HttpLD SDK Documentation

V4.9.2

Larson Davis – PCB Piezotronics, Inc.

Confidential

Contents

Change Log

Initial draft	4/10/2014
Additional information on streaming	7/18/2014
Inclusion of the Table Of Floats for the types and added types for Measurement Properties.	10/10/2014
Update to the Data Streaming Rates and functional descriptions.	1/14/2015
Corrected getDataFileList instructions	3/12/2015
V1.2.4 – Updated downloadSettingsFile info, and getSessionValue and setSessionValue functions.	4/9/2015
V1.2.5 – Updated HttpLD usage; show all command line switches in doc; update port number usage; spelling and duplication corrections; encryption section;	4/16/2015
V1.2.6 – Update for TAG_DATE_TIME_EX calling. Updated information on Closing.	5/18/2015
V1.2.7 – Added the changeLang func	8/1/2015
V1.2.8 – Moved TAG_DAILY_STORE_TIME next to TAG_DAILY_STORE	11/4/2015
V1.2.9 – Added in TAG_NETWORK_PASSWORD info and further directions to the Properties.	11/5/2015
V1.3.0 – Several Additions to descriptions of the TAGS with references back to the 831 manual.	11/6/2015
V1.3.1 – Updated and cleaned up the TAGS. Added some enumeration descriptions.	12/2/2015
V1.3.2 – Added Info on Data Streaming in JSON	12/15/2015
V1.3.3 – Added documentation for Weather Streaming Data Improved Data Streaming	3/21/2016
V1.3.4 – Updated uploadPropertiesFromCache section and added a NOTE on length of filename.	4/29/2016
V1.3.5 – Updated and revised the Data Streaming in JSON section for clarity. Renamed enums to be consistent with actual code in LDCommon.	5/3/2016
V1.3.6 – Update info on the IntervalTime and RunMode. Some small organizational changes to the location of the System Properties list.	7/1/2016
V1.3.7 – Added getMeterList; more info on CMD_LATCH_SETTINGS	8/3/2016
V1.3.8 – Added Connection information for HVM200 and Device Category description in Usage.	8/11/2016
V1.3.9 – Correction to the Property List for Tag ID. Some headings were promoted.	11/17/2016
V1.4.0 – Addition of how to call CMD_STORE	12/7/2016
V1.4.1 – Addition of the flag table for Streaming Data	1/17/2017
V1.4.2 – Update for info on JSON parsing of settings.	1/27/2017
V1.4.3 – Fixed Units in Streaming Data table to Volts squared.	2/3/2017
V3.0.0 – Inclusion of the 831C functionality, updates to most of the document with related changes and addition of the Appendix with updated example code.	2/14/2017
V3.0.3 – Addition of Command Line and Web Socket information	5/12/2017

V3.0.4 – SFTP and Dropbox support and Updates to the File List functions with associated source examples.	6/20/2017
V3.0.5 – nErroFlags added,	8/31/2017
V3.0.6 – Update to the Commands for performing Calibrations and Cal Checks.	1/2/2018
V3.0.7 – Addition of AudCal to SDK	2/28/18
V3.2.1.0 – Addition of the new Properties: Static IP, Running LEQ, and Event Triggers. G4 export instructions. Time History Options defined.	9/19/2018
V3.2.1.1 – Updated Date Time functions and example code,	11/27/2018
V3.3.0.0 – Changed some of the code associated with examples in the commands and Time functions. Added missing System Property.	4/5/2019
V3.5.0.0 – Updated details on the TAG_DAILY_STORE Auto Store option values.	9/13/2019
V3.5.0.1 – Updated "Command-line Exporting from G4" section to include a chart that details command-line commands and arguments.	10/7/2019
V4.0.8 – Moved the GetJSONStream up next to the rest of the data streaming functions. - Added description of the Event State - Added more examples of setProperty to include setting up Time History - Fixed wrong ID for Event Min Duration	12/19/2019
V4.5.0 – Added Requesting Time History Records to the appendix	4/28/2020
V4.5.0 – Added login and keep alive explanations and examples.	06/08/2020
V4.5.1 – Added description of the SetActiveIndex in the TAG_IO_EVENT_HIST_LOAD_RECORD and TAG_IO_MEAS_HIST_LOAD_RECORD	9/1/2020
V4.5.2 – Added New Weather Time History IDs Added examples of how to Delete Files using C# - see DeleteFile/DeleteFiles section.	10/12/2020
V4.6.0 – Added CalHistJson information to the calibration section Added new Measurement and System Properties for Schedule and FFT.	11/11/2020
V4.6.0 – Added tags for FFT and removed deprecated ZModem instructions. Additional clarifcations and updates to LDBin file, AudCal tags, and SoundAdvisor data tags.	11/24/2020
V4.6.0 – Added information about Tonality functions in LDFirmwareUtilities.	01/12/2021
V 4.6.0 – Added File access information on the meter.	01/28/2021
V 4.6.0 – Included the HVM200 Properties and Streaming Data and added details to the sendCommand section.	02/23/2021
V 4.6.2 – Update for info on the internal units for weather. Removed duplicate page in the AudCal Test information. Updated the SysProps with the Random Correction property.	4/6/2021 5/3/2021 5/17/2021
V 4.6.5 – Updated the dlls and packages; updated the System properties to include External Shutoff and USB Storage	10/17/2021

V 4.7.0 – Update to qualify the usage to 831C unless specifically stated. Updated calHistJson to designate it is for 831C only.	4/29/2022
V 4.8.1 – Added the libraries to use C# translator in C++ Windows applications. Updated HttpLD to latest version.	9/4/2022
V 4.9.0 – Updated libraries and associated files and HLD.	6/15/2023
V 4.9.2 – Documenting existing support for 721/821 meters.	8/11/2023

Introduction

The SDK has the following parts:

1. This document which describes the functionality of the SDK and executable.
2. Code examples found in the SampleCode folder.
    a. There are examples in C++, JavaScript/HTML and C#.
        i. In CppSamples, the SdkTestApp and StreamingDataDemo folders contain C++ code examples which can be compiled in Windows and Linux. Note for Linux you must be using an 831-INT-ET or have purchased a version of the HLD that runs on your platform.
        ii. HTML5-JavaScript folder contains the JavaScript examples.
        iii. CSharpSamples contains several C# examples like:
            1. HvmStreamingData
            2. LiveStreamingDemo
            3. SdkConnect
            4. SdkDownload
            5. TimeHistoryDemo
    b. This document contains information and code snippets that can be used for C++, C# and JavaScript.
3. Firmware files, for both Model 831 and 831-INT-ET, which are compatible with the SDK.
    a. Please ensure that you have updated to the latest firmware before proceeding with the development using this SDK.
4. The HttpLD.exe and supporting files that is the core of the SDK functionality.
    a. All of the files located in the bin\Win32 folder are required to use the SDK.

Using HttpLD.exe for Communication

The HttpLD.exe (HLD) is an executable for Windows® based OS's. It is the method of communication with the updated SoundAdvisor© Model 831C (SoundAdvisor, or 831C), Model 831A (831A), 831-INT-ET (INT-ET), SoundTrack LxT® (LxT) and HVM200 meters. You may launch HLD from the command line to talk to a USB connected meter. This method of connection is based on standard HTTP requests and responses. Much of the data that comes from the meter is in JSON.

Note: Unless specifically called out with an example or noted, the bulk of the document is for the 831C. When in doubt, make the request to your specific meter. If you receive an error verify the call from an example. If the error persists, the meter most likely doesn't support the operation.

There are several samples demonstrating the communication that are installed with the HLD application. We will include some of the code from these examples for descriptive purposes in this document.

If you are not developing in a Windows environment, the 831-INT-ET allows you to connect using the same methods described here in, with the exception of needing to launch the HLD from the command line. If communicating via an 831-INT-ET an HLD will not need to be started as it is integrated into the 831-INT-ET.

Linux

There are two packages currently available in Linux pre-built: the Raspberry Pi and the BeagleBone Black. There are separate instructions to aid in installing these packages on to your device along with verification to test that the installation was successful.

Note: We use port 2508 on the PC to distinguish from a connection to an 831-INT-ET (dock/cradle) on port 2001.

Deployment

Depending on your license agreement, you may wish to deploy an application in which you may choose to include the current HLD and associated documentation. You may add HLD (based on license agreement) to any location you choose and your application may continue to communicate through this channel even if newer HLDs are installed through other LD packages such as G4.

First Example

It is very easy to connect to a meter.

831 via USB

Start HttpLD with the correct parameters.

1. Ensure that an updated 831 meter is attached via USB.
2. From a command prompt execute:
    a. **HttpLD.exe -p 2508 -c USB;0;300**
3. There are a few other parameters that are available but for now this will provide you connection to the first meter that is connected via USB. See the

4. Usage section below for a description of all the available parameters.
5. You may test access by using a browser and navigating to:
   a. http://127.0.0.1:2508/sdk?func=getData&id=100.
   b. This will return the Measurement Properties of the Instrument.
6. For a Model 831A, you may also display the screen of the meter in Chrome by similar navigation.
   a. http://127.0.0.1:2508/sdk?func=getData&id=4000

Note: For each instrument you have connected via USB you will need to have a unique HLD, if you desire to communicate with multiple meters simultaneously. For 831-INT-ET, you should not launch the HLD as it is already a part of the 831-INT-ET. You may connect to it directly as described in **831-INT-ET** section. (e.g., http://ipAddress[:port]/sdk?func=getData&id=100)

Note: ipAddress can be "localhost". You may use "127.0.0.1" as the local loopback instead of *localhost* as *localhost* may not always be set to the standard loopback.

831-INT-ET

If you wish to connect to an 831-INT-ET follow these steps:

1. Obtain IP Address of the 831-INT-ET.
2. You may test access by using a browser and navigating to:
   a. http://IpAddress:2001/sdk?func=getData&id=100.
   b. This will return the Measurement Properties of the Instrument.
   c. http://IpAddress:2001/sdk?func=getData&id=200.
   d. This will return the System Properties of the Instrument.

HVM200 via USB

If you wish to connect to an HVM, the process is very similar to an 831 via USB:

1. Ensure that an updated HVM200 meter is attached via USB.
2. From a command prompt execute:
   a. *httpld -port 2508 -d 1 -c USB;S0000510MHVM200;300*
3. See the **Usage** section below for a description of all the available parameters.
4. You may test access by using a browser and navigating to:
   a. http://127.0.0.1:2508/sdk?func=getProperties.
   b. This will return the Measurement Properties of the Instrument in JSON.

Making Requests

You may now begin making http requests. Follow the examples below to talk to your meter. We try to use the same colors to represent the 3 different languages used throughout the document. Below you will notice, we start with C#, then C++ and finally JavaScript.

Example in C#:

```
WebRequest request;
```

```
request = WebRequest.Create("http://"  + ipAddress +  ":" + port +
"/sdk?func=getData&id=100");

request.Method = "GET";
WebResponse response = request.GetResponse();
Stream dataStream = response.GetResponseStream();
data = new byte[response.ContentLength];

int offset = 0;
while (offset < response.ContentLength)
{
        offset += dataStream.Read(data, offset, (int)(response.ContentLength - offset));
}

dataStream.Close();
response.Close();
```

Using Newtonsoft.Json.Linq.JObject, we can reduce some of the effort in C# for parsing the JSON which is contained in the data object.

```
var jsonMsg = System.Text.Encoding.UTF8.GetString(data);
JObject json;
try
{
        json = JObject.Parse(jsonMsg);
}
catch
{
        return;
}
```

Example in C++:

```
SOCKET create_socket(string ip_address, int port)
{

#ifdef _WIN32
   WSADATA wsaData;

   if (WSAStartup(MAKEWORD(2,2), &wsaData) != 0)
   {
      cout << "WSAStartup failed.\n";
      sys_pause();
      return 0;
   }
#endif

   SOCKET Socket=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
   struct hostent *host;
   host = gethostbyname( ip_address.c_str() );
   SOCKADDR_IN SockAddr;
   SockAddr.sin_port=htons(port);
   SockAddr.sin_family=AF_INET;
   SockAddr.sin_addr.s_addr = *((unsigned long*)host->h_addr);

   if(connect(Socket,(SOCKADDR*)(&SockAddr),sizeof(SockAddr)) != 0)
   {
      cout << "Could not connect";
      sys_pause();
      return 0;
   }

   return Socket;
}

void close_socket(SOCKET socket)
{

   closesocket(socket);

#ifdef _WIN32
   WSACleanup();
#endif

}

#define MAX_HLD_COMMAND (256)
SOCKET send_HLD_command( string ip_address, int port, string command)
{
   SOCKET socket;

   socket = create_socket(ip_address, port);

   char buf[MAX_HLD_COMMAND];

#ifdef _WIN32
   sprintf_s(buf,"GET /sdk?func=%s HTTP/1.1\r\nHost: %s:%d\r\nConnection: clos
e\r\nAccept: */*\r\n\r\n", command.c_str(), ip_address.c_str(), port);
#else
   sprintf(buf,"GET /sdk?func=%s HTTP/1.1\r\nHost: %s:%d\r\nConnection: close\
r\nAccept: */*\r\n\r\n", command.c_str(), ip_address.c_str(), port);
#endif

   send(socket, buf, strlen(buf),0);

   return socket;
}

// Return value is the http response length, the data is stored in response_da
ta and http_response_header
#define MAX_BUF_SIZE (1024)
int get_HLD_response( SOCKET socket, string &http_response_header, stringstrea
m &response_data)
{
    http_response_header.clear();

    char buf[MAX_BUF_SIZE];
    int response_length = -1;

    if ( http_header_recv(socket, http_response_header) )
    {
        response_length = html_header_get_content_length(http_response_header)
;

        int remaining_response = response_length;

        while( remaining_response > 0 )
        {
            int response = recv(socket, buf, MAX_BUF_SIZE, 0);

            if( response > 0)
            {
                response_data.write(buf, response);
                remaining_response -= response;
            }
            else
            {
               cout << "Wrong amount of data received, expected: " << remainin
g_response;
               cout << " more bytes" << endl;
               break;
            }
        }
    }

    if ( -1 == response_length )
    {
        response_length = 0;
        cout << " Error - Http header: " << http_response_header << endl;
    }
    return response_length;
}
```

Example JavaScript with the aid of jQuery:

```
var MeasurementProperties = new Object();
var SystemProperties = new Object();

function GenerateCache(data) {
       var obj = function () { };
       obj.prototype = data;
       return new obj();
}

function GetProperties(val) {
$.getJSON('/sdk?func=getData?id=' + val)
 .done(function (dataVal) {
   if (val == 100) {
      MeasurementProperties = GenerateCache(dataVal["MeasProperties"]);
      SystemProperties = GenerateCache(dataVal["SysProperties"]);
   } else if (val == 200) {
      MeasurementProperties = GenerateCache(dataVal["MeasProperties"]);
      SystemProperties = GenerateCache(dataVal["SysProperties"]);
   }
 });
}
GetProperties(100);
```

Usage

From the command line you may call HLD as follows:
httpld [-port <port>[,<port>]] [-d 0|1|2] [-r <ResourceFolderPath>]
[-c USB;(<deviceId>|S<serialnumber>M<model>);<connectTimeout>] [-ignorePassword]

As an example:

SLMs: Model 831A, Sound Advisor Model 831C, and SoundTrack LxT

httpld -port 2508 -c USB;S0001065M831;300 -ignorePassword

HVM200

httpld -port 2508 -d 1 -c USB;S0000510MHVM200;300

Parameters

-port is used to specify the port number(s).
<port> can be any port with most likely candidates being ports 2001, and 2508. We use port 2508 on the PC to distinguish from a connection to an 831-INT-ET (dock/cradle) on port 2001.

-c is used to specify the connection string. See more on the **Connections** main section below. You may use the <deviceId>, which is usually a zero based index to the meter and is not very stable. Normally you would specify the <serialnumber> (the serialnumber of the meter to which you wish to connect) and <model> (831, future expansion of model will include other models). Be sure to include a <connectionTimeout> value greater than 0 or your meter may not connect and remain stable. A Connection Timeout of 300 means 300 milliseconds.

The following connection strings are valid:

1. USB;0;300
2. USB;S0001244M831;400

-d specifies device category. Default is 0.

SLM: 0
HVM: 1
X21/730: 2

-r allows you to specify the Resource path to any HTML5/JavaScript files you may work with. <ResourceFolderPath> string may be relative to the location from which HLD resides and is launched. Examples:

1. .\Resources
2. C:\MyHTML5Files

If you do not have a password on the device (found in the System Properties Network tab) or if you wish to ignore the password that has been set, then you may use -ignorePassword as a parameter.

Closing HLD

When you wish to communicate with your meter outside of G4 over USB, you will need to run HLD. Once your communication has completed and you are ready to close your application, you will want to close your instantiated HLD. You can simply terminate the process or call the 'closedown' function to have the HLD close nicely. If left running HLD will prevent other connections to the same device from working correctly, for instance G4.

Or C#:

```
        request = WebRequest.Create("http://ipAddress:port/sdk?func=closeDown");
```

JavaScript:

```
$.getJSON('/sdk?func=closeDown')
```

Functions

In the above C# example you may note that the request was made to:

```
        request = WebRequest.Create("http://ipAddress:port/sdk?func=getData&id=100");
```

'func=getData&id=100' represents the actual data request. Func is used to determine which function you are requesting, e.g., getData or sendKeyCommand. Each function will have their own list of parameters. getData requires an id. In the above example setting it to 100 will return all of the Measurement Properties along with a few of the System Properties (Preferences) that are needed when evaluating the Properties. Some properties have requirements on other properties from both Measurement and System Properties, for example the USB must be on (System property) if the Vaisala weather (Measurement property) is used.

With an id=200, the System Properties are returned. Similarly, there are some Measurement Properties contained in the returned JSON. It is not always necessary to retrieve all of the System or Measurement properties from these two calls if you only need to read the values. It is only important when you expect to set some of the values that may have some interaction.

The ipAddress and port represent the connection to the HttpLD. The connections can be on the local PC for direct connections or on an 831-INT-ET. The response from the meter will come back in the form of JSON for most getData calls. There is much documentation on JSON online and a discussion here is out of scope. We may show some examples though we will not explain all of the syntax and usage of JSON here. You wish to visit some of the following sources for further info:

1. http://www.w3schools.com/json/
2. http://en.wikipedia.org/wiki/JSON
3. http://www.json.org/

The calls to the HLD are rather simple and straightforward. When a function is called it is performed as follows:

> http://ipAddress:port/sdk?func=sendKeyCommand&id=32

In subsequent sections we will further discuss the makeup of this request. The core is a call to the sdk with the function you wish to perform. In this case, we are sending a Key Command to the meter with an id of 32, which on an 831 means the “Enter” button being pressed.

Change Language

You may decide that you need a different language for any strings or Html that may be used or sent back from the HLD. To change to another language you simply call:
http://IpAddress:2001/sdk?func=changeLang&lang=fr-FR

Available languages currently supported and character lang value:

1. English: en-US
2. French: fr-FR
3. Italian: it-IT
4. German: de-DE

On the SoundAdvisor, you may use any of the following for testing if the associated language pack has been uploaded to the meter: cs-CZ, de-DE, en-US, es-ES, fr-FR, hu-HU, it-IT, ja-JA, nn-NO, pl-PL, pt-BR, pt-PT, ro-RO, ru-RU, sv-SE, th-TH, tr-TR. These are standard language monikers.

Connections

When trying to connect to a meter in Windows you may use the USB;0;300 connection string as you pass in the parameters from the command line. And for further information, here is the breakdown of the possible connection string parameters.

USB

`USB;(`*`<device ID>`*`|S`*`<serial number>`*`M`*`<model number>`*`);`*`<connecting timeout in milliseconds>`*

Examples:

`USB;0;300` `(connects to first device)`

`USB;4;300` `(connects to fourth device)`

`USB;S0001065M831;300` `(connects to a Model 831 with serial 0001065)`

The connection string is formatted as such: string.Format("USB;S{0}M{1};300", this.SerialNumber, "831"); Where the first parameter is the Serial Number to which meter you are trying to connect. The second parameter is the Model, in this case 831. This is the main method to connect through USB.

```
string connect = "";
if (ConnectionType == ConnectionTypes.USB)
{
        connect = string.Format("USB;S{0}M{1};300", this.SerialNumber, "831");
        LaunchCommandLineApp(connect);
}

…

public Process LaunchCommandLineApp(string ConnectionString)
{
  // Use ProcessStartInfo class
  ProcessStartInfo startInfo = new ProcessStartInfo();
  startInfo.CreateNoWindow = true;
  startInfo.UseShellExecute = true;
  startInfo.FileName = "HttpLD.exe";
  startInfo.WindowStyle = ProcessWindowStyle.Hidden;
  startInfo.Arguments = ConnectionString;
  try
  {
        // Start the process with the info we specified.
        // Call WaitForExit and then the using statement will close.
        mHLDProcess = Process.Start(startInfo);
        mHLDProcess.WaitForExit(1000);
```

```
            processJob.AddProcess(mHLDProcess.Handle);


            return mHLDProcess;
 }
 catch
 {
            // handle error.
 }
}
```

Or if you know you will only ever have one meter connected you may try:

```
connect = string.Format("USB;{0};300", this.DeviceID);
```

TAPI

`TAPIEX;`*`<device id>`*`;`*`<baud rate>`*`;`*`<numeric password>`*`;`*`<phone number>`*

Example: `TAPIEX;0;115200;11111111;8015551234`

Or

`TAPI;`*`<device handle>`*`;`*`<numeric password>`*

Example: `TAPI;132;11111111`

MODEM

`MODEM;`*`<com port>`*`;`*`<baud rate>`*`;`*`<numeric password>`*`;`*`<phone number>`*

Example: MODEM;1;115200;11111111;8015551234

RS232

`RS232;`*`<com port>`*`;`*`<baud rate>`*

Example: `RS232;1;115200`

BLE

```
BLE connections are only supported via the
LDConnectionServer/LDConnectionService. Currently there is no SDK
support for BLE connections outside of the G4 LD Utility software.
```

Special Case for getMeterList

You may need to get a list of meters attached by USB to your local machine. First you must launch an HLD with the following parameter: -c DIRECT;0;800

Example:

httpld -port 2505 -c DIRECT;0;800

This will start an HLD that can look for devices connected via USB on your PC. Then you may call `http://localhost:2505/sdk?func=getmeterlist`. You will receive JSON that represents a set a possible meters to which you may connect. Once you have your list you may close this HLD using `/sdk?func=closedown` (See SDK manual for Closing HLD function).

You might review the following C# code as to how one might process the list of devices (This is only an example). This function processes the JSON received by `getMeterList`.

```
private static void ProcessUsbMeterList(IList<UsbDevice> list, JObject json)
{
        string serial = string.Empty;
        string product = string.Empty;
        int id = 0;

        foreach (var meter in json["Meters"])
        {
                if (null != meter["serialNumber"]) serial = meter["serialNumber"].ToString();
                if (null != meter["product"]) product = meter["product"].ToString();
                if (null != meter["id"]) int.TryParse(meter["id"].ToString(), out id);

                MeterType meterType =
                        (product.StartsWith("831")) ? MeterType.M831 :
                        (product.StartsWith("LxT")) ? MeterType.LxT :
                        (product.StartsWith("HVM")) ? ((product.Contains("200")) ? MeterType.HVM200 :
 MeterType.HVM100) : MeterType.Unknown;

                // for now, 831C's are reported as 831
                if (MeterType.M831C == meterType) meterType = MeterType.M831;

                if (MeterType.Unknown != meterType)
                {
                        list.Add(new UsbDevice(serial, meterType, id));
                }
        }
}
```

Sending Commands and Keys

This section describes the methods used to send commands with and without values as well as the command for sending key presses to the meter.

Example in C#:

```
request = WebRequest.Create("http://ipAddress:port/sdk?func= sendKeyCommand&id=32");
```

or in JavaScript with jQuery:

```
$.getJSON('/sdk?func=sendKeyCommand&id=32);
```

Most all of the commands will respond with some form of Ack. The most common is:

```
{"Result": "Success" }
```

You may want to check the Result to ensure that there was no error in the transmission of the command and to verify that it was successful. Here in JavaScript, we send a message to the console if there is an error. You could instead call a message box or other popup mechanism to warn your user.

```
function errorHandling(functionName, data) {
   var message = "Error in " + functionName + " with: ";
   if (data["Result"] != undefined) {
      if (data["Result"].indexOf("Error") >= 0) {
         //do something with the error
       message += data["Result"];
       if (data["Message"] != undefined) {
          message += " - Message: " + data["Message"];
       }
       console.log("%c " + message, "color:rgb(200, 10, 10); font-
size: 11pt;");
       return true;
      }
   }
   return false;
}

function stopMeter() {
   $.getJSON('/sdk?func=sendCommand&id=' + CMD_STOP)
    .done(function (dataVal) {
      errorHandling("stopMeter", dataVal);
   });
}
```

sendCommand

The operational commands will be described here in the following format:
OpId: The name of the setting.

Command ID: The value that corresponds to the OpId.

Description: A short description of the command and/or its purpose.

Type: The type for Allowed Values.

```
var TypeUnknown = 0;
var TypeInt = 1;
var TypeUInt = 2;
var TypeFloat = 3;
var TypeFloatSeries = 4;
var TypeFloatWithFlags = 5;
var TypeString = 6;
var TypeByteSeries = 7;
var TypeUIntSeries = 8;
var TypeEnum = 9;
var TypeUIntWithFlags = 10;
var TypeFloatWithFlagsSeries = 11;
```

Allowed Values: Allowed values, min/max, or string length.

SetOperation Commands (831C, 831, LxT, HVM200)

These commands are used to change the state of the meter. These commands can be run for the SLMs and HVM200.

Example: CMD_RUN
http://ipAddress:port/sdk?func=sendCommand&id=256

Example: CMD_STOP
http://ipAddress:port/sdk?func=sendCommand&id=257

OpId: CMD_RUN
Command ID: 256
Description: Start a measurement.
Type: NA
Allowed Values: NA

OpId: CMD_STOP
Command ID: 257
Description: Stop a measurement.
Type: NA
Allowed Values: NA

OpId: CMD_DATA_LATCH
Command ID: 262
Description: Latches the Properties to the active measurement setup.
Type: NA
Allowed Values: NA

OpId: CMD_REBOOT
Command ID: 0x83A3
Description: Reboot the meter. The meter must be stopped.
Type: NA
Allowed Values: NA

OpId: CMD_FORMAT
CMD_FORMAT_831_BY_SBC *(use when connected via INT_ET)*
Command ID: 0x9110; 0x9116
Description: Reset meter to factory settings (erase all internal data, clear files, reset all settings, etc.). The instrument must be in boot mode when sending this command. Use CMD_REBOOT to put the meter into boot mode.
Type: NA
Allowed Values: NA

SetOperationVal Commands

OpId: CMD_CONNECT
Command ID: 0x00030001
Description: Connect to a meter. Only use this command if you need to interact in boot mode.
Type: *TypeString*
Allowed Values: * *(see below for various connection string examples)*

OpId: CMD_CONNECT_EX
Command ID: 0x00030006
Description: Connect to a meter. This version will wait for meter to be out of boot mode to connect.
Type: *TypeString*
Allowed Values: * *(see below for various connection string examples)*

OpId: CMD_SETLXTUI
Command ID: 0x00030005
Description: Tells the meter that an external process is connected and may make setting changes. This will cause the meter to present a warning dialog to the user if any setting dialogs are opened on the meter directly.
Type: *TypeInt*
Allowed Values: 0 = not connected, 1 = connected

OpId: CMD_DISCONNECT
Command ID: 0x00030002
Description: Disconnect from a meter.
Type: *TypeInt*
Allowed Values: 0

OpId: CMD_LATCH_SETTINGS
Command ID: 0x9114

Description: Tells the meter to begin using new settings after updating system settings, preference settings, control settings, or calibration settings. This is very important if you are changing the settings, system properties or preferences.
Type: *TypeUInt* - unsigned int
Allowed Values: 0 = system setting(s) changed
1 = preference setting(s) changed
2 = control setting(s) changed
3 = calibration setting(s) changed

OpId: CMD_RENAME
Command ID: 0x9113
Description: Rename a data file or a setup file on the meter.
Type: *TypeByteSeries*
Allowed Values: Set BYTE array as follows as a hex string:
Byte 0: 0 = data directory, 1 = setup directory
Bytes 1 – m: new file name of length 'm' *(max 15 chars)*
Byte m+1: NULL terminator *('\0')*
Bytes m+2 – m+2+n: old file name of length 'n' *(max 15 chars)*
Byte m+2+n+1: NULL terminator *('\0')*
The maximum length of the byte array is 33 bytes (1 directory + 2*15 strings + 2 terminators). Be sure to check in the software that the two strings are properly terminated.

OpId: CMD_DELETE
Command ID: 0x9112
Description: Delete a data file or a setup file from the meter.
Type: *TypeString*
Allowed Values: Name of file to delete as a string with maximum length of 15 characters.

OpId: CMD_RESET
Command ID: 1
Description: Reset data on the meter.
Type: *TypeInt*
Allowed Values: data=1 means reset overall data and filters,
data=0 means reset overall data only
Example: http://localhost:2508/sdk?func=sendCommand&type=1&data=1&id=1

OpId: CMD_STORE
Command ID: 263
Description: Store data on the meter to the specified filename.
Type: *TypeString (name of file to store, file extension is required)*
Allowed Values: Max length: 15

In the following example, 'file' is constructed with 8.3.1, e.g., "filename.001.s" where 's' stands for SLM mode; TypeString = 6; CMD_STORE = 263. We are also checking for an error result to see if the meter responded that it could not save the file.

```
$.getJSON("/sdk?func=sendCommand&id=" + CMD_STORE + "&type=" + TypeString + "&
data=" + file)
```

```
.done(function (data) {
        if (ErrDup == data.ResultCode) {
                alert("IDS_HLD_ALERT", "IDS_CANNOT_STORE");
        }
        console.log(data);
        if (Sess.reset != undefined) Sess.reset();
});
```

OpId: TAG_DATE_TIME
Command ID: 0x44543031
Description: Sets the date and time. Meter uses GMT time so make the proper adjustments from local time.
Type: *TypeUInt* - unsigned int
Allowed Values: Min: 0, Max: 2147483648 (2^32-1)*
* All date and datetime values are represented as the number of seconds since 01/01/70 00:00:00 GMT.

OpId: TAG_DATE_TIME_DELTA
Command ID: 0x44543033
Description: Sets the date and time with milliseconds precision. Meter uses GMT time so make the proper adjustments from local time.
Type: *TypeFloat* – float of milliseconds to offset
Allowed Values: Min: 0, Max: 2147483648 (2^32-1)*
* All date and datetime values are represented as the number of seconds and milliseconds since 01/01/70 00:00:00 GMT.

OpId: TAG_DATE_TIME_EX
Command ID: 0x44543032
Description: Sets the date and time with microsecond precision. Meter uses GMT time so make the proper adjustments from local time.
Type: *TypeUIntSeries* - unsigned int series two values:
TimeInSeconds, MicroSecondsInCurrentSecond
Allowed Values: Min: 0, Max: 2147483648 (2^32-1)*
* All date and datetime values are represented as the number of seconds and microseconds since 01/01/70 00:00:00 GMT.

```
TimeSpan ts = DateTime.Now - SlmConstants.Epoch;
int timeInSeconds = (int)ts.TotalSeconds;
int microSeconds = 0; //set to the microseconds of your NTP or PC
string cmd =
string.Format("/sdk?func=setProperty&tagid={0}&value={1},{2}&type={3}&index={4}",
(int)PropertyTag.DateTimeEx, timeInSeconds, microSeconds, 8, 0);
```

PropertyTag.DateTimeEx is found in the C# include file: SlmInclude.cs. This include file contains definitions for the available tags.

A round trip check is made to see if there is any latency or lag in the connection to the meter. Average several request/responses of the status to see how long it takes over the connection and then use that in the calculation of the time pushing the time sent to the meter into the future enough that the meter and pc are synced correctly.

C# example of calling the DateTimeEx and DateTimeDelta functions:

```
public void SendDateTimeToMeter(DateTime? dateTime)
{
    WrtCalResult wrtResult;
    JObject jObj = null;
    bool result = false;
    uint secs = 0;
    uint usecs = 0;
    long cmdStart = 0;
    long cmdEnd = 0;
    TimeSpan ts = DateTime.Now - LDConstants.Epoch;

    string cmd = "/sdk?func=setProperty";

    if (null == dateTime)
    {
         double delta = TimeSpan.FromTicks(DeltaAverage).TotalMilliseconds;

    bool useDelta = ((delta < (30 * LDConstants.SecondsPerMinute * LDConstants
.MilliPerSecond)) && (DeltaCount > MinStableDeltaCount)) ? true : false;

         if (meterModel.Is831C() && useDelta)
         {

    cmd += string.Format("&tagid={0}&type={1}&index={2}&value={3}", (int)Prope
rtyTag.DateTimeDelta, (int)DataType.Float, 0, delta);
         }
         else
         {
             string tmz = "";
             int tmzHrs = -6;
             int tmzMin = 0;

             GetMeterZoneInfo(ref tmz, ref tmzHrs, ref tmzMin);

             var pcLocal = LocalDateTime.FromDateTime(DateTime.Now);
             var pcZone = DateTimeZoneProviders.Tzdb.GetSystemDefault();
             var pcZoned = pcLocal.InZoneLeniently(pcZone);

             var meterZone = DateTimeZoneProviders.Tzdb.GetZoneOrNull(tmz);
             var meterZoned = pcZoned.WithZone(meterZone);

             GetTimeInSecondsAndMicroSeconds(meterZoned, out secs, out usecs);


    cmd += string.Format("&tagid={0}&type={1}&index={2}&value={3},{4}", (int)P
ropertyTag.DateTimeEx, (int)DataType.UIntSeries, 0, secs, usecs);
         }
    }
    else
    {
         ts = (DateTime)dateTime - LDConstants.Epoch;

         GetTimeInSecondsAndMicroSeconds(ts, out secs, out usecs);


    cmd += string.Format("&tagid={0}&type={1}&index={2}&value={3},{4}", (int)P
ropertyTag.DateTimeEx, (int)DataType.UIntSeries, 0, secs, usecs);
    }

    cmdStart = Stopwatch.GetTimestamp();
    result = meterModel.PostCommandToHLD(cmd, "Success", out jObj);
    cmdEnd = Stopwatch.GetTimestamp();

    ResetDelta();

    if (cmdEnd - cmdStart > 0)
    {
         lock (_timeLock)
         {
             ++CommandCount;
             CommandSum += (ulong)(cmdEnd - cmdStart);
             CommandAverage = CommandSum / CommandCount;
         }
    }

    wrtResult = result ? WrtCalResult.WriteSuccess : WrtCalResult.WriteFailed;

    if (wrtResult != WrtCalResult.WriteSuccess)
    {

    string msg = "Date and time were not successfully written to the meter.";
         LogFile.WriteLog(msg);
#if DEBUG

    MsgBox.Show(msg, Strings.GIDS_ERROR, MessageBoxButton.OK, MessageBoxImage.
Error);
#endif
    }
}

private void GetTimeInSecondsAndMicroSeconds(TimeSpan ts, out uint secs, out u
int usecs)
{
    secs = (uint)ts.TotalSeconds;
    usecs = (uint)(ts.Milliseconds + (int)(CommandAverage / TimeSpan.TicksPerM
illisecond / 2)) * 1000;  // * 1000 to get micro seconds
    while (usecs > LDConstants.MicroPerSecond)
    {
         ++secs;
         usecs -= LDConstants.MicroPerSecond;
    }
}
```

```
private void GetTimeInSecondsAndMicroSeconds(ZonedDateTime zdt, out uint secs,
 out uint usecs)
{
    TimeSpan ts = zdt.ToDateTimeUnspecified() - LDConstants.Epoch;

    GetTimeInSecondsAndMicroSeconds(ts, out secs, out usecs);
}
```

Sending Commands to 730, 721 and 821 (Including SE)

The request to send a command is
"http://ipAddress:port/sdk?func=cmd&op=send&message={message}.

