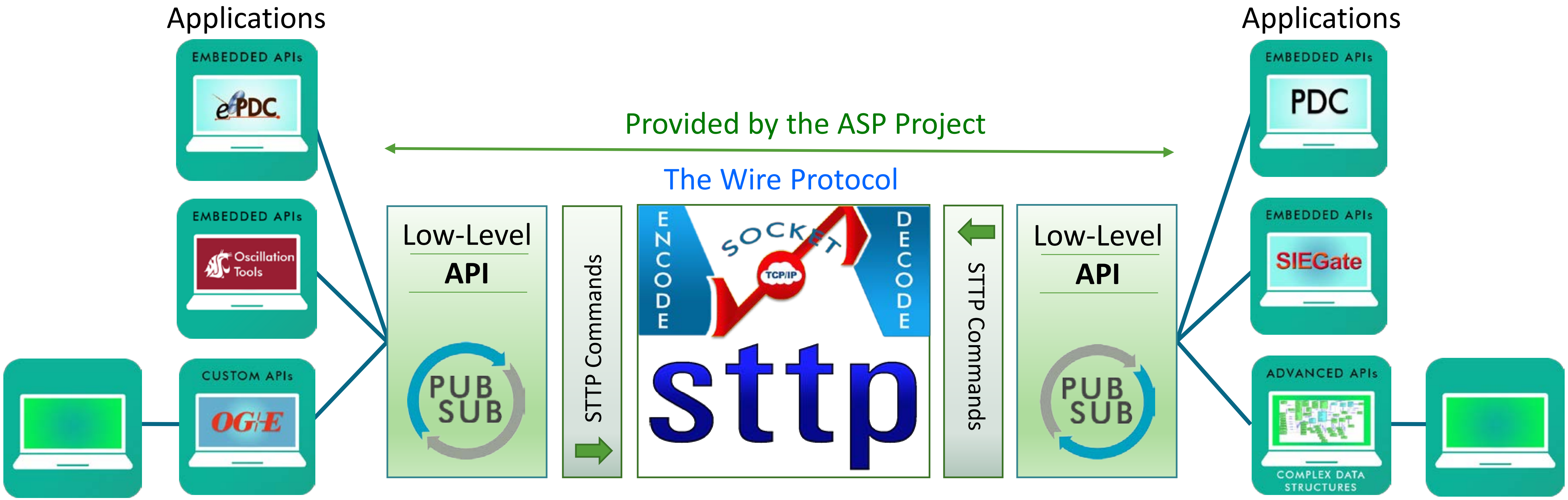
Low-Level API

STTP Commands

STTP Commands

Low-Level API

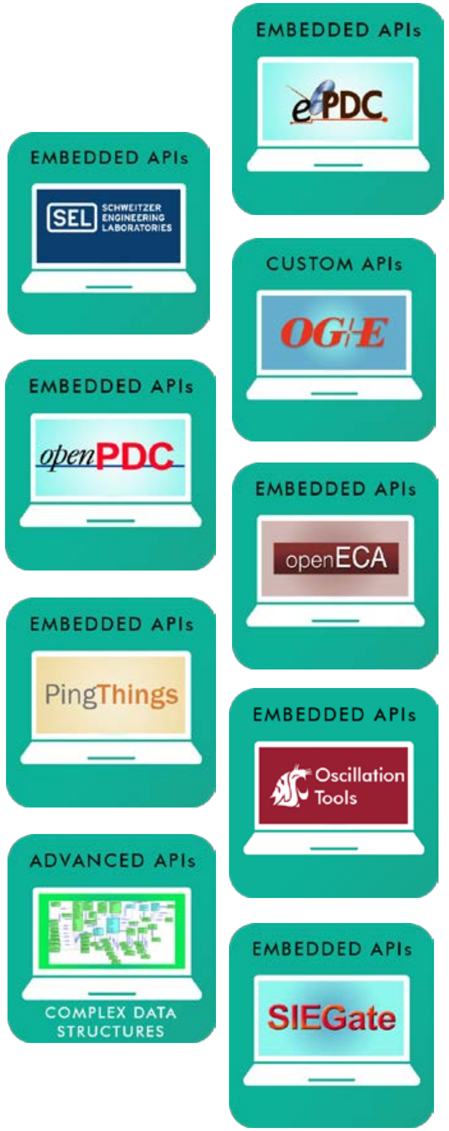
# Application Layer Enables More Complex Integrations



