Time-Series Library (TSL) Components

GPA's Grid Solutions Framework (GSF) is an extensive collection of open source .NET code. It consists of hundreds of class libraries that extend or expand the functionality included in the .NET Framework. The GSF makes many of the more complex .NET features (e.g., sockets and encryption) easier to use and adds functionality not included in the .NET. For more information see: https://gsf.codeplex.com/

Included in the GSF is the Time-Series Library, or "TSL". This collection of code contains many components that can be used to create open source products that manage time-series data, such as synchrophasor data. GPA has used the TSL as the basis for the openPDC, SIEGate, and the substationSBG among other products. In creating these products, as much functionality as possible is embedded in the TSL so that this functionality can easily be shared by all GPA time-series data processing products.

In addition, GPA has created a generic time-series product template called "TSL Project Alpha". This project template provides a major jump start to those wanting to create their own custom products based on the TSL.

The major components that are included in the TSL are shown in Figure 1 below and they are described in detail on the following pages.

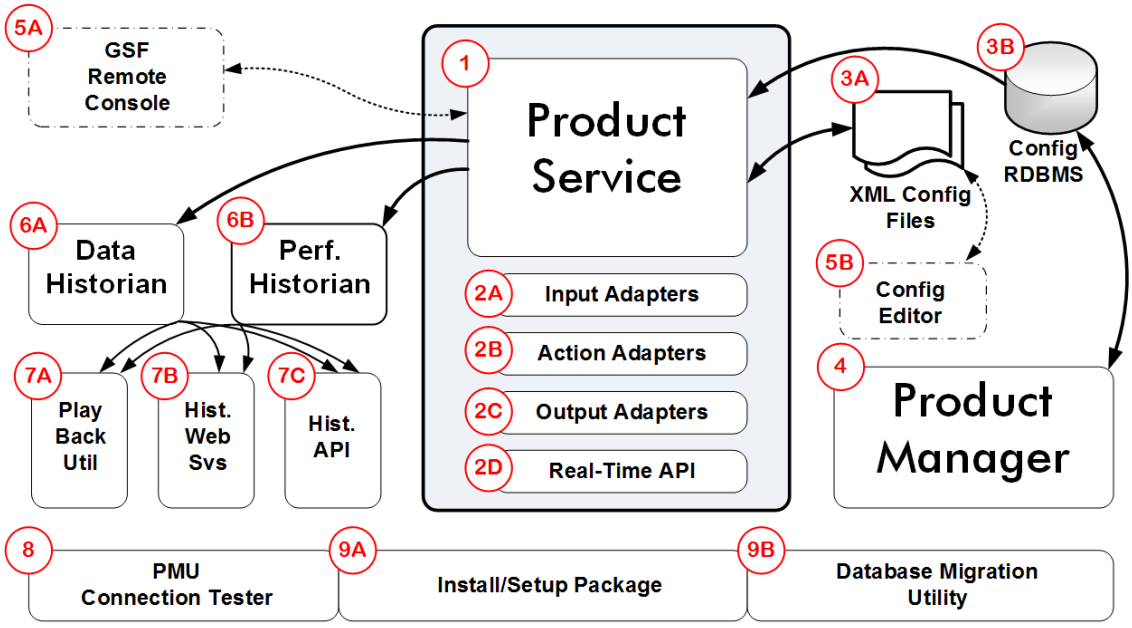

Figure 1. Time-Series Library Components

Product Service 

The "Product Service[1]" is a Windows service that runs in the background on the host server. The Product Service is a time-series measurement processor and routing engine that includes an extensible collection of standard input, action and output adapters that provide its core functionality, such as time-synchronization. The Product Service receives streaming data from time-series measurement devices, such as a Phasor Measurement Units (PMUs) or a Phasor Data Concentrators (PDCs), parses that data into individual measurements, concentrates the data by time, performs predefined analysis and produces multiple data output streams comprised of selected devices at configurable sampling rates.

Input Adapters 

The input adapter layer of the extensible TSL provides a large collection of protocol parsers to allow the Product Service to consume data from external sources. For example, IEEE C37.118 and SEL Fast-Message are parsed by TSL input adapters for the Product Service to obtain synchrophasor data. New input adapters can be easily coded and compiled to add new protocol parsing support to the system.