In the above example message should contain one of the following supported messages, the response returned is dependent on the meter's current state. Some commands require an additional suffix in the form of "*nnn"

Measurement Commands

Code	Description
M1	Starts a measurement.
M2	Stops the current measurement. On the 730 this will always store and put the meter in a reset state. On the X21 this will simply end the current measturement but future measurements will continue to be stored in the same file until the Store command (M11) is sent.
M3	Pause the current measurement.
M10	Discards the current measurement file. Does not apply to the 730. The meter must be in a stopped state.
M11	Stores the current measurement file. Does not apply to the 730. The meter must be in a stopped state.
D10	Gets the Time History graph data.
D11	Gets the number of Time History records.
D9*125	Deletes all data files currently stored on the meter.

Calibration Commands

Code	Description
M4	Triggers a manual calibration
R90	Read the calibration delta
R91	Read the calibration level
R92	Read the calibration history

System Commands

Code	Description
M5*130	Initiates the meter shutdown.
M8*133	Initiates the meter reboot.

Calibration and Cal Check

In performing a Calibration, the meter will reset the filters and reset the data. The meter must be stopped.

In order to perform a Cal Check, the meter must have an Environmental Preamp (PRM2103 or 426A12) attached. In order to use this method your meter must be paused or stopped. You can set your settings on the meter to perform a Cal Check automatically while running in continuous mode. See the Auto Cal Check (**114**) setting to configure the meter.

id: integer, may be either 279 or 280

1. `CMD_CALIBRATE = (279)`
    a. `Full Calibration to be performed.`
    b. `Full reset required.`
2. `CMD_CHECK_CALIBRATION = (280)`
    a. `Cal Check to be performed.`
    b. `Meter must be in paused or stopped state.`

p1: integer,

May be one of the following values for **Calibration**: 1-3.

1. `PRM_CAL_START = (1)`
    a. `Starts a calibration`
2. `PRM_CAL_APPLY = (2)`
    a. `Ends the calibration and stores the result`
    b. `Use this after the status has returned done and you wish to store the Calibration and apply the offset.`
3. `PRM_CAL_ABORT = (3)`
    a. `Ends the calibration without storing the result`

May be one of the following values for **Cal Check**: 1-4.

1. `PRM_CAL_CHECK_START = 1`
    a. `Starts a Cal Check`
2. `PRM_CAL_CHECK_END_1 = 2`
    a. `Ends the Cal Check, stores the history, and stores new std. level`
    b. `Use this value for p1 after the status has returned done and you wish to store this Cal Check in the history`
3. `PRM_CAL_CHECK_END_2 = 3`
    a. `Ends the Cal Check, stores the history, but does not store new std. level`
4. `PRM_CAL_CHECK_ABORT = 4`
    a. `Ends the Cal Check without storing any results`

Example of how to start a Cal Check:

/sdk?func=cmd&id=280&p1=1&format=ack:i4

Response:
{ "Response":{"ack":0},"Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }

Where ack is the return response for the start of performing the Cal Check. ack == 0 means success.

```
$.getJSON("/sdk?func=cmd&id=280&p1=1&format=ack:i4", function (data) {
                        // send Start Calibration
        if (data["Response"]["ack"] == "0") {
                console.log("Successfully started the Cal Check");
        }
        else
        {
                console.log("Failed to start the Cal Check");
        }
});
```

You can abort the Cal check with:

/sdk?func=cmd&id=280&p1=4&format=ack:i4

If the ack == -65534 then you might try calling the abort function to try to set the meter in the right state to perform the Cal Check.

Calibration Status and Recorded Level

Once you have started a Cal Check you may monitor the Stability and Level of the meter. The **FUNC_READ_LIVE_CALIBRATION_LEVEL** is used to monitor the progress of a Cal Check. When the *stab* value returns -1 the Cal Check is complete; see below.

```
FUNC_READ_LIVE_CALIBRATION_LEVEL = (0xA9)   ///< (169)
```

Example:

/sdk?func=get&f=169&format=level:f4,stab:i4,ack:i4

Response:
{ "Response":{"level":1.558578e+06,"stab":0,"ack":0},"Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }

level: calibration level that will be recorded.

ack: integer, 0 = successful call, other values = invalid state.

stab: integer, stability indicator valid values are:

- `CS_TIMEOUT              = -2,`
- `CS_DONE                 = -1,`
- `CS_FALLING_RAPIDLY      =  0,`
- `CS_FALLING_MODERATELY   =  1,`
- `CS_FALLING_SLOWLY       =  2,`
- `CS_STABLE               =  3,`
- `CS_RISING_SLOWLY        =  4,`
- `CS_RISING_MODERATELY    =  5,`
- `CS_RISING_RAPIDLY       =  6`

The Standard Electrostatic Actuator Level for Cal Check

The System Property TAG_STD_EA_LEVEL is used to get the Level used by the Cal Check. You obtain the value from the System Properties:

sdk?func=getProperties&subset=sys

You may also use the standard methods for Getting and Setting Properties. For more details on this tag, review the Appendix D System Property Descriptions.

Calibration and Cal Check Data

You may use the **getcalspectrum** function to obtain the data from a calibration or cal check.

Parameters:

index – specifies which of the spectra to retrieve. Valid values are 0-9

check – optional, if included will specify that you want Cal Check data otherwise the spectrum will be for Calibration. Valid value is true.

Example:

/sdk?func=getcalspectrum&check=true&index=0

Response of a potential Cal Check spectrum:

```
{"Data":[5.327987e+01,4.296607e+01,5.462307e+01,4.877151e+01,5.726844e+
01,5.519419e+01,4.057322e+01,9.354131e+01,5.068034e+01,4.607697e+01,3.9
70083e+01,3.474782e+01,3.683249e+01,4.311489e+01,2.795535e+01,4.193641e
+01,9.356870e+01,4.068466e+01,3.183941e+01,4.270462e+01,3.880296e+01,4.
748045e+01,9.357658e+01,4.792123e+01,4.273923e+01,4.623689e+01,4.625741
e+01,4.970310e+01,9.378906e+01,5.142803e+01,4.966982e+01,9.418066e+01,5
.119566e+01,5.404717e+01,5.194556e+01,4.682945e+01],
"Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Example:

/sdk?func=getcalspectrum&index=8

Response of a potential Calibration spectrum at index 8:

```
{ "Data":[5.605194e-45,-4.910856e-01,1.087457e-38,1.075638e-38,-
4.910856e-01,1.087457e-38,1.075638e-38,1.075638e-38,-4.910856e-
01,1.087457e-38,1.087457e-38,-4.911737e-01,-6.371012e-06,-6.436654e-
06,-6.431714e-06,-9.999900e+03,1.088545e-38,0.000000e+00,-4.910831e-
01,-4.910791e-01,1.413938e-39,1.298970e-39,0.000000e+00,1.088544e-38,-
5.727792e-06,1.087457e-38,-4.911737e-01,-4.911081e-01,-4.911157e-01,-
4.910852e-01,1.075638e-38,-4.910856e-01,1.087457e-38,1.087457e-38,-
4.911737e-01,-6.308173e-06],"Result" : "Success: 0
","ResultCode":0,"ResultName":"Success" }
```

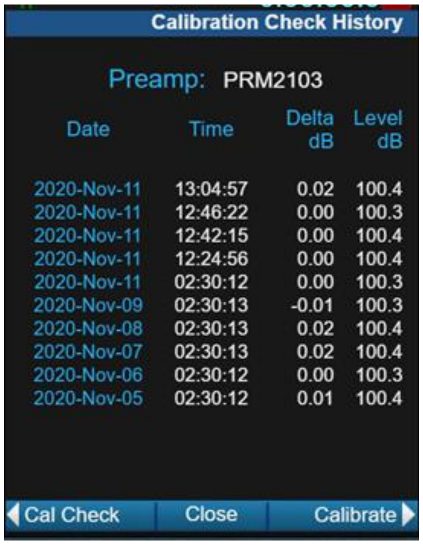
Array of the Spectrum Entries – calHistJson – 831C only

Obtain the list of currently stored array of the spectrum entries with the following command:

Example: http://ipAddress:port/sdk?func=calhistjson&check=true

```
{ "Entries":
[{"timeStamp":1605099897,"scale":0.0221176,"delta":100.369},
{"timeStamp":1605098782,"scale":0,"delta":100.346},
{"timeStamp":1605098535,"scale":0,"delta":100.372},
{"timeStamp":1605097496,"scale":0,"delta":100.361},
{"timeStamp":1605061812,"scale":0.00050354,"delta":100.348},
{"timeStamp":1604889013,"scale":-0.00800323,"delta":100.34},
{"timeStamp":1604802613,"scale":0.0185165,"delta":100.366},
{"timeStamp":1604716213,"scale":0.0179825,"delta":100.366},
{"timeStamp":1604629812,"scale":0.000396729,"delta":100.348},
{"timeStamp":1604543412,"scale":0.00856018,"delta":100.356}
],"Result":"Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Which is the data that corresponds to this view in G4:



Preamp: PRM2103

Date	Time	Delta dB	Level dB
2020-Nov-11	13:04:57	0.02	100.4
2020-Nov-11	12:46:22	0.00	100.3
2020-Nov-11	12:42:15	0.00	100.4
2020-Nov-11	12:24:56	0.00	100.4
2020-Nov-11	02:30:12	0.00	100.3
2020-Nov-09	02:30:13	-0.01	100.3
2020-Nov-08	02:30:13	0.02	100.4
2020-Nov-07	02:30:13	0.02	100.4
2020-Nov-06	02:30:12	0.00	100.3
2020-Nov-05	02:30:12	0.01	100.4

The following command returns the spectrum for the most recent calCheck:

http://ipAddress:port/sdk?func=getcalspectrum&check=true&index=0

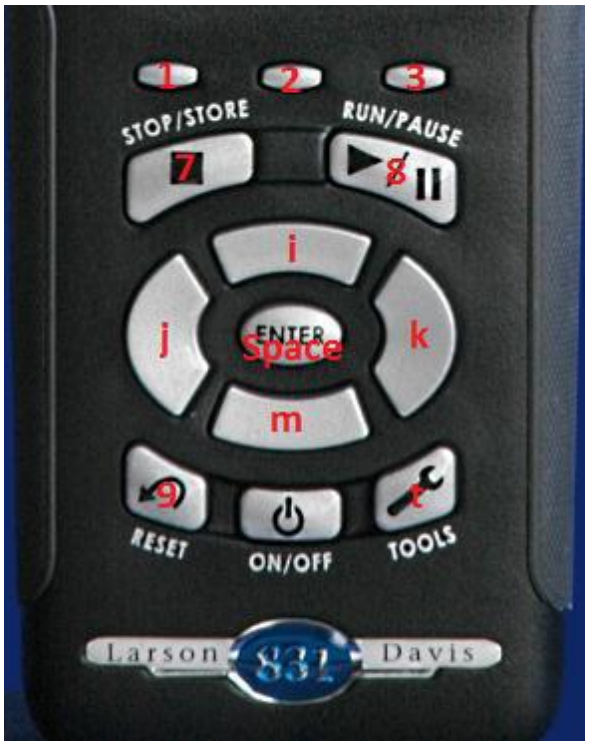
And this returned the calCheck for Nov 6th as shown in the G4 view above:

http://ipAddress:port/sdk?func=getcalspectrum&check=true&index=8

sendKeyCommand (Legacy SLMs only)

Send a Key Command directly to the meter. The chart below describes the key code values and the associated key.

```
//Enter Key
http://ipAddress[:port]/sdk?func=sendKeyCommand&id=32
```



Decimal	Hex	Char	Command
49	0x31	1	S Right
50	0x32	2	Menu
51	0x33	3	S Left
32	0x20	Space	Enter
55	0x37	7	Stop/Store
56	0x38	8	Run/Pause
57	0x39	9	Reset
105	0x69	i	Arrow Up
106	0x6A	j	Arrow Left
107	0x6B	k	Arrow Right
109	0x6D	m	Arrow Down
116	0x74	t	Tools

Status Functions – Get System Info

These functions will provide varying amounts of status about the meter. They are provided to allow for more granular selection of the flags. The three main flags are described below (uiRunStatus, uiStatusFlags and uiLxTFlags).

On the 831C, the meter's state is also returned. This can be used to determine the run state of the meter.

getPageStatus

Description: Reads the three status flags as well as other useful information. This is the most complete of the Status requests.
Sample call:

http://ipAddress[:port]/sdk?func=getPageStatus

Example of **831A**:

```
{ "Status":{"sSerialNumber":"0001472","sModel":"831
","fw_ver":"2.403","sHldVersion":"4.600","INT_ET":0,"LockMode":0,"fBatV
olt":0,"fEventLevel":0,"fExtVolt":4.7535,"fPeak":1.65645e+06,"fSPL":704
.009,"fTemperature":34.675,"nEventNum":0,"nEventTime":0,"nMode":0,"tv_s
ec":1604758915,"tv_usec":600000,"uiRunFlags":2147483776,"nErrorFlags":0
,"uiLxTFlags":4035838085,"uiStatusFlags":4,"file_count":67,"free_mem_kb
":1016,"total_mem_kb":1883628,"timeRemaining":0,"sPreamp":"PRM831"},"Re
sult":"Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Example of **831C:**

```
{
"Status":{"sSerialNumber":"0000050","sModel":"831C","fw_ver":"04.6.0T10
8","sHldVersion":"4.600","INT_ET":0,"sPreamp":"PRM831","fBatVolt":0.004
12598,"fExtVolt":12.0518,"fUsbVolt":5.00481,"fTemperature":32,"upTimes"
:{"app":1115869,"sys":1115896},"loads":{"s1":2.55566,"s5":2.26953,"s15"
:2.25684},"tz":"US\/Mountain","nMode":0,"tv_sec":1605537750,"tv_usec":4
00000,"total_mem_kb":1914764,"free_mem_kb":1852184,"file_count":16,"uiM
eterState":3,"uiLxTFlags":4027188228,"uiRunFlags":2415919232,"nErrorFla
gs":0,"uiStatusFlags":2,"indicators":33570840,"indicators2":0,"notifID"
:0,"notifFlags":0,"LockMode":0,"fPeak":1.31406e+07,"fSPL":1426.22},"Res
ult":"Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Example of **HVM200**:

```
{ "Status":{
"sSerialNumber":"0000057","sModel":"HVM200","fw_ver":"4.6.0R0","sHldVer
sion":"4.600","INT_ET":0,"fBatVolt":100.000000,"fEventLevel":0.000000,"
fExtVolt":0.000000,"fPeak":0.003048,"fSPL":0.001422,"fTemperature":0.00
0000,"nErrorFlags":0,"nEventNum":0,"nEventTime":0,"nMode":1,"tv_sec":16
14083705,"tv_usec":662580,"uiLxTFlags":0,"uiRunFlags":268435456,"uiStat
usFlags":0,"LockMode":0,"indicators":1879048192,"indicators2":0,"notifI
D":0,"notifFlags":0,"CalDate":1561528800,"ManufDate":1437976800,"nBatSt
ate":3,"nBatTime":30522,"TEDS":{"X":{"Valid":"true","Man":"PCB","MN":"3
56","VL":"B","VN":"18","SN":"16389","Sen":"95.710556","Units":"mV\/(m\/
s^2)"},"Y":{"Valid":"true","Man":"PCB","MN":"356","VL":"B","VN":"18","S
N":"16389","Sen":"95.595779","Units":"mV\/(m\/s^2)"},"Z":{"Valid":"true
","Man":"PCB","MN":"356","VL":"B","VN":"18","SN":"16389","Sen":"96.8077
85","Units":"mV\/(m\/s^2)"}},"file_count":2,"free_mem_kb":15000992,"tot
al_mem_kb":15001168,"timeRemaining":17778950}, "Result":"Success: 0
","ResultCode":0,"ResultName":"Success" }
```

Example of **721/821**:
Note that communicating over BLE to the meter requires a running instance of the LDConnectionServer or the LDConnectionSerivce

```
{
 "Status": {
  "Device": "Larson Davis
SoundExpert 821",
  "Serial Number": "0000005",
  "Firmware Revision": "1.001R27",
  "Hardware Revision": "X1",
  "Battery Voltage": 4.2,
  "Estimated Run Time": 46816,
  "Free Memory": 30031,
  "Total Memory": 30286,
  "LAeq": 40.6,
  "LCeq": 54.61,
  "LZeq": 62.3,
  "LAS": 47.6,
  "LAF": 40.9,
  "LAI": 60.6,
  "LCS": 57.1,
  "LCF": 52.8,
  "LCI": 67.4,
  "LZS": 73.4,
  "LZF": 60.6,
  "LZI": 83.1,
  "LApeak": 66.1,
  "LCpeak": 70.7,
  "LZpeak": 73.5,
  "Mode": 3,
  "Overload": 0,
  "Time": 1691572902,
  "Start Time": 1691572814,
  "Stop Time": 1691572902,
  "Runtime": 88,
  "Motion Percentage": 0.00,
  "LAeq Overall": 38.9,
  "LCeq Overall": 51.6,
  "LZeq Overall": 65.6,
  "LASMin": 26.9,
  "LAFMin": 24.1,
  "LAIMin": 30.2,
  "LCSMin": 48.6,
  "LCFMin": 45.8,
  "LCIMin": 51.8,
  "LZSMin": 56.8,
  "LZFMin": 52.6,
"LZIMin": 60.9,
  "TASMin": 1691572846,
  "TAFMin": 1691572865,
  "TAIMin": 1691572872,
  "TCSMin": 1691572861,
  "TCFMin": 1691572850,
  "TCIMin": 1691572861,
  "TZSMin": 1691572861,
  "TZFMin": 1691572865,
  "TZIMin": 1691572862,
  "LASMax": 59.0,
  "LAFMax": 60.8,
  "LAIMax": 71.9,
  "LCSMax": 63.1,
  "LCFMax": 70.0,
  "LCIMax": 74.2,
  "LZSMax": 80.0,
  "LZFMax": 85.8,
  "LZIMax": 88.4,
  "TASMax": 1691572814,
  "TAFMax": 1691572879,
  "TAIMax": 1691572814,
  "TCSMax": 1691572900,
  "TCFMax": 1691572899,
  "TCIMax": 1691572814,
  "TZSMax": 1691572900,
  "TZFMax": 1691572900,
  "TZIMax": 1691572900,
  "LApeakMax": 88.5,
  "LCpeakMax": 88.9,
  "LZpeakMax": 91.4,
  "TApeakMax": 1691572900,
  "TCpeakMax": 1691572900,
  "TZpeakMax": 1691572899,
  "Excd Count SPL1": 0,
  "Excd Dur SPL1": 0,
  "Excd Count SPL2": 0,
  "Excd Dur SPL2": 0,
  "Excd Count Peak1": 0,
  "Excd Dur Peak1": 0,
  "Excd Count Peak2": 0,
  "Excd Dur Peak2": 0,
  "Excd Count Peak3": 0,
  "Excd Dur Peak3": 0,
  "SEL": 58.4,
  "SEL Pa2H": 0.000000,
  "SEL8 Pa2H": 0.000025,
  "SEL40 Pa2H": 0.000125,
  "SEL Pa2S": 0.000275,
  "SEL8 Pa2S": 0.089919,
  "SEL40 Pa2S": 0.449596,
  "Dose1 Lavg": -999.9,
  "Dose1 Ltwa": -999.9,
  "Dose1 P.Ltwa": -999.9,
  "Dose1 LEP'd": 13.8,
  "Dose1 P.LEP'd": 38.9,
  "Dose1 Dose": 0.0,
  "Dose1 P.Dose": 0.0,
  "Dose2 Lavg": -999.9,
  "Dose2 Ltwa": -999.9,
  "Dose2 P.Ltwa": -999.9,
  "Dose2 LEP'd": 13.8,
  "Dose2 P.LEP'd": 38.9,
  "Dose2 Dose": 0.0,
  "Dose2 P.Dose": 0.0,
  "Dose3 Lavg": -999.9,
  "Dose3 Ltwa": -999.9,
  "Dose3 P.Ltwa": -999.9,
  "Dose3 LEP'd": 13.8,
  "Dose3 P.LEP'd": 38.9,
  "Dose3 Dose": 0.0,
  "Dose3 P.Dose": 0.0,
  "Dose4 Lavg": -999.9,
  "Dose4 Ltwa": -999.9,
  "Dose4 P.Ltwa": -999.9,
  "Dose4 LEP'd": 13.8,
  "Dose4 P.LEP'd": 38.9,
  "Dose4 Dose": 0.0,
  "Dose4 P.Dose": 0.0,
  "OBA Live Leq": [],
  "OBA Meas Leq": [],
  "OBA Lmax": [],
  "OBA Lmin": [],
  "Percentage": 82,
  "Battery Capacity": 4967.0,
  "Current": 234,
  "Avg Curr": 234,
  "Cal Delta": 0.00,
  "Stored Cal Records": 3,
  "Cal History Number": 3,
  "Timer Next Event Date": 0,
  "Timer Next Event Time": 0,
  "Meas Timer:": 9498,
  "Alarm1 Units": "%",
  "Alarm1 LED": "LED Off",
  "Alarm2 Units": "%",
  "Alarm2 LED": "LED Off",
  "Error Flags": 196608,
  "Accelerometer Data": "0,0,0",
  "Indicators": 512,
  "File Count": 65,
  "Temperature": 25,
  "Alert State": 0,
  "BLE FW Ver": "0.001",
  "Options": 29,
  "Model": "821SE",
  "LAIeq": 62.1,
  "Overall LAIeq": 55.9,
  "Ln 1": 49.7,
  "Ln 2": 44.4,
  "Ln 3": 33.0,
  "Ln 4": 31.0,
  "Ln 5": 29.8,
  "Ln 6": 28.3,
  "LDN": 38.9,
  "LDN Day": 38.9,
  "LDN Night": -99.9,
  "LDEN": 38.9,
  "LDEN Day": 38.9,
  "LDEN Evening": -99.9,
  "LDEN Night": -99.9,
  "Meas Hist Intervals": 0,
  "Pause Time": 0,
  "Overall Graph Data": [34.4, 35.2,
31.3, 31.7, 33.1, 35.7, 35.3, 33.1, 29.6,
30.9, 28.4, 27.4, 32.6, 28.1, 29.1, 29.3,
28.9, 29.6, 29.7, 27.9, 41.0, 34.9, 33.1,
27.1, 28.6, 27.3, 26.2, 31.9, 29.2, 27.2,
27.5, 27.3, 29.9, 31.0, 29.3, 27.3, 30.2,
30.8, 32.6, 31.9, 27.6, 29.2, 30.3, 38.3,
28.2, 25.9, 31.1, 30.9, 31.2, 27.7, 28.6,
26.7, 31.4, 27.8, 31.7, 29.2, 26.1, 28.3,
27.1, 31.7, 28.6, 31.9, 35.4, 34.8, 35.4,
52.5, 32.3, 45.1, 39.0, 31.7, 31.0, 38.3,
33.2, 31.2, 29.7, 31.3, 31.7, 30.5, 31.7,
33.1, 28.1, 33.8, 28.0, 32.6, 43.7, 51.2,
52.4, 40.6],
  "FW Date": 1687533445,
  "FW Install Date": 1687514842,
  "Overload Duration": 0,
  "Overload Count": 0,
  "Live Graph Data": [33.0, 34.8, 31.9,
33.4, 29.0, 40.9, 44.5, 34.6, 33.7, 35.3,
32.1, 31.4, 31.5, 32.4, 31.2, 34.1, 30.2,
34.6, 36.5, 33.7, 34.7, 43.7, 38.4, 34.9,
35.7, 37.1, 38.5, 39.5, 38.3, 60.0, 53.2,
62.0, 34.4, 35.2, 31.3, 31.7, 33.1, 35.7,
35.3, 33.1, 29.6, 30.9, 28.4, 27.4, 32.6,
28.1, 29.1, 29.3, 28.9, 29.6, 29.7, 27.9,
41.0, 34.9, 33.1, 27.1, 28.6, 27.3, 26.2,
31.9, 29.2, 27.2, 27.5, 27.3, 29.9, 31.0,
29.3, 27.3, 30.2, 30.8, 32.6, 31.9, 27.6,
29.2, 30.3, 38.3, 28.2, 25.9, 31.1, 30.9,
31.2, 27.7, 28.6, 26.7, 31.4, 27.8, 31.7,
29.2, 26.1, 28.3, 27.1, 31.7, 28.6, 31.9,
35.4, 34.8, 35.4, 52.5, 32.3, 45.1, 39.0,
```

31.7, 31.0, 38.3, 33.2, 31.2, 29.7, 31.3, 31.7, 30.5, 31.7, 33.1, 28.1, 33.8, 28.0, 32.6, 43.7, 51.2, 52.4, 40.6], "Est Run Time Screen Off": 115362	}, "hldVer": "4.900", "Result": "Success: 0 ", "ResultCode": 0, "ResultName": "Success" }

getMinStatus

Description: Minimal information is returned on this request.

http://ipAddress[:port]/sdk?func=getMinStatus

Example of 831A:

```
{
"Status":{"nMode":0,"tv_sec":1605017264,"tv_usec":0,"uiRunFlags":214748
3776,"nErrorFlags":0,"uiLxTFlags":4035838085,"uiStatusFlags":4,"file_co
unt":67,"free_mem_kb":1016,"total_mem_kb":1883628,"timeRemaining":0},"R
esult":"Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Example of **831C**:

```
{
"Status":{"nMode":0,"tv_sec":1605537893,"tv_usec":0,"total_mem_kb":1914
764,"free_mem_kb":1852184,"file_count":16,"uiMeterState":3,"uiLxTFlags"
:4027188228,"uiRunFlags":2415919232,"nErrorFlags":0,"uiStatusFlags":2,"
indicators":33570840,"indicators2":0,"notifID":0,"notifFlags":0},"Resul
t":"Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

getVersionStatus

Description: The request retrieves Version information as well as two of the main flags. Used to verify the firmware version and compatibility.

Example of return on 831A:

```
{"Status": { "LockMode": 0, "nMode": 0, "tv_sec": 1481151222,
"tv_usec": 100000, "fw_ver": "2.900", "uiRunFlags": 2147483776,
"uiStatusFlags": 0  }, "Result" : "Success: 0
","ResultCode":0,"ResultName":"Success" }
```

getStatusEx

Description: Reads three status flags, named StatusFlags, LxTFlags, and RunFlags from the meter.

http://ipAddress[:port]/sdk?func=getStatusEx, i.e., http://10.3.3.100/sdk?func=getStatusEx

Example of return:

```
{ "Status": { "fBatVolt": 4.984180, "fEventLevel": 0.000000,
"fExtVolt": 11.693930, "fPeak": 4347298.500000, "fSPL": 61413.792969,
"fTemperature": 34.000000, "nErrorFlags": 0, "nEventNum": 0,
"nEventTime": 0, "nMode": 0, "tv_sec": 1481140270, "tv_usec": 900000,
"uiLxTFlags": 4027187204, "uiRunFlags": 2147483776, "uiStatusFlags": 0
}, "Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Description of Return Values

The following tags are used in each of the Status functions. Please refer to them to understand the return values.
Tag ID: sModel
Description: Returns the model name as a string.

Tag ID: LockMode
Description: Whether or not the meter is Locked: 0 = unlocked; 1 = locked.

Tag ID: fBatVolt
Description: Voltage of the Battery

Tag ID: fEventLevel
Description: Level of Event

Tag ID: fExtVolt
Description: Voltage of external source

Tag ID: fPeak
Description: Peak

Tag ID: fSPL
Description: Model 831 and LxT - L_{AS} may not line up with a stored value.
SoundAdvisor 831C – Level follows weighting (A, C or Z) and detector (F, S or I) selected in the SLM tab or Measurement Properties
If you desire to obtain a particular level, use the Streaming Data functions.

Tag ID: fTemperature
Description: Current temperature of meter.

Tag ID: nErrorFlags
Description: See chart below.

Tag ID: nEventNum
Description: Number of Events in current Measurement

Tag ID: nEventTime
Description: Time of most recent Event

Tag ID: nMode
Description: Mode of the meter: 0=SLM; 1=RA/RT60; 2=FFT

Tag ID: tv_sec
Description: Time Value in Seconds since Epoch. You may wish to include tv_usec for more accurate time.

Tag ID: tv_usec

Description: Time Value in micro seconds in the current tv_sec.

Tag ID: uiLxTFlags
Description: See Chart Below

Tag ID: uiRunFlags
Description: See Chart Below

Tag ID: sPreamp
Description: String of the currently connected Preamp

Tag ID: fw_ver
Description: Firmware Version of the Meter

Tag ID: free_mem_kb
Description: Memory available in Kilobytes.

Tag ID: total_mem_kb
Description: Total memory in Kilobytes

Tag ID: file_count
Description: Number of Measurement files on meter.

Tag ID: uiStatusFlags
Description: See Chart Below.

Tag ID: uiMeterState
Description: **Only available on the 831C.** This is used to determine the meter's valid running state. See Chart Below.

Status Flags

uiStatusFlags uses the following bits to represent different conditions in the meter:

Status Flag	Value	Description:
UI_SETNG_SEMAPHORE	0x00000001	The SLM user interface is active, you may not want to make any changes to settings which may be overwritten when the users closes the SLM user interface.
USB_SETNG_SEMAPHORE	0x00000002	The SLM is connected via USB.
UI_AT_BASE_LEVEL	0x00000004	The SLM User interface is at base level. I.e. no menu or system panels are open. Used to let software know that the user is not in the middle of possibly changing settings which could be lost if the software makes differing changes.
UI_NOT_READY	0x00000008	UI is not ready on the SLM.
ANALOG_SETNG_SEMAPHORE	0x00000010	The SLM is connected via Analog modem.

RS232_SETNG_SEMAPHORE	0x00000020	The SLM is connected via RS232 cable.
EDGE_SETNG_SEMAPHORE	0x00000040	The SLM is connected via Wireless modem.
UI_DATA_EXP_OPEN	0x00000080	Data Explorer is open on the SLM

LxT Flags

uiLxTFlags uses the following bits to represent different conditions in the SLM:

LxT Flag	Value	Description
SLM_RUN	0x00000002	SLM is Running
SLM_VALID	0x00000004	Data is Valid
SLM_RESET	0x00000400	Data is Reset
SLM_PAUSED	0x00000800	SLM is Paused
SLM_RECORD	0x00001000	SLM is recording a voice or audio file
SLM_PLAY	0x00002000	SLM is playing a voice or audio file
SLM_SLEEP	0x00004000	SLM is in power-save mode, analog power is off...
SLM_TIMER_END	0x00008000	A Timed run or “until stable” run is completed
SLM_STORED	0x00010000	SLM data has been Stored to file
SLM_STARTED	0x00020000	Instrument restarted
SLM_SLM_APP_RDY	0x00080000	SLM has finished file system checks and App is ready
SLM_OBA_UNDER_NOW	0x02000000	OBA is now under range. Set for 1 second minimum.
SLM_UNDER_NOW	0x04000000	SLM is now under range. Set for 1 second minimum.
SLM_PREAMP_LOW_RG	0x08000000	The preamp is designed for low range input. (LxT)
SLM_PREAMP_TYPE_1	0x10000000	The preamp is a type 1 preamp (LxT)
SLM_PREAMP	0x20000000	A preamp is connected (else Direct input)
SLM_THIRD_ENABLE	0x40000000	The Overall Third Octave is enabled
SLM_OCTAVE_ENABLE	0x80000000	The Overall Octave is enabled

Run Flags

uiRunFlags uses the following bits to represent different conditions in the SLM:

Run Flag	Value	Description
RUNNING	0x00000001	SLM is in a Running state.
POST_RUN	0x00000004	SLM is in a Run Pending state.
PAUSED	0x00000200	SLM is in a Paused state.
POWERING_ON	0x01000000	SLM is in Powering Up state. (bootloader)
DATA_RESET	0x10000000	SLM date is in Reset state. (no data to store)
DATA_STORED	0x20000000	SLM data is in Stored state. (saved but not reset)

Meter Status

uiMeterState: See the descriptions below to understand the value. This is only available on the 831C. You can request the meter state directly with:

http://localhost:2508/sdk?func=get&f=0x82&p2=0xB0&format=state:i4

State	Value	Description
state_Error	0	STATUS: An error occurred! On startup will start DSP and go to Stop.
state_Initializing	1	STATUS: The meter is initializing. Valid Cmd: cmd_Stop
state_Stopped	2	STATUS: Stopped with unsaved data Valid Cmd: cmd_Run, cmd_Reset, cmd_Store,cmd_Calibrate, cmd_Record, cmd_Play
state_StopReset	3	STATUS: Stopped and Reset Valid Cmd: cmd_Run, cmd_Calibrate, cmd_Record, cmd_Play
state_StopStored	4	STATUS: Stopped and Stored Valid Cmd: cmd_Run, cmd_Reset, cmd_Calibrate, cmd_Record, cmd_Play
state_Waiting	5	STATUS: Waiting for valid data before going to run mode Valid Cmd: cmd_Stop, cmd_Reset (resets and stops)
state_Running	6	STATUS: Running Valid Cmd: cmd_Status, cmd_Update, cmd_Pause, cmd_Stop, cmd_Reset (resets & continues running)
state_Paused	7	STATUS: Paused Valid Cmd: cmd_Run, cmd_Stop
state_PrevOk	7	NOTE: *** Only states equal or less than cmd_PrevOK can be saved as previous state
state_Sleeping	8	STATUS: Sleep, power save mode, not processing data and analog system off Valid Cmd: cmd_Stop, cmd_Reset, cmd_Run, cmd_Calibrate, cmd_Record, cmd_Play
state_Calibrating	9	STATUS: Calibrating Mode Valid Cmd: cmd_CalEnd
state_Recording	10	STATUS: Recording a voice or audio record Valid Cmd: cmd_RecordEnd
state_RecordDone	11	STATUS: Recording done Valid Cmd: cmd_Play, cmd_RecordDiscard, cmd_RecordSave
state_Playing	12	STATUS: Playing a voice or audio record Valid Cmd: cmd_PlayEnd
state_CheckingCal	13	
state_USBRC0	14	Status: Recovering data from inserted USB driver Valid Cmd: cmd_Stop
state_WaitFileRecovery	15	Status: State to keep SlmApp from running until file save is received and processed. Valid Cmd: cmd_Stop

state_Maintenance	16	Status: State for maintenance like file system checks to prevent running Valid Cmd: cmd_Maintenance
state_RestorePrev	17	Special internal state to restore previous mode

nErrorFlags