<u>Key STTP Requirements:</u>

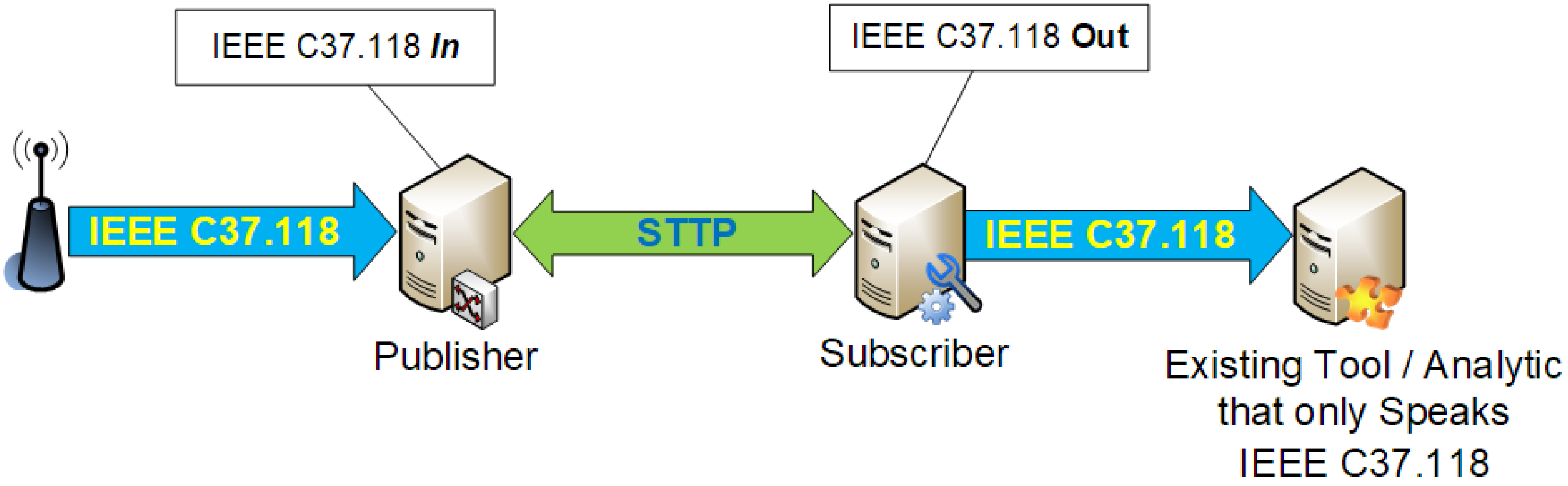
- Performant Data Exchange at Scale
- Extensible Metadata
- Access Control and Security
- Bidirectional Connectivity