Action Adapters 

The action adapter layer of the TSL enables time-aligned analytics and calculations at sub-second intervals. Included in the Product Service are time-series data concentration and power calculation methods, such as frequency excursion and loss of field detection. Using Action Adapters "calculated values" can be easily created from measured values. Real-time calculation results are made available to output streams and other calculations at their full sample rate (e.g., 30 samples per second). Custom Action Adapters can be developed to support specialized real-time analytics.

Output Adapters 

The output adapter layer of the TSL buffers streaming time-series data for output to data archiving systems. Buffering is only used if the receiving system goes offline such as occurs during system updates and patching. Included in the TSL is archiving support for relational data systems and commercial historians such as OSI-PI. New output adapters can be easily coded and compiled to support other historical archiving systems.

Real-Time API 

The real-time data API is a high-speed, full resolution publish/subscribe data access mechanism that allows applications to receive point-based data directly from the Product Service. This API is available in .NET, Java, C++ and the Unity 3D platform and can be used by applications requiring high-resolution, streaming time-series data.

[1] For example, the openPDC is considered a "product" as is SIEGate, among others.

Two API calls exist to subscribe to real-time data:

1) `bool SynchronizedSubscribe(bool compactFormat, int framesPerSecond, double lagTime, double leadTime, string filterExpression, bool useLocalClockAsRealTime = false, bool ignoreBadTimestamps = false, bool allowSortsByArrival = true, long timeResolution = Ticks.PerMillisecond, bool allowPreemptivePublishing = true, DownsamplingMethod downsamplingMethod = DownsamplingMethod.LastReceived)`

2) `bool UnsynchronizedSubscribe(bool compactFormat, bool throttled, string filterExpression, double lagTime = 10.0D, double leadTime = 5.0D, bool useLocalClockAsRealTime = false)`

These two API calls provide the following possible real-time data subscriptions:

- A synchronized (i.e., concentrated by time) set of subscribed data points:
```
subscriber.SynchronizedSubscribe(true, 30, 0.5D, 1.0D, "FILTER
ActiveMeasurements WHERE Device = 'SHELBY' AND SignalType = 'VPHM'");
```
- An on-change unsynchronized set of subscribed data points:
```
subscriber.UnsynchronizedSubscribe(true, false, "PPA:87;PPA:92");
```
- A throttled (e.g., down-sampled to every few seconds) set of subscribed data points:
```
subscriber.UnsynchronizedSubscribe(true, true, "POINT:1;POINT:2", 5.0D,
1.0D, false);
```

A web service also exists to extract data in real-time from the Project Service.

Configuration Files 

The Product Service uses XML files for configuration settings and it caches the last known system configuration so that the Product Service can be successfully restarted without a dependency on the health of the relational configuration data base (3B). The primary XML configuration settings file contains settings that control fundamental, slow to change, system behaviors such as the primary configuration source and archiving locations of the openHistorian. The cached XML system configuration file (SystemConfiguration.xml) contains the last known operational system configuration state for information related to input devices, output streams and the active system measurements.

Configuration RDMBS 

The Product Service Configuration Relational Data Base Management System (RDBMS) is typically a main stream data base such as MS SQL Server but can be implemented as a web service or as an XML file. The configuration RDMS is used to store the information necessary to run the Product Service and it includes data such as connection information for time-series measurement devices, output data streams and for adapter configuration. The Product Service can access configuration data stored in one of many types of relational systems and it includes schemas for SQL Server, MySQL, Oracle and SQLite. The last known configuration for the Product Service is always cached to an XML file so the system can continue to function even if access to primary configuration source is unavailable.

Product Manager (4)

The Product Manager is a Windows GUI application used to remotely configure and manage multiple deployed instances of the Product Service from a single workstation. Secured through role-based security integrated with Windows authentication, configuration of system devices, input, action and output adapters and their operational parameters are saved to a RDMBS (3B) for use by the Product Service. All configuration changes are logged and auditable by time and user. The system also allows real-time diagnostic viewing of live streaming data, system statistics and device connection status.