```
None                        : 0x00000000,  // NONE
FlashReadFailure            : 0x00000001,  // LDERR_FLASH_READ_FAILURE
FlashReadDismissed          : 0x00000002,  // LDERR_FLASH_READ_DISMISSED
FileSystemCorrupt           : 0x00000004,  // LDERR_FILE_SYSTEM_CORRUPT
SerialNumberError           : 0x02000000,

UpgradeStat3                : 0x04000000,
UpgradeStat2                : 0x08000000,
UpgradeStat1                : 0x10000000,
UpgradeType                 : 0x20000000,

NoUserSDCard                : 0x40000000,
NoSensorConnected           : 0x80000000,

/// MSB of nErrorFlags:  t : type, where 0 = options, 1 = firmware
UC                 : 0x3C000000,   /// x x M M - M M x x
UCMask             : 0x1C000000,   /// x x t M - M M x x
UCUpgStarted       : 0x04000000,   /// x x t 0 - 0 1 x x
UCUpgSuccess       : 0x08000000,   /// x x t 0 - 1 0 x x
UCDowngrade        : 0x0C000000,   /// x x t 0 - 1 1 x x
UCSameVersion      : 0x10000000,   /// x x t 1 - 0 0 x x
UCError            : 0x1C000000    /// x x t 1 - 1 1 x x
```

Masked Options

These are the values you can mask to turn off installed options if you wish to not see in the UI or not use them. They can be queried to know if the meter has these features.
System Properties: TAG_MASK_OPTION and TAG_OPTION_FLAGS.

TAG_MASK_OPTION can be read or set. Making changes to the Mask Options requires a meter reboot.

TAG_OPTION_FLAGS can be read. They are the purchased options of the meter.

Option	Value	Description	Instrument
OPT_THIRD_OCTAVE	0x00000001	third octave	LxT, 831
OPT_ONE_OCTAVE	0x00000002	full octave	LxT, 831
OPT_MESSAGE	0x00000004	voice recording	LxT, 831(s)
OPT_AUDIO	0x00000008	audio recording	831
OPT_EVENTS	0x00000010	exceedance event Time history	831
OPT_DATALOGGING	0x00000020	Data logging (Time History)	LxT, 831
OPT_DOSE	0x00000040	dose	LxT(s), 831
OPT_ANYDATA	0x00000080	a,c,z data	831(s)
OPT_INTV	0x00000100	Intervals, Daily, Timed measurement controls	831
OPT_WEATHER	0x00000200	Wind spd, wind dir., Temperature, Humidity	831
OPT_COMMUNITY	0x00000400	community noise (environmental)	831(s)
OPT_GPS	0x00000800	Global-positioning	831
OPT_HSLOG	0x00001000	High speed Time Hist (adds 100ms, 200ms, and 500ms th periods)	LxT
OPT_ENV	0x00002000	Environmental	831
OPT_FST	0x00004000	831 fast Time History (adds 10ms, 5ms, and 2.5ms th periods)	831
OPT_WIRELESS_MDM	0x00008000	Wireless Modem (EDGE)	831
OPT_ANALOG_MDM	0x00010000	Analog Modem	831
OPT_ROOMS	0x00020000	Room acoustics (SIL)	831
OPT_QC_TONALITY	0x00040000	Corrected dBA for Quebec	LxT, 831
OPT_RT60	0x00080000	RT60 (subset of OPT_ROOMS)	831
OPT_FFT	0x00100000	FFT	831
OPT_RS232	0x00200000	RS232 communication	831
OPT_FILE_AVERAGING	0x00400000	File Averaging	831
OPT_NF30_STATS	0x00800000	NF30-101 (France Ln of 25-2k Hz)	831
OPT_FAST_WEATHER	0x01000000	Fast Weather Time History 831	

Managing Secure Sessions

Available as of firmware version 4.5 on the 831C.

Logging In to an 831C

To log in simply post to the “login” function via the SDK with an appropriate username and password combination in the body, be sure to use TLS to do this (use HTTPS instead of HTTP), otherwise the password will be sent in plain text.

Example:

/sdk?func=login

POST BODY: {"User":"johndoe@email.com", "Pass":"really good password"}

Response:

```
{
    "UserInfo": {"Name": "John", "Perm": 7, "FailCount": 0, "FailLogin": 0},
    "Result": "Success: 0 ",
    "ResultCode": 0,
    "ResultName": "Success"
}
```

Set-Cookie Header:

```
sessionid=ewKjs59kFqSb2KPp9GJdoBHbGnhmQJmG1JNBAQ2sjDtulvaP4LymJSL157YSY3lNj6y/
fQ9WXd/FFVyh8u+ymvNtygy0aCoM5GRHx00dvwlOXPluddURYIG6aEBgqhggXZVcNoof9T8Opk+
FITtmlBwxlYNcyx1nB2vnUFgweoNgGWW5xVQ9WVVEJcqfG2HuySd3sQzaEqgKhz54LuhkwH+R
WfOjXJWVTTXZVe4q1v/UtKLaJ6BVsrxRaJk8J2MT3O5cOF/pOJ0U6AVT4D2MVvXBs8aWQcG91
+9BAKF4Jfioq/spKNTo5C8VsXjmi6sObIcSn/rbXLb//EK0PsWypw==; Max-Age=900; HttpOnly
```

The UserInfo returned in the response JSON provides the Nickname, what permissions the user has, and some login failure metrics. The Set-Cookie header contains the secure session id. This cookie must be added to all future requests until after a logout has been sent. Requests sent without it will result in an “Unauthorized” error.

Session Timout and Keep Alive

The session will automatically timeout after 15 minutes, regardless of activity unless a “keepalive” call is made. It is possible call this on a timer but it is easier to call it in response to a session id update which happens a few minutes before the session expires.

Example:

/sdk?func=keepalive

Response:

```
{ "Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

In addition to resetting the session timer, this call also resets the modem powersave timer. If you have configured your system with the modem off for power save feature a keepalive will also keep the modem awake.

Managing and Downloading Data Files

Getting the Data File List

Before you can download any data files you must first retrieve the file list from the meter. This section documents how to do this. Which steps are required depends on the type of meter with which you are communicating.

	API Path	Description	Meter Support
1	/sdk?func=startFileListBuild	Tells the meter to prepare its file list for enumeration.	Only required for Legacy SLMs and the the 831c with Firmware older than 4.8.1. Not supported on 730, 721, 821, and HVM200
2	/sdk?func=queryFileListStatus	Queries if the meter is ready to provide a file list. Will return a magic value of 2468 when ready for legacy support, check the **complete** token.	Only needed if "startFileListBuild" was used.
3	/sdk?func=getDataFileList	Returns the prepared list of files.	Supported by all meter types.
4	/sdk?func=getManifestList	Returns the prepared list, similar to "getDataFileList" but has additional details for each file and the total file count.	Only supported by 831c with Firmware 4.8.1 or newer.

1. startFileListBuild

This is used to tell the meter to prepare a file list. The meter will go and search the internal memory for files stored on the meter. If the meter is a SoundAdvisor Model 831C then a parameter can be sent to also get files stored on the USB. The other requests should not be made to the meter except **queryFileListStatus** in order for the process to complete as quickly as possible. The meter may not respond to other func calls.

API path and parameters: "/sdk?func=startFileListBuild&dpath={**pathType**}"

pathType Values	Description
0	Internal flash only. (831A and Legacy LxT ignore other values)
1	USB flash drive only. (Only for 831c)
2	Both internal flash and USB flash. (Only for 831c, *recommended*)+

Response:
```
{ "Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

2. queryFileListStatus

Description: After calling startFileListBuild, the meter may need time to sort through all the files found on the internal and optionally USB device. Files which contain a large number of Sound Records will increase the time it takes to generate the file list. This function is used to check to see if the meter is ready with the file list so that **getDataFileList** may be called to retrieve the list.

API path and parameters: "/sdk?func=queryFileListStatus"

Response:
```
{ "Progress":{"ReturnValue":2468,"complete":true},"Result" : "Success:
0 ","ResultCode":0,"ResultName":"Success" }
```

Note: You will need to query the status until you receive "ReturnValue: 2468" or "complete: true", as this denotes the meter has finished building the list. Depending on your configuration, the build file list may take some time. The 831C has improved the speed of this process.

3. getDataFileList

Description: Used to get the list of data files on the meter. This will return all the filenames and associated data to allow a listing for the user.

Note: The file list is limited to 3200.

API path and parameters: "/sdk?func=getDataFileList&sindex={**startIndex**}&count={**count**}"

> **startIndex**: The index to start the block at, must be a value less than the file count returned from the page and min status calls.
>
> **count**: [Optional] The maximum number of files to include in the response. Defaults to 500 if not specified. Generally, we advise not specifying this and letting it default to 500.

Response:
Example with startIndex: 0, count: 500 and 3 files on the meter.
```
{ "DataFiles": [ {"name": "831_Data.001.s", "index": 0, "startTime":
1493827922, "size": 43484,"location": 2},{"name": "831_Data.002.s",
"index": 1, "startTime": 1493899879, "size": 355448,"location":
2},{"name": "831_Data.003.s", "index": 2, "startTime": 1493901806,
"size": 331516,"location": 2} ] }
```

Note: If the meter has less than the ***count*** number files it will only send you the number of entries that are actually on the meter. And the response will be empty if you have an ***sindex*** that is greater than the total number of files, ex: { "DataFiles": [ ] }.

4. getManifestList

Only supported by the 831c.

API path and parameters: “/sdk?func=getManifestList&index={**startIndex**}&count={**count**}”

index={**startIndex**}: The index to start the block at, must be a value less than the file count returned from the page and min status calls.

count=count: [Optional] The maximum number of files to include in the response. Defaults to 500 if not specified. Generally we advise not specifying this and letting it default to 500.

Examples

Getting the file list from 831c

```
JObject fileList = null;

bool result = meterModel.PostCommandToHLD(IpAddress, Port, "/sdk?func=getManifestList", "
DataFiles", out fileList);
```

Getting the file list HVM200, 730, 721, 821

```
JObject fileList = null;

bool result = meterModel.PostCommandToHLD(IpAddress, Port, "/sdk?func=getDataFileList", "D
ataFiles", out fileList);
```

Getting the file list from legacy meters

The process of getting the file list on legacy meters is a three step process, 1) Inform the meter to prepare the file list, 2) Ask the meter if it has finished building the list, 3) Request the file list once we have learned that the meter has finished building the list.

Sample code in C#:

1) Inform the meter to prepare the file list with: func=startFileListBuild

```
JObject fileList = null;

bool result = meterModel.PostCommandToHLD(IpAddress, Port, "/sdk?func=startFileListBuild", "
Success", out fileList);
```

2) Ask the meter if it has finished building the list with func=queryFileListStatus

```
public static bool CheckIfFileListReady(MeterModel model, ref bool isClosing, bool prevTaskErre
d)
{
        bool err = prevTaskErred;

        if (!err)
        {
                int timeout = 120000;
                JObject jsonReturn = null;
                bool done = false;

                while (!done && timeout > 0)
                {
                        if (isClosing)
                        {
                                break;
                        }


        if (model.PostCommandToHLD(model.IpAddress, model.DownloadPort, "/sdk?func=que
ryfileliststatus", "Success", out jsonReturn))
                        {

        if (jsonReturn != null && jsonReturn["Progress"] != null && jsonReturn["Progress"]["Ret
urnValue"] != null)
                                {

        string value = (string)jsonReturn["Progress"]["ReturnValue"];
                                        if (value == "2468")
                                        {
                                                done = true;
                                                continue;
                                        }
                                }

                                timeout -= 1000;
                                MessagePump.SleepWithPump(1000);
                        }
                }

                if (timeout == 0) err = true;
                MessagePump.SleepWithPump(200);
        }

        return err;
}
```

3) Request the file list with: func=getDataFileList.

```
JObject fileList = null;

bool result = meterModel.PostCommandToHLD(IpAddress, Port, "/sdk?func=getDataFileList", "D
ataFiles", out fileList);
```

Example return: { "DataFiles": [{"name": "RA_Data.001.r", "index": 0, "startTime": 1394014880, "size": 407284},{"name": "831_Data.001.s", "index": 1, "startTime": 1394014942, "size": 43196},{"name": "831_Data.002.s", "index": 2, "startTime": 1394015351, "size": 43220},{"name": "831_Data.006.s", "index": 3, "startTime": 1395138823, "size": 150028},{"name": "831_Data.004.s", "index": 4, "startTime": 1394464889, "size": 780136},{"name": "831_Data.005.s", "index": 5, "startTime": 1394721379, "size": 81304},{"name": "831_Data.007.s", "index": 6, "startTime": 1397028790, "size": 69736}] }

Downloading a Data File

Downloading Data File with a Stream (Legacy Slm, 831c, Hvm200)

The following steps are needed to correctly download a file that is compatible with the ldbin format.

1. Create a file for writing.
2. Create web request the "/download" endpoint.
    a. /download?index={**index**}&size={**size**}" can be used to download from 831, 831c, HVM200, and legacy LxT meters.
        i. **index** is the index from the getDataFileList function for the desired file.
        ii. **size** is the filesize returned in the getDataFileList for the desired file.
        iii. NOTE: If a file was deleted between getting the file list and downloading a meter all indeces after the one deleted will no longer be valid and attempting to use them may lead to unexpected behavior or corrupted downloads. You can aliviate this by downloading from the highest index first and then deleting that file so the other files' indeces are unaffected.
    b. The 831c also supports downloading by name as of firmware version 4.8.1. /download?name={**name**}&size={**size**}&path={**pathType**}
        i. **name** is the file name from the file list.
        ii. **size** is the file size as returned from the file list.
        iii. **pathType** identifies if the file is located on the internal flash or an attached thumb thumb drive.
3. If you wish the file to be translatable, write the LdBin headers to the file before writing any data from the download.
    a. The headers include:
        i. The "LD" Tag (Int32)0x00004C44
        ii. The "Bin" Tag (Int32)0x0042494E
        iii. The Version Number (Int32)0x00000001
        iv. The Record Count (Int32) count
            1. Multiple records in one file is DEPRECATED, always use 1
        v. Record Size (Int32) size

   b. Note for Legacy files: If the file has a record count other than 1, the file contains multiple records and there will be one Record Size located immediately before each individual record’s data.

4. Then while there are still remaining bytes on the stream read them and append to the file or till the filesize of read bytes have been read, which should be the same.

Sample code C# (see C# CS_SDK_Example):

```
private void StreamDownload(FileInfo localFile, int size, int index)
{
	Console.WriteLine("Starting to stream download.");

	using (var targetFile = new BinaryWriter(localFile.Create()))
	{

	string cmd = string.Format("{0}/download?index={1}&size={2}", RootUri, index, size);
		WebRequest request = WebRequest.Create(cmd);

		using (WebResponse response = request.GetResponse())
		{
			WriteLDBinHeader(targetFile, size);
			using (Stream dataStream = response.GetResponseStream())
			{
				byte[] buffer = new byte[4096];
				int totalRead = 0;
				int read = dataStream.Read(buffer, 0, buffer.Length);

				while (read > 0)
				{
					targetFile.Write(buffer, 0, read);
					read = dataStream.Read(buffer, 0, buffer.Length);
					totalRead += read;
					Console.WriteLine("{0} bytes downloaded", totalRead);
				}
			}
		}
	}
}
```

Downloading Data Files in Chunks (730, 721, 821)

Download files from a 730, 721, or 821 by following this process.

1. Retrieve the current file list from the meter, see Getting the Data File List, specifically 3. getDataFileList.

2. Identify the file to download; you will use the **name** and **size** properties to perform the download.
3. Determine the download **chunkSize**, larger chunks will increase latency but will reduce the overall time it take to download the entire file slightly. G4 uses a 150 KB block size for USB connections.
4. Make repeated calls to "/v2?func=download&name={**name**}&size={**blockSize**}&offset={**offset**}". Each response will have a payload containing the data file chunk.
    a. **blockSize** is to the minimum of **chunkSize** or **(size - offset)**.
    b. Initially **offset** should be 0 and should be increased by the number of bytes returned in each chunk.
    c. It is possible to resume an interrupted download by providing the correct offset to where the download was inturrpted.
5. Once the response has been read you can close the file being written to and the download is complete.

No additional file header is required to make it translatable by G4.

G4 uses the extension ".ld7" to identify a file from the 730, 721, or 821 meters.

Deleting Data Files

deleteDataFile

Description: Delete a Data file from the SLM.

Example in C#:

```
string mode = string.Empty;
switch (file.Mode)
{
        case InstrumentMode.Slm:
                mode = ".s";
                break;
        case InstrumentMode.Fft:
                mode = ".f";
                break;
        case InstrumentMode.RT60:
                mode = ".r";
                break;
        case InstrumentMode.AudCal:
                mode = ".a";
                break;
        case InstrumentMode.Hvm:
                mode = "." + file.Extension;
                break;
}
string strDelete = string.Format("/sdk?func=deleteDataFile&filename={0}{1}", file.FileName, mo
de);
```

```
success = meterModel.PostCommandToHLD(IpAddress, Port, strDelete, "Success", out jsonObjec
t);
```

DeleteFile/DeleteFiles

This code is in C# and used to delete files from the meters. The PathType is used to distinguish between USB and Internal.

```
public enum PathType {
        /// <summary>
        /// The file name only
        /// </summary>
        FileNameOnly,
        /// <summary>
        /// Internal or USB, based on setting
        /// </summary>
        Primary,
        /// <summary>
        /// Internal file
        /// </summary>
        Internal,
        /// <summary>
        /// File is on usb
        /// </summary>
        USB,
        /// <summary>
        /// Merged (not valid in fileList)
        /// </summary>
        Merged
}
```

Delete file from Model 831/LxT:

C# sample code to delete a file from an 831 or LxT. You can delete one file at a time.

filename = name of the file without extension
type = “.s”, “.f”, or “.r” same as the **mode** above
storage = PathType.USB or PathType.Internal

```
public override async Task<JObject> DeleteFile(string filename, string type, i
nt storage)
{
        JObject result = new JObject();
        string fileWithExt = filename + "." + type;
        if (FileList.FirstOrDefault(f => fileWithExt == (string)f["name"] && s
torage == (int)f["storage"]) is JObject file)
        {

        JsonResponse response = await Connection.GetJsonAsync($"/sdk?func=file
cmd&fn={filename}.{type}&fs={(storage == (int)(PathType.USB) ? 1 : 0)}&cmd=3&i
ndex={(int)file["index"]}");
                if (response.Success)
                {
                        result = response.Json;
                        IsFileListUpdateNeeded = true;
                }
                else
                {
                        result["Result"] = "Failed to delete file";
                }
        }
        else
        {
                result["Result"] = "Failed to delete file: not in file list";
        }

        return result;
}
```

Delete file from 831C:

C# sample code to delete files from the 831C. You may send in more than one file at a time to delete with one call to HLD.

file.name = name of the file without extension
file.type = ".s", ".f", or ".r" same as the **mode** above.
file.storage = PathType.USB or PathType.Internal

```
public override async Task<JObject> DeleteFiles(List<FileNode> toDelete)
{
        JObject result = new JObject();

        StringBuilder json = new StringBuilder();
        json.Append("{\"files\":[");
        foreach (var file in toDelete)
        {
        json.Append($"{{\"name\":\"{file.name}.{file.type}\",\"fs\":{((file.st
orage == (int)PathType.USB) ? 3 : 2)}}},");
        }
        json.Remove(json.Length - 1, 1);
        json.Append("]}");
        JsonResponse response = await Connection.GetJsonAsync("/sdk?func=filec
md&cmd=12", UTF8Encoding.UTF8.GetBytes(json.ToString()));
        if (response.Success && response.Json.ToString().Contains("Success"))
        {
                result = response.Json;
        }
        else
        {
                result["Result"] = "Failed to delete files";
        }
        return result;
}
```

Delete file from 730, 721, and 821 Meters:

C# sample code that shows how to delete files from a Spartan.

filename = name of the file without extension
type = unused
storage = unused

```
public override async Task<JObject> DeleteFile(string filename, string type, i
nt storage)
{
        JObject result = null;
        RawResponse response = await SendSpartanRawCommand(“D8” + Path.GetFile
NameWithoutExtension(filename) + ".LD7");

        if (response.Success)
        {
                result = new JObject { {"Result", "Success" } };
        }
        else
        {
                result = new JObject { { "Result", "Failed" } };
        }

        return result;
}
```

Delete file from HVM200:

C# sample code to show how to a delete file from an HVM200.

filename = name of the file without extension
type = unused
storage = unused

```
public override async Task<JObject> DeleteFile(string filename, string type, i
nt storage)
{
        var resp = await Connection.GetJsonAsync($"/sdk?func=filecmd&fn={filen
ame}{type}&cmd=3");
        JObject result = null;

        if (resp.Success)
        {
                result = resp.Json;
        }
        else
        {

        result = new JObject { ["Result"] = "Failed to delete file" };
        }

        return result;
}
```

Measurement and System Property Files

These functions will allow you to get the list of Property files. You can load them so that you may make modifications and then save them back. You can even set the modified Properties as the active configuration. Some of the examples, in the Visual Studio Projects, demonstrate how to use Properties.

CODE: See the **Appendix: Transfer Settings** for code samples for transferring Measurement and System Property files. The example code can be used to better understand the calling of the functions defined in the following sections.

CODE: Example code for Updating Properties can be found in the **Appendix: Update Properties.**

Measurement Property Files

These functions can be used to manipulate the Measurement Settings Files on the meter. For some of these functions you will need to prepare HLD with some session data. The following post session data is required for deleteSettingsFile and renameSettingsFile:

measPropFiles

Description: Returns a list of Measurement Settings/Property files from the meter.

downloadSettingsToCache

Description: Used to prepare the meter to allow you to change the currently selected settings file. You will need to use the query string to send in this request the value is the index of the file from the FileInfoList:

/sdk?func=downloadSettingsToCache&id=3

Example in JavaScript:

```
///One function to download either Settings to Cache
function downloadPropertiesToCache(nDx) {
       var getDataLoc = "func=downloadSettingsToCache&index=" + nDx;
       $.getJSON('/sdk?' + getDataLoc)
       .done(function (dataVal) {
         //error handling
         errorHandling("downloadSettingsToCache", dataVal);
       });
}
```

After performing this action you may post changes to the settings as described in the **Appendix: Update Properties**. You may then perform uploadSettingsFromCache to store the settings back to the meter.

uploadSettingsFromCache

Description: Saves the settings that have been changed to the meter. You will need to use the query string to send in this request:

/sdk?func=uploadSettingsFromCache

If you wish to save the file to a different name you may provide one by:

/sdk?func=uploadSettingsFromCache&fileName=YourFileNewName

Note: Do not include a file extension and the filename length may not exceed 10 characters.

Example in JavaScript:

```
function uploadPropertiesFromCache(filename) {
       if (filename == undefined) {
          filename = "";
       }
       if (filename != "") {
          // Verify that the filename is not longer than 10
          if (filename.length > 10) {
              filename = filename.substr(0, 10);
          }
          if (filename.indexOf(".") > 0) {
            filename = filename.substring(0, filename.indexOf("."));
          }
          filename = encodeURI(filename);
          filename = "&filename=" + filename;
       }
       var getDataLoc = "func=uploadSettingsFromCache";
       $.getJSON('/sdk?' + getDataLoc + filename)
       .done(function (dataVal) {
          //error handling
          errorHandling("uploadPropertiesFromCache", dataVal);
       });
}
```

downloadSettingsFile

Description: Downloads a selected settings file. You will receive the content of the http request as the binary Settings file. You may save this off directly to a file. The file can contain the filename as part of the header in the file. See below on how to change or add the filename to the setup header in C#.

To prep the HLD to use the desired Settings file from the list previously acquired, you will need to post the filename and index to the session of the HLD as shown here in JavaScript.

```
var SessionID;
function setSessionValue(key, value) {
      var keyPairs = {};
      keyPairs[key] = value;
      if (SessionID == undefined || SessionID == 0) {
        SessionID = Date.now();
      }
      $.post("sdk?setSessionValue=" + SessionID, keyPairs)
      .done(function (data) {
            //console.log(data);
      });
}
// Set the index in the session
setSessionValue("nDx", 0);
setSessionValue("filename", "MySettings.s");
```

uploadSettingsFile

Description: You can upload a settings file that you previously downloaded. You must include post data with this call, which will only contain the binary Settings file. You may not upload a Settings file with the name “Active”. If you need to update the Active settings use downloadSettingsToCache (above) or setToActive (described below).

deleteSettingsFile

Description: Used to delete a selected settings file on the meter.

Example in JavaScript:

```
function deletePropertiesFile(filename) {
      if (filename != undefined && filename != "") {
        filename = "&filename=" + filename;
        var getDataLoc = "func=deleteSettingsFile";
        $.getJSON('/sdk?' + getDataLoc + filename)
             .done(function (dataVal) {
             //error handling
             errorHandling("deletePropertiesFile", dataVal);
        });
      } else {
        console.log("When Deleting, you must specifiy a new Filename.");
      }
}
```

renameSettingsFile

Description: Rename a settings file on the meter.
Parameters:

1. filename – the new file name to which you wish to change the currently selected file.
    a. You must not rename the settings file to “Active”.

Example in JavaScript:

```
function renameSettingsFile(newFilename) {
       if (newFilename != undefined && newFilename != "")
       {
         newFilename = "&filename=" + newFilename;
         var getDataLoc = "func=renameSettingsFile";
         $.getJSON('/sdk?' + getDataLoc + newFilename)
         .done(function (dataVal) {
              //error handling
              errorHandling("renameSettingsFile", dataVal);
         });
       } else {
         console.log("When Renaming, you must specifiy a new Filename.");
       }
}
```

System Property Files

These functions are similar to the Measurement Property functions. Review that section on how to use them.

1. downloadSysPropToCache – Same as downloadSettingsToCache above except this is used for System Properties or Preferences.
2. uploadSysPropFromCache – Same as uploadSettingsToCache above except this is used for System Properties or Preferences.
3. downloadSysPropFile – Download a Preferences File. You will receive the content of the http request as the binary file.
4. uploadSysPropFile – See Settings.

Property Helpers

These functions are used with the above Property functions.

setToActive

Description: Sets the currently selected (using func=setSessionValue) Settings file as the active configuration. Usually, one would call this function after uploading a file.

Using the calls from the defined function ***setSessionValue*** below, which uses a Ajax post command to set the values for the nDx and filename.

```
setSessionValue("nDx",  selectedIndex);
setSessionValue("filename",  selectedSettingsFilename);
```

commitSettings

Description: Used to modify an individual setting. – Deprecated 2016/10/1

clearCache

Description: This function clears the cache when working with Property Files. If you do not wish to keep changes that you made, and you haven't used "uploadSettingsFromCache", call func=clearCache.

Miscellaneous Functions

setSessionValue

Description: May be used to set any key value pair at the server level. We use this value to keep track of which Property file is currently selected. The SessionID (id) must be greater than 2000, otherwise the values will not be stored.

Parameter in Query String: id – int, your SessionID, a value greater than 2000.

Post values: key – string, name of object to store;
value – any, the value associated with the key. May be any type.