# Applications & Advanced APIs

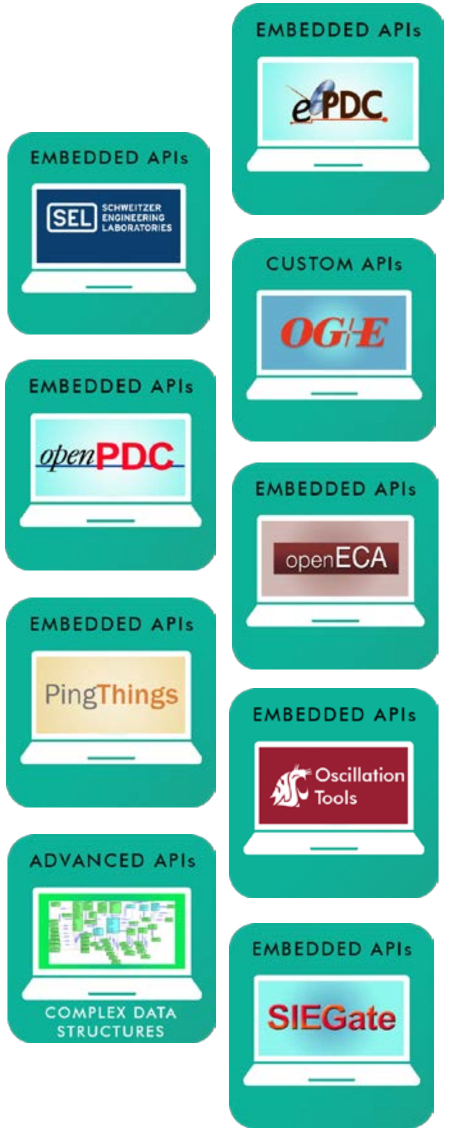


## Initial Common Use Case

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



# Applications & Advanced APIs



- 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

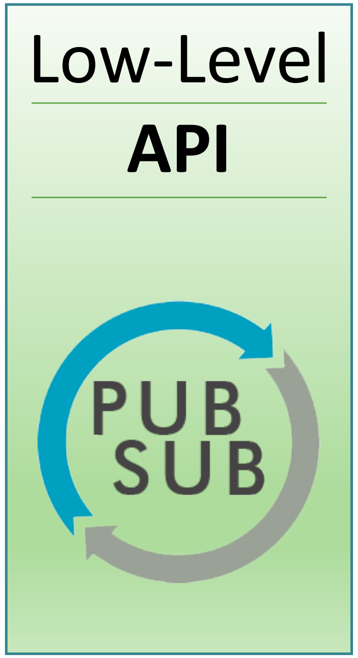
Low-Level

**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

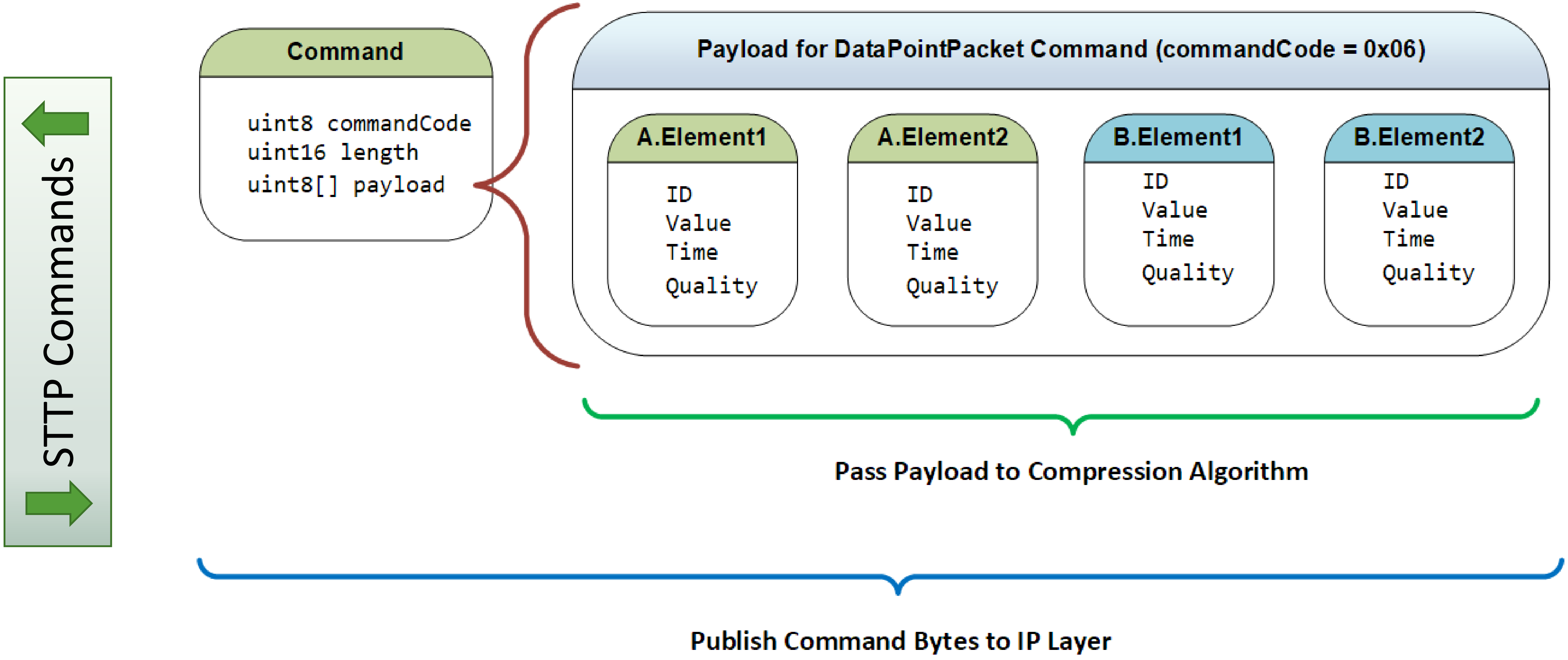
## Commands

| | |
|---|---|
| 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



STTP Commands

Command

```
uint8 commandCode
uint16 length
uint8[] payload
```

Payload for DataPointPacket Command (commandCode = 0x06)

| A.Element1 | A.Element2 | B.Element1 | B.Element2 |
|---|---|---|---|
| ID<br>Value<br>Time<br>Quality | ID<br>Value<br>Time<br>Quality | ID<br>Value<br>Time<br>Quality | ID<br>Value<br>Time<br>Quality |