GSF Remote Console (5A)

The Product Service can be remotely monitored with the system diagnostic console monitor. Access to this application requires authentication (e.g., via Active Directory) and is secured through TLS. This low-level application monitors messages and operation of a deployed Product Service. System health, usage statistics and errors are constantly updated with support for optional requests, where rights are available, for actions and more detail on individual adapters.

Core Configuration Editor (5B)

The Product Service Core Configuration Editor is a Windows GUI application that allows system administrators to modify low-level Product Service settings fundamental to system behavior. These settings are stored in an XML configuration settings file. The Core Configuration Editor provides setting descriptions to simplify system changes. It also automatically restarts the Product Service to apply new settings.

Data Historian (6A)

The Data Historian is the primary data archive defined for the Product Service to locally store time-series measurement data, such as synchrophasor data. This data archival system can be GPA's openHistorian, a commercial historian or an RDBMS. When using the openHistorian binary ".d" files (an open format) are created that can be accessed through the Historian Trending Tool (7A), near real-time web-services (7B) and a full-featured API (7C).

Performance Statistics Historian (6B)

The Product Service uses a local instance of the openHistorian to store system performance statistics every ten seconds. A default set of statistics are defined for each input connection, output stream and system status. New custom statistics can be dynamically added to the system and then viewed in real-time with the Product Manager (4).

Historian Trending Tool (7A)

The Historian Trending Tool is Windows GUI application that can be used to export a selected time-range of archived data in either COMTRADE or CSV formats. As its name implies, it can also be used to plot selected measurements or calculated values for comparison.

*open*Historian Web Services 

Data in the openHistorian can be requested via restful web services (REST). This web service implements actions such as requesting real-time or historic data for sets of points matching a filter expression. The REST services are applied to each instance of the openHistorian that is running in-process with the Product Service. The web service also supports SOAP, JSON and WS-Security formats.

Historian API 

The openHistorian includes a low-level code-based API for directly reading and writing time-series data. This real-time data access API would be used when the web service was not sufficient because high-speed access to large volumes was necessary or the system needed to write data as fast as possible. The API parameters include options for time range, desired points and down-sampling options.

PMU Connection Tester 

The PMU Connection Tester is a Windows GUI application used to validate, test and trouble-shoot connections and data streams from phasor measurement devices as well as graphically visualize their synchrophasor data in real-time.

Installation/Setup Package 

The Product Service Installation/Setup Package is a Windows GUI application used to initially install and configure the Product Service., IT guides the user through a step-by-setup setup process with an easy to navigate wizard. Validation and error checking is performed on configuration information to get an Product Service instance setup in minutes taking care of database schema creation, system access rights, windows service installation and registration.

Database Migration Utility 

The Product Service Data Migration Utility is a Windows GUI application that is used to migrate RDBMS configurations from older schema definitions to a newer version as may be required by updates to the Product Service.

TSL Glossary

Blob: a binary object, e.g. a file, which can traverse the TSL, but is not converted to Measurements

Failover TSL Instance: a TSL instance which exists for purposes of backup and disaster recovery. May be local or remote, depending on context.

GEP: Gateway Exchange Protocol - an open source measurement-based publish/subscribe transport protocol used to securely exchange time-series data and automatically synchronize meta-data between two applications.

GSF: Grid Solutions Framework – http://gsf.codeplex.com/

Measurement Stream: a sequence of measurements for a specific Point ID

Measurement: the value of a point at a specific time, and associated per-measurement metadata, e.g., 12:22:23.033 122 38000 (Good Quality)

Message: either in the form of a command (between TSLs instances) or a notification (within the TSL), is information which has semantic meaning to the TSL, it may carry data.

Notification: an internal message in the TSL, used between modules to transmit state information, or cause state changes, in a coordinate fashion.

Point: a measurement point in the system. e.g., voltage on transmission line 1, or current on line 2. Note that points represent scalars only so that a voltage magnitude and angle are comprised of two points.

Point ID: the unique numerical identifier for a point.

Point Tag: Then alpha-numeric name associated with a Point ID.

TSL: Time-Series Library component of the Grid Solutions Framework.

TSL Instance: A machine running TSL.

Stream: a data object representing a blob, contains information such as size, deadline, destination, etc. May or may not carry the actual blob itself (TBD –perhaps instead carrying a pointer a socket or stored copy of the blob)