```
var SessionID;
function setSessionValue(key, value) {
        var keyPairs = {};
        keyPairs[key] = value;
        if (SessionID == undefined || SessionID == 0) {
          SessionID = Date.now();
        }
        $.post("sdk?func=setSessionValue&id=" + SessionID, keyPairs)
          .done(function (data) {
               errorHandling("setSessionValue ", dataVal);
          });

}

setSessionValue("nDx",  selectedIndex);
setSessionValue("filename",  selectedSettingsFilename);
```

getSessionValue

Description: May be used to retrieve a key value pair previously set at the server level (previous SessionID must be sent). JavaScript example:

```
SessionData = {};
function getSessionValue() {
        if (SessionID == undefined || SessionID == 0) {
          SessionID =Date.now();
        }
        $.getJSON("sdk?getSessionValue=" + SessionID)
          .done(function (data) {
               SessionData = GenerateCache(data);
               console.log(SessionData);
          });

}
```

Get and Set Properties (Deprecated)

This is legacy support. Use Get and Set Properties with JSON for all future development.

In order to get and set Measurement Settings or System Preferences, the functions getProperty and setProperty have been defined. They take at least the parameters of tag (integer value) and type. When setting the property (setting or preference), use the value and index parameters.

Important Note:

If you change settings, system properties or preferences on a meter, you must remember to call CMD_LATCH_SETTINGS from the SetOperation Commands section. You send 2 for measurement properties and 0 for system and 1 for preferences.

CODE: The JavaScript example for setting properties can be found in the **Appendix, Update Properties.** The code examples describe how to set multiple properties simultaneously, instead of changing one by one. If you have to set more than one property or need to set a string you should use func=setProperties. If you need to set a single number type property then func=setProperty is a good choice.

Entry Description and Usage Example

The following is the description of the entries in the Property Tables below:

- Name = the name of the property you wish to get/set
- Description = brief explanation of property
- Tag Name = used in enums or #defines
- Tag ID = number to be sent as part of the get or set property functions
- Type = one of the 9 valid types

Type	Value	Notes
int	1	integer numeric values - TAG_DISPLAY_CONTRAST
uint	2	Unsigned integer - TAG_EVENT_TH_PRE_TRIG Including enums - TAG_OBA_RANGE
float	3	Float values like - TAG_EVENT_MIN_DURATION
floatSeries	4	A series of floats, where you must specify the size TAG_FULL_OCTAVE_REFERENCE_SPECTRA1
tableOfFloats	4	Deprecated – same as float series.
floatwithflags	5	TAG_IO_ANYLEVEL_SPL
string	6	Must be URL Encoded when setting this property and when retrieving, it will also be URL Encoded. TAG_OVERALL_TITLE
byteseries	7	
uintseries	8	Unsigned integer series
uintwithflags	10	Unsigned integer with flags

- Value = the new value you are setting for the given tag.

- **Index** = some of the properties require an index into its array of values. Ln Percentiles is an example of an array of floats. Index defaults to 0. And only needs to be used when accessing an array.

Set Properties:

You will need to call latch settings after performing a setProperty call.

Examples of Setting an Individual Property:

1. http://ipAddress:port/sdk?func=setProperty&type=string&value=I%27m%20the%20man&tagid=1431515185

Example to Setup Time history on the meter with 1 minute period and only the LAeq set:

1. Enable Time History:
   http://ipAddress:port/sdk?func=setProperty&type=int&tagid=0x54483031&value=1
2. Set TH Period:
   http://ipAddress:port/sdk?func=setProperty&type=int&tagid= 0x54483032&value=12
3. Set the TH Options:
   http://ipAddress:port/sdk?func=setProperty&type=int&tagid= 0x54483033&value=1024

Be sure to Latch the Properties:

Should use the following instead where data=2 means Measurement Properties
http://ipAddress:port /sdk?func=LATCHSETTINGS&data=2&marksettingschanged=true

Get Properties:

Examples of Getting an Individual Property:

1. http://localhost/sdk?func=getProperty&type=string&tagid=1431515185
2. http://localhost/sdk?func=getProperty&type=string&tagid=0x55533031
3. http://localhost/sdk?func=getProperty&type=string&tag=US01
4. http://localhost/sdk?func=getProperty&tagid=0x4C4E3032&type=floatseries&size=6

Measurement Properties or Control Settings

The following Measurement Properties (settings) are available for use on the 831. Note the Special Handling information which may require a reset or stop before saving the changes. You may wish to refer to the **SlmInclude.cs, LxT831.h** or **Tags.js** in the **Include** folder for the values for the Tag IDs. To see the complete list of Properties refer to: **Tags.js** with their associated rules.

> **Important Note**: Each string field is modified through the SDK, which can allow special characters like the degree symbol, which may not be parsable by the JSON parser. This means that when a JSON request is performed the result may throw an error. In this case do a “search and replace” on the returned string before parsing with JSON. It is

possible to replace the offending characters with alternates or remove them completely.

The descriptions of the Measurement Properties can be found in the **Appendix: <u>Measurement Property Descriptions</u>.**

System Properties/Preferences

The following System Properties (preferences) are available for use on the 831. Note the Special Handling information which may require a reset or stop before saving the changes. You may wish to refer to the **SlmInclude.cs, LxT831.h** or **Tags.js** in the **Include** folder for the values for the Tag IDs. To see the complete list of System Properties refer to: **Tags.js** with their associated rules.

> **Important Note**: Each string field is modified through the SDK, which can allow special characters like the degree symbol, which may not be parsable by the JSON parser. This means that when a JSON request is performed the result may throw an error. In this case do a "search and replace" on the returned string before parsing with JSON. It is possible to replace the offending characters with alternates or remove them completely.

The descriptions for the System Properties can be found in the **Appendix: <u>System Property Descriptions</u>.**

Allowed Values

Some of the Properties used defined values, like Display Options as described in the next sub-section. The Enumerations can be found in the LxT831.h, SdkEnums.h, SlmInclude.cs and Tags.js.

Other Defined Enums

```
public enum HvmType
{
       HVM100 = 0,
       HVM200 = 1
}

public enum ObaType
{
       Full = 0,
       Third = 1
}

public enum SeriesName
{
       RMS = 0,
       Peak = 1,
       Max = 2,
}
```

```
public enum AveragingMode
{
        Slow = 0,
        A1Sec = 1,
        A2Sec = 2,
        A5Sec = 3,
        A10Sec = 4,
        A20Sec = 5,
        A30Sec = 6,
        A60Sec = 7,
        A2Min = 8,
        A5Min = 9,
        A10Min = 10,
        A20Min = 11,
        A30Min = 12,
        A60Min = 13,
};

public enum OperatingMode
{
        Vibration = 0,
        HandArm = 1,
        WholeBody = 2
};

public enum AutoStoreMode
{
        Off = 0,
        On = 1,
        AutoStop = 2
};

public enum SensorType
{
        Direct = 0,
        ICP = 1,
        Charge = 2
};

public enum HvmUnits
{
        M_PerSecSq = 0,
        Cm_PerSecSq = 1,
        Ft_PerSecSq = 2,
        Inch_PerSecSq = 3,
        g = 4,
        dB = 5
}

public enum ReferenceLevel
{
        R10e_5 = 0,
        R10e_6 = 1
};

public enum AxisGain                    // HVM100 only
{
		G0 = 0,
		G20 = 1,
		G40 = 2,
		G60 = 3
	};

	public enum IntMethod				// integration
	{
		None = 0,
		Single = 1,
		Double = 2
	};

	public enum Weighting
	{
		Ws = 0,					// Severity
		Fa = 1,					// 0.4-100 Hz
		Fb = 2,					// 0.4-1250 Hz
		Fc = 3,					// 6.3-1250 Hz
		Wh = 4,					//
		Wm = 5,					// Buildings
		Wb = 6,					// Railways-Z
		Wc = 7,					// Seatback-X
		Wd = 8,					// Horizontal-XY
		We = 9,					// Rotational
		Wg = 10,				// BS 8461-Z
		Wj = 11,				// Head-XMeasProps
		Wk = 12					// Vertical-Z
	};

	public enum AcDcOut
	{
		AC_Weighted = 0,
		AC_Bandlimit = 1,
		DC_Rms = 2,
		DC_Min = 3,
		DC_Max = 4,
		DC_Peak = 5,
		DC_RmsSum = 6,
		DC_MinSum = 7,
		DC_MaxSum = 8,
		DC_PeakSum = 9
	};

	public enum StorePeakEn
	{
		None = 0,
		Peak = 1
	};

	public enum ExposureRef
	{
		R2_8 = 0,
		R2_5 = 1,
		R4_0 = 2,
		R5_0 = 3
	};
```

Over/Under range values:

```
                X-axis   Y-axis   Z-axis
                =====    =====    =====
Overload:       0x01     0x02     0x04          (bit values)
Under Range:    0x08     0x10     0x20
```

Start Delay

```
1 = 5 seconds
2 = 10 seconds
3 = 20 seconds
4 = 30 seconds
5 = 60 seconds
default = 0 seconds
```

Save mask / save flags

```
#define SAVE_THIRD   0x01
#define SAVE_FULL    0x02
#define SAVE_RAW     0x04
```

Display Options Notes

To control the display of pages in the instrument, simply set or clear the appropriate bit in the display option flag. For example, to only display the LiveProfile and Live11 screen, pass a value of 3 (DISPLAY_BIT0 + DISPLAY_BIT1) to SetControl, using TAG_DISPLAYOPTIONS1 for the tag. To determine the value of the display bit to set or clear, simply raise 2 to the bit number. For example, the value of DISPLAY_BIT3 is 2^3 = 8. The value of DISPLAY_BIT6 is 2^6 = 64.

Note: Only TAG_DISPLAYOPTIONS1 through TAG_DISPLAYOPTIONS4 are currently used.

Page	Display Option Tag	Option Bit
Live		
LiveProfile	TAG_DISPLAYOPTIONS1	MASK_BIT_0_
LiveEnfMkt	TAG_DISPLAYOPTIONS1	MASK_BIT_25
Live11	TAG_DISPLAYOPTIONS1	MASK_BIT_1_
Live13	TAG_DISPLAYOPTIONS1	MASK_BIT_2_
LiveTrigger	TAG_DISPLAYOPTIONS1	MASK_BIT_3_
LiveWeather	TAG_DISPLAYOPTIONS1	MASK_BIT_4_
LiveVaisalaWeather	TAG_DISPLAYOPTIONS3	MASK_BIT_22
LivePreamp	TAG_DISPLAYOPTIONS1	MASK_BIT_23
LiveGPS	TAG_DISPLAYOPTIONS1	MASK_BIT_5_
LivePower	TAG_DISPLAYOPTIONS1	MASK_BIT_6_
Overall		
OverallGraph	TAG_DISPLAYOPTIONS1	MASK_BIT_7_
OverallEnfMkt	TAG_DISPLAYOPTIONS1	MASK_BIT_26
OverallLeq	TAG_DISPLAYOPTIONS1	MASK_BIT_8_
Overall11	TAG_DISPLAYOPTIONS1	MASK_BIT_9_
OverallQTonality	TAG_DISPLAYOPTIONS3	MASK_BIT_25
Overall13	TAG_DISPLAYOPTIONS1	MASK_BIT_10
OverallPercentiles	TAG_DISPLAYOPTIONS1	MASK_BIT_14

OverallSpectralLn	TAG_DISPLAYOPTIONS1	MASK_BIT_22
OverallExceedences	TAG_DISPLAYOPTIONS1	MASK_BIT_15
OverallOverload	TAG_DISPLAYOPTIONS1	MASK_BIT_17
OverallCommunity	TAG_DISPLAYOPTIONS1	MASK_BIT_19
OverallMisc	TAG_DISPLAYOPTIONS1	MASK_BIT_20
OverallTALarm	TAG_DISPLAYOPTIONS1	MASK_BIT_24
OverallSEL	TAG_DISPLAYOPTIONS1	MASK_BIT_13
OverallDose1	TAG_DISPLAYOPTIONS1	MASK_BIT_11
OverallDose2	TAG_DISPLAYOPTIONS1	MASK_BIT_12
OverallSEA	TAG_DISPLAYOPTIONS1	MASK_BIT_16
OverallWeather	TAG_DISPLAYOPTIONS1	MASK_BIT_21
OverallVaisalaWeather	TAG_DISPLAYOPTIONS1	MASK_BIT_27
OverallMemory	TAG_DISPLAYOPTIONS1	MASK_BIT_18
OverallGPS	TAG_DISPLAYOPTIONS3	MASK_BIT_20
Current		
CurrentGraph	TAG_DISPLAYOPTIONS2	MASK_BIT_0_
CurrentEnfMkt	TAG_DISPLAYOPTIONS3	MASK_BIT_13
CurrentLeq	TAG_DISPLAYOPTIONS2	MASK_BIT_1_
Current11	TAG_DISPLAYOPTIONS2	MASK_BIT_2_
CurrentQTonality	TAG_DISPLAYOPTIONS3	MASK_BIT_26
Current13	TAG_DISPLAYOPTIONS2	MASK_BIT_3_
CurrentPercentiles	TAG_DISPLAYOPTIONS2	MASK_BIT_7_
CurrentSpectralLn	TAG_DISPLAYOPTIONS2	MASK_BIT_13
CurrentExceedences	TAG_DISPLAYOPTIONS2	MASK_BIT_8_
CurrentOverload	TAG_DISPLAYOPTIONS2	MASK_BIT_10
CurrentMisc	TAG_DISPLAYOPTIONS2	MASK_BIT_12
CurrentSEL	TAG_DISPLAYOPTIONS2	MASK_BIT_6_
CurrentTALarm	TAG_DISPLAYOPTIONS2	MASK_BIT_11
CurrentDose1	TAG_DISPLAYOPTIONS2	MASK_BIT_4_
CurrentDose2	TAG_DISPLAYOPTIONS2	MASK_BIT_5_
CurrentSEA	TAG_DISPLAYOPTIONS2	MASK_BIT_9_
CurrentWeather	TAG_DISPLAYOPTIONS2	MASK_BIT_14
CurrentVaisalaWeather	TAG_DISPLAYOPTIONS3	MASK_BIT_23
CurrentGPS	TAG_DISPLAYOPTIONS3	MASK_BIT_21
Measurement		
MeasurementHistoryGraph	TAG_DISPLAYOPTIONS2	MASK_BIT_15
MeasurementProfileGraph	TAG_DISPLAYOPTIONS2	MASK_BIT_16
MeasurementEnfMkt	TAG_DISPLAYOPTIONS3	MASK_BIT_14
MeasurementMetrics	TAG_DISPLAYOPTIONS2	MASK_BIT_17
MeasurementRemainingLeq	TAG_DISPLAYOPTIONS3	MASK_BIT_29
Measurement11	TAG_DISPLAYOPTIONS2	MASK_BIT_18
MeasurementQTonality	TAG_DISPLAYOPTIONS3	MASK_BIT_27
Measurement13	TAG_DISPLAYOPTIONS2	MASK_BIT_19
MeasurementPercentiles	TAG_DISPLAYOPTIONS2	MASK_BIT_23
MeasurementSpectralLn	TAG_DISPLAYOPTIONS2	MASK_BIT_29
MeasurementExceedences	TAG_DISPLAYOPTIONS2	MASK_BIT_24
MeasurementOverload	TAG_DISPLAYOPTIONS2	MASK_BIT_26
MeasurementMisc	TAG_DISPLAYOPTIONS2	MASK_BIT_27
MeasurementSEL	TAG_DISPLAYOPTIONS2	MASK_BIT_22
MeasurementTALarm	TAG_DISPLAYOPTIONS2	MASK_BIT_30
MeasurementDose1	TAG_DISPLAYOPTIONS2	MASK_BIT_20
MeasurementDose2	TAG_DISPLAYOPTIONS2	MASK_BIT_21
MeasurementSEA	TAG_DISPLAYOPTIONS2	MASK_BIT_25
MeasurementWeather	TAG_DISPLAYOPTIONS2	MASK_BIT_28

MeasurementVaisalaWeather	TAG_DISPLAYOPTIONS3	MASK_BIT_24
MeasurementGPS	TAG_DISPLAYOPTIONS3	MASK_BIT_19
Exceedance		
ExceedanceTrigStatusDisplay	TAG_DISPLAYOPTIONS3	MASK_BIT_17
ExceedenceMetrics	TAG_DISPLAYOPTIONS3	MASK_BIT_1_
Exceedence11	TAG_DISPLAYOPTIONS3	MASK_BIT_2_
Exceedence13	TAG_DISPLAYOPTIONS3	MASK_BIT_3_
ExceedenceTimeHistory	TAG_DISPLAYOPTIONS3	MASK_BIT_4_
ExceedenceSpectralTimeHistory	TAG_DISPLAYOPTIONS3	MASK_BIT_5_
ExceedenceSpectralTimeHistoryByTime	TAG_DISPLAYOPTIONS3	MASK_BIT_18
Time History		
TimeHistoryGraph	TAG_DISPLAYOPTIONS3	MASK_BIT_7_
TimeHistory11	TAG_DISPLAYOPTIONS3	MASK_BIT_8_
TimeHistory13	TAG_DISPLAYOPTIONS3	MASK_BIT_9_
TimeHistoryFST11	TAG_DISPLAYOPTIONS3	MASK_BIT_15
TimeHistoryFST13	TAG_DISPLAYOPTIONS3	MASK_BIT_16
Session Log		
SessionLogPage	TAG_DISPLAYOPTIONS3	MASK_BIT_10

SFTP and Cloud Storage (831C only)

The use of "SFTP" in the tag names in this section represents both SFTP and Cloud Storage. In order to set the properties for SFTP/Cloud storage, you will need to follow these steps:

1. Download *rclone* for your platform (https://rclone.org/downloads/).
2. Extract the *rclone* executable to a usable location.
3. Open a command prompt or terminal to the location where you extracted *rclone*.
4. Run "rclone --config FILENAME config" and where FILENAME is a name you would like to use for the file
5. Follow the prompts with two constraints.
    a. For the service name you must enter "sftp" if you intend to set up sftp or "dropbox" for a dropbox service.
    b. On the type of storage the 831c currently only supports SFTP and Dropbox, any other service type will not be found by the meter.
6. Once the entry has been completed find the file FILENAME and post its contents to the meter
    a. Use "IPADDRESS/sdk?func=setproperty&tagid=0x46545048&type=7" for the url and "POST" for the method and send the contents.
7. Set the properties for `TAG_SFTP_PUSH_CTRL`, `TAG_SFTP_SERVICE_TYPE`, `TAG_SFTP_PATH`, `TAG_SFTP_EMAIL_CONTROL`, and `TAG_SFTP_MOVE_COPY`.
    a. `TAG_SFTP_PUSH_CTRL`
        i. 0 means the meter will not push.
        ii. 1 means the meter will push on store.
    b. `TAG_SFTP_SERVICE_TYPE` has two values
        i. default of SFTP = 0
        ii. Dropbox = 1
    c. `TAG_SFTP_PATH`
        i. This value is the path on the remote location to where you wish the files pushed
        ii. This value must be begin with a backslash, \
        iii. If the location does not exist on the remote, then the meter will try to create the folder location. If the remote doesn't allow the creation of the location then the push will fail.
        iv. Default is empty and the meter will attempt to send the file to the base folder of your remote storage.
    d. `TAG_SFTP_EMAIL_CONTROL`
        i. Defaults to Never = 0
        ii. On Failure = 1 – if there is an error during a push the meter will send an email alerting those setup to receive email
        iii. Always = 2 – the meter will always send an email on success or failure
    e. `TAG_SFTP_MOVE_COPY`
        i. Defaults to Copy = 0 – the meter will copy the file to the remote storage
        ii. Move = 1 – the meter will delete the file after it has successfully pushed to the remote storage

   f. Additional properties are available to store true SFTP settings such as `TAG_SFTP_HOST_ADDRESS`, `TAG_SFTP_USER_NAME`, and `TAG_SFTP_PASSWORD` but their use is purely for storing and recalling settings and are not directly used by rclone on the meter.

8. After the above steps are complete and `TAG_SFTP_PUSH_CTRL` is set to 1 then the meter will send all newly created files to your specified remote storage.

Get and Set Properties with JSON

All meter types support /sdk?func=getproperties and /sdk?func=setproperties.

URL	Description
/sdk?func=getproperties&subset=meas	Gets, as a JSON object, all measurement properties and any system properties need to provide context for the measurement properties
/sdk?func=getproperties	Same as /sdk?func=getproperties&subset=meas
/sdk?func=getproperties&subset=sys	Gets, as a JSON object, all the system properties and any measurement properties needed to provide context for the system properties.
/sdk?func=setproperties	Accepts a JSON object with two possible sub-objects named “MeasProperties” and “SysProperties”. All properties to set can be included in a single POST to this end point.

Buildling up the JSON for Setting Properties

Refer to the list of supported properties for each meter type and ensure that you are setting each property to a supported data type. Supported types are strings, integers, floats, and, in some situations, arrays of floats or integers.

Some settings are read-only and will not be changed even if it is included in the JSON being sent to the meter.

Example JSON to enable Timer 3, adjust the backlight, and change the device description:

{ "MeasProperties": { "TAG_TIMER3_ENABLE": 1}, "SysProperties": {"TAG_BACKLIGHT":25, "TAG_DEVICE_DESCRIPTION": "North East Corner"}}

SoundAdvisor Get Data

This section describes how to obtain data from the SoundAdvisor Model 831C that is not available in other meters. There are two functions that provide access to data: getDataNew and getDataMulti. See **Appendix: New Tags for Data – SoundAdvisor Only** for details on each of the available TAGS.

The TAG descriptions are broken down as follows:

TAG_IO_ANYLEVEL_SPL - `Tag ID: 0x4C505341`
`Type: TypeFloatWithFlags, MinVal: 0, MaxVal: 140`

http://localhost:2508/sdk?func=getDataNew&group=live&tagid=0x4C505341&type=5

TAG_IO_ANYLEVEL_SPL is the name of the TAG
Tag ID: is the value sent in the query string as id (i.e., id=0x4C505341)
Type: is the expected type as an enum (type=5)
MinVal: Expected Minimum Value
MaxVal: Expected Maximum Value
nDx: index of TAG
Prec: is the precision to be displayed.

Another example:

http://localhost:2508/sdk?func=getdatanew&group=oneseclive&tagid=0x4B504C41&type=5

This retrieves the ANY LEVEL L$_{Peak}$ from the OneSecLive group as the following:

```
{ "Data":7.436529e+01,"Flags":0,"Result" : "Success: 0
","ResultCode":0,"ResultName":"Success" }
```

TAG_IO_ANYLEVEL_LPEAK - `Tag ID: 0x4B504C41`
`Type: TypeFloatWithFlags, MinVal: 0`

Another Example: ACZ LEQ

http://ld831c.pcb.com/sdk?func=GetDataNew&type=11&tagid=0x414C3031&group=overall

```
{
"value":[{"val":58.6724,"dt":0,"flags":0},{"val":64.9848,"dt":0,"flags"
:0},{"val":74.968,"dt":0,"flags":0}],"Result":"Success: 0
","ResultCode":0,"ResultName":"Success" }
```

getDataNew

This function requires:

1. Group: the group from which you expect the data
    a. Live
    b. Overall
    c. Current
    d. Measurement
    e. Events
    f. Time
2. Type: the enumerated value
    a. var TypeInt = 1;
    b. var TypeUInt = 2;
    c. var TypeFloat = 3;
    d. var TypeFloatSeries = 4;

   e. var TypeFloatWithFlags = 5;
   f. var TypeString = 6;
   g. var TypeByteSeries = 7;
   h. var TypeUIntSeries = 8;
   i. var TypeEnum = 9;
   j. var TypeUIntWithFlags = 10;
   k. var TypeFloatWithFlagsSeries = 11;
3. TagId: The ID of the TAG of which you wish to obtain the data.
   a. Ex: 0x4C505341 for TAG_IO_ANYLEVEL_SPL
4. Optional:
   a. nDx: is the index of the TAG; if not specified defaults to 0.

Here is an example of how to use getDataNew in JavaScript:

```
function getValue(strTag, group) {
     $.getJSON("/sdk?func=getDataNew&group=" + group + "&type=" + DataTags[str
Tag].Type + "&tagid=" + DataTags[strTag].ID)
     .done(function (data) {
        if (data["Data"] != undefined) {
           if (data["Data"] == -9999.0) {
              pageData[strTag] = "---";
           }
           else {
              if (DataTags[strTag].Prec != undefined) {
              pageData[strTag] = formD(data["Data"], DataTags[strTag].Prec);
              } else {
                 pageData[strTag] = data["Data"];
              }
           }
        } else {
           pageData[strTag] = 1.1;
        }
     });
}
```

getDataMulti

getDataMulti is what it sounds like, a way to get multiple TAGs simultaneously from the meter. This function has more requirements than its single counterpart above.

1. Group: all of the requested values must come from the same group
    a. These values are the same as above.
2. Tagids: is a comma separated list of the TAGS you wish to request (no spaces)
3. Types: another comma delimited list of types as defined above.
4. Indices: the indexes of the requested TAGS

Example of getDataMulti for TAG_IO_ANYLEVEL_SPL, TAG_IO_OBA_SERIES_LIVE_1_3:

http://localhost:2508/sdk?func=getdatamulti&group=live&tagids=0x4C505341,0x334C424F&types=5,4&indices=0,0

Returns:

```
{
      "Data": [{
            "tagId": 1280332609,
            "value": 5.055847e+01,
            "flags": 0
      }, {
            "tagId": 860635727,
            "value": [5.468252e+01, 5.886111e+01, 5.302473e+01,
5.510790e+01, 4.963264e+01, 6.432265e+01, 5.638278e+01, 4.917971e+01,
4.854479e+01, 4.381086e+01, 4.730509e+01, 4.058179e+01, 4.149162e+01,
3.710800e+01, 3.405349e+01, 3.426625e+01, 3.731847e+01, 4.112594e+01,
3.218463e+01, 3.212697e+01, 3.015873e+01, 2.913863e+01, 2.415304e+01,
2.376599e+01, 2.152117e+01, 2.423798e+01, 3.277596e+01, 2.649420e+01,
2.862195e+01, 2.770446e+01, 2.877575e+01, 3.034726e+01, 3.651043e+01,
3.360079e+01, 2.868397e+01, 2.834005e+01]
      }],
      "Result": "Success: 0 ",
      "ResultCode": 0,
      "ResultName": "Success"
}
```

Review the **Appendix: New Tags for Data – SoundAdvisor Only** for the details of each TAG.

get

The get function allows you to request a multi-parameter TAG that can return multiple data values. These TAGs are usually related to Time History. To see some examples of how this call is made see **Examples of Using the get Function**

/sdk?func=get&f=128&p1=207&vals=1093882946,22,0,18&format=tag:x4,sizeOfSeries0:i4,series0:af18,sizeOfSeries1:i4,series1:af18,sz:i4,series2:af18,flagtag:x4,flagsize:i4,thFlags:ai18,timetag:x4,timesz:i4,thTime:ai18,actiontag:x4,actionsz:i4,thAction:ai18

This specific request is made up of the following:

The function is the get function. The f=128 is required to get data. p1 is parameter 1 which we send our group, in this case GROUP_TIME_HISTORY (207). The vals parameter is broken up into 4 parts: the first is the Tag ID, second is frequency index (22 = 1kHz), the third is the start index, and the fourth is the count (in this case must be less than 120).

This will return the following, currently requesting the first 18 items:

tagID – in hex of 4 bytes
sizeOfSeries0 – int
series0 – array of floats with 18 items
sizeOfSeries1 – int
series1 – array of floats with 18 items
sizeOfSeries2 – int
series2 – array of floats with 18 items
flagtag – hex of 4 bytes
flagsize – int
thFlags – array of ints representing the data flags for each entry with 18 items
timetag – hex of 4 bytes
timesz – size of time series int
thTime – Time Array of ints (number of seconds from epoch) for each entry with 18 items
actiontag – hex of 4 bytes
actionsz – int for the size of actions
thAction – array of ints for the actions for each time history entry with 18 items

Retrieve File Data (831C) using getData and getDataMulti

In order to obtain data from files stored on the meter you can use the same commands as getData and getDataMulti if you load the file into the meters cached memory. Follow these step to get the Measurement data from a file. You can review the Error! Reference source not found. t o obtain Time History, Event History, etc., in a similar manner. See **New Tags for Data – SoundAdvisor Only** for more details about the available tags.

1. Open the file with the exact filename (fn=21012600.LD0.s in this case)
    a. /sdk?func=filecmd&cmd=7&fs=0&fn=21012600.LD0.s

Then you will make the same calls as you did before except with the File Group as defined in the Error! Reference source not found. section.

Measurement = 203

FileMeasurement = 205

2. Query number of measurement history records
    a. http://ld831c.pcb.com/sdk?func=getDataNew&Group=FileMeasurement&tagid=0x5443484D&type=2
3. Select the desired measurement history record that is equal to the number returned by the previous query minus 1.
    a. http://ld831c.pcb.com/sdk?func=setactiveindex&group=205&tag=0x524C484D&idx=n
4. Query LAeq, L10 and L90 using
    a. http://ld831c.pcb.com/sdk?func=getDataMulti&Group=FileMeasurement&tagids=0x414C3031,0x564C4E4C,0x564C4E4C&types=5,3,3&indices=0,1,5
5. Close the file when you are done.
    a. /sdk?func=filecmd&cmd=9

Web Socket

This feature allows you to receive updates to data as it changes. There are several function calls to make to setup the web socket and we suggest you have a strong understanding of web sockets before continuing.

There is a Sample application in the

You must create your WebSocket like the following JavaScript example:

```
var ws = new WebSocket("ws://" + location.host + "/ws");
```

Set Data Needed

Next set the data for which you wish to receive updates using a command *setdata*. Each call to *setdata* will overwrite previous calls to set the data.

```
ws.send(JSON.stringify(json));
```

Example asking for 4 TAGs:

```
{"Registering":[ {"g":"overall","i":0,"tags":[
        {"tg":1414681924,"tp":3,"n":"TAG_IO_DURATION"}]},
{"g":"oneseclive","i":0, "tags":[
        {"tg":860639315,"tp":5,"n":"TAG_IO_ANYLEVEL_SPL_3"},
        {"tg":1263553601,"tp":5,"n":"TAG_IO_ANYLEVEL_LPEAK"},
        {"tg":1280332609,"tp":5,"n":"TAG_IO_ANYLEVEL_SPL"}]}],
"command":"setdata"}
```

g: is for group and requires one of the group names be passed as its value:

live
current
overall
fileoverall
measurement
filemeasurement
session
filesession
events
fileevents
time
filetime
sysinfo
oneseclive

i: is for index; all TAGs must agree on the index requested as part of the group <optional>.

tags: is the array of objects for the group.

The objects are defined as follows:

tg: is for the Tag ID in decimal format

tp: is for type as defined in the **Types** section.

i: is for index of the individual TAG <optional>

n: is for the name of the data as returned in the JSON structure, can be any name you wish to use.

command: must be “setdata”

Set Data Rate

The command for setting the data rate is *dataon*. You only need to call this once unless you wish to change the rate.

Example to set the data rate to on at rate of 500ms:

```
{"rate":500,"command":"dataon"}
```

Suspend Data Updates

The command for setting the data rate is *dataoff*. Call the command *dataon* to resume data stream.

Example to set the data rate to off:

```
{"command":"dataoff"}
```

Status Rate

The command for setting the data rate is *statuson*. You only need to call this once unless you wish to change the rate.

Example to set the status data rate to 500ms period:

```
{"rate":500,"command":"statuson"}
```

The web socket will return status values that have changed. The first status will be a complete min status JSON object like the following:

```
{"uiRunFlags":2147483776,"uiLxTFlags":4027187204,"tv_sec":1494599308,"tv_usec":600000,"free_mem_kb":1822684,"total_mem_kb":1891188,"file_count":175,"uiMeterState":2,"indicators":33566748,"Force":0}
```

Suspend Status Updates

The command for setting the data rate is *statusoff*. Call the command *statuson* to resume data stream.

Example to set the status data rate to off:

```
{"command":"statusoff"}
```

Optional Functions

Audio Streaming

HLD now supports a simpler audio streaming call. Starting with HLD version 4.6.5 a single call to **<host:port>/audiostream** will return a continuous stream of bytes that are transferred using the HTTP Chunked sub protocol. This also supports up to 8 clients streaming simultaneously.

Using a standard HTTP client to make the request should keep the transaction simple, the request should be made in such a way as to read the headers immediately and not timeout. The headers need to be read before playback on the client can begin. To end the audio stream simply close the response stream.

The response headers contain several items, two of which are important for audio streaming.

Content-Type contains either “audio/ogg” or “audio/wav” since compression for this version of audio streaming is controlled by the meter’s measurement setting instead of a parameter on the client. The audio/ogg format will contain ogg/vorbis packets while the audio/wav format will contain raw PCM data, a WAV header will need to be generated if the Wave data is to be stored as a .wav file.

X-SampleRate contains the sample rate used to collect the audio on the meter coming from the meter’s settings. The value will be 8000, 16000, 24000, 48000, or 51200.

```
HttpWebRequest req = (HttpWebRequest) HttpWebRequest.Create(BaseURL +
                                                            "/audiostream");
using (WebResponse res = req.GetResponse())
{
        string format = res.Headers["Content-Type"];
        int sampleRateStr = int.Parse(res.Headers["X-SampleRate"]);

        using (Stream s = res.GetResponseStream())
        {
                …// Read the response stream here and use the bytes
        }
}
```

Browsers that support the Ogg format can use Audio elements with their source set to “/audiostream” if the html was served from the HLD’s host and the meter is set to stream in ogg. These elements would need to be destroyed or have their source cleared in order to end the stream.

Legacy Audio Streaming

This is still available and is the recommended method on INT-ET systems.

These functions are used to Stream Audio: **startAudioStream**, **getAudioStream** and **endAudioStream**. There are examples included in the SDK to show how to call them. The **getAudioStream** will return binary data that can be sent directly to a buffer for audio playback.

Audio Streaming is only compatible with SLM mode without Fast Time History. You should ensure that the setting for Time History period is set to 20ms or slower. If the period is faster (like 10ms, 5ms, or 2.5ms), then Audio Streaming will not work. Similarly, if you use Streaming Data, Time History period should not be 10ms or faster.

The SoundAdvisor Model 831C has the ability to compress the stream before it is sent. This reduces the overall data usage of the connection. Compression does create a lossy stream, so only use this form of audio streaming when you do not need to reprocess the audio data.

Streaming Data - SLM

These functions are used to Stream Data: **startDataStreaming**, **getStreamingData**, **stopDataStreaming**. The examples below show how to call them. The **getStreamingData** will return binary data that can be sent directly to a buffer based on the flags you set with the start call.

There are several samples on streaming data from the meter. Please review the code found in these samples to better understand how to use the stream functions.

Important Note:

You will need to ensure that you have the options installed for the data for which you wish to stream. If you send *flags* that represent data that is either not installed or that is masked, you will not receive any data for these *flags*. Further, if you only have *flags* that are masked or non-purchased options then you will not receive any return value when a request is made.

Note for Calculation:

The calculation for the maximum time of the buffer is 32 samples * length of sample time, for example, 80ms interval for the 32 samples at 2.5ms sample rate or 32 seconds with 1 second sample rate. If your system has any lag or delay in the request, samples will be lost if the request is made at 80ms intervals using the 2.5ms sample rate so an interval of 70ms might be more appropriate. The faster the system making the requests would, theoretically, be able to do so closer to the 80ms boundary while slower machines would need to reduce the interval. This further implies that an inferior system may need to be restricted from trying to sample at 2.5ms as the system may not be capable of making requests fast enough to keep the buffer from overrunning.

Example Stream Interval request times in milliseconds by Streaming Rate:

```
        var StreamInterval =
        {
                I20ms: 200,
                I50ms: 200,
                I100ms: 200,
                I200ms: 400,
                I500ms: 600,
                I1s: 1000,
                I2s: 2000,
                I5s: 5000,
                I10s: 10000,
                I15s: 15000,
                I20s: 20000,
                I30s: 30000,
                I1m: 60000,
                I2500us: 70,
                I5ms: 100,
                I10ms: 180
        }
```

You may review **StreamingData.html** in the included **SampleCode/Resources** folder for further details on how to execute and retrieve the data through streaming. There are other examples in C++ and C# included.

Important Note:
The Streaming data and Time History data do not sync to boundaries. The streaming data starts when the stream is initiated, while the Time History data starts when the run is processed. In the 831C, we made changes so that the Streaming data will "interval time sync" and line up with the Time History.

Start Data Streaming:

http://ipAddress[:port]/sdk?func=startDataStreaming&rate=<rate>&flags=<flags>

Example to start streaming ObaLeq13:
http://127.0.0.1:2508/sdk?func=startDataStreaming&rate=5&flags=131072

Example to start streaming LZeq and ObaSpl13:
http://127.0.0.1:2508/sdk?func=startDataStreaming&rate=5&flags=263168

Get Streaming Data:

http://ipAddress[:port]/sdk?func=getStreamingData

The actual date, float and integer data will be returned in the response as binary. Each value is 4 bytes of data. Most of the returned flags represent a single float, where others like **ObaLeq11** (32768=0x8000) returns 12 floats or **ObaLeq13** (131072=0x20000) returns 36 floats. The **Timestamp** and **TMS** are the only values that are returned as Integers all others are returned as floats. See the **StreamOption** enum below for the values and number of returned bytes. The first 4 bytes will always be the integer timestamp (in seconds from epoch) for each sample. You will need to calculate the size of your data request, and then use the response size to determine the number of samples to iterate over.

For example, if flags = `0x00000001` then the sample size should be 8 (4 from the timestamp and 4 byte float for the ALeq). Further, if the response size is 32 bytes then there are 4 samples to be processed with a timestamp and ALeq for each of the 4 samples.

The values for most of the data are in V^2, which means a conversion to dB may be required. The chart below describes which units are for which flags/data.

You can also use the streaming data to create a longer Leq average.

1. You can sum the values (V^2) for the rate you wish.
2. Then divide by the number of samples you have taken.
3. Convert to dB with the standard formula for converting V^2 to dB.

For example:

Select the rate of 1s for L_{Aeq} and sum the streamed data for one hour. Take the sum and divide by 3600 for number of samples taken in the hour to get the average V^2. Then 10 * Log_{10} (average) = L_{Aeq} in dB.

> **Important Note:** When using fast data streaming (2.5, 5 or 10ms), only `ObaLeq11 or ObaLeq13` may be selected for data, not both and only one other option is currently available - `TMS` which will provide nanoseconds of the current second resolution of the timestamp.
>
> **Important Note for LxT**: The Leq and Peak weightings are set in the settings on the meter and you will only receive a value for the configured weightings. If you wish ZPeak then you must set the LxT to use the Z weighting for Peak and use the correct flag when starting the streaming data. The same goes for A and C weightings.
>
> Further, the LxT does not support fast rates 13, 14, and 15 (2500us, 5ms, and 10ms respectively).

Stop Data Streaming:

http://ipAddress[:port]/sdk?func=stopDataStreaming

This will tell the meter to stop streaming. If Streaming is left on there will be increased power usage. It is advised to limit streaming (use 1s or slower) when using battery power.

Streaming Enumerations for aid in programming:

```
//[Rates]
public enum StreamHistoryPeriod
{
	P20ms                              = 0,
	P50ms                              = 1,
	P100ms                             = 2,
	P200ms                             = 3,
	P500ms                             = 4,
	P1s                                = 5,
	P2s                                = 6,
	P5s                                = 7,
	P10s                               = 8,
	P15s                               = 9,
	P20s                               = 10,
	P30s                               = 11,
	P1m                                = 12,
	P2500us                            = 13,
	P5ms                               = 14,
	P10ms                              = 15
}

[Flags]
public enum StreamOption : uint
{
	LAeq              = (0x00000001),   // 4 bytes ID_STH_ALEQ
```

```
        APeak              = (0x00000002),    // 4 bytes ID_STH_APEAK
        ASlowSpl           = (0x00000004),    // 4 bytes ID_STH_ASLOWSPL
        AFastSpl           = (0x00000008),    // 4 bytes ID_STH_AFASTSPL
        AImplSpl           = (0x00000010),    // 4 bytes ID_STH_AIMPLSPL
        LCeq               = (0x00000020),    // 4 bytes ID_STH_CLEQ
        CPeak              = (0x00000040),    // 4 bytes ID_STH_CPEAK
        CSlowSpl           = (0x00000080),    // 4 bytes ID_STH_CSLOWSPL
        CFastSpl           = (0x00000100),    // 4 bytes ID_STH_CFASTSPL
        CImplSpl           = (0x00000200),    // 4 bytes ID_STH_CIMPLSPL
        LZeq               = (0x00000400),    // 4 bytes ID_STH_ZLEQ
        ZPeak              = (0x00000800),    // 4 bytes ID_STH_ZPEAK
        ZSlowSpl           = (0x00001000),    // 4 bytes ID_STH_ZSLOWSPL
        ZFastSpl           = (0x00002000),    // 4 bytes ID_STH_ZFASTSPL
        ZImplSpl           = (0x00004000),    // 4 bytes ID_STH_ZIMPLSPL
        ObaLeq11           = (0x00008000),    // 12X4 bytes ID_STH_OBALEQ11
        ObaSpl11           = (0x00010000),    // 12X4 bytes ID_STH_OBASPL11
        ObaLeq13           = (0x00020000),    // 36X4 bytes ID_STH_OBALEQ13
        ObaSpl13           = (0x00040000),    // 36X4 bytes ID_STH_OBASPL13
        Flags              = (0x00080000),    // 4 bytes uint ID_STH_FLAGS
        WSlowSpl           = (0x00100000),    // 4 bytes ID_STH_WSLOWSPL
        WFastSpl           = (0x00200000),    // 4 bytes ID_STH_WFASTSPL
        WImplSpl           = (0x00400000),    // 4 bytes ID_STH_WIMPLSPL
        Tms                = (0x00800000),    // 4 bytes Integer ID_STH_TMS
        Wind               = (0x01000000),    // 2x4 bytes ID_STH_WEATHER_WIND
        TempHumidityPressure = (0x02000000),   // 3x4 bytes ID_STH_WEATHER_THP
        Precip             = (0x04000000),    // 6x4 bytes ID_STH_WEATHER_PRECIP
}
```

Unless otherwise specified in the Stream Option flags, 4 bytes represents a float in the streaming data flow, i.e., ***ObaLeq13*** has 36 floats.

Definition of Streaming Flags

Flag Name	Flag Value	Data Type	Order	Data units (SI)	Notes
Time	N/A	4 byte Int	0	Secs	Reports timestamp – This comes with every request and is not requested with a flag. For times faster than 1 sec. use the TMS flag to augment this timestamp to millisecond precision.
LAeq	0x01	4 byte float	1	V^2	Reports the LAeq
APeak	0x02	4 byte float	2	V^2	Reports the APeak
ASlowSpl	0x04	4 byte float	3	V^2	Reports the A weighted Slow Sound Pressure Level (SPL)
AFastSpl	0x08	4 byte float	4	V^2	Reports the A weighted Fast Sound Pressure Level (SPL)
AImplSpl	0x10	4 byte float	5	V^2	Reports the A weighted Impulse Sound Pressure Level (SPL)

LCeq	0x20	4 byte float	6	V^2	Reports the LCeq
CPeak	0x40	4 byte float	7	V^2	Reports the CPeak
CSlowSpl	0x80	4 byte float	8	V^2	Reports the C weighted Slow Sound Pressure Level(SPL)
CFastSpl	0x100	4 byte float	9	V^2	Reports the C weighted Fast Sound Pressure Level (SPL)
CImplSpl	0x200	4 byte float	10	V^2	Reports the C weighted Impulse Sound Pressure Level (SPL)
LZeq	0x400	4 byte float	11	V^2	Reports the LZeq
ZPeak	0x800	4 byte float	12	V^2	Reports the ZPeak
ZSlowSpl	0x1000	4 byte float	13	V^2	Reports the Z weighted Slow Sound Pressure Level (SPL)
ZFastSpl	0x2000	4 byte float	14	V^2	Reports the Z weighted Fast Sound Pressure Level (SPL)
ZImplSpl	0x4000	4 byte float	15	V^2	Reports the Z weighted Impulse Sound Pressure Level (SPL)
ObaLeq11	0x8000	12 4-byte floats	16-27	V^2	Reports the 12 Octave Band Leq's for frequencies in the Full Octave
ObaSpl11	0x10000	12 4-byte floats	28-39	V^2	Reports the 12 Octave Band SPL's for frequencies in the Full Octave
ObaLeq13	0x20000	36 4-byte floats	40-75	V^2	Reports the 36 Octave Band Leq's for frequencies in the Third Octave
ObaSpl13	0x40000	36 4-byte floats	76-111	V^2	Reports the 36 Octave Band SPL's for frequencies in the Third Octave
Flags	0x80000	4 byte Unsigned Int	112	Bits	Flags used to determine the quality of the OBA data
WSlowSpl	0x100000	4 byte float	113	V^2	Reports the W weighted Slow Sound Pressure Level (SPL)
WFastSpl	0x200000	4 byte float	114	V^2	Reports the W weighted Fast Sound Pressure Level (SPL)
WImplSpl	0x400000	4 byte float	115	V^2	Reports the W weighted Impulse Sound Pressure Level (SPL)
TMS	0x800000	4 byte Unsigned Int	116	Bits	Number of milliseconds (ms) since the last second as part of the time

Weather

All of the weather data will come through as 4 byte floats. The order follows the names described below. See the table below for streaming with the SEN03x. ***Wind*** will come in a pair of floats. ***TempHumidityPressure*** comes in a set of three floats. And ***Precip*** comes as a set of 6 floats. These are further described in the table below.

Flag	Description	Value	Data Type	Order	Data units (SI)	Notes
Wind	Wind Speed	0x1000000	4 byte float	1	m/s	Reports the average for the 1s period just prior to the end of the interval
	Wind Direction		4 byte float	2	Degrees	Reports the average for the 1s period just prior to the end of the interval
Temp Humidity Pressure	Temperature	0x2000000	4 byte float	1	Degrees Celsius	Reports the temperature just prior to the end of the interval
	Relative Humidity		4 byte float	3	Percent	Reports the relative humidity just prior to the end of the interval
	Barometric Pressure		4 byte float	2	hPa	Reports the barometric pressure just prior to the end of the interval
Precip	Rain total	0x4000000	4 byte float	1	mm	
	Rain rate		4 byte float	2	mm/hr	Reports the average over the last 5s just prior to the end of the interval
	Rain duration		4 byte float	3	Seconds	
	Hail total		4 byte float	4	mm	
	Hail rate		4 byte float	5	Hits/cm^2 hr	Reports the average over the last 5s just prior to the end of the interval
	Hail duration		4 byte float	6	Seconds	

Note about Precip – The SEN03x reports this information once every 5 seconds. If the streaming rate is more often than once every 5 seconds then the 831 will report the most recent data from the SEN030x until new data is available. If the streaming rate is greater than 5 seconds then the 831 will report the last 5 second data sample from the SEN03x.

Data Streaming in JSON – SLM

Just as with the other 831 Streaming Data functionality, you will need to call **startstreamingdata**. Next, instead of calling **getstreamingdata**, you will call **getjsonstream** to get the same data but formatted as JSON. Call **stopstreamingdata** to end the streaming data session. **startstreamingdata** requires two parameters, “rate” and “flags”; “rate” tells the meter the time period for each sample. Acceptable values are defined in the StreamHistoryPeriod enum above. “flags” tells the meter which metrics to collect for each sample. Each ‘bit’ in the flags value corresponds to a specific metric as defined in the StreamOptions enum above. In order to select multiple metrics, you will need to perform a bitwise 'or' with the desired values.

For example, to select LAeq, LAPeak, and 1/1 OBA Leq, set flags = (StreamOption.ALeq | StreamOption.APeak | StreamOption.ObaLeq11). The resulting flags value should be 0x00008003.

Examples

This:

http://<ipAddress>:<port#>/sdk?func=startstreamingdata&rate=4&flags=0x00018000

Tells the meter to collect 1/1 OBA Leq & SPL data every 500ms.

Or:

http://<ipAddress>:<port#>/sdk?func=startstreamingdata&rate=5&flags=0x07018000

Tells the meter to collect 1/1 OBA Leq & SPL and all the weather metrics (Wind, TempHumidityPressure, Precip) at a 1s sample period.

```
{ "Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

getJsonStream

This:

http://<ipAddress>:<port#>/sdk?func=getjsonstream

Returns a JSON object containing an array named "Data" that contains an element for every completed sample period collected since **startstreamingdata** or the last **getjsonstream** was called. Depending on the sample period selected and how often **getjsonstream** is called, there may either be multiple array entries (one for each sample) or an empty array. Also note that the meter will return a maximum of 32 sample periods per call to **getjsonstream**. If calling **getjsonstream** slower than the sample period x 32, only the most recent 32 samples will be returned.

The following JSON examples correspond to the expected output based on the **startstreamingdata** examples above.

```
{
"Data":
      [{
      "Timestamp": 1450182844,
      "oLEQ11": [56.51, 52.81, 42.64, 35.62, 39.05, 32.09, 29.36,
24.72, 16.18, 8.81, 8.26, 9.29],
      "oSPL11": [52.68, 53.47, 49.08, 39.06, 39.71, 31.98, 29.22,
25.69, 17.36, 11.28, 8.79, 9.45]
      }],
"Result": "Success: 0 ",
"ResultCode": 0,
"ResultName": "Success"
}
```

Or:

```
{
"Data":
      [{
      "Timestamp": 1457441173,
      "oLEQ11": [58.10825, 63.596405, 57.37233, 47.275627, 44.830612,
38.263531, 39.328156, 37.437599, 41.673195, 37.423504, 37.422607,
30.651896],
      "oSPL11": [58.50135, 62.569576, 56.52282, 47.952728, 44.759975,
38.977798, 41.549232, 43.490059, 34.943645, 31.883785, 32.795578,
27.871021],
      "wind": {
            "speed": 0,
            "dir": -99.900002
      },
      "temp": 30.900002,
      "rh": 55.900002,
      "pres": 30.000002,
      "rain": {
            "total": 0,
            "rate": 0,
            "dur": 0
      },
      "hail": {
            "total": 0,
            "rate": 0,
            "dur": 0
      }
      }]
}
```

This:

http://<ipAddress>:<port#>/sdk?func=stopstreamingdata

Tells the meter to stop storing intervals in its buffer so once this is called then no new data can be retrieved using **getjsonstream** until streaming is started again.

```
{ "Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }
```

Data Streaming – HVM200

Streaming data for HVM200 comes as JSON. See the JavaScript example below on how to access the data.

```
var liveDt;
$.getJSON('/sdk', {
                func: "streamdata",
                last: (stopped && !liveState) ? 1 : 0
        }).done(function (jData) {
                liveDt = jData["LiveData"][0];
});
```

It takes as parameters:

func: “streamdata”
Must be called as func = streamdata.

last: whether or not to get the last one. Used when stopped.

Example Data:

```
    liveDt contains the following:
    A1Sum: 0.00266669
    A1x: 0.000798377
    A1y: 0.000805933
    A1z: 0.00241336
    A2Sum: 0.00188564
    A2x: 0.000564538
    A2y: 0.000569881
    A2z: 0.0017065
    A4Sum: 0.00133335
    A4x: 0.000399189
    A4y: 0.000402967
    A4z: 0.00120668
    A8ActSum: 5677630
    A8Actx: 63342700
    A8Acty: 62160500
    A8Actz: 6932170
    A8ExpSum: 99.1
    A8Expx: 99.1
    A8Expy: 99.1
    A8Expz: 99.1
    A8Sum: 0.000942818
    A8x: 0.000282269
    A8y: 0.00028494
    A8z: 0.000853251
    AeqSum: 0.00296757
    Aeqx: 0.000888458
    Aeqy: 0.000896866
    Aeqz: 0.00268566
    AmaxSum: 0.00945947
    Amaxx: 0.00521155
    Amaxy: 0.00573585
    Amaxz: 0.00709492
    AminSum: 0.00180625
    Aminx: 0.000533368
    Aminy: 0.000508089
    Aminz: 0.00163985
    AmpSum: 0.0208816
    Ampx: 0.0133796
    Ampy: 0.0124297
    Ampz: 0.0204787
    ArawSum: 0.00335558
    BatLevel: 100
    BatState: 3
    Duration: 2907
    Mode: 1
    OverUnderBit: 0
    OverUnderLatchBit: 24
    PESum: 0.0000142225
    PeakSum: 0.00635998
    Peakx: 0.00368179
    Peaky: 0.00242384
    Peakz: 0.00627863
    RMSSum: 0.00321663
    RMSx: 0.00113326
    RMSy: 0.000979637
    RMSz: 0.00284653
    Rawx: 0.00109188
    Rawy: 0.00100359
    Rawz: 0.00301007
    RunStarted: 1614089752
    lastCount: 7161
    lastOvlCount: 20818359
    timeStamp: 1614092660
    uiRunFlags: 1
```

Command Line Arguments to Export Sound Files using G4

The following commands may be used to export Audio Files from an ldbin file using G4.

Command	Parameter(Optional)
"-exportAudio" or "-ea"	"all" or any combination of "v", "m", "e" (Voice, Measurement, Event)
"-inputFile" or "-if"	file name, include complete path or no path (partial path not allowed!)
"-destFolder" or "-df" (optional)	path for destination folder (will be created if doesn't exist)

G4 Options, Command Line Options Tab

- set pattern to define the output file name
- set folder where input file will be found if path is not specified in "-inputFile" or "-if"
- set output folder if "-destFolder" or "-df" are not present

Results log file will be written to: C:\ProgramData\PCB Piezotronics\G4\Logs\AudioExportLog.txt

Use:

G4.exe -exportAudio {all|m|v|e} –inputFile {[path and] filename} -destFolder {output directory}

Examples:
-exportAudio all -file "C:\Temp\#Projects\MSP Project\DST Spring 2017\Ldbin Files\01_20170311_030635__20170310-00.LD0.s.ldbin" -destFolder "C:\Temp\PCB Piezotronics\G4\SoundRecords\"

-exportAudio all -inputFile "C:\Temp\PCB Piezotronics\G4\LDbin\07_831A_All_SR_Types.ldbin" -destFolder "C:\Temp\PCB Piezotronics\G4\SoundRecords\"

-exportAudio all -inputFile "C:\Temp\PCB Piezotronics\G4\LDbin\07_831A_All_SR_Types.ldbin" -destFolder "C:\Temp\PCB Piezotronics\G4\SoundRecords\"

-exportAudio all -inputFile "C:\Temp\PCB Piezotronics\G4\LDbin\07_831A_All_SR_Types.ldbin" -df "C:\Temp\PCB Piezotronics\G4\SoundRecords\"

Using all the G4 options:
-ea all -if "07_831A_All_SR_Types.ldbin"

Encryption (Model 831-INT-ET)

We suggest that you use other methods of security, such as using your modems "Allowed IP List" or firewall. If you are using the meter on a LAN your router can be setup to restrict access. These are both beyond the scope of this document.

Encryption with an 831-INT-ET may be performed using port 443. The main function, Start Secure Session, prepares both HLD and your application to communicate if a password exists or is needed on the meter. Other elements are needed for the communication in order to keep it secure. SSL is the mode of secure communication.

startSecureSession

Description: Prepare the secure communication by calling this function and passing the password as a parameter.

Parameters: password or pw – the password that was previously set on the meter.

```
$.getJSON('/sdk', {
        func: "startSecureSession",
        password: myPassword
});
```

```
public string getSessionString(string protocol, string ip, int port, string
pass, string newLoc)
{
        string connString =
string.Format(@"{0}://{1}:{2}/sdk?func=startSecureSession&password={3}",
protocol, ip, port, pass);
        string script = @"var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
          if (xmlhttp.readyState == 4 &&
xmlhttp.status == 200) {
          }
        }
          xmlhttp.open('GET', '" + connString + @"',
true);
                xmlhttp.send();";
        return script;
}
```

NOTE: The certificate is not signed by a Certificate Authority (CA). The certificate may need to be added to your root Certificate Store to allow the connection to work correctly.

In order to start a secure session, you may do the following in a browser:

1. https://IpAddress:Port/sdk?func=startsecuresession&pw=password
    a. Example: https://10.3.122.79/sdk?func=startsecuresession&pw=A
    b. Where my address to the 831-INT-ET is 10.3.122.79 and my password is "A".
2. You may now make standard calls over https port 443.

a. You may need to map the port 443 on a Cell Modem to meet your needs.

You can test the connection in a browser. From there you can review the certificate that is used on the device. After sending the password in the browser navigate to the URL below to see if you have a connection, as it will return the Measurement Properties.

https://ipAddress:port/sdk?func=getdata&id=100

Appendix

A. Transfer Settings

```
/// <summary>
/// 'which' will determine which type of file to transfer:
/// 0 = 'Settings' File
/// 1 = 'System' Properties File
/// </summary>
/// <param name="sourceHLD"></param>
/// <param name="destinationHLD"></param>
/// <param name="propType">PropertyTypes 0 = 'Settings'; PropertyTypes 1 = 'System' Properti
es</param>
public void TransferPropertyFile(string sourceHLD, string destinationHLD, PropertyType propTyp
e, TransferDirection dir, CefSharp.Wpf.ChromiumWebBrowser browser, string model = "831", st
ring filename = null)
{
  Application.Current.MainWindow.Cursor = System.Windows.Input.Cursors.Wait;
  bool fromCache = false;
  string command = string.Empty;

  //check to see if this is from PC to meter
  if (dir == TransferDirection.ToMeter) //sourceHLD.Contains(":2508"))
  {
    RunRulesChanged = false;
    try
    {
      string theTag = "TAG_FILE_NAME";

      if (PropertyType.System == propType)
      {
        command = sourceHLD + "/sdk?func=downloadSysPropToCache&name=" + PropertiesObj
ects[(int)PropsIndex.PcSysProps].FileName;
        theTag = "TAG_MASK_OPTION";
      }
      else
      {
        command = sourceHLD + "/sdk?func=downloadSettingsToCache";
      }

      WebRequest request = WebRequest.Create(command);
      //Cause Properties to move to Cache
      request.Method = "GET";

      //WebResponse response = request.GetResponse();
      using (WebResponse response = request.GetResponse())
      {
        //check for success
        using (Stream dataStream = response.GetResponseStream())
        {
          long len = response.ContentLength;
          int offset = 0;
          byte[] data = new byte[len];

          while (offset < len)
          {
            offset += dataStream.Read(data, offset, (int)(len - offset));
          }

          //Run Rules on Cached Properties
          string dstring = Encoding.ASCII.GetString(data);
          if (dstring.Contains("Success") && ("TAG_FILE_NAME" != theTag || !model.StartsWith("
HVM")))
          {
            string script = "getAllPropertyData('" + theTag + "', wrapFunction(RunRules, this, ['" + (
propType == PropertyType.System ? "System" : "Measure") + "', '" + model + "', true]));"; //"dow
nloadPropertiesToCache(SelectedFile['nDx'], emptyFun, 'Measure');"

                //object obj = browser.EvaluateScript(script, TimeSpan.FromSeconds(15));
            object obj = browser.EvaluateScriptAsync(script, TimeSpan.FromSeconds(25));
            //script = "RunRules('Measure', '" + model + "');";
            //browser.EvaluateScriptAsync(script, TimeSpan.FromSeconds(15));
            Thread.Sleep(1000);
            //display message that changes were made if necessary
            if (RunRulesChanged)
            {
              if (MsgBox.Show("Changes are needed to allow these properties to be transfered to
your meter. Continue?", "Changes", MessageBoxButton.OKCancel, MessageBoxImage.Informati
on) == MessageBoxResult.Cancel)
                return;
            }
            //Change the download to use the cache instead.
            fromCache = true;
          }
          else
          {
            //Don't use cache
            fromCache = false;
          }
        }
      }
    }
    catch (Exception ex)
    {
      Debug.WriteLine(ex.Message);
    }
  }

  //------------------------------------------
  // Get the file from the Source HLD
  //------------------------------------------
  try
  {
    // Create a request using a URL that can receive a post.
    if (PropertyType.System == propType)
    {
      command = sourceHLD + "/sdk?func=downloadSysPropFile&fromCache=" + fromCache.ToS
tring();
    }
    else
    {
      command = sourceHLD + "/sdk?func=downloadSettingsFile&fromCache=" + fromCache.ToS
tring();
    }

    WebRequest request = WebRequest.Create(command);

    // Set the Method property of the request to GET.
    request.Method = "GET";
    if (dir == TransferDirection.ToPc)
    {
      if (!string.IsNullOrEmpty(activeMeter.meterModel.SessionCookie))
      {
        request.Headers.Add("Cookie", activeMeter.meterModel.SessionCookie);
      }
    }

    // Get the response.
    using (WebResponse response = request.GetResponse())
    {
      // Get the stream containing content returned by the server.
      using (Stream dataStream = response.GetResponseStream())
      {
        int offset = 0;
        int start = 0;
        long len = response.ContentLength;
        byte[] data = new byte[len];

        while (offset < len)
        {
          offset += dataStream.Read(data, offset, (int)(len - offset));
```

```
}

if (PropertyType.System == propType)
{
  command = destinationHLD + "/setData.js?uploadSysPropFile=1";
}
else
{
  command = destinationHLD + "/setData.js?uploadSettingsFile=1";
}

if (!string.IsNullOrWhiteSpace(filename))
{
  command += "&name=" + filename;
}

HttpWebRequest post = (HttpWebRequest)WebRequest.Create(command);
post.Method = "POST";
post.KeepAlive = true;
post.ContentType = "application/octet-stream; charset=ISO-8859-1";
post.ContentLength = len;
post.Headers.Set("cache-control", "no-cache");
if (dir == TransferDirection.ToMeter)
{
  if (!string.IsNullOrEmpty(activeMeter.meterModel.SessionCookie))
  {
    post.Headers.Add("Cookie", activeMeter.meterModel.SessionCookie);
  }
}

using (Stream reqStream = post.GetRequestStream())
{
  reqStream.Write(data, start, (int)len);

  using (WebResponse postResponse = post.GetResponse())
  {
    using (Stream resStream = postResponse.GetResponseStream())
    {
      using (StreamReader sr = new StreamReader(resStream))
      {
        string postResult = sr.ReadToEnd();
        try
        {
          JObject json = JObject.Parse(postResult);

          if (((int?)json["ResultCode"]) == (int)LDError.IoErrorModified)
          {
```

```
                    MsgBox.Show(LDCommon.Properties.Resources.GIDS_SettingResetToDefaultVa
lue);
                  }
                }
                catch
                {
                }
              }
            }
          }
        }
      }
    }

    if (fromCache)
    {
      string script = "clearCache();";
      browser.EvaluateScriptAsync(script, TimeSpan.FromSeconds(25));
    }
  }
  catch (Exception ex)
  {
    MsgBox.Show(ex.Message);
  }
  finally
  {
    Application.Current.MainWindow.Cursor = System.Windows.Input.Cursors.Arrow;
  }
}

public bool UploadFile(string address, byte[] byteArray)
{
  int loc = address.IndexOf(":", 6) > 0 ? address.IndexOf(":", 6) : address.Length;
  string ipadd = address.Substring(7, loc - 7);
  int port = Port;
  port = int.Parse(address.Substring(loc + 1));

  return Utility.UploadFile(ipadd, port, byteArray);
}

public bool SavePropertyFile(SlmRecord rec, PropertyType propType, string hld, string filename)
{
  bool result = false;
  int siz = 0;
  byte[] data = null;
  string response = String.Empty;
  switch (propType)
  {
    case PropertyType.Measurement:
      {
        BareMeasPropFile_t propFile = new BareMeasPropFile_t();

        propFile.PropHeader.BlockHead.tag = (int)BlockTag.Settings;
        propFile.PropHeader.BlockHead.length = (uint)(Marshal.SizeOf(propFile.PropHeader) + M
arshal.SizeOf(propFile.MeasProps));
        propFile.PropHeader.BlockHead.records = 1;
        propFile.PropHeader.name = Path.GetFileName(filename);
        //propFile.PropHeader.name = Path.GetFileNameWithoutExtension(filename);  // this stri
ps the mode extension and all setups get saved as SLM
        if (propFile.PropHeader.name.Length > 10) propFile.PropHeader.name = propFile.PropHe
ader.name.Substring(0, 10);
        propFile.MeasProps = rec.MeasProperties;

        siz = Marshal.SizeOf(propFile);
        data = new byte[siz];

        data = SlmConvert.StructToByteArray<BareMeasPropFile_t>(propFile, (int)ByteOffset.Zer
o);
        response = "POST /setData.js?uploadSettingsFile=1 HTTP/1.0\r\n";
      }
      break;

    case PropertyType.System:
      {
        BareSystemPreferencesFile_t propFile = new BareSystemPreferencesFile_t();

        propFile.PropHeader.BlockHead.tag = (int)BlockTag.Prefer;
        propFile.PropHeader.BlockHead.length = (uint)(Marshal.SizeOf(propFile.PropHeader) + M
arshal.SizeOf(propFile.SysPrefs));
        propFile.PropHeader.BlockHead.records = 1;
        propFile.PropHeader.name = Path.GetFileName(filename);
        //if (propFile.PropHeader.name.Length > 10) propFile.PropHeader.name = propFile.PropH
eader.name.Substring(0, 10);
        propFile.SysPrefs.userPropSiz = Marshal.SizeOf(propFile.SysPrefs.UserProps);
        propFile.SysPrefs.sysPropSize = Marshal.SizeOf(propFile.SysPrefs.SysProps);
        propFile.SysPrefs.UserProps = rec.UserProperties;
        propFile.SysPrefs.SysProps = rec.SysProperties;

        siz = Marshal.SizeOf(propFile);
        data = new byte[siz];

        data = SlmConvert.StructToByteArray<BareSystemPreferencesFile_t>(propFile, (int)ByteO
ffset.Zero);
```

```
        response = "POST /setData.js?uploadSysPropFile=1 HTTP/1.0\r\n";
      }
      break;

    default:
      break;
  }

  if (null != data)
  {
    response += String.Concat("Content-Length: ", data.Length.ToString(), "\r\n");
    response += String.Concat("Content-Type: application/octet-stream; charset=ISO-8859-
1", "\r\n");
    response += String.Concat("cache-control: no-cache", "\r\n");
    response += String.Concat("Connection: Keep-alive", "\r\n\r\n");

    List<byte> content = new List<byte>(response.Length + data.Length);

    content.InsertRange(0, Encoding.ASCII.GetBytes(response));
    content.InsertRange(content.Count, data);

    result = UploadFile(hld, content.ToArray());
  }

  return result;
}
```

B. Update Properties

This section has code examples in JavaScript for changing Properties in the meter.

```
function UpdateProperties(whichProps) {
     try {
         if (inFileDsp) {
             return $.when({});
         }
     } catch (e) { }
     var index = 0;
     var properties, propTags;
     if (whichProps == "System") {
         properties = SystemProperties;
         propTags = SystemTags;
     } else {
         whichProps = "Measure";
         properties = MeasurementProperties;
         propTags = PropertyTags;
         clearThFlags();
     }
     index++;
     posterChild = {};
     var itemList = {};
     var hasData = false;
     for (var keyItem in properties) {
         if (properties.hasOwnProperty(keyItem) && keyItem !== "TAG_MODEL"
             && propTags[keyItem] != undefined && !propTags[keyItem].readOnly
             ) {

     if ((!(OnSystemTabs || IsHvm())) && keyItem === "TAG_MASK_OPTION" || keyI
tem === "TAG_OPTION_FLAGS") continue;
             LatchWhich[whichProps] = 1;

     console.log("G4~ Sending: " + keyItem + ": " + properties[keyItem]);

     if (keyItem !== "" && (typeof propTags[keyItem].ID !== "undefined")) {
         var keyPairs = {};
         if (propTags[keyItem].Type === TypeFloatSeries) {
             keyPairs["Value"] = properties[keyItem];
         } else if (propTags[keyItem].Type === TypeFloat) {
         keyPairs["Value"] = minMaxVal(parseFloat(properties[keyItem]), propTag
s[keyItem]);
         } else if (propTags[keyItem].Type === TypeUInt) {
         keyPairs["Value"] = minMaxVal(parseInt(properties[keyItem], 10) >>> 0,
 propTags[keyItem]);
         } else if (propTags[keyItem].Type === TypeInt || propTags[keyItem].Typ
e === TypeEnum) {
         keyPairs["Value"] = minMaxVal(parseInt(properties[keyItem], 10), propT
ags[keyItem]);
     } else {
         keyPairs["Value"] = properties[keyItem];
     }
     if (propTags[keyItem].Type === TypeFloatSeries) {
     //need to get the tag and all of its siblings.
     var shortTag = keyItem.substr(0, keyItem.length - 2);
     if (typeof propTags[keyItem].num !== "undefined") {
        var arr = [];
     //we must send the entire table
     for (var i = 0, ii = propTags[keyItem].num; i < ii; i++) {
        var subTag = shortTag + "_" + i;
        arr[i] = minMaxVal(parseFloat(properties[subTag]), propTags[keyItem]);
     }
               keyPairs["Value"] = arr;
               }
            }
               itemList[keyItem] = keyPairs["Value"];
               hasData = true;
            }
        }
     }

     //check to see if we have anything to send.
    if (hasData) {
        CommitCompleteCnt++;
        var struct = {};
        if (whichProps === "System") {
           struct["SysProperties"] = itemList;
        } else {
           struct["MeasProperties"] = itemList;
        }
        var str = JSON.stringify(struct);
        console.log(str);
        posterChild = $.post("/sdk?func=setProperties", str)
        .done(function (data) {
           console.log("CommitSettings: " + data.responseText);
        })
        .fail(function (data) {
           console.log("error Sending Properties");
        })
        .always(function (data) {
           testData = data;
           console.log("Settings Finale");
           CommitCompleteCnt--;
        });
        return posterChild;
     }
     return $.when({});
}
var posterChild = {};
```

After Setting Properties you must call the appropriate LatchSettings along with the markSettingsChanged to ensure that the meter updates its view and internal processes.

```
var LatchWhich = new Object();
//You must set LatchWhich before calling this function.
//Param: noReld (optional) value of true specifies to NOT reload. noReld == un
defined or false -> ReloadTab.
```

```
function cmdLatchSettings(noReld, cb) {
     var func;
     //LatchWhich is used to determine which Properties to Latch.  Only Latch
Properties that have changed.  See UpdateProperties() above.
     var states = [];
     if (resetFilter) {

     $.getJSON('/sdk?func=sendCommand&type=1&data=1&id=' + CMD_RESET).done(fun
ction () { console.log("FCFR") });
         resetFilter = false;
     }
     if (LatchWhich["Measure"] == 1) {
         CommitCompleteCnt = 1;

     states.push($.getJSON('/sdk?func=LATCHSETTINGS&data=2&marksettingschanged
=true')
         .done(function (dataVal) {
             errorHandling("cmdLatchSettings Settings", dataVal);
         })
         .always(function () {
             var cnt = 0;

     states.forEach(function (item) { if (item.state() !== "pending") cnt++; }
);
             if (cnt + 1 >= states.length) {
                 if (func != undefined) clearTimeout(func);
                 func = clearCache(cb);
             }
             CommitCompleteCnt = 0;
         }));
     }
     if (LatchWhich["System"] == 1) {
         CommitCompleteCnt++;

     states.push($.getJSON('/sdk?func=LATCHSETTINGS&data=0&marksettingschanged
=false')
         .done(function (dataVal) {
             CommitCompleteCnt++;
             errorHandling("cmdLatchSettings Sys", dataVal);

     return $.getJSON('/sdk?func=LATCHSETTINGS&data=1&marksettingschanged=true
')
             .done(function (dataVal) {
                 errorHandling("cmdLatchSettings Prefs", dataVal);
             }).always(function () {
                 CommitCompleteCnt = 0;
             });
         }).always(function () {
             var cnt = 0;

     states.forEach(function (item) { if (item.state() != "pending") cnt++; })
;
             if (cnt + 1 >= states.length) {
                 console.log("G4~ SaveComplete");
```

```
               if (func != undefined) clearTimeout(func);
               func = clearCache(cb);
            }
            if (noReld === undefined || noReld === false) {
               ReloadTab();
            }
         }));
     }
     LatchWhich = new Object();
}

function MarkSettingsChanged() {
     $.getJSON('/sdk?func=MarkSettingsChanged')
     .done(function (dataVal) {
        errorHandling("MarkSettingsChanged", dataVal);
     });
}
```

C. Measurement Property Descriptions 831, LxT and 831C

Name: ADC1
Description: ADC1 description string.
Type: string
Tag Name: TAG_ADC1_DESCRIPTION
Tag ID: 0x41443134
Allowed Values: Max length: 255
Special Handling: RESET_REQ.