Pass Payload to Compression Algorithm

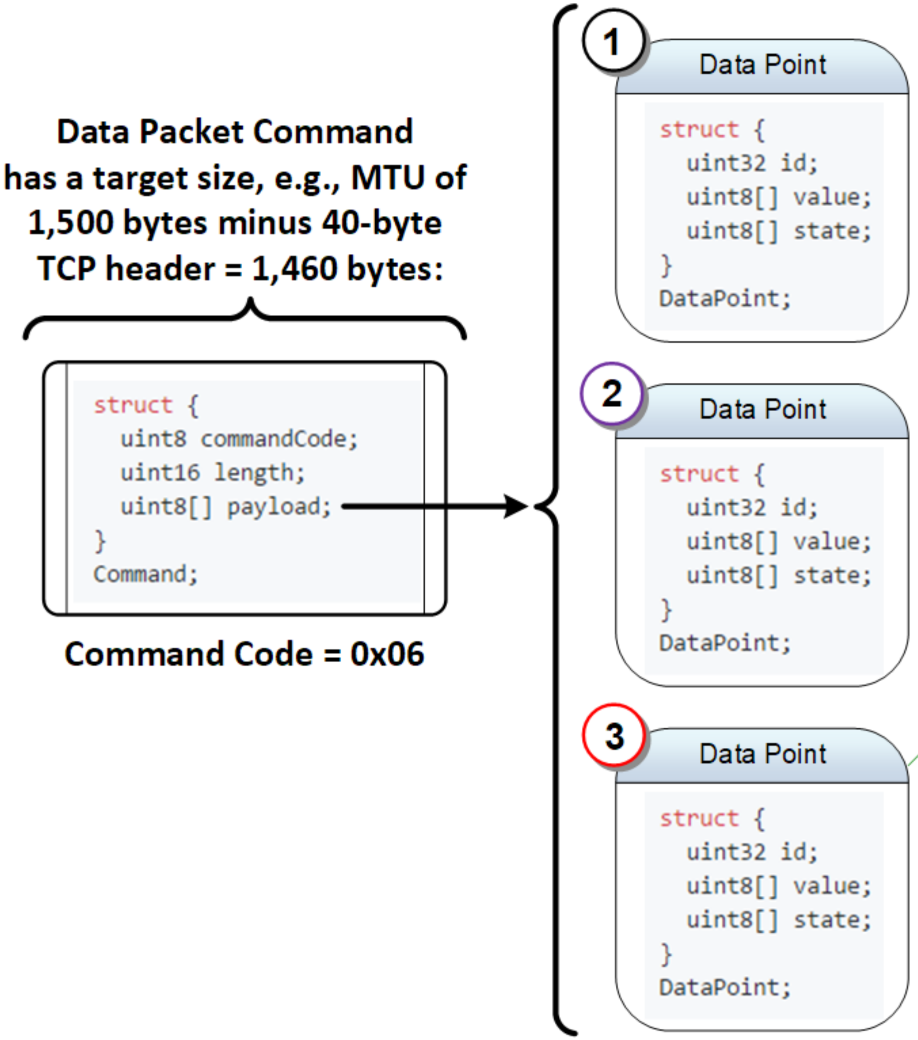
Publish Command Bytes to IP Layer

# Data Packet Payload: Data Points

STTP Commands

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

Data Packet Command payload is a set of Data Points:



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

Command Code = 0x06

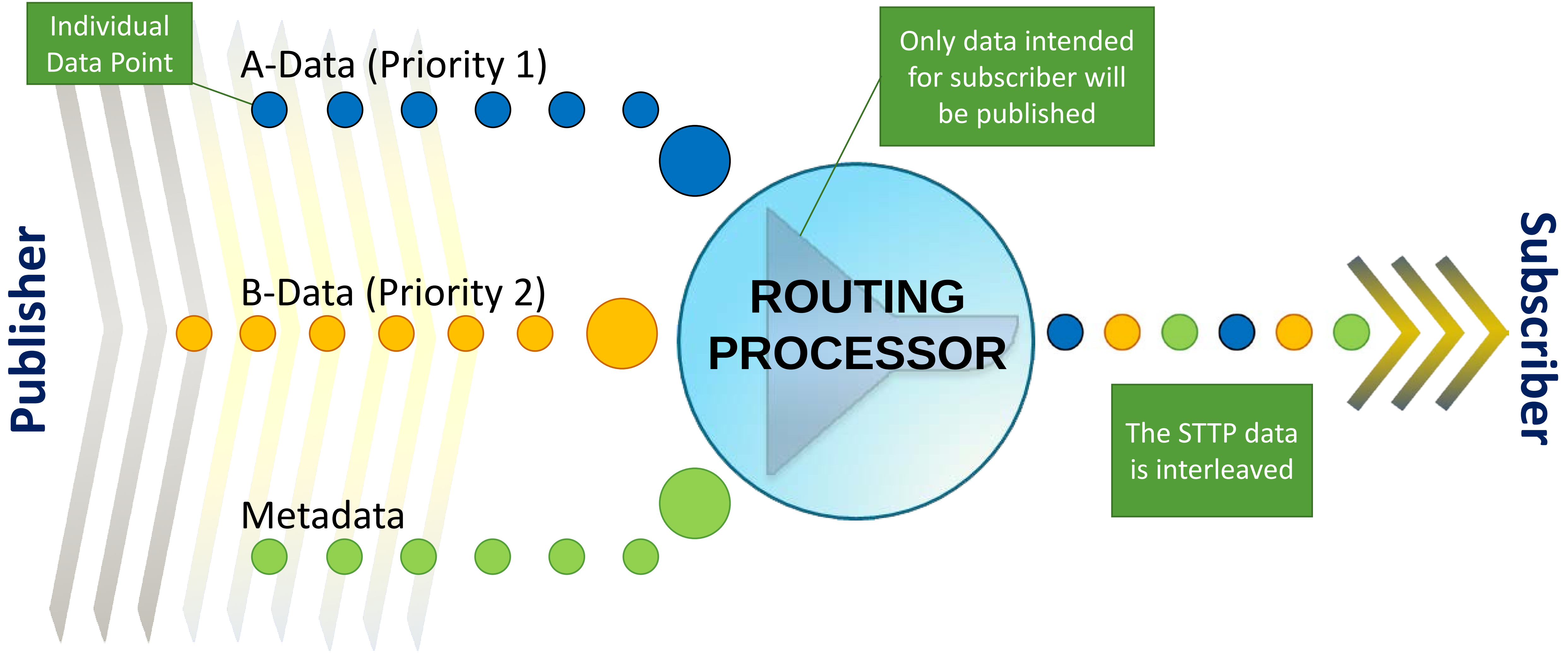
```
struct {
  uint32 id;
  uint8[] value;
  uint8[] state;
}
DataPoint;
```

```
struct {
  uint32 id;
  uint8[] value;
  uint8[] state;
}
DataPoint;
```

```
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



# The Wire Protocol

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

The Wire Protocol

- 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

The Wire Protocol

- 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

# Wire Protocol Connections

The Wire Protocol

- 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

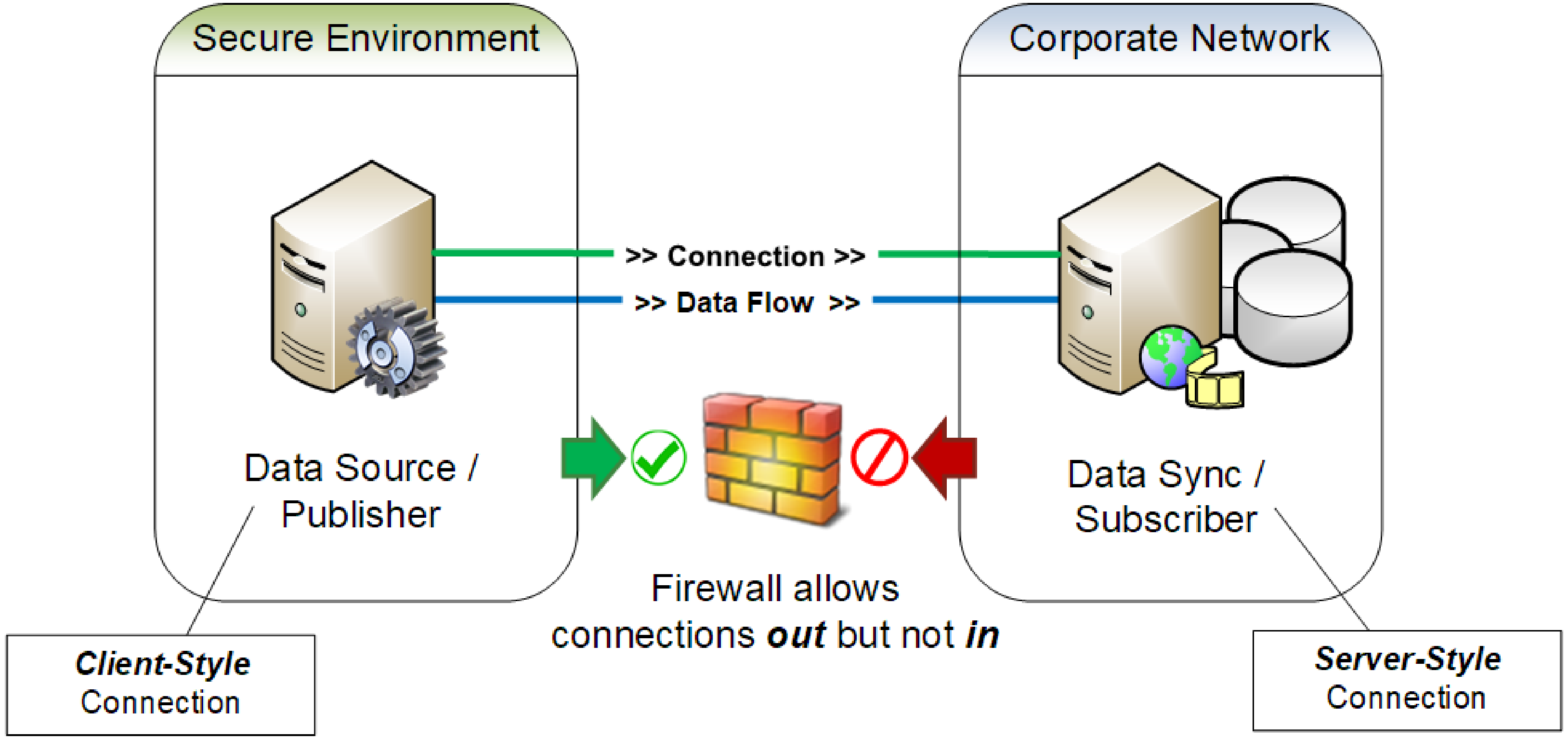
- Bidirectional Communications Allowed
  - Once connection is established, publisher/subscriber functions can operate in either direction over the single connection

# Reverse Connection Use Case

- Publisher and Subscriber operations are “*functions*” in STTP – not “*objects*”
