GRID PROTECTION ALLIANCE

openXDA Open Interface

{API}

A self-hosted API is embedded in the openXDA Windows service.

GPA's openHistorian also includes easy-to-use APIs.

The purpose of this paper is to document the features of the JSON Application Programming Interfaces that are built into *open*XDA.

Introduction

*open*XDA is an automation engine that is used to process event and trending data files produced by substation devices. *open*XDA parses the files, analyzes the data and places it into an open relational database, the *open*XDA database, for visualization and enterprise use. *open*XDA has a meter-centric data structure where meters are installed in substations and collect data on lines and other elements of the transmission or distribution system. *open*XDA stores both the raw waveform data that is captured from events, the frequency domain representation of the waveform data and the results from analytics that use these two data sets to provide information on current or voltage events. For meters that return time-series (trending) data, *open*XDA can store this data in a utility's enterprise data historian — or in GPA's *open*Historian.

For event data in the *open*XDA database, an open, JSON interface is provided along with the *open*XDA Windows service (*for openXDA Versions 2.0 and later*). JSON (JavaScript Object Notation) is a lightweight data-interchange format that is text-based, programming language-independent and human readable. The *open*XDA API allows other enterprise systems to easily query the *open*XDA data base. It also makes it very easy to develop and/or modify browser based applications to display event data processed and saved by *open*XDA.

For trending data that *open*XDA saves in the *open*Historian. This high-performance historian includes multiple data access interfaces.

This paper provides information on the *open*XDA API method calls plus an overview of the API's for the *open*Historian.

The *open*XDA API

The *open*XDA API returns disturbance data within three major categories:

- Asset Data and Meter Configuration Data (Config)
- Event Attribute Data (Events)
- Event Data Detail (EventData)

JSON makes it simple.

JSON uses JavaScript syntax, but the JSON format is text only. This Text can be read and used as a data format by any programming language.

JSON objects are key-value pairs.

JSON is like XML.. Both are self-describing, human readable, hierarchical, and most languages have built-in parsers available.

JSON is better than XML. JSON is less verbose; it's quicker; and it can incorporate arrays.

JSON objects are surrounded by curly braces {}.

JSON Parameter Objects

Each JSON function can include one or more limiting filters. These include:

- ID - *the integer data base key for this item*
- AssetKey - *the meter's string key, e.g., "CalvertKY1"*
- Name - *the full name of the meter, e.g., "CalvertCityKY-DFR1"*
- MeterIDList / MeterAssetKeyList - *The array of meter IDs or meter asset keys of interest.*
- EventID / EventIDList - *The event ID, or an array of Event IDs of interest*
- LineIDList / LineAssetKeyList - *The array of line ID or line asset keys of interest*
- StartDate / EndDate - *The start and/or end date of data requested.*

Asset Data and Meter Configuration Data (Config)

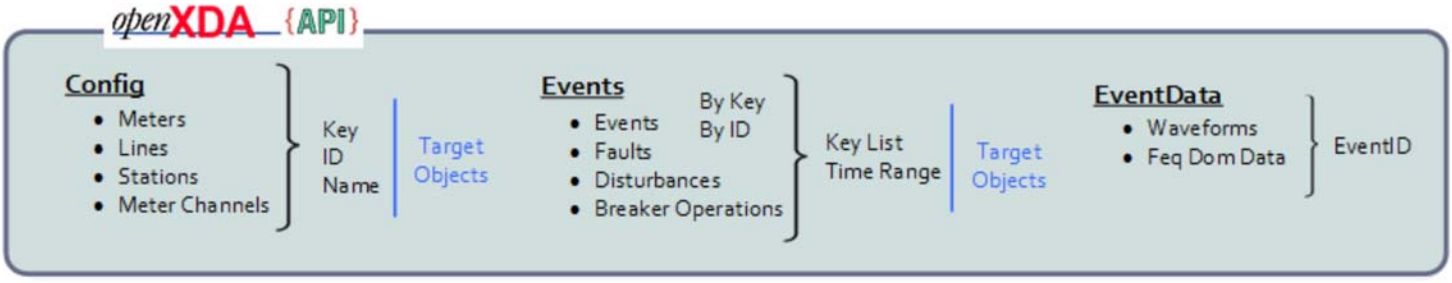
The following functions return metadata from *open*XDA..

- GetStations - *Returns station ID, lat, long,*
- GetLines - *Returns line D, length, nominal kV, and thermal rating*
- GetMeters - *Returns meter ID, make, model, location ID and time zone*
- GetChannelsByMeter - *For each channel in the meter(s), returns measurement type, measurement characteristic ID, line ID, meter ID, Phase ID, among others.*

Event Attribute Data (Events)

The following functions return data about events in *open*XDA

- GetEvents - *Returns event start and end time, event type, file group ID, line ID, meter ID, samples, time zone offset*
- GetFaults - *Returns fault type, fault location, calculation method*
- GetDisturbances (including faults) - *Returns event ID, event type ID, duration, start and end time, magnitude, phase ID*
- GetBreakerOperations - *Returns breaker ID plus an array of date-times that are within the sequence of a breaker's operation (if available from the DFR)*



Event Data Detail (EventData)

The API also includes functions to return time-domain or frequency domain data for the full event waveform

- GetEventWaveformData - *Returns an array of voltages and currents*
- GetEventFrequencyDomainData - *Returns an array of RMS and sequence values.*

Overview of *open*Historian APIs

*open*XDA saves trending data returned from substation devices in a historian. The *open*Historian 2.0 has multiple data access interfaces. There include:

- .NET API (socket) - This is the highest performance and fastest API available for the openHistorian.
- SQL Server - This collection of MS SQL-Server scripts allows the *open*Historian to be accessed via SQL like a relational database
- GEP/STTP - GPA's pub/sub protocol is ideal for streaming real-time or historical data to applications or for reliably and securely exchanging high -volumes of trending data with external entities
- Web Service API - Both JSON and XML output are supported
- Grafana Web Service - This JSON API supports the Grafana® interface standard
- SignalR - A powerful framework that supports asynchronous applications

Grafana
Beautiful metric & analytic dashboards

The *open*Historian includes an interface to this easy-to-use dashboard development system.

See: grafana.net

Example Grafana Dashboard