Name: ADC1
Description: ADC1 offset.
Type: float
Tag Name: TAG_ADC1_OFFSET
Tag ID: 0x41443132
Allowed Values: Min: -9999.99, Max: 9999.99
Special Handling: RESET_REQ.

Name: ADC1
Description: ADC1 scale.
Type: float
Tag Name: TAG_ADC1_SCALE
Tag ID: 0x41443131
Allowed Values: Min: -9999.99, Max: 9999.99
Special Handling: RESET_REQ.

Name: ADC1
Description: ADC1 units string.
Type: string
Tag Name: TAG_ADC1_UNITS
Tag ID: 0x41443133
Allowed Values: Max length: 255
Special Handling: RESET_REQ. May contain special characters like the degree symbol which may not be parsable by the JSON parser. In this case use a search and replace on the returned string before parsing with JSON. By default this field contains a degree symbol.

Name: ADC2
Description: ADC2 description string.
Type: string
Tag Name: TAG_ADC2_DESCRIPTION
Tag ID: 0x41443234
Allowed Values: Max length: 255
Special Handling: RESET_REQ.

Name: ADC2
Description: ADC2 offset.
Type: float
Tag Name: TAG_ADC2_OFFSET
Tag ID: 0x41443232
Allowed Values: Min: -9999.99, Max: 9999.99
Special Handling: RESET_REQ.

Name: ADC2
Description: ADC2 scale.
Type: float
Tag Name: TAG_ADC2_SCALE
Tag ID: 0x41443231
Allowed Values: Min: -9999.99, Max: 9999.99
Special Handling: RESET_REQ.

Name: ADC2
Description: ADC2 units string.
Type: string
Tag Name: TAG_ADC2_UNITS
Tag ID: 0x41443233
Allowed Values: Max length: 255
Special Handling: RESET_REQ.

Name: AutoCalCheck
Description: Enable automatic daily cal check. Must have Environmental Preamp (PRM2103 or 426A12) attached and RUN_MODE must be set to continuous and the meter must be running. See the Command **Sending Commands** to 730, 721 and 821 (Including SE)

The request to send a command is "http://ipAddress:port/sdk?func=cmd&op=send&message={message}.

In the above example message should contain one of the following supported messages, the response returned is dependent on the meter's current state. Some commands require an additional suffix in the form of "*nnn"

Measurement Commands

Code	Description
M1	Starts a measurement.
M2	Stops the current measurement. On the 730 this will always store and put the meter in a reset state. On the X21 this will simply end the current measturement but future measurements will continue to be stored in the same file until the Store command (M11) is sent.
M3	Pause the current measurement.

M10	Discards the current measurement file. Does not apply to the 730. The meter must be in a stopped state.
M11	Stores the current measurement file. Does not apply to the 730. The meter must be in a stopped state.
D10	Gets the Time History graph data.
D11	Gets the number of Time History records.
D9*125	Deletes all data files currently stored on the meter.

Calibration Commands

Code	Description
M4	Triggers a manual calibration
R90	Read the calibration delta
R91	Read the calibration level
R92	Read the calibration history

System Commands

Code	Description
M5*130	Initiates the meter shutdown.
M8*133	Initiates the meter reboot.

Calibration and Cal Check sub-section to perform Cal Check manually.
Type: int
Tag Name: TAG_DAILY_CAL_CHECK
Tag ID: 0x524D3230
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: AutoCalCheckTime
Description: Set the time of day in seconds for the automatic daily cal check.
Type: int
Tag Name: TAG_CAL_CHECK_TIME
Tag ID: 0x524D3231
Allowed Values: 0 - 86399
Special Handling: RESET_REQ.

Name: ClockSync
Description: Enable Intervals to sync with the hour. See the Model 831 Manual – Continuous and Timer Modes – Interval Time Sync. The interval time sync feature ensures that all measurement records, except the first, will begin at a time of day equal to a multiple of the measurement time selected.
Type: int
Tag Name: TAG_INTERVAL_CLOCK_SYNC
Tag ID: 0x524D3039
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: CurrentFileIndex
Description: Set the file index for the specified directory.
Type: int
Tag Name: TAG_CURRENT_FILE_INDEX
Tag ID: 0x464E3032
Allowed Values: Min: 0, Max: 2147483647 (2^31-1)
Special Handling: None.

Name: DailyAutoStore
Description: Enable automatic storing of data at set intervals.
Type: int
Tag Name: TAG_DAILY_STORE
Tag ID: 0x524D3038
Allowed Values: 0 = Off/Never,
1 = 1/day (24 hours),
2 = 2/day (12 hours),
3 = 4/day (6 hours),
4 = 6/day (4 hours),
5 = 8/day (3 hours),
6 = 12/day (2 hours),
7 = 24/day (1 hour)
8 = 48/day (30 min)
9 = 96/day (15 min)
10 = 144/day (10 min)
11 = 288/day (5 min)
12 = 720/day (2 min)
13 = 1440/day (1 min)

Special Handling: RESET_REQ.

Name: AutoStoreTime
Description: Time offset from midnight, in seconds, to perform a daily auto store.

Type: int
Tag Name: TAG_DAILY_STORE_TIME
Tag ID: 0x524D3139
Allowed Values: Min: 0 (midnight), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: DayTime
Description: Time offset from midnight, in seconds, for LDN and LDEN start of day calculations.
Type: int
Tag Name: TAG_DAY_TIME
Tag ID: 0x444E3031
Allowed Values: Min: 0 (midnight), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: Detector Weighting
Description: Sets the Slm detector.
Type: int
Tag Name: TAG_DET_WEIGHING (misspelling is correct)
Tag ID: 0x44573031
Allowed Values: 0 = Impulse, 1 = Fast, 2 = Slow
Special Handling: RESET_REQ.

Name: DoseName
Description: Name of the Dose settings (i.e. OSHA-1)
Type: string
Tag Name: TAG_DOSE_NAME
Tag ID: 0x44533036
Allowed Values: Max length: 8.
Special Handling: None.

Name: DoseCriterionLevel
Description: Dose criteria level.
Type: float
Tag Name: TAG_DOSE_C_LEVEL
Tag ID: 0x44533034
Allowed Values: Min: 20.0, Max: 200.0
Special Handling: None.

Name: DoseCriterionTime
Description: Dose criteria time.
Type: float
Tag Name: TAG_DOSE_C_TIME
Tag ID: 0x44533035

Allowed Values: Min: 0.1, Max: 24.0
Special Handling: None.

Name: DoseExchangeRate
Description: Dose exchange rate.
Type: int
Tag Name: TAG_DOSE_EXCHANGE_RATE
Tag ID: 0x44533033
Allowed Values: 0 = 3dB,
1 = 4dB,
2 = 5dB,
3 = 6dB
Special Handling: RESET_REQ.

Name: DoseThreshold
Description: Dose threshold level.
Type: float
Tag Name: TAG_DOSE_THRESHOLD
Tag ID: 0x44533032
Allowed Values: Min: 20.0, Max: 200.0
Special Handling: RESET_REQ.

Name: DoseThresholdEnable
Description: Enable dose threshold.
Type: int
Tag Name: TAG_DOSE_THRESHOLD_ENABLE
Tag ID: 0x44533037
Allowed Values: 0 = Off, 1 = On
Special Handling: None.

Name: DynamicTriggerOffsetLevel
Description: Dynamic response trigger level.
Type: float
Tag Name: TAG_EVENT_DYN_TRIG_OFFSET
Tag ID: 0x45563037
Allowed Values: Min: 3.0, Max: 99.0
Special Handling: RESET_REQ.

Name: DynamicResponse
Description: Dynamic response.
Type: int
Tag Name: TAG_EVENT_DYNAMIC_RESPONSE
Tag ID: 0x45563038

Allowed Values: Min: 1 (slowest), Max: 5 (fastest)
Special Handling: RESET_REQ.

Name: EveningPenalty
Description: Penality added to Leq during the evening time period for LDN and LDEN calculations.
Type: float
Tag Name: TAG_EVENING_PENALTY
Tag ID: 0x444E3034**Allowed Values:** Min: 0.0, Max: 99.9
Special Handling: RESET_REQ.

Name: EveningTime
Description: Time offset from midnight, in seconds, for LDN and LDEN start of evening calculations.
Type: int
Tag Name: TAG_EVENING_TIME
Tag ID: 0x444E3032
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: Event ContinuationPeriod
Description: Amount of time, in seconds, that must be exceeded before next event is recorded.
Type: int
Tag Name: TAG_EVENT_CONT_PERIOD
Tag ID: 0x45563036
Allowed Values: Min: 0, Max: 9 – for the 831A
831C – Min: 0.1, Max: 999 and is based on whether Event Time History is enabled or not and the value of the Event Time History Post Trigger value. Max: 199 when Post Trigger is 0. Subtract Postrigger from the max of 199.
Special Handling: RESET_REQ.

Name: Event Email Notification Only available on 831C
Description: If 1 then event notifications will be emailed to the configuration in System Properties. 0 if you do not wish to send notifications. 2,3 are reserved.
Type: unsigned int
Tag Name: TAG_EVENT_NOTIFICATION
Tag ID: 0x45563230
Allowed Values: Min: 0, Max: 16 – only available on the **831C**
Special Handling: RESET_REQ.

Name: EventLevel
Description: Event trigger levels. The GroupId specifies which trigger is being set.
Tag Name: TAG_EVENT_LEVEL

Tag ID: 0x45563032

Type: float

Allowed Values: Min: 0.0, Max: 200.0

Special Handling: RESET_REQ.

Name: Event MinimumDuration
Description: Minimum exceedence duration, in seconds, reguired to trigger an event.
Type: float

Tag Name: TAG_EVENT_MIN_DURATION

Tag ID: 0x45563035

Allowed Values: Min: 0.1, Max: 9.9 – For 831A

831C – Min: 0.1, Max: 60.0. This is also dependent on the Sound recording settings. The snapshot time can be as high as 999. If Event History is disabled then the Min Duration is zero in this equation:

Sample Rate	Minimum Duration + Pre-Trigger ≤
8 ksps	60 s
16 ksps	30 s
24 ksps	20 s
48 ksps	10 s
51.2 ksps	9.375 s

Special Handling: RESET_REQ.

Name: ExceedenceTimeHistoryEnable

Description: Enable storing of event time histories.

Type: int

Tag Name: TAG_EVENT_TH_ENABLE

Tag ID: 0x45563134

Allowed Values: 0 = Off, 1 = On

Special Handling: RESET_REQ.

Name: EventTHCount

Description: Number of event time history samples to record.

Type: int

Tag Name: TAG_EVENT_TH_MAX_SAMPLES

Tag ID: 0x45563137

Allowed Values: Min: 10, Max: 9999

Special Handling: RESET_REQ.

Name: EventTHPostCount

Description: Number of post event time history samples to record.

Type: int

Tag Name: TAG_EVENT_TH_POST_TRIG
Tag ID: 0x45563139
Allowed Values: Min: 0, Max: 99 default is 10.
This value will cause changes to the Continue Duration max value.
Special Handling: RESET_REQ.

Name: EventTHPreCount
Description: Number of pre event time history samples to record.
Type: int
Tag Name: TAG_EVENT_TH_PRE_TRIG
Tag ID: 0x45563138
Allowed Values: Min: 0, Max: 99
Special Handling: RESET_REQ.

Name: ExceedAudioSnapshotPeriod
Description: The duration, in seconds, to record an exceedence audio snapshot.
Type: int
Tag Name: TAG_EVENT_SNAPSHOT_PERIOD
Tag ID: 0x45563131
Allowed Values: Min: 1, Max: 999
Special Handling: RESET_REQ.

Name: ExceedAudioSnapshotPreTriggerPeriod
Description: For Sound Recordings, this is the duration, in seconds, of the pre-trigger audio to save.
Type: int
Tag Name: TAG_EVENT_PRETRIGGER_PERIOD
Tag ID: 0x45563132
Allowed Values: Min: 1, Max: 999
Special Handling: RESET_REQ.

Name: ExceedenceHistoryEnable
Description: Enable storing of event histories.
Type: int
Tag Name: TAG_EVENT_HISTORY_ENABLE
Tag ID: 0x45563133
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: Event SaveExceedAudioSnapshot
Description: Enable storing a sound recording snapshot for triggered events.
Type: int
Tag Name: TAG_EVENT_SNAPSHOT_ENABLE
Tag ID: 0x45563135

Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: ExceedenceSpectralTimeHistoryEnable
Description: Enable storing of time history spectra for an event.
Type: int
Tag Name: TAG_EVENT_TH_SPECTRA_ENABLE
Tag ID: 0x45563136
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: ExceedenceTimeHistoryPeriod
Description: Enable automatic storing of data at set intervals.
Type: int
Tag Name: TAG_EVENT_TH_PERIOD
Tag ID: 0x45563130
Allowed Values: 0 = 20ms, 1 = 50ms, 2 = 100ms, 3 = 200ms, 4 = 500ms, 5 = 1 second, 6 = 2 second, 7 = 5 second, 8 = 10 second
Special Handling: RESET_REQ.

Name: FFT_WindowType
Description: Set window type for FFT operations.
Type: int
Tag Name: TAG_FFT_WINDOW_TYPE
Tag ID: 0x46465431
Allowed Values: 0 = Hanning 1 = Flat Top 2 = Rectangle
Special Handling: FILTER_RESET_REQ

Name: FFT_FreqSpan
Description: Set frequency span for FFT operations.
Type: int
Tag Name: TAG_FFT_FREQSPAN
Tag ID: 0x46465432
Allowed Values: 0 = 20,000 Hz 1 = 10,000 Hz 2 = 5,000 Hz 3 = 2,000 Hz 4 = 1,000 Hz 5 = 500 Hz 6 = 200 Hz 7 = 100 Hz
Special Handling: FILTER_RESET_REQ

Name: FFT_NumLines
Description: Set number of lines (resolution) for FFT operations.
Type: int
Tag Name: TAG_FFT_NUM_LINES
Tag ID: 0x46465433

Allowed Values: 0 = 6400   1 = 3200   2 = 1600   3 = 800   4 = 400
Special Handling: FILTER_RESET_REQ

Name: File Name
Description: The default name to use when storing a data file.
Type: string
Tag Name: TAG_FILE_NAME
Tag ID: 0x464E3031
Allowed Values: Max length: 8
Special Handling: None.

Name: InstrumentMode
Description: Sets the operating mode of the instrument.
Type: int
Tag Name: TAG_INSTRUMENT_MODE
Tag ID: 0x494D3031
Allowed Values: 0 = SLM,   1 = RA,   2 = FFT
Special Handling: FILETER_RESET_REQ.

Name: IntegrationInput
Description: Sets the integration method.
Type: int
Tag Name: TAG_INTEGRATION_INPUT
Tag ID: 0x44573032
Allowed Values: 0 = Linear, 1 = Exponential
Special Handling: RESET_REQ.

Name: IntervalTime
Description: The time, in seconds, of each interval measurement. The Interval Time (TAG_INTERVAL_TIME) is used in conjunction with Measurements. If you have chosen the Run Mode (TAG_RUN_MODE) as Continuous, Single Block Timer or Daily Timer, then the Interval Time tells the meter how often to close a Measurement Record and start a new Measurement Record (this is different than storing). If you are not using one of these Run Modes then the Interval Time has no bearing on the rest of the meter.
Type: int
Tag Name: TAG_INTERVAL_TIME
Tag ID: 0x524D3037
Allowed Values: Min: 60 = 1 Minute; Max: 86340 = 23:59 hh:mm
Special Handling: RESET_REQ.

Name: MarkerAudioRecord
Description: Enables sound recordings when marker is placed.
Type: int

Tag Name: TAG_MARKER_AUDIO_RECORD
Tag ID: 0x4D4B3032
Index: array index of desired marker 0-9
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: MarkerNames
Description: Name of marker. An array of up to ten markers may be defined.
Type: string
Tag Name: TAG_MARKER_NAMES
Tag ID: 0x4D4B3031
Index: array index of desired marker 0-9
Allowed Values: Max length: 32.
Special Handling: None.

Name: MarkerPreRecordPeriod
Description: The duration, in seconds, of the pre-marker audio to save.
Type: int
Tag Name: TAG_MARKER_PRERECORD_PERIOD
Tag ID: 0x4D4B3033
Allowed Values: Min: 0, Max: 9
Special Handling: RESET_REQ.

Name: MarkerRecordPeriod
Description: The duration, in seconds, to record a marker audio record.
Type: int
Tag Name: TAG_MARKER_RECORD_PERIOD
Tag ID: 0x4D4B3034
Allowed Values: Min: 1, Max: 9999
Special Handling: RESET_REQ.

Name: MEAS_ALERT_TRIG_SRC // min firmware version 4.5.0
Type: Int
Tag Name: TAG_MEAS_ALERT_TRIG_SRC
Tag ID: 0x4D415453
Allowed Values: MinVal: 0, MaxVal: 10
Index: 1

Name: MEAS_ALERT_TRIG_LVL // min firmware version 4.5.0
Type: Float
Tag Name: TAG_MEAS_ALERT_TRIG_LVL
Tag ID: 0x4D41544C
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2

Name: MeasurementCounter
Description: The number of times to run a measurement when RunMode is set to Timed Stop.
Type: int
Tag Name: TAG_MEASUREMENT_COUNTS
Tag ID: 0x524D3036
Allowed Values: Min: 1, Max: 99999
Special Handling: RESET_REQ.

Name: MeasurementHistoryEnable
Description: Enable storing of measurement histories.
Type: int
Tag Name: TAG_MEASUREMENT_HISTORY
Tag ID: 0x524D3035
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: MeasurementAudioSnapshotPeriod
Description: The duration, in seconds, to record a sound snapshot at the beginning of a measurement.
Type: int
Tag Name: TAG_MEASUREMENT_SNAPSHOT_PERIOD
Tag ID: 0x4D533032
Allowed Values: Min: 1, Max: 9999
Special Handling: RESET_REQ.

Name: NightPenalty
Description: Penality added to Leq during the night time period for LDN and LDEN calculations.
Type: float
Tag Name: TAG_NIGHT_PENALTY
Tag ID: 0x444E3035
Allowed Values: Min: 0.0, Max: 99.9
Special Handling: RESET_REQ.

Name: NightTime
Description: Time offset from midnight, in seconds, for LDN and LDEN start of night calculations.
Type: int
Tag Name: TAG_NIGHT_TIME
Tag ID: 0x444E3033
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: EnableTimer2
Description: Enable second block timer when RunMode is set to Daily Timer
Type: int

Tag Name: TAG_NUM_BLOCK_TIMERS
Tag ID: 0x524D3138
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: EnableTimer3
Description: Enable third block timer when RunMode is set to Daily Timer.
Type: int
Tag Name: TAG_NUM_BLOCK_TIMERS
Tag ID: 0x524D3138
Allowed Values: 0,1 = Off, 2 = On
Special Handling: RESET_REQ.

Name: ObaBandwidth
Description: Sets the Oba Bandwidth.
Type: int
Tag Name: TAG_OBA_BANDWIDTH
Tag ID: 0x4F423032

Allowed Values: 0 = None,
1 = 1/1 Octave,
2 = 1/3 Octave,
3 = Both
Special Handling: FILTER_RESET_REQ.

Name: ObaMaxSpectrumMode
Description: Sets which method to use to determine max Oba spectra.
Type: int
Tag Name: TAG_OBA_ MAX_MODE
Tag ID: 0x4F423037
Allowed Values: 0 = Lmax, 1 = Bin max
Special Handling: FILTER_RESET_REQ.

Name: ObaRange
Description: Sets the Oba range to high or low.
Type: int
Tag Name: TAG_OBA_RANGE
Tag ID: 0x4F423035
Allowed Values: 0 = High, 1 = Low
Special Handling: FILTER_RESET_REQ.

Name: ObaWeighting
Description: Sets the Oba weighting filter.
Type: int
Tag Name: TAG_OBA_WEIGHTING
Tag ID: 0x4F423036

Allowed Values: 0 = A, 1 = C, 2 = Z
Special Handling: FILTER_RESET_REQ.

Name: OverallTitle
Description: Overall Title string
Type: string
Tag Name: TAG_OVERALL_TITLE
Tag ID: 0x4F543031
Allowed Values: Max length: 30.
Special Handling: None.

Name: PeakFrequencyWeighting
Description: Sets the Peak weighting filter.
Type: int
Tag Name: TAG_PEAK_WEIGHTING
Tag ID: 0x50573130
Allowed Values: 0 = A, 1 = C, 2 = Z
Special Handling: RESET_REQ.

Name: Percentiles
Description: A six element array of Ln distribution levels. (Table of Floats)
Type: floatseries
Tag Name: TAG_LN
Tag ID: 0x4C4E3032
Index: 0-5
Allowed Values: Min: 0.0, Max: 100.0
Special Handling: RESET_REQ.

Name: RAExitTime
Description: Time, in seconds, to allow the user to exit the room before starting a RT60 measurement.
Type: int
Tag Name: TAG_RA_EXIT_TIME
Tag ID: 0x45583031
Allowed Values: Min: 0, Max: 99
Special Handling: STOP_REQ.

Name: RAHighestFreq
Description: Sets the highest allowed frequency for RT60 measurements. Must be > RALowestFreq.
Type: int
Tag Name: TAG_RA_HIGHEST_FREQ
Tag ID: 0x48463031
Allowed Values: Min: 0, Max: 37 (see Trigger Source #defines in LxT831.h)
Special Handling: RESET_REQ.

Name: RALowestFreq

Description: Sets the lowest allowed frequency for RT60 measurements. Must be < RAHighestFreq.
Type: int
Tag Name: TAG_RA_LOWEST_FREQ
Tag ID: 0x4C463031
Allowed Values: Min: 0, Max: 37 (see Trigger Source #defines in LxT831.h)
Special Handling: RESET_REQ.

Name: RegressionSpan
Description: Width of a regression line. The value is a percentage of the critical bandwidth to either side of the center frequency.
Type: int
Tag Name: TAG_REGRESSION_SPAN
Tag ID: 0x46465437
Allowed Values: 0 = 0.50 1 = 0.75 2 = 1.0 3 = 1.5 4 = 2.0
Special Handling: None

Name: RT60Bandwidth
Description: Sets the Oba Bandwidth.
Type: int
Tag Name: TAG_RT60_BANDWIDTH
Tag ID: 0x42573031
Allowed Values: 0 = None, 1 = 1/1 Octave, 2 = 1/3 Octave, 3 = Both
Special Handling: FILTER_RESET_REQ.

Name: RT60BuildTime
Description: Reverberation build time, in seconds.
Type: int
Tag Name: TAG_RT60_BUILD_TIME
Tag ID: 0x42543031
Allowed Values: Min: 2, Max: 9
Special Handling: STOP_REQ.

Name: RT60NoiseAttenuation
Description: Adjusts the amplitude of the output signal.
Type: int
Tag Name: TAG_RT60_NOISE_ATTENUATION
Tag ID: 0x4E413031
Allowed Values: Min: 0.0, Max: 50.0
Special Handling: STOP_REQ.

Name: RT60NoiseType
Description: Sets the output signal noise type (Off, White, or Pink). Only for Interrupted method.
Type: int
Tag Name: TAG_RT60_NOISE_TYPE
Tag ID: 0x4E543031
Allowed Values: 0 = Off, 1 = White, 2 = Pink
Special Handling: STOP_REQ.

Name: RT60Method
Description: Sets the method of noise generation for the RT60 measurement.
Type: int
Tag Name: TAG_RT60_METHOD
Tag ID: 0x52544D31
Allowed Values: 0 = Impulse, 1 = Interrupted
Special Handling: RESET_REQ.

Name: RT60RunCount
Description: Sets the run count at a single location.
Type: int
Tag Name: TAG_RT60_BUILD_TIME
Tag ID: 0x42543031
Allowed Values: Min: 1, Max: 99
Special Handling: STOP_REQ.

Name: RT60RunTime
Description: Time, in seconds, for each run.
Type: int
Tag Name: TAG_RT60_RUN_TIME
Tag ID: 0x52543032
Allowed Values: Min: 2, Max: 19* (dependent on Sample Period) 4 @ 2.5ms, 9 @ 5ms, 18 @ 10ms, 19 @ 20ms (max allowed run time)
Special Handling: STOP_REQ.

Name: RT60SamplePeriod
Description: Sets the RT60 time sample period.
Type: int
Tag Name: TAG_RT60_SAMPLE_PERIOD
Tag ID: 0x53503031
Allowed Values: 0 = 2.5ms, 1 = 5ms, 2 = 10ms, 3 = 20ms
Special Handling: RESET_REQ.

Name: RT60TriggerLevel
Description: Level required to trigger a RT60.
Type: float
Tag Name: TAG_RT60_TRIGGER_LEVEL
Tag ID: 0x54524733

Allowed Values: Min: 0.0, Max: 99.9
Special Handling: STOP_REQ.

Name: RT60TriggerSource
Description: Sets the trigger source frequency. Limited to >= RALowestFreq and <= RAHighestFreq.
Type: int
Tag Name: TAG_RT60_TRIGGER_SOURCE
Tag ID: 0x54524732
Allowed Values: Min: 0, Max: 37 (see Trigger Source #defines in LxT831.h)
Special Handling: RESET_REQ.

Name: RunMode
Description: Sets the run mode for the measurement. Note: The Interval Time (TAG_INTERVAL_TIME) is used in conjunction with Measurements. If you have chosen the Run Mode (TAG_RUN_MODE) as Continuous, Single Block Timer or Daily Timer, then the Interval Time tells the meter how often to close a Measurement Record and start a new Measurement Record (this is different than storing). If you are not using one of these Run Modes then the Interval Time has no bearing on the rest of the meter.
Type: int
Tag Name: TAG_RUN_MODE
Tag ID: 0x524D3031
Allowed Values: 0 = Manual Stop, 1 = Timed Stop, 2 = Stop when Stable,
3 = Continuous, 4 = Single Block Timer, 5 = Daily Timer
Special Handling: STOP_REQ.

Name: RunTime1
Description: Sets the start time of block 1 when run mode is Single Block Timer or Daily Timer.
Type: int
Tag Name: TAG_TIMER_START_TIME_1
Tag ID: 0x524D3132
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: RunTime2
Description: Sets the start time of block 2 when run mode is Daily Timer.
Type: int
Tag Name: TAG_TIMER_START_TIME_2
Tag ID: 0x524D3133
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: RunTime3
Description: Sets the start time of block 3 when run mode is Daily Timer.
Type: int
Tag Name: TAG_TIMER_START_TIME_3
Tag ID: 0x524D3134

Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: RunTimer
Description: Measurement time, in seconds, when Run Mode is Timed Stop.
Type: int
Tag Name: TAG_RUN_TIME
Tag ID: 0x524D3032
Allowed Values: Min: 1, Max: 4294967295 (2^32-1)
Special Handling: STOP_REQ.

Name: SaveAllTimeSeries
Description: Sets the option to save all time histories for each RT60 measurement.
Type: int
Tag Name: TAG_SAVE_ALL_TIME_SERIES
Tag ID: 0x53565453
Allowed Values: 0 = No, 1 = Yes
Special Handling: RESET_REQ.

Name: SaveMeasurementAudioSnapshot
Description: Enable storing a sound recording snapshot at the beginning of a new measurement.
Type: int
Tag Name: TAG_MEASUREMENT_SNAPSHOT_ENABLE (deprecated, no longer available)
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: SaveMsmtLevelDistributionTable
Description: Enable storing the level distribution table for the measurement.
Type: int
Tag Name: TAG_MEASUREMENT_LEVEL_TABLE
Tag ID: 0x4D533033
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: SlmFrequencyWeighting
Description: Sets the Slm weighting filter.
Type: int
Tag Name: TAG_FREQ_WEIGHTING
Tag ID: 0x46573031
Allowed Values: 0 = A, 1 = C, 2 = Z
Special Handling: RESET_REQ.

Name: SlmGain
Description: Enable a +20dB gain.
Type: int

Tag Name: TAG_SLM_GAIN
Tag ID: 0x474E3030
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: SoundRecSampleRate
Description: Sets the sample rate for all sound recordings.
Type: int
Tag Name: TAG_SNAPSHOT_SAMPLING_RATE
Tag ID: 0x4D533034
Allowed Values: 0 = 48kHz, 1 = 24kHz, 2 = 16kHz, 3 = 8kHz
Special Handling: RESET_REQ.

Name: SpectralLnMode
Description: Enable storing a spectra with Ln's.
Type: int
Tag Name: TAG_SPECTRAL_LN
Tag ID: 0x4C4E3033
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: SRRange
Description: Sets the sound recording range to high or low.
Type: int
Tag Name: TAG_SR_RANGE
Tag ID: 0x53523032
Allowed Values: 0 = Low, 1 = High
Special Handling: RESET_REQ.

Name: StabledB
Description: Max level deviation to be considered stable when Run Mode is Stop when Stable.
Type: float
Tag Name: TAG_STABLE_LEVEL
Tag ID: 0x524D3034
Allowed Values: Min: 0.0, Max: 10.0
Special Handling: STOP_REQ.

Name: StableTimer
Description: Time, in seconds, to allow for a stable stop when Run Mode is Stop when Stable.
Type: int
Tag Name: TAG_STABLE_TIME
Tag ID: 0x524D3033
Allowed Values: Min: 1, Max: 4294967295 (2^32-1)
Special Handling: STOP_REQ.

Name: StreamHistoryOptions

Description: Streaming history option flags. Each bit represents a metric in the streamed data.
Type: int
Tag Name: TAG_STREAM_HISTORY_OPTIONS
Tag ID: 0x54483233
Allowed Values: Min: 0x00000000, Max: 0xFFFFFFFF* *(reference the Streaming Data Option flag #defines in LxT831.h of the SDK)
Special Handling: None.

Name: StreamHistoryPeriod
Description: Sets the time history sample period for streaming live data.
Type: int
Tag Name: TAG_STREAM_HISTORY_PERIOD
Tag ID: 0x54483232
Allowed Values: 0 = 20ms, 1 = 50ms, 2 = 100ms, 3 = 200ms, 4 = 500ms,
5 = 1 sec, 6 = 2 sec, 7 = 5 sec, 8 = 10 sec , 9 = 15 sec, 10 = 20 sec, 11 = 30 sec,
12 = 1 min, 13 = 2.5ms, 14 = 5ms, 15 = 10ms
Special Handling: None.

Name: Timer Start Date
Description: Start date for block mode timers when RunMode is Single Block Timer or Daily Timer.
Type: int
Tag Name: TAG_TIMER_START_DATE
Tag ID: 0x524D3130
Allowed Values: Min: 0, Max: 4294967295 (2^32-1)*
Special Handling: RESET_REQ. * Max value must be < EndDate

Name: Timer Stop Date
Description: End date for block mode timers when RunMode is Single Block Timer or Daily Timer.
Type: int
Tag Name: TAG_TIMER_STOP_DATE
Tag ID: 0x524D3131
Allowed Values: Min: *, Max: 4294967295 (2^32-1)
Special Handling: RESET_REQ. * Min value must be > StartDate

Name: Timer StopTime1
Description: Sets the stop time of block 1 when Run Mode is Single Block Timer or Daily Timer.
Type: int
Tag Name: TAG_TIMER_STOP_TIME_1
Tag ID: 0x524D3135
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: Timer StopTime2

Description: Sets the stop time of block 2 when Run Mode is Daily Timer.
Type: int
Tag Name: TAG_TIMER_ STOP_TIME_2
Tag ID: 0x524D3136
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: Timer StopTime3
Description: Sets the stop time of block 3 when Run Mode is Daily Timer.
Type: int
Tag Name: TAG_TIMER_ STOP_TIME_3
Tag ID: 0x524D3137
Allowed Values: Min: 0 (00:00:00), Max: 86399 (23:59:59)
Special Handling: RESET_REQ.

Name: TimeHistoryEnable
Description: Enable storing time histories.
Type: int
Tag Name: TAG_TIME_HISTORY_ENABLE
Tag ID: 0x54483031
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ.

Name: TimeHistoryPeriod
Description: Sets the time history sample period.
Type: int
Tag Name: TAG_TIME_HISTORY_PERIOD
Tag ID: 0x54483032
Allowed Values: 0 = 20ms, 1 = 50ms, 2 = 100ms, 3 = 200ms, 4 = 500ms, 5 = 1 sec, 6 = 2 sec, 7 = 5 sec, 8 = 10 sec , 9 = 15 sec, 10 = 20 sec, 11 = 30 sec, 12 = 1 min, 13 = 2 min, 14 = 5 min, 15 = 10 min, 16 = 15 min, 17 = 20 min, 18 = 30 min, 19 = 1 hr, 20 = 24 hr, 21 = 2.5ms,
22 = 5ms, 23 = 10ms
Special Handling: RESET_REQ.