- As such, a publisher “*sends*” data and a subscriber “*receives*” data – always

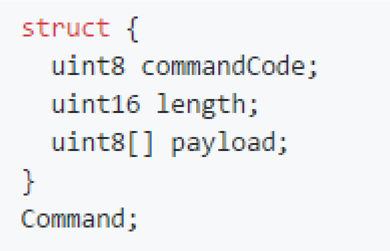
The Wire Protocol

Crossing Security Zone

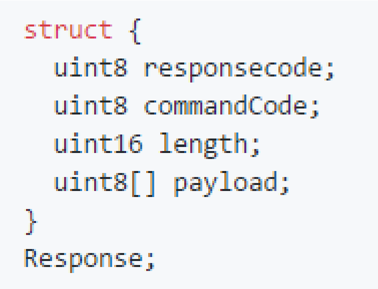

Secure Environment
Corporate Network
>> Connection >>
>> Data Flow >>
Data Source / Publisher
Data Sync / Subscriber
Firewall allows connections ***out*** but not ***in***
***Client-Style*** Connection
***Server-Style*** Connection

# Wire Level Structure and Payload Examples

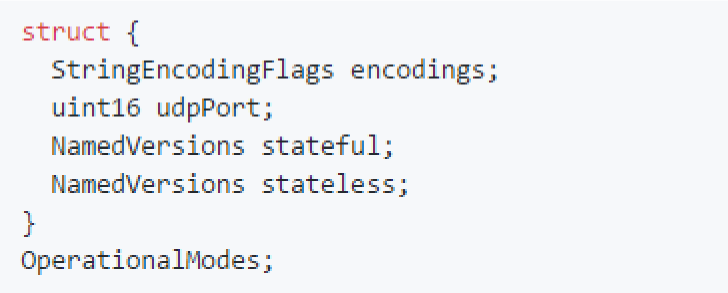
The Wire Protocol



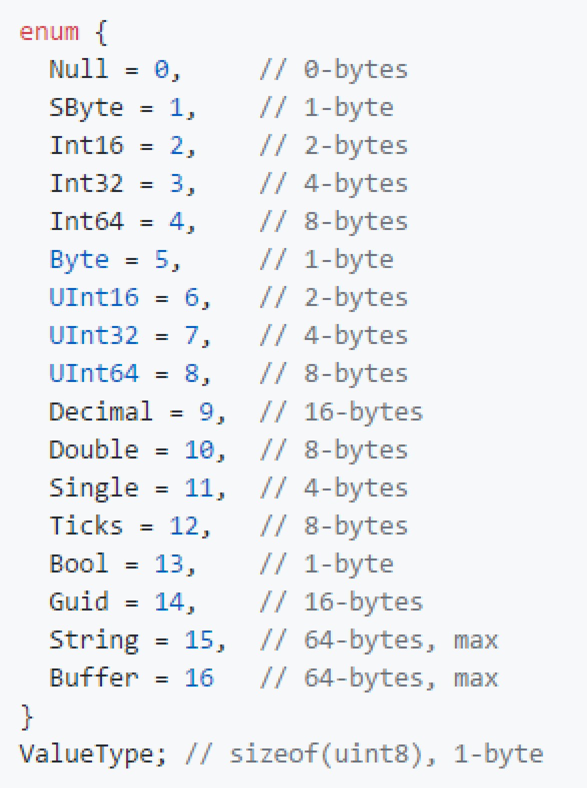
```
struct {
  uint8 commandCode;
  uint16 length;
  uint8[] payload;
}
Command;
```



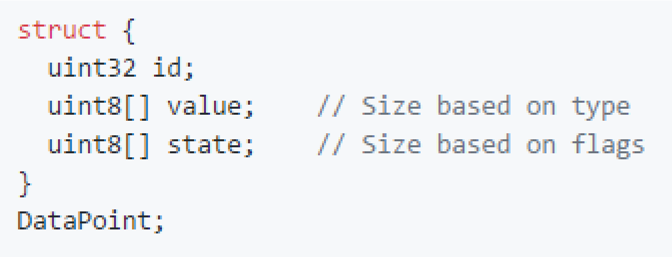
```
struct {
  uint8 responsecode;
  uint8 commandCode;
  uint16 length;
  uint8[] payload;
}
Response;
```



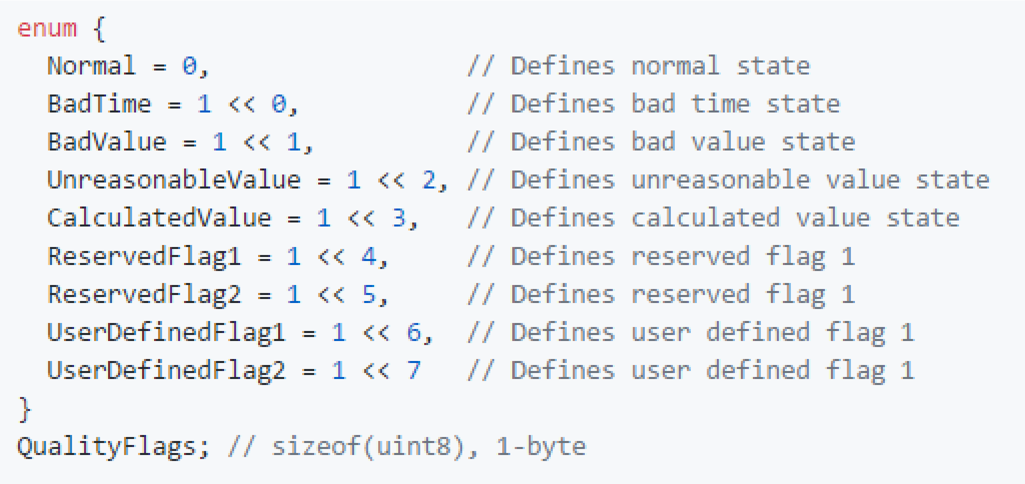
```
struct {
  StringEncodingFlags encodings;
  uint16 udpPort;
  NamedVersions stateful;
  NamedVersions stateless;
}
OperationalModes;
```