Time History Details

The following 3 Tags are further defined by their options in Appendix **F. Request Time History**

With the use of the SDK you may now request Time History data from the meter during a run. Using the SDK call to obtain the first 120 entries using index=0:

/sdk?func=timehistjson&group=time&metric=1&index=0

You will receive data similar to the following:

```
{ "thLeq":[-448.535,
29.5812,32.3678,30.1272,29.4511,30.4587,30.3081,31.655,37.2697,29.3061,
```

30.1012,29.714,29.7077,29.3336,29.2633,29.2923,29.0833,29.1234,29.3179,
29.6077,29.6463,29.386,29.6642,30.2315,29.9971,30.1576,29.6707,29.2993,
29.3808,29.7397,29.4757,29.2464,29.2979,29.0867,29.2615,31.5266,29.3122
,29.0952,28.9625,28.7053,28.9837,31.6638,31.6003,29.2019,29.2885,29.052
3,29.1137,29.1724,29.4513,29.058,29.1468,29.1526,29.0469,29.1198,29.286
1,29.1281,29.4697,29.1244,29.3935,29.3841,29.0686,29.1948,29.8057,28.88
17,28.9994,29.1419,29.2544,29.1981,29.3095,29.873,34.5869,31.2319,31.91
27,29.7356,29.4183,29.4456,30.5363,29.0682,29.0811,29.2183,29.1999,29.0
677,30.5028,33.0639,29.2716,28.9886,28.9953,29.0321,29.5134,29.7697,29.
279,29.1794,29.0362,29.657,29.6611,29.1339,32.6172,40.1821,43.1842,40.2
935,29.264,29.062,29.2707,29.7054,29.2747,29.0494,29.2564,40.4,29.8031,
41.0553,39.1903,35.8073,33.5164,39.4362,28.8922,35.9437,33.7467,29.3648
,29.1172,29.078],

"thMetrics":[-1e+12,
30.5462,34.1595,31.5015,30.2019,31.4555,31.6151,35.3168,41.9486,30.5075
,31.2145,30.8437,30.5442,29.8954,29.4887,29.66,29.3351,29.4749,29.6527,
29.9383,29.9621,29.9621,29.9289,30.8826,30.4007,30.8308,30.2732,29.6122
,29.8146,31.5467,29.8932,29.5392,29.8149,29.4545,29.9245,36.3694,33.062
2,29.5615,29.1198,29.3978,29.477,36.5639,36.4588,32.5068,29.8758,29.273
,29.421,29.4998,30.0999,29.3878,29.4353,29.6863,29.5447,29.3848,29.7719
,29.4741,31.0145,29.3863,29.7131,30.482,29.5234,29.4345,31.5408,30.9489
,29.255,29.4747,29.6142,29.6397,29.8622,31.6811,37.3492,33.7302,33.5459
,33.0956,29.9533,30.0892,32.242,30.1973,29.6235,29.5016,29.627,29.3825,
33.3443,37.1244,29.7552,29.4064,29.441,29.3816,30.8179,30.6801,29.7126,
29.7596,29.3286,30.3273,30.1711,29.6068,35.7212,45.8777,47.4929,45.896,
30.6949,29.4513,29.5762,30.659,29.6591,29.4517,29.788,43.3341,36.2022,4
5.1855,44.6324,42.1437,38.6118,45.1543,31.0692,39.122,37.3118,30.8909,2
9.3443,29.4667],

"thFlags":[2147483648,8192,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

"thTime":[1588007943,1588007943,1588007944,1588007945,1588007946,158800
7947,1588007948,1588007949,1588007950,1588007951,1588007952,1588007953,
1588007954,1588007955,1588007956,1588007957,1588007958,1588007959,15880
07960,1588007961,1588007962,1588007963,1588007964,1588007965,1588007966
,1588007967,1588007968,1588007969,1588007970,1588007971,1588007972,1588
007973,1588007974,1588007975,1588007976,1588007977,1588007978,158800797
9,1588007980,1588007981,1588007982,1588007983,1588007984,1588007985,158
8007986,1588007987,1588007988,1588007989,1588007990,1588007991,15880079
92,1588007993,1588007994,1588007995,1588007996,1588007997,1588007998,15
88007999,1588008000,1588008001,1588008002,1588008003,1588008004,1588008
005,1588008006,1588008007,1588008008,1588008009,1588008010,1588008011,1
588008012,1588008013,1588008014,1588008015,1588008016,1588008017,158800
8018,1588008019,1588008020,1588008021,1588008022,1588008023,1588008024,
1588008025,1588008026,1588008027,1588008028,1588008029,1588008030,15880
08031,1588008032,1588008033,1588008034,1588008035,1588008036,1588008037

,1588008038,1588008039,1588008040,1588008041,1588008042,1588008043,1588008044,1588008045,1588008046,1588008047,1588008048,1588008049,1588008050,1588008051,1588008052,1588008053,1588008054,1588008055,1588008056,1588008057,1588008058,1588008059,1588008060,1588008061],

"thAction":[514,1063675496,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217],"Result":"Success: 0
","ResultCode":0,"ResultName":"Success" }

Changing the index will allow you to query any of the time history data collected thus far in the measurement. Once the Meter has stored the current run, the Time History for that run will no longer be available. You will be most interested in the thLeq, the thMetrics and the thTime entries.

Return Array Values Definitions

thLeq:

Values in dB of the defined Leq for Time History

Exception values less than -99 are invalid (usually means that it is a run, stop or some other record.)

thMetrics:

Values in dB representing the selected metric (in this case 1, the first metric after the Leq). If Time History is configured with only LAFmax then this will be 1. See G Time History Options

Exception values less than -99 are invalid (usually means that it is a run, stop or some other record.)

thFlags:

See the Functional description below, Flags and Their Use

Exception: The value 0 represents no flag.

If Flags = `2147483648 = 0x80000000` then refer to the thAction otherwise ignore thAction

In the case 514 = 0x0202 => Run

`8192` => Time History Partial Record, means the record did not align with the time history period (if the time history period is one minute then this would appear for the first and last record of a run if you did not start and end them exactly on a one minute boundary)

Most all other values hold no meaning.

thActions:

See below description below, Actions

thTime:

Seconds since epoch:

`1588007943 =>` **GMT**: Monday, April 27, 2020 5:19:03 PM

Flags and Their Use

Functional descriptions of flags. The function shows the ones most used or seen during standard environmental monitoring.

```
var TIMEHIST_MARKER1 = (0x00000001);
var TIMEHIST_MARKER2 = (0x00000002);
var TIMEHIST_MARKER3 = (0x00000004);
var TIMEHIST_MARKER4 = (0x00000008);
var TIMEHIST_MARKER5 = (0x00000010);
var TIMEHIST_MARKER6 = (0x00000020);
var TIMEHIST_MARKER7 = (0x00000040);
var TIMEHIST_MARKER8 = (0x00000080);
var TIMEHIST_MARKER9 = (0x00000100);
var TIMEHIST_MARKER10 = (0x00000200);
var TIMEHIST_OVERLOAD = (0x00000400);
var TIMEHIST_OBA_OVLD = (0x00000800);
var TIMEHIST_EXCDED = (0x00001000);
var TIMEHIST_T2READY = (0x00002000);
var TIMEHIST_PARTIAL = (0x00002000);
var TIMEHIST_SR = (0x00004000);
var TIMEHIST_BACKERASE = (0x00008000);
var TIMEHIST_MANUAL = (0x00010000);
var TIMEHIST_SESSION_LOG = (0x80000000);
var TIMEHIST_MARKER_MASK = (TIMEHIST_MARKER1 | TIMEHIST_MARKER2 | TIMEHIST_MAR
KER3 |
 TIMEHIST_MARKER4 | TIMEHIST_MARKER5 | TIMEHIST_MARKER6 | TIMEHIST_MARKER7 |

TIMEHIST_MARKER8 | TIMEHIST_MARKER9 | TIMEHIST_MARKER10 | TIMEHIST_BACKERASE |
 TIMEHIST_MANUAL);

function transFlag(flag, noOBA)
{
        var strDetail = "";
        if (flag & TIMEHIST_OVERLOAD) {
                strDetail = "Overload";
        }
        if ((flag & TIMEHIST_OBA_OVLD) && !noOBA) {
                strDetail = "IDS_OBA_OVLD";
        }
        if (flag & TIMEHIST_EXCDED) {
                strDetail = "Exceeded";
        }
        if (flag & TIMEHIST_T2READY) {
                strDetail = "IDS_T2READY";
        }
        if (flag & TIMEHIST_PARTIAL) {
                strDetail = "Partial Record ";
        }
        return strDetail;
}
```

Actions

thActions array contains values for every entry, though most are not used. If the thFlags contains 0x80000000, then the Action will have a meaning as described below.

```
function actionCause(action) {
        var retVal = false;
        for (var i = 0; i < 8; ++i) {
                topElems[i].html("&nbsp;");
        }
        if (lastCount > 0) {
                var act = (action & 0x00FF).toString();
```

```
                 var cause = (action & 0xFF00).toString();
                 var actVal = Sess.Actions[act]
                 if (actVal !== undefined) {
                  topElems[0].html("<img src='/images/firmware/" + actVal.i +
".png' class='hdrImages'/> &nbsp;" + actVal.n);
                         topElems[2].html(Sess.Causes[cause]);
                         retVal = true;
                 }
        }
        return retVal;
}
```

```
        var Actions = {};
        Actions["0"] = { n: "IDS_ERROR", i: "warning" };
                      //     // Err action performed
        Actions["1"] = { n: "IDS_STOP", i: "bmp_stop" };
                      //var PRM_ACTION_STOP = (0x0001);    // Stop action perf
ormed
        Actions["2"] = { n: "IDS_RUN", i: "bmp_run1" };
                      //var PRM_ACTION_RUN = (0x0002);    // Run
        Actions["4"] = { n: "IDS_PAUSE", i: "bmp_pause" };
                      //var PRM_ACTION_PAUSE = (0x0004);    // Pause
        Actions["8"] = { n: "IDS_RESUME", i: "bmp_runnext" };
                      //var PRM_ACTION_RESUME = (0x0008);    // Resume from Pa
use
        Actions["16"] = { n: "IDS_VOICE", i: "audio" };
              //var PRM_ACTION_VOICE = (0x0010);    // Voice Recording
        Actions["32"] = { n: "IDS_SOUND", i: "audio" };
              //var PRM_ACTION_AUDIO = (0x0020);    // Audio Recording {reserv
ed for future}
        Actions["64"] = { n: "IDS_CALCHECK", i: "bmp_cal" };
     //var PRM_ACTION_CAL = (0x0040);    // Calibration record, need deviation
 stored also! {reserved for future}
        Actions["128"] = { n: "IDS_RESET", i: "bmp_stopreset" };
                      //var PRM_ACTION_RESET = (0x0080);    // Reset {reserved
 for future}
        Actions["129"] = { n: "IDS_GPSTIMESYNC", i: "gps" };
     //var PRM_ACTION_GPS_SYNC = (0x0081);    // GPS Time Sync
        Actions["130"] = { n: "IDS_BACKERASE", i: "back_erase" };
              //var PRM_BACK_ERASE = (0x0082);    // Back Erase Session Log
        Actions["131"] = { n: "IDS_MARK", i: "x_mark" };
                      //var PRM_ACTION_MARKER = (0x0083);    // Marker set
        Actions["132"] = { n: "IDS_CALCHANGE", i: "bmp_cal" };
              //var PRM_ACTION_CALCHG = (0x0084);    // Calibration Change Per
formed
        Actions["133"] = { n: "IDS_PREAMP_DISCONNECT", i: "warning" };     //v
ar PRM_ACTION_PREAMPOFF = (0x0085);    // Preamp Removed =(show type of preamp
 removed);
```

```
        Actions["134"] = { n: "IDS_PREAMP_CONNECT", i: "warning" };
     //var PRM_ACTION_PREAMPON = (0x0086);    // Preamp Connected =(show type
of preamp connected);
        Actions["135"] = { n: "IDS_CREATED_AVERAGE", i: "bmp_CreateAvg" };
     //var PRM_ACTION_FILE_AVG = (0x0087);    // Create a new average or add t
o average in data explorer
        Actions["136"] = { n: "IDS_COMMS_WATCHDOG", i: "warning" };
     //var PRM_ACTION_WATCHDOG = (0x0088);    // Phoenix watchdog reset
        Actions["137"] = { n: "USB", i: "bmp_usb" };
                               //var PRM_ACTION_USB_FAULT = (0x0089);    // US
B controller fault detected
        Actions["138"] = { n: "Panic", i: "warning" };
                               //var PRM_ACTION_PANIC = (0x008a);    // Panic
restart detected
        Actions["139"] = { n: "IDS_HLD_CHG_FAULT", i: "warning" };
                                       //var PRM_ACTION_PANIC = (0x008a);    /
/ Panic restart detected
        Actions["140"] = { n: "IDS_HLD_NTP_SYNC", i: "ntpSync" };
                                       //var PRM_ACTION_NTP_SYNC            (0
x008C)
        Actions["141"] = { n: "IDS_HLD_TIMEADJUST", i: "clock" };
                                       //var PRM_ACTION_TIME_CHANGE
 (0x008D)

Sess.Actions = Actions;

        var Causes = [];
        Causes["256"] = "IDS_KEY";
         //var PRM_CAUSE_KEY                  = (0x0100);    // Action caused
by keyboard command            // SR Events Cause
        Causes["257"] = "IDS_EVENT";
         //var PRM_CAUSE_KEY                  = (0x0100);    // Action caused
by keyboard command            // SR Events Cause
        Causes["258"] = "IDS_HLD_SYNC_POS";
                 //var PRM_CAUSE_TIME_SYNC_POSITIVE   (0x0100)    // Action ca
used by keyboard command            // SR Events Cause
        Causes["512"] = "IDS_IO";
         //var PRM_CAUSE_IO                   = (0x0200);    // Action caused
by I/O command                 // SR Measurement
        Causes["513"] = "IDS_MEASUREMENT";
                 //var PRM_CAUSE_IO                   = (0x0200);    // Action
 caused by I/O command                 // SR Measurement
        Causes["514"] = "IDS_HLD_SYNC_NEG";
                 //var PRM_CAUSE_TIME_SYNC_NEGATIVE   (0x0200)    // Action ca
used by keyboard command            // SR Events Cause
        Causes["1024"] = "IDS_TIMER";
//var PRM_CAUSE_TIMER                  = (0x0400);    // Action caused by the ru
n or stable timer    // SR Markers
        Causes["1025"] = "IDS_TABMARKER";                    //var PRM_CAUSE_TIM
ER                  = (0x0401);    // Action caused by the run or stable timer
   // SR Markers

Causes["1280"] = "IDS_HLD_PRM_CAUSE_SCHEDULE";//var PRM_CAUSE_SCHEDULE
     = (0x0500);    // Action caused by schedule
```

```
        Causes["2048"] = "IDS_POWER";
//var PRM_CAUSE_POWER                = (0x0800);    // Stop due to power failu
re
        Causes["4096"] = "IDS_OUTOFMEMORY";                                 //v
ar PRM_CAUSE_MEMORY               = (0x1000);    // Stop due to out-of-memory
        Causes["8192"] = "IDS_PREAMP_CONNECT";              //var PRM_CAUSE_P
REAMP_CONNECT       = (0x2000);    // Stop due to preamp connect
        Causes["16384"] = "IDS_PREAMP_DISCONNECT";          //var PRM_CAUSE_P
REAMP_DISCONNECT    = (0x4000);    // Stop due to preamp disconnect
        Causes["32768"] = "IDS_STABLE";
//var PRM_CAUSE_STABLE               = (0x8000);    // Stop on STABLE
        Causes["33024"] = "IDS_COMMS_WATCHDOG";             //var PRM_CAUSE_8
31_INT_ET           = (0x8100);    // Phoenix communications watchdog via 831
INTET
        Causes["33280"] = "IDS_COMMS_WATCHDOG";             //var PRM_CAUSE_A
NALOGMODEM          = (0x8200);    // Phoenix UsbHost watchdog via Analog Mode
m
        Causes["33536"] = "Other";
         //var PRM_CAUSE_COUNTER_WRAP         = (0x8300);    // uClinux tick c
ounter is a signed 32 bit value that will wrap around in less than 248 days
        Causes["33792"] = "Internal Fault";                                 /
/var PRM_CAUSE_INTERNAL_FAULT       = (0x8400);    // Internal fault
        Causes["34048"] = "IDS_HLD_METER_UPDATE";           //var PRM_CAUSE_M
ETER_UPDATE         = (0x8500);    // Meter update (FW, Options, etc.)
        Causes["34304"] = "IDS_HLD_CHG_FAULT_TEMP";                //var PRM_
CAUSE_CHG_TEMP_FAULT       (0x8600)    // Charger Temperature fault
        Causes["34560"] = "IDS_HLD_CHG_FAULT_OVERV";        //var PRM_CAUSE_CHG
_OVER_V_FAULT     (0x8700)    // Charger Over Voltage fault
        Causes["34816"] = "IDS_HLD_CHG_FAULT_TYPE";
//var PRM_CAUSE_CHG_BATT_TYPE_FAULT  (0x8800)    // Charger Battery Type fault
        Causes["35072"] = "IDS_HLD_CHG_LOW_PWR";   //var PRM_CAUSE_CHG_NO_CURR
ENT_FAULT  (0x8900)    // Charger can't supply current fault

Sess.Causes = Causes;
```

Time History Options.

Name: TimeHistoryOptions1
Description: Time history option flags, set one. Each bit represents a time history metric to store.
Type: int
Tag Name: TAG_TIME_HISTORY_OPTIONS
Tag ID: 0x54483033
Allowed Values: Min: 0x00000000, Max: 0xFFFFFFFF* *(reference the TimeHistoryOptions1 #defines in LxT831.h of the SDK)
Special Handling: RESET_REQ.

Name: TimeHistoryOptions2

Description: Time history option flags, set two. Each bit represents a time history metric to store.
Type: int
Tag Name: TAG_TIME_HISTORY_MISC
Tag ID: 0x54483034
Allowed Values: Min: 0x00000000, Max: 0xFFFFFFFF* *(reference the TimeHistoryOptions2 #defines in LxT831.h of the SDK)
Special Handling: RESET_REQ.

Name: TimeHistoryOptions3
Description: Time history option flags, set three. Each bit represents a time history metric to store.
Type: int
Tag Name: TAG_TIME_HISTORY_OPTIONS3
Tag ID: 0x54483035
Allowed Values: Min: 0x00000000, Max: 0xFFFFFFFF* *(reference the TimeHistoryOptions3 #defines in LxT831.h of the SDK Or Tags.js)
Special Handling: RESET_REQ.

Name: Tonality_1996_2
Description: Enable tonality measurements.
Type: int
Tag Name: TAG_TONE_1996_2
Tag ID: 0x46465435
Allowed Values: 0 = Off, 1 = On
Special Handling: None

Name: ToneSeekDelta
Description: Parameter used in the calculation of a tone.
Type: int
Tag Name: TAG_TONE_SEEK_DELTA
Tag ID: 0x46465436
Allowed Values: 0 = 1 dB 1 = 2 dB 2 = 3 dB 3 = 4 dB
Special Handling: None

Name: TriggerMethod
Description: Sets the event trigger method.
Type: int
Tag Name: TAG_EVENT_TRIG_METHOD
Tag ID: 0x45563039
Allowed Values: 0 = Level, 1 = Dynamic
Special Handling: RESET_REQ.

Name: VaisalaHeaterDisTime
Description: Vaisala weather station end time for Timed state.
Type: int

Tag Name: TAG_VHEAT_DIS_TIME
Tag ID: 0x56483033
Allowed Values: Min: 0 (12:00:00 am), Max: 86399 (11:59:59 pm)
Special Handling: None

Name: VaisalaHeaterEnaTime
Description: Vaisala weather station start time for Timed state.
Type: int
Tag Name: TAG_VHEAT_ENA_TIME
Tag ID: 0x56483032
Allowed Values: Min: 0 (12:00:00 am), Max: 86399 (11:59:59 pm)
Special Handling: None

Name: VaisalaHeaterState
Description: Vaisala weather station heater state.
Type: int
Tag Name: TAG_VHEAT_STATE
Tag ID: 0x56483031
Allowed Values: 0 = Off 1 = Enabled 2 = Timed
Special Handling: None

Name: Weather Enable
Description: Sets which, if any, weather option is enabled.
Type: int
Tag Name: TAG_WEATHER_ENABLE
Tag ID: 0x41443031
Allowed Values: 0 = Off, 1 = Weather-INT, 2 = Vaisala
Special Handling: RESET_REQ.

Name: WindDirection
Description: Set wind direction reporting format.
Type: int
Tag Name: TAG_WIND_DIRECTION
Tag ID: 0x574E4433
Allowed Values: 0 = Compass 1 = Degrees 2 = Percent 3 = Volts
Special Handling: None

Name: WindExcdLevel
Description: If WindPause is set to “Yes” and wind speeds exceed this level, then sound exceedances are held off.
Type: float
Tag Name: TAG_WIND_LEVEL
Tag ID: 0x574E4435
Allowed Values: Min: 0.0, Max: 1,000.0
Special Handling: RESET_REQ

Name: WindHysteresis

Description: If sound exceedances are paused due to a wind exceedance, wind speed must drop below "WindExcdLevel – WindHysteresis" before sound exceedances are resumed.
Type: float
Tag Name: TAG_WIND_HYSTERESIS
Tag ID: 0x574E4436
Allowed Values: Min: 0.0, Max: 1,000.0
Special Handling: RESET_REQ

Name: WindPause
Description: If set to "Yes" and wind speeds exceeds WindExcdLevel, then sound exceedances are held off.
Type: int
Tag Name: TAG_WIND_PAUSE
Tag ID: 0x574E4437
Allowed Values: 0 = No, 1 = Yes
Special Handling: RESET_REQ

Name: WindScale
Description: Wind sensor scale factor (from design parameters of the wind sensor).
Type: float
Tag Name: TAG_WIND_SCALE
Tag ID: 0x574E4431
Allowed Values: Min: 0.0001, Max: 1,000.0
Special Handling: RESET_REQ

Name: WindThreshold
Description: Sets the wind threshold. Values exceeding this are considered "windy".
Type: float
Tag Name: TAG_WIND_THRESHOLD
Tag ID: 0x574E4434
Allowed Values: Min: 0.0, Max: 1,000.0
Special Handling: RESET_REQ

Name: WindUnits
Description: Set wind units ("mi/h", "m/s", "km/h", "knots", or "fps").
Type: string
Tag Name: TAG_WIND_UNITS
Tag ID: 0x574E4432
Allowed Values: Max length: 31 characters + null terminator
Special Handling: RESET_REQ

Email Settings (831C Only)

Email alert System Properties that allow you to set up the email options for when to send out emails. Only use this with firmware greater than 3.1.1.0.

Name: EVENT NOTIFICATION

Type: String
Tag Name: TAG_EVENT_NOTIFICATION
Tag ID: 0x45563230
Tag String: 'EV20'
Allowed Values: MinVal: 0, MaxVal: 15
Any combination of the following are allowed:
0=No Notification for Events or Measurements
1=Event Email
2=Event Text
4=Measurement Event
8=Measurement Text

Event Triggering Settings (831C Only)

These settings are available on firmware 3.2.1.0 and greater. When setting these values remember to include the index. The 1_1 represents the On/Off toggle; use index 1. The 1_2 represents the trigger's value; use index 2.

Name: RUNNING_LEQ_PERIOD
Type: Int
Tag Name: TAG_RUNNING_LEQ_PERIOD
Tag ID: 0x45563232
Tag String: 'EV22'
Allowed Values: MinVal: 1, MaxVal: 3600 in seconds
Index: 0
Special Handling: RESET_REQ

Name: SPL_1_1
Type: Int
Tag Name: TAG_ETS_SPL_1_1
Tag ID: 0x314C5053
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: SPL_1_2
Type: Float
Tag Name: TAG_ETS_SPL_1_2
Tag ID: 0x314C5053
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: PEAK_1_1
Type: Int

Tag Name: TAG_ETS_PEAK_1_1
Tag ID: 0x31304B50
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: PEAK_1_2
Type: Float
Tag Name: TAG_ETS_PEAK_1_2
Tag ID: 0x31304B50
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: RUNNING_LEQ_1
Type: Int
Tag Name: TAG_ETS_RUNNING_LEQ_1
Tag ID: 0x51454C52
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: RUNNING_LEQ_2
Type: Float
Tag Name: TAG_ETS_RUNNING_LEQ_2
Tag ID: 0x51454C52
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: A_LINR_1
Type: Int
Tag Name: TAG_ETS_A_LINR_1
Tag ID: 0x4C414C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: A_LINR_2
Type: Float
Tag Name: TAG_ETS_A_LINR_2
Tag ID: 0x4C414C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: A_FAST_1
Type: Int
Tag Name: TAG_ETS_A_FAST_1
Tag ID: 0x46414C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: A_FAST_2
Type: Float
Tag Name: TAG_ETS_A_FAST_2
Tag ID: 0x46414C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: A_SLOW_1
Type: Int
Tag Name: TAG_ETS_A_SLOW_1
Tag ID: 0x53414C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: A_SLOW_2
Type: Float
Tag Name: TAG_ETS_A_SLOW_2
Tag ID: 0x53414C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: A_IMPL_1
Type: Int
Tag Name: TAG_ETS_A_IMPL_1
Tag ID: 0x49414C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: A_IMPL_2
Type: Float
Tag Name: TAG_ETS_A_IMPL_2
Tag ID: 0x49414C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2

Special Handling: RESET_REQ

Name: A_PEAK_1
Type: Int
Tag Name: TAG_ETS_A_PEAK_1
Tag ID: 0x50414C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: A_PEAK_2
Type: Float
Tag Name: TAG_ETS_A_PEAK_2
Tag ID: 0x50414C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: C_LINR_1
Type: Int
Tag Name: TAG_ETS_C_LINR_1
Tag ID: 0x4C434C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: C_LINR_2
Type: Float
Tag Name: TAG_ETS_C_LINR_2
Tag ID: 0x4C434C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: C_FAST_1
Type: Int
Tag Name: TAG_ETS_C_FAST_1
Tag ID: 0x46434C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: C_FAST_2
Type: Float
Tag Name: TAG_ETS_C_FAST_2
Tag ID: 0x46434C41

Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: C_SLOW_1
Type: Int
Tag Name: TAG_ETS_C_SLOW_1
Tag ID: 0x53434C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: C_SLOW_2
Type: Float
Tag Name: TAG_ETS_C_SLOW_2
Tag ID: 0x53434C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: C_IMPL_1
Type: Int
Tag Name: TAG_ETS_C_IMPL_1
Tag ID: 0x49434C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: C_IMPL_2
Type: Float
Tag Name: TAG_ETS_C_IMPL_2
Tag ID: 0x49434C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: C_PEAK_1
Type: Int
Tag Name: TAG_ETS_C_PEAK_1
Tag ID: 0x50434C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: C_PEAK_2
Type: Float
Tag Name: TAG_ETS_C_PEAK_2

Tag ID: 0x50434C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: Z_LINR_1
Type: Int
Tag Name: TAG_ETS_Z_LINR_1
Tag ID: 0x4C5A4C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: Z_LINR_2
Type: Float
Tag Name: TAG_ETS_Z_LINR_2
Tag ID: 0x4C5A4C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: Z_FAST_1
Type: Int
Tag Name: TAG_ETS_Z_FAST_1
Tag ID: 0x465A4C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: Z_FAST_2
Type: Float
Tag Name: TAG_ETS_Z_FAST_2
Tag ID: 0x465A4C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: Z_SLOW_1
Type: Int
Tag Name: TAG_ETS_Z_SLOW_1
Tag ID: 0x535A4C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: Z_SLOW_2

Type: Float
Tag Name: TAG_ETS_Z_SLOW_2
Tag ID: 0x535A4C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: Z_IMPL_1
Type: Int
Tag Name: TAG_ETS_Z_IMPL_1
Tag ID: 0x495A4C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: Z_IMPL_2
Type: Float
Tag Name: TAG_ETS_Z_IMPL_2
Tag ID: 0x495A4C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

Name: Z_PEAK_1
Type: Int
Tag Name: TAG_ETS_Z_PEAK_1
Tag ID: 0x505A4C41
Allowed Values: 0=Off, 1=On
Index: 1
Special Handling: RESET_REQ

Name: Z_PEAK_2
Type: Float
Tag Name: TAG_ETS_Z_PEAK_2
Tag ID: 0x505A4C41
Allowed Values: MinVal: 0.1, MaxVal: 200.00
Index: 2
Special Handling: RESET_REQ

FFT Properties

These FFT properties are only available on the 831C with min firmware 4.6.0. You will only be able to access them if you have the FFT Option installed.

Name: FFT_MEASUREMENT_COUNT
Type: UInt
Tag Name: TAG_FFT_MEASUREMENT_COUNT
Tag ID: 0x46464D43

Allowed Values: MinVal: 1, MaxVal: 99999
Special Handling: RESET_REQ

Name: FFT_HISTORY_CONTROL
Type: UInt
Tag Name: TAG_FFT_HISTORY_CONTROL
Tag ID: 0x46464843
Allowed Values: MinVal: 0, MaxVal: 1
Special Handling: RESET_REQ

Name: FFT_INTEGRATION
Type: Enum
Tag Name: TAG_FFT_INTEGRATION
Tag ID: 0x46465449
Allowed Values: MinVal: 0, MaxVal: 2
Special Handling: RESET_REQ

Name: RUN_ON_TRIGGER
Type: UInt
Tag Name: TAG_RUN_ON_TRIGGER
Tag ID: 0x524D3233
Allowed Values: MinVal: 0, MaxVal: 1
Special Handling: RESET_REQ

D. System Property Descriptions

Name: AutoStoreMode
Description: Set the Auto-Store mode, which determines whether or not pressing the Stop key also causes a data file to be stored.
Tag Name: TAG_AUTO_STORE
Tag ID: 0x41533031
Allowed Values: 0 = none (do not automatically save a file) 1 = prompt (ask if a file should be saved) 2 = store (automatically store a file)
Special Handling: None

Name: AutoSyncDateTime
Description: Allow the SLM time to be set to the PC time when the unit is connected to the 831 G3 Utility software .
Tag Name: TAG_AUTOSYNC
Tag ID: 0x44503034
Allowed Values: 0 = Off, 1 = On
Special Handling: None

Name: BackLight
Description: Sets the brightness of the backlight or turns it off.
Tag Name: TAG_POWER_LIGHT
Tag ID: 0x50573034
Allowed Values: 0 = off 1 = dim 2 = bright
Special Handling: None

Name: BackLightTimer
Description: Duration of display backlight after key press activity.
Tag Name: TAG_POWER_LIGHT_TIME
Tag ID: 0x50573033
Allowed Values: 0 = 5 seconds 1 = 10 seconds 2 = 30 seconds 3 = 60 seconds 4 = always on
Special Handling: None

Name: BatteryType
Description: Set the type of battery that will be installed into the SLM.
Tag Name: TAG_POWER_BATT
Tag ID: 0x50573035
Allowed Values: 0 = Alkaline 1 = NiMH 2 = Lithium
Special Handling: None

Name: DateFormat
Description: Select date format.
Tag Name: TAG_DISPLAY_DATE
Tag ID: 0x44503033
Allowed Values: 0 = dd mmm yyyy 1 = yyyy mmm dd
Special Handling: None

Name: DecimalFormat
Description: Select decimal symbol (period or comma).
Tag Name: TAG_DISPLAY_DECIMAL
Tag ID: 0x44503032
Allowed Values: 0 = period, 1 = comma
Special Handling: None

Name: DisplayContrast
Description: Adjust LCD display contrast.
Tag Name: TAG_DISPLAY_CONTRAST
Tag ID: 0x44503031
Allowed Values: Min: -9, Max: 9
Special Handling: None

Name: DisplayOptions1 – DisplayOptions8
Description: Display options (see table below).
Tag Name: TAG_DISPLAYOPTIONS1 – TAG_DISPLAYOPTIONS8
Tag ID: 0x44535031, 0x44535032, 0x44535033, 0x44535034, 0x44535035, 0x44535036, 0x44535037, 0x44535038
Allowed Values: See notes below.
Special Handling: None

Name: FullOctaveReferenceSpectra1
Description: Set full octave (1/1) reference spectra 1.
Tag Name: TAG_FULL_OCTAVE_REFERENCE_SPECTRA1
Tag ID: 0x46525331
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 12 floats

Name: FullOctaveReferenceSpectra2
Description: Set full octave (1/1) reference spectra 2.
Tag Name: TAG_FULL_OCTAVE_REFERENCE_SPECTRA2
Tag ID: 0x46525332
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 12 floats

Name: FullOctaveReferenceSpectra3
Description: Set full octave (1/1) reference spectra 3.
Tag Name: TAG_FULL_OCTAVE_REFERENCE_SPECTRA3
Tag ID: 0x46525333
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 12 floats

Name: FullOctaveReferenceSpectra4
Description: Set full octave (1/1) reference spectra 4.
Tag Name: TAG_FULL_OCTAVE_REFERENCE_SPECTRA4
Tag ID: 0x46525334

Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 12 floats

Name: Language
Description: Select the language used on the SLM menus.
Tag Name: TAG_LANGUAGE
Tag ID: 0x4C473031
Allowed Values: 0 = English 1 = French 2 = German 3 = Italian 4 = Portuguese 5 = Spanish 6 = Swedish 7 = Norwegian 8 = Portuguese (BR)
Special Handling: None

Name: LockCal
Description: Allows or disallows calibration while in any "locked" mode.
Tag Name: TAG_LOCK_CAL
Tag ID: 0x4C4B3032
Allowed Values: 0 = Off, 1 = On
Special Handling: None

Name: LockCombination
Description: Four-digit combination to unlock the tamper-proof lock feature on the SLM.
Tag Name: TAG_LOCK_COM
Tag ID: 0x4C4B3034
Allowed Values: Min: 0000, Max: 9999
Special Handling: None

Name: LockMode
Description: Select lock mode, which specifies behavioral conditions while SLM is locked.
Tag Name: TAG_LOCK_MODE
Tag ID: 0x4C4B3031
Allowed Values: 0 = unlocked 1 = locked with manual stop 2 = locked with auto-store 3 = fully locked
Special Handling: None

Name: NetworkPassword (831-INT-ET only, Not available on the 831C)
Description: Network Password setting.
Type: string
Tag Name: TAG_NETWORK_PASSWORD
Tag ID: 0x4E505744
Allowed Values: Max length: 30 characters. Once set you will need to reconnect with a secure connection as described in the

Encryption (Model 831-INT-ET) section.
Special Handling: None

Name: Outputs
Description: Set the function of the jack output.
Tag Name: TAG_OUTPUTS
Tag ID: 0x4F553031
Allowed Values: 0 = Off 1 = AC/DC 2 = Headset
Special Handling: STOP_REQ

Name: PowerExternalShutOff
Description: The External Voltage Level at which the meter will automatically shutoff.
Tag Name: TAG_POWER_EXT_SHUTOFF
Tag ID: 0x50573036
Type: float
Allowed Values: min: 10.0 and max: 25.0
Special Handling: None
Get the property:
```
$.getJSON("/sdk?func=getProperty&type=float&tagid=0x50573036", console.log);
```
Set the property:
```
$.getJSON("/sdk?func=setProperty&type=float&tagid=0x50573036&value=10", console.log);
$.getJSON("/sdk?func=LATCHSETTINGS&data=1&marksettingschanged=true");
```

Name: PowerAutoOff
Description: The duration of time the instrument will stay on when no activity is occurring, where activity includes button presses, running a measurement, USB communications, etc..
Tag Name: TAG_POWER_OFF_TIME
Tag ID: 0x50573031
Allowed Values: 0 = 5 minutes 1 = 10 minutes 2 = 30 minutes 3 = 60 minutes 4 = never powers off automatically
Special Handling: None

Name: PowerAutoSleep
Description: Set the"Power-Save Time" after which battery power is significantly reduced by shutting down the display and the analog ciruitry and ceasing signal processing activities.
Tag Name: TAG_POWER_SLEEP_TIME
Tag ID: 0x50573032
Allowed Values: 0 = 5 minutes 1 = 10 minutes 2 = 30 minutes 3 = 60 minutes 4 = never enters Power-Save mode
Special Handling: None

Name: Random Correction (Mic Correction)
Description: Random or Mic Corrections that are applied in the meter
Tag Name: TAG_RANDOMCORRECTION
Tag ID: 0x52443031
Allowed Values: 0 = Off,
10=WS001 - 3/5" Windscreen (Default)

1=RI2FF
2=FF2RI
3=FF2RI_2106_8
4=FF2FF_2106_8
5=FF290_2106_8
6=FF2RI_2116
7=FF2FF_2116
8=FF290_2116
9=PRI_377A15
Special Handling: None

Name: ResetPrompt
Description: Enable or disable the reset prompt, which, if enabled, will pop up an "Are you sure?" message whenever the RESET key is pressed.
Tag Name: TAG_RESET_PROMPT
Tag ID: 0x52503031
Allowed Values: 0 = Off, 1 = On
Special Handling: None

Name: StartupTab
Description: ID of tab to display at startup.
Tag Name: TAG_STARTUP_TAB
Tag ID: 0x53545431
Allowed Values: 9106 = Live
9107 = Overall
9108 = SessionLog
9109 = Current
9110 = Measurement
9111 = Events
9112 = TimeHistory
Special Handling: None

Name: Standard Electrostatic Actuator Level
Description: The level in dB for the Cal-Check.
Tag Name: TAG_STD_EA_LEVEL
Tag ID: 0x53544541
Special Handling: None

Name: TaktMetricEnable
Description: Enable Takt Maximal data.
Tag Name: TAG_LTM5_ENABLE
Tag ID: 0x4C544D35
Allowed Values: 0 = Off, 1 = On
Special Handling: RESET_REQ

Name: ThirdOctaveReferenceSpectra1
Description: Set third octave (1/3) reference spectra 1.
Tag Name: TAG_THIRD_OCTAVE_REFERENCE_SPECTRA1
Tag ID: 0x54525331
Type: floatSeries
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 36 floats

Name: ThirdOctaveReferenceSpectra2
Description: Set third octave (1/3) reference spectra 2.
Tag Name: TAG_THIRD_OCTAVE_REFERENCE_SPECTRA2
Tag ID: 0x54525332
Type: floatSeries
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 36 floats

Name: ThirdOctaveReferenceSpectra3
Description: Set third octave (1/3) reference spectra 3.
Tag Name: TAG_THIRD_OCTAVE_REFERENCE_SPECTRA3
Tag ID: 0x54525333
Type: floatSeries
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 36 floats

Name: ThirdOctaveReferenceSpectra4
Description: Set third octave (1/3) reference spectra 4.
Tag Name: TAG_THIRD_OCTAVE_REFERENCE_SPECTRA4
Tag ID: 0x54525334
Type: floatSeries
Allowed Values: Min: -20.0, Max: 140.0
Special Handling: Index 36 floats

Name: USBStorage
Description: Enable data storage to USB thumbdrive.
Tag Name: TAG_USB_STORAGE
Tag ID: 0x55534231
Type: int
Allowed Values: 0 = Internal storage, or 2 = USB Primary
Special Handling: None

Email Settings (831C Only)

Email alert System Properties that allow you to set up the email options for when to send out emails.