```
enum {
  Null = 0,      // 0-bytes
  SByte = 1,     // 1-byte
  Int16 = 2,     // 2-bytes
  Int32 = 3,     // 4-bytes
  Int64 = 4,     // 8-bytes
  Byte = 5,      // 1-byte
  UInt16 = 6,    // 2-bytes
  UInt32 = 7,    // 4-bytes
  UInt64 = 8,    // 8-bytes
  Decimal = 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
```

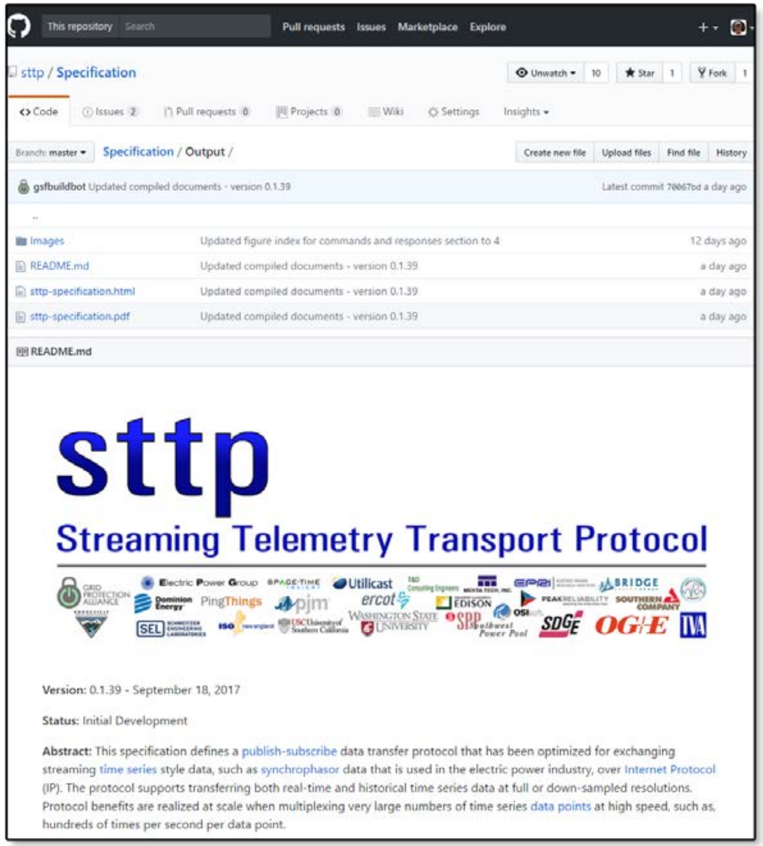


```
struct {
  uint32 id;
  uint8[] value;    // Size based on type
  uint8[] state;    // Size based on flags
}
DataPoint;
```



```
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
```

# The STTP Specification



sttp / Specification

Specification / Output /

gsfbuildbot Updated compiled documents - version 0.1.39 — Latest commit 70067bd a day ago

| | | |
|---|---|---|
| .. | | |
| Images | Updated figure index for commands and responses section to 4 | 12 days ago |
| README.md | Updated compiled documents - version 0.1.39 | a day ago |
| sttp-specification.html | Updated compiled documents - version 0.1.39 | a day ago |
| sttp-specification.pdf | Updated compiled documents - version 0.1.39 | a day ago |

README.md

**sttp**

**Streaming Telemetry Transport Protocol**

**Version:** 0.1.39 - September 18, 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.

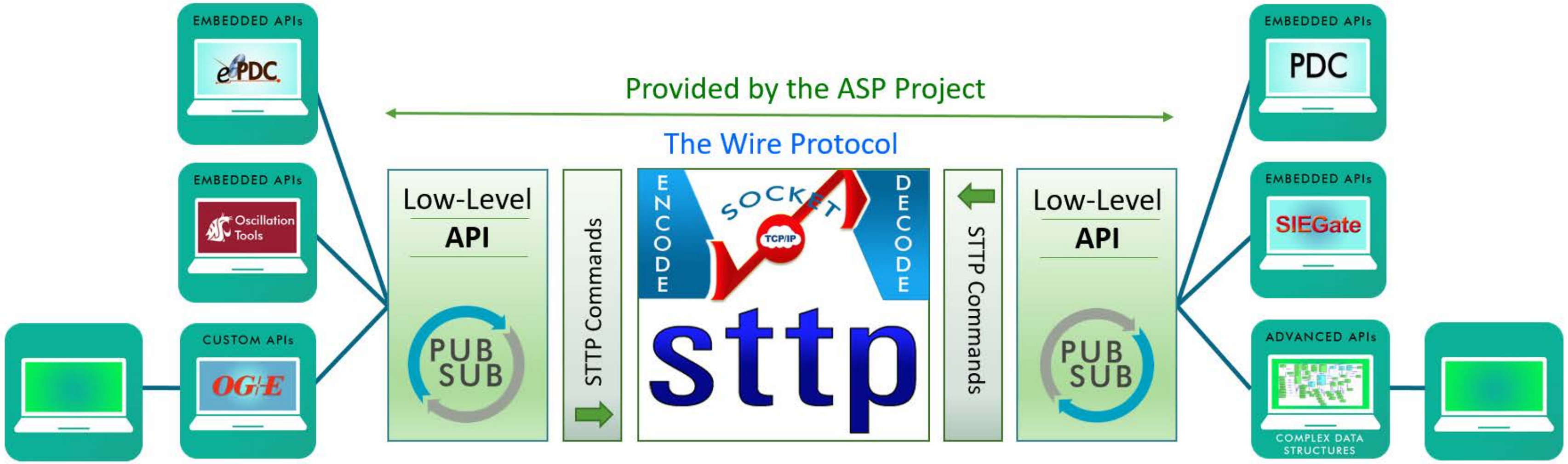
**https://github.com/sttp**

- Specification development is open on GitHub:
  - https://github.com/sttp/Specification
- 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

# 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



## Demonstrations

- WSU Tools at:
  - TVA, SDG&E, SPP, OG&E
- EPG Tools at:
  - PJM, Dominion