Name: SMTP_OPTIONS
Type: UInt

Tag Name: TAG_SMTP_OPTIONS
Tag ID: 0x454D4C38
Tag String: 'EML8'
Allowed Values: MinVal: 0, MaxVal: 7
Index: 0

Name: EMAIL_ALERT_CONTROL
Description: Used to flag whether or not to send Text or Email for the various Alerts
Type: UInt
Tag Name: TAG_EMAIL_ALERT_CONTROL
Tag ID: 0x454D4130
Tag String: 'EMA0'
Allowed Values: MinVal: 0, MaxVal: 16384 – bitmask taken from the following definitions

```
var EmailAlertNotification =
{
ALERT_OFF: 0,          // Do not send any system health emails
ALERT_SETTINGS_HTML: (1 << 0), // Send Settings Changed HTML formatted email
ALERT_SETTINGS_TEXT: (1 << 1), // Send Settings Changed Text formatted email
ALERT_MEMORY_HTML: (1 << 2), // Send Memory Status HTML formatted email
ALERT_MEMORY_TEXT: (1 << 3), // Send Memory Status Text formatted email
ALERT_TEMPERATURE_HTML: (1 << 4), // Send Temperature HTML formatted email
ALERT_TEMPERATURE_TEXT: (1 << 5), // Send Temperature Text formatted email
ALERT_POWER_HTML: (1 << 6), // Send Power Status HTML formatted email
ALERT_POWER_TEXT: (1 << 7), // Send Power Status Text formatted email
ALERT_SLM_STATE_HTML: (1 << 8), // Send SLM State HTML formatted email
ALERT_SLM_STATE_TEXT: (1 << 9), // Send SLM State Text formatted email
ALERT_SECURITY_HTML: (1 << 10), // Send Security HTML formatted email
ALERT_SECURITY_TEXT: (1 << 11), // Send Security Text formatted email
ALERT_CERT_HTML: (1 << 12), // Certification reminder HTML formatted email
ALERT_CERT_TEXT: (1 << 13), // Certification reminder Text formatted email
}
```

Static IP Address (831C Only)

Only valid when communicating through USB.

Name: STATIC_IP_ENABLE
Type: Int
Tag Name: TAG_STATIC_IP_ENABLE
Tag ID: 0x53495045
Tag String: 'SIPE'
Allowed Values: MinVal: 0, MaxVal: 1
Special Handling:

Name: STATIC_IP_ADDRESS
Type: String
Tag Name: TAG_STATIC_IP_ADDRESS
Tag ID: 0x53495041
Tag String: 'SIPA'
Allowed Values: MinVal: 0, MaxVal: 16
Special Handling:

Name: STATIC_IP_MASK
Type: String
Tag Name: TAG_STATIC_IP_MASK
Tag ID: 0x5349504D
Tag String: 'SIPM'
Allowed Values: MinVal: 0, MaxVal: 16
Special Handling:

Name: STATIC_IP_GATEWAY
Type: String
Tag Name: TAG_STATIC_IP_GATEWAY
Tag ID: 0x53495047
Tag String: 'SIPG'
Allowed Values: MinVal: 0, MaxVal: 16
Special Handling:

4.5.0 Scheduling and Security

You will need 4.5.0 firmware or greater on your 831C in order to use these properties. Some of these properties could lock you out of your system remotely. Please use them carefully.

Name: SCHEDULING_ENABLE
Type: Int
Tag Name: TAG_SCHEDULING_ENABLE
Tag ID: 0x53434845
Allowed Values: MinVal: 0, MaxVal: 15

Security Authentication required

Name: SECURITY_ENABLE
Type: Int
Tag Name: TAG_SECURITY_ENABLE
Tag ID: 0x53433030
Allowed Values: MinVal: 0, MaxVal: 3
0=no auth; 1=ssl_req; 2=no_guest; 3=with_guest
Sets the level of security on the meter and can lock you out if set. You need to add an administrator to the meter as a user to ensure that the security works correctly. You will not want to set this level without an admin.

Name: SECURITY_TIMEOUT
Type: Int
Tag Name: TAG_SECURITY_TIMEOUT
Tag ID: 0x53433031
Allowed Values: MinVal: 60, MaxVal: 3600
How long the session will remain active after the last request.

RV50 Modem power Control
Name: RV50_LOW_POWER_ENABLE
Type: Int
Tag Name: TAG_RV50_LOW_POWER_ENABLE
Tag ID: 0x52563031
Allowed Values: MinVal: 0, MaxVal: 1

Name: RV50_ENA_TIME
Type: Int
Tag Name: TAG_RV50_ENA_TIME
Tag ID: 0x52563032
Allowed Values: MinVal: 0, MaxVal: 86399

Name: RV50_ENA_DURATION
Type: Int
Tag Name: TAG_RV50_ENA_DURATION
Tag ID: 0x52563033
Allowed Values: MinVal: 0, MaxVal: 86400/60 or 1440

Name: RV50_ENA_ALERT_TIMEOUT
Type: Int
Tag Name: TAG_RV50_ENA_ALERT_TIMEOUT
Tag ID: 0x52563034
Allowed Values: MinVal: 0, MaxVal: 86400/60 or 1440

4.6.0 System Properties

FFT Properties
Name: FFT_CURSOR_TYPE
Type: Int
Tag Name: TAG_FFT_CURSOR_TYPE
Tag ID: 0x46464354
Allowed Values: MinVal: 0, MaxVal: 1

Name: FFT_NUM_HARMONICS
Type: Int
Tag Name: TAG_FFT_NUM_HARMONICS
Tag ID: 0x46464E48
Allowed Values: MinVal: 0, MaxVal: 24

Name: FFT_GRAPH_BOTTOM
Type: Int
Tag Name: TAG_FFT_GRAPH_BOTTOM
Tag ID: 0x46464742
Allowed Values: MinVal: -20, MaxVal: 200

Name: FFT_GRAPH_HEIGHT
Type: Int
Tag Name: TAG_FFT_GRAPH_HEIGHT
Tag ID: 0x46464748
Allowed Values: MinVal: 20, MaxVal: 140

Name: FFT_X_AXIS_UNITS
Type: Int
Tag Name: TAG_FFT_X_AXIS_UNITS
Tag ID: 0x46465855
Allowed Values: MinVal: 0, MaxVal: 1

Name: FFT_Y_AXIS_SCALE
Type: Int
Tag Name: TAG_FFT_Y_AXIS_SCALE
Tag ID: 0x46465955
Allowed Values: MinVal: 0, MaxVal: 1

Transducer Units

Name: FFT_USER_XDUCER_UNITS
Type: Int
Tag Name: TAG_FFT_USER_XDUCER_UNITS
Tag ID: 0x46465555
Allowed Values: MinVal: 0, MaxVal: 0x00030002

Name: XDUCER_ENABLE
Type: UInt
Tag Name: TAG_XDUCER_ENABLE
Tag ID: 0x58454E41
Allowed Values: MinVal: 0, MaxVal: 1
Special Handling: RESET_REQ

Name: XDUCER_UNITS
Type: UInt
Tag Name: TAG_XDUCER_UNITS
Tag ID: 0x58554E49
Allowed Values: MinVal: 0, MaxVal: 0x30002
Special Handling: RESET_REQ

Name: XDUCER_UNITS_CUSTOM
Type: String
Tag Name: TAG_XDUCER_UNITS_CUSTOM
Tag ID: 0x5843544D
Allowed Values: Valid string representing the custom units like: mm, cm, in
Special Handling: RESET_REQ

Name: XDUCER_SENSITIVITY
Type: Float

Tag Name: TAG_XDUCER_SENSITIVITY
Tag ID: 0x5853454E
Allowed Values: MinVal: 0.001, MaxVal: 9999.9
Special Handling: RESET_REQ

E. Measurement Property Defaults (831C)

This section lists the default values for Measurement Properties.

```
// AlarmDateTime (Deprecated)
#define TAG_DEF_ALARM                              (0u)

// OverallTitle
#define TAG_DEF_OVERALL_TITLE                      ""

// StableTimer
#define TAG_DEF_STABLE_TIMER                       (20u)

// RunTimer
#define TAG_DEF_RUN_TIMER                          (20u)

// RunMode
#define TAG_DEF_RUN_MODE                           (RUNMODE_MANUAL)

// StableDB
#define TAG_DEF_STABLE_DB                          (0.2f)

// MeasurementHistoryEnable
#define TAG_DEF_MEASUREMENT_HISTORY_ENA            (CTRL_OFF)

// MeasurementCounter
#define TAG_DEF_MEASUREMENT_COUNTER                (1u)

// IntervalTime
#define TAG_DEF_INTERVAL_TIME                      (3600u)

// ClockSync
#define TAG_DEF_INTERVAL_CLOCK_SYNC                (CTRL_OFF)

// DailyAutoStore
#define TAG_DEF_DAILY_AUTOSTORE                    (DAS_NEVER)

// AutoStoreTime
#define TAG_DEF_DAILY_AUTOSTORE_TIME               (0u)

// AutoCalCheck
#define TAG_DEF_DAILY_CAL_CHECK                    (CTRL_OFF)

// AutoCalCheckTime
#define TAG_DEF_DAILY_CAL_CHECK_TIME               (9000u)
///< 2:30 am, the quietest time of the day

// StartDate
#define TAG_DEF_BLOCK_TIMER_START_DATE             (DATEINT - (DATEINT %
SECONDS_IN_YEAR))
///< Jan 01 of firmware build year

// EndDate
#define TAG_DEF_BLOCK_TIMER_STOP_DATE              ((SECONDS_IN_YEAR * 5) +
TAG_BLOCK_TIMER_START_DATE)
```

```
// RunTime1
#define TAG_DEF_RUN_TIME_1                    (28800u)
///< 8 AM

// StopTime1
#define TAG_DEF_STOP_TIME_1                   (43200u)
///< Noon

// EnableTimer2
#define TAG_DEF_ENABLE_TIMER_2                (CTRL_OFF)

// RunTime2
#define TAG_DEF_RUN_TIME_2                    (46800u)
///< 1 PM

// StopTime2
#define TAG_DEF_STOP_TIME_2                   (61200u)
///< 5 PM

// EnableTimer3
#define TAG_DEF_ENABLE_TIMER_3                (CTRL_OFF)

// RunTime3
#define TAG_DEF_RUN_TIME_3                    (64800u)
///< 6 PM

// StopTime3
#define TAG_DEF_STOP_TIME_3                   (82800u)
///< 11 PM

// NumBlockTimers
#define TAG_DEF_NUM_BLOCK_TIMERS              (0u)
///< Enable only timer 1

// LnTableEnable
#define TAG_DEF_LN_TABLE_ENABLE               (CTRL_ON)

// TimeHistoryEnable
#define TAG_DEF_TIME_HISTORY_ENABLE           (CTRL_OFF)

// TimeHistoryPeriod
#define TAG_DEF_TIME_HISTORY_PERIOD           (TH_PERIOD_1S)
///< 1 second (see g_time_history_periods_int)

// TimeHistoryOptions1
#if (ANY_LEVEL)
#define TAG_DEF_TIME_HISTORY_OPTIONS_1        (ID_TH_ALEQ | ID_TH_CLEQ
| ID_TH_CPEAK)
#else
#define TAG_DEF_TIME_HISTORY_OPTIONS_1        (ID_TH_LEQ | ID_TH_LPEAK)
#endif // ANY_LEVEL
```

```
// TimeHistoryOptions2
#if (ANY_LEVEL)
#define TAG_DEF_TIME_HISTORY_OPTIONS_2         (ID_TH_ZPEAK)
#else
#define TAG_DEF_TIME_HISTORY_OPTIONS_2         (0u)
#endif // ANY_LEVEL

// StreamHistoryPeriod
#define TAG_DEF_STREAM_HISTORY_PERIOD          (TH_PERIOD_10PS)
///< 0.1 second period (see m_stream_period_count)

// StreamHistoryOptions
#define TAG_DEF_STREAM_HISTORY_OPTIONS         (0u)

// EventHistoryEnable
#define TAG_DEF_EVENT_HISTORY_ENABLE           (CTRL_OFF)

// EventTimeHistoryEnable
#define TAG_DEF_EVENT_TH_ENABLE                (CTRL_OFF)

// EventSpectralTimeHistoryEnable
#define TAG_DEF_EVENT_TH_SPECTRA_ENABLE        (CTRL_OFF)

// MinimumDuration
#define TAG_DEF_EVENT_MIN_DURATION             (3.0f)
///< Seconds

// TriggerMethod
#define TAG_DEF_EVENT_TRIG_METHOD              (TRIGGER_LEVEL)

// ContinuationPeriod
#define TAG_DEF_EVENT_CONT_PERIOD              (2u)
///< Seconds

// EventTimeHistoryPeriod
#define TAG_DEF_EVENT_TH_PERIOD                (ID_1S)
///< 1 second (see g_time_history_periods_int)

// MarkerNames[NUM_MARKERS]
#define TAG_DEF_MARKER_NAMES_0                "Truck"
#define TAG_DEF_MARKER_NAMES_1                "Automobile"
#define TAG_DEF_MARKER_NAMES_2                "Motorcycle"
#define TAG_DEF_MARKER_NAMES_3                "Aircraft"
#define TAG_DEF_MARKER_NAMES_4                "Exclude"
#define TAG_DEF_MARKER_NAMES_5                "#6"
#define TAG_DEF_MARKER_NAMES_6                "#7"
#define TAG_DEF_MARKER_NAMES_7                "#8"
#define TAG_DEF_MARKER_NAMES_8                "#9"
#define TAG_DEF_MARKER_NAMES_9                "#10"

// MarkerAudioRecord[NUM_MARKERS]
#define TAG_DEF_MARKER_AUDIO_RECORD_0          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_1          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_2          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_3          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_4          (0u)
```

```
#define TAG_DEF_MARKER_AUDIO_RECORD_5          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_6          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_7          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_8          (0u)
#define TAG_DEF_MARKER_AUDIO_RECORD_9          (0u)

// MarkerRecordPeriod
#define TAG_DEF_MARKER_RECORD_PERIOD           (7u)
///< Seconds

// MarkerPreRecordPeriod
#define TAG_DEF_MARKER_PRERECORD_PERIOD        (4u)
///< Seconds

// DayTime
#define TAG_DEF_DAY_TIME                       (7u * 60u * 60u)
///< 07:00

// EveningTime
#define TAG_DEF_EVENING_TIME                   (19u * 60u * 60u)
///< 19:00

// NightTime
#define TAG_DEF_NIGHT_TIME                     (22u * 60u * 60u)
///< 22:00

// EveningPenalty
#define TAG_DEF_EVENING_PENALTY                (5.0f)
///< dB

// NightPenalty
#define TAG_DEF_NIGHT_PENALTY                  (10.0f)
///< dB

// SaveEventAudioSnapshot
#define TAG_DEF_EVENT_SNAPSHOT_ENABLE          (CTRL_OFF)

// EventAudioSnapshotPeriod
#if (EXCEEDANCE)
#define TAG_DEF_EVENT_SNAPSHOT_PERIOD          (7u)
///< Seconds
#else
#define TAG_DEF_EVENT_SNAPSHOT_PERIOD          (0u)
///< Seconds
#endif

// EventAudioSnapshotPreTriggerPeriod
#if (EXCEEDANCE)
#define TAG_DEF_EVENT_PRETRIGGER_PERIOD        (4u)
///< Seconds
#else
#define TAG_DEF_EVENT_PRETRIGGER_PERIOD        (0u)
///< Seconds
#endif

// SaveMsmtAudioSnapshot
```

```
#define TAG_DEF_MEASUREMENT_SNAPSHOT_ENABLE    (CTRL_OFF)

// MsmtAudioSnapshotPeriod
#define TAG_DEF_MEASUREMENT_SNAPSHOT_PERIOD    (5u)
///< Seconds

// UNUSED_01
#define TAG_DEF_UNUSED_01                      (0)

// DefaultFileName
#define TAG_DEF_DEFAULT_FILENAME              "831_Data"

// SlmFrequencyWeighting
#define TAG_DEF_FREQ_WEIGHTING                 (FREQWT_A)

// PeakFrequencyWeighting
#define TAG_DEF_PEAK_WEIGHTING                 (PEAKWT_Z)

// SlmDetector
#define TAG_DEF_DET_WEIGHTING                  (DETECTOR_SLOW)

// SlmGain
#define TAG_DEF_SLM_GAIN                       (GAIN_0)

// IntegrationInput
#define TAG_DEF_INTEGRATION_INPUT              (INTMETHOD_LINEAR)

// HistPeriod
#define TAG_DEF_HIST_PERIOD                    (1)
// Control "HistPeriod" is unused

// Percentiles
#define TAG_DEF_LN_0                           (5.0f)
#define TAG_DEF_LN_1                           (10.0f)
#define TAG_DEF_LN_2                           (33.3f)
#define TAG_DEF_LN_3                           (50.0f)
#define TAG_DEF_LN_4                           (66.6f)
#define TAG_DEF_LN_5                           (90.0f)

// ObaRange
#define TAG_DEF_OBA_RANGE                      (OBA_RANGE_LOW)

// ObaBandwidth
#define TAG_DEF_OBA_BANDWIDTH                  (OBA_BW_THIRD_OCTAVE)

// ObaMaxSpectrumMode
#define TAG_DEF_OBA_MAX_MODE                   (OBA_BINMAX)

// ObaWeighting
#define TAG_DEF_OBA_WEIGHTING                  (OBA_FREQWTZ)

// DoseName[NUM_DOSE_CHANNELS]
#define TAG_DEF_DOSE_NAME_0                   "OSHA-1"
#define TAG_DEF_DOSE_NAME_1                   "OSHA-2"

// DoseThreshold[NUM_DOSE_CHANNELS]
```

```
#define TAG_DEF_DOSE_THRESHOLD_0               (90.0f)
///< dB
#define TAG_DEF_DOSE_THRESHOLD_1               (80.0f)
///< dB

// DoseThresholdEnable[NUM_DOSE_CHANNELS]
#define TAG_DEF_DOSE_THRESHOLD_ENABLE_0        (CTRL_ON)
#define TAG_DEF_DOSE_THRESHOLD_ENABLE_1        (CTRL_ON)

// DoseExRate[NUM_DOSE_CHANNELS]
#define TAG_DEF_DOSE_EXCHANGE_RATE_0           (dose_ex_rate_5dB)
#define TAG_DEF_DOSE_EXCHANGE_RATE_1           (dose_ex_rate_5dB)

// DoseCriterionLevel[NUM_DOSE_CHANNELS]
#define TAG_DEF_DOSE_C_LEVEL_0                 (90.0f)
///< dB
#define TAG_DEF_DOSE_C_LEVEL_1                 (90.0f)
///< dB

// DoseCriterionTime[NUM_DOSE_CHANNELS]
#define TAG_DEF_DOSE_C_TIME_0                  (8.0f)
///< Hours
#define TAG_DEF_DOSE_C_TIME_1                  (8.0f)
///< Hours

// EventWeighting[NUM_EVENT_COUNTERS]
#define TAG_DEF_EVENT_FREQ_WEIGHTING_SPL_0     (FREQWT_A)
#define TAG_DEF_EVENT_FREQ_WEIGHTING_SPL_1     (FREQWT_A)
#define TAG_DEF_EVENT_FREQ_WEIGHTING_PK_0      (FREQWT_A)
#define TAG_DEF_EVENT_FREQ_WEIGHTING_PK_1      (FREQWT_A)
#define TAG_DEF_EVENT_FREQ_WEIGHTING_PK_2      (FREQWT_A)

// EventDetector[NUM_EVENT_COUNTERS]
#define TAG_DEF_EVENT_DET_WEIGHTING_SPL_0      (0)
///< *** NOT USED
#define TAG_DEF_EVENT_DET_WEIGHTING_SPL_1      (0)
///< *** NOT USED
#define TAG_DEF_EVENT_DET_WEIGHTING_PK_0       (0)
///< *** NOT USED
#define TAG_DEF_EVENT_DET_WEIGHTING_PK_1       (0)
///< *** NOT USED
#define TAG_DEF_EVENT_DET_WEIGHTING_PK_2       (0)
///< *** NOT USED

// EventLevel[NUM_EVENT_COUNTERS]
#define TAG_DEF_EVENT_LEVEL_SPL_0              ( 65.0f)
#define TAG_DEF_EVENT_LEVEL_SPL_1              ( 85.0f)
#define TAG_DEF_EVENT_LEVEL_PK_0               (135.0f)
#define TAG_DEF_EVENT_LEVEL_PK_1               (137.0f)
#define TAG_DEF_EVENT_LEVEL_PK_2               (140.0f)

// DynamicTriggerOffsetLevel
#define TAG_DEF_EVENT_DYN_TRIG_OFFSET          (20.0f)

// DynamicResponse
#define TAG_DEF_EVENT_DYNAMIC_RESPONSE         (3)
```

```
// WindScale
#define TAG_DEF_WIND_SCALE                      (0.21920f)

// WindUnits
#define TAG_DEF_WIND_UNITS                      "mi/h"

// WindDirection
#define TAG_DEF_WIND_DIRECTION                  (ID_WIND_COMPASS)

// WindThreshold
#define TAG_DEF_WIND_THRESHOLD                  (5.0f)

// WindExcdLevel
#define TAG_DEF_WIND_EXCD_LEVEL                 (40.0f)

// WindHysteresis
#define TAG_DEF_WIND_HYSTERESIS                 (10.0f)

// WindPause
#define TAG_DEF_WIND_PAUSE                      (CTRL_OFF)

// ADC1Scale
#define TAG_DEF_ADC1_SCALE                      (184.32f)
///< default for -40 to +144.32 degree F sensor

// ADC1Offset
#define TAG_DEF_ADC1_OFFSET                     (-40.0f)

// ADC1Units
#define TAG_DEF_ADC1_UNITS                      "F"

// ADC1Description
#define TAG_DEF_ADC1_DESCRIPTION                "Temperature"

// ADC2Scale
#define TAG_DEF_ADC2_SCALE                      (102.4f)
///< default for 0 to 102.4% RH

// ADC2Offset
#define TAG_DEF_ADC2_OFFSET                     (0.00f)

// ADC2Units
#define TAG_DEF_ADC2_UNITS                      ("% RH")

// ADC2Description
#define TAG_DEF_ADC2_DESCRIPTION                "Humidity"

// SoundRecSampleRate
#define TAG_DEF_SNAPSHOT_SAMPLING_RATE          (SR_SAMPLERATE8)

// TimeHistoryOptions3
#define TAG_DEF_TIME_HISTORY_OPTIONS_3          (ID_TH_DUR_FRAC)

// SpectralLnMode
#define TAG_DEF_SPECTRAL_LN                     (SPECLN_OFF)
```

```
// EventTHCount
#define TAG_DEF_EVENT_TH_MAX_SAMPLES          (1000u)

// EventTHPreCount
#define TAG_DEF_EVENT_TH_PRE_TRIG             (10u)

// SRRange
#define TAG_DEF_SR_RANGE                      (SRRANGE_LOW)

// GraphRelative
#define TAG_DEF_GRAPH_RELATIVE                (CTRL_OFF)

// EventTHPostCount
#define TAG_DEF_EVENT_TH_POST_TRIG            (10u)

// EnableWeather
#define TAG_DEF_WEATHER_ENABLE                (ID_WEATHER_NONE)

// InstrumentMode
#define TAG_DEF_INSTRUMENT_MODE               (MODE_SLM)

// ExitTime
#define TAG_DEF_RA_EXIT_TIME                  (10)
///< Seconds

// LowestFrequency
#define TAG_DEF_RA_LOWEST_FREQ                (MIN_FILTER_11)

// HighestFrequency
#define TAG_DEF_RA_HIGHEST_FREQ               (MAX_FILTER_11)

// LinkSourceToControl
#define TAG_DEF_TIE_SOURCE_TO_CONTROL         (LINK_TO_NONE)

// RT60_NoiseType
#define TAG_DEF_RT60_NOISE_TYPE               (NOISE_NONE)

// UNUSED_02
#define TAG_DEF_UNUSED_02                     (0)

// RT60_NoiseAttenuation
#define TAG_DEF_RT60_NOISE_ATTENUATION        (0.0f)

// RT60_Method
#define TAG_DEF_RT60_METHOD                   (RT60_METHOD_IMPULSE)

// RT60_TriggerSource
#define TAG_DEF_RT60_TRIGGER_SOURCE           (F_4000)

// RT60_TriggerLevel
#define TAG_DEF_RT60_TRIGGER_LEVEL            (80.0f)
///< dB

// RT60_Bandwidth
#define TAG_DEF_RT60_BANDWIDTH                (OBA_BW_OCTAVE)
```

```
// RT60_BuildTime
#define TAG_DEF_RT60_BUILD_TIME                 (2)
///< Seconds

// RT60_RunTime
#define TAG_DEF_RT60_RUN_TIME                   (4)
///< Seconds

// RT60_RunCount
#define TAG_DEF_RT60_RUN_COUNT                  (1)

// RT60_ByTimeSamplePeriod
#define TAG_DEF_RT60_SAMPLE_PERIOD              (RT60_SAMPLE_PERIOD_5_MS)

// RT60_SaveAverageTimeSeries
#define TAG_DEF_SAVE_AVERAGE_TIME_SERIES        (CTRL_ON)

// RT60_SaveAllTimeSeries
#define TAG_DEF_SAVE_ALL_TIME_SERIES            (CTRL_ON)

// FFT_WindowType
#define TAG_DEF_FFT_WINDOW_TYPE                 (WIN_HANNING)

// FFT_FreqSpan
#define TAG_DEF_FFT_FREQSPAN                    (BW_20K)

// FFT_NumLines
#define TAG_DEF_FFT_NUM_LINES                   (LINES_6400)

// FFT_RunMode
#define TAG_DEF_FFT_RUN_MODE                    (FFT_MANUAL_STOP)

// Tonality_1996_2
#define TAG_DEF_TONE_1996_2                     (CTRL_OFF)

// ToneSeekDelta
#define TAG_DEF_TONE_SEEK_DELTA                 (DB_1)
///< 1 dB

// RegressionSpan
#define TAG_DEF_REGRESSION_SPAN                 (SPAN_75)
///< +/- 75%

// VaisalaHeaterState
#define TAG_DEF_VHEAT_STATE                     (HEATER_ENA)

// VaisalaHeaterEnaTime
#define TAG_DEF_VHEAT_ENA_TIME                  (43200u)
///< 12:00 (noon)

// VaisalaHeaterDisTime
#define TAG_DEF_VHEAT_DIS_TIME                  (46800u)
///< 13:00

// AudioCompression
```

```
#define TAG_DEF_AUDIO_COMPRESSION             (CTRL_OFF)

// EventNotification
#define TAG_DEF_EVENT_NOTIFICATION            (NOTIFY_OFF)

// SMTPEnable
#define TAG_DEF_SMTP_ENABLE                   (CTRL_OFF)
```

F. Request Time History

With the use of the SDK you may now request Time History data from the meter during a run. Using the SDK call to obtain the first 120 entries using index=0:

/sdk?func=timehistjson&group=time&metric=1&index=0

You will receive data similar to the following:

```
{ "thLeq":[-448.535,
29.5812,32.3678,30.1272,29.4511,30.4587,30.3081,31.655,37.2697,29.3061,
30.1012,29.714,29.7077,29.3336,29.2633,29.2923,29.0833,29.1234,29.3179,
29.6077,29.6463,29.386,29.6642,30.2315,29.9971,30.1576,29.6707,29.2993,
29.3808,29.7397,29.4757,29.2464,29.2979,29.0867,29.2615,31.5266,29.3122
,29.0952,28.9625,28.7053,28.9837,31.6638,31.6003,29.2019,29.2885,29.052
3,29.1137,29.1724,29.4513,29.058,29.1468,29.1526,29.0469,29.1198,29.286
1,29.1281,29.4697,29.1244,29.3935,29.3841,29.0686,29.1948,29.8057,28.88
17,28.9994,29.1419,29.2544,29.1981,29.3095,29.873,34.5869,31.2319,31.91
27,29.7356,29.4183,29.4456,30.5363,29.0682,29.0811,29.2183,29.1999,29.0
677,30.5028,33.0639,29.2716,28.9886,28.9953,29.0321,29.5134,29.7697,29.
279,29.1794,29.0362,29.657,29.6611,29.1339,32.6172,40.1821,43.1842,40.2
935,29.264,29.062,29.2707,29.7054,29.2747,29.0494,29.2564,40.4,29.8031,
41.0553,39.1903,35.8073,33.5164,39.4362,28.8922,35.9437,33.7467,29.3648
,29.1172,29.078],

"thMetrics":[-1e+12,
30.5462,34.1595,31.5015,30.2019,31.4555,31.6151,35.3168,41.9486,30.5075
,31.2145,30.8437,30.5442,29.8954,29.4887,29.66,29.3351,29.4749,29.6527,
29.9383,29.9621,29.9621,29.9289,30.8826,30.4007,30.8308,30.2732,29.6122
,29.8146,31.5467,29.8932,29.5392,29.8149,29.4545,29.9245,36.3694,33.062
2,29.5615,29.1198,29.3978,29.477,36.5639,36.4588,32.5068,29.8758,29.273
,29.421,29.4998,30.0999,29.3878,29.4353,29.6863,29.5447,29.3848,29.7719
,29.4741,31.0145,29.3863,29.7131,30.482,29.5234,29.4345,31.5408,30.9489
,29.255,29.4747,29.6142,29.6397,29.8622,31.6811,37.3492,33.7302,33.5459
,33.0956,29.9533,30.0892,32.242,30.1973,29.6235,29.5016,29.627,29.3825,
33.3443,37.1244,29.7552,29.4064,29.441,29.3816,30.8179,30.6801,29.7126,
29.7596,29.3286,30.3273,30.1711,29.6068,35.7212,45.8777,47.4929,45.896,
30.6949,29.4513,29.5762,30.659,29.6591,29.4517,29.788,43.3341,36.2022,4
5.1855,44.6324,42.1437,38.6118,45.1543,31.0692,39.122,37.3118,30.8909,2
9.3443,29.4667],

"thFlags":[2147483648,8192,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

"thTime":[1588007943,1588007943,1588007944,1588007945,1588007946,158800
7947,1588007948,1588007949,1588007950,1588007951,1588007952,1588007953,
1588007954,1588007955,1588007956,1588007957,1588007958,1588007959,15880
07960,1588007961,1588007962,1588007963,1588007964,1588007965,1588007966
,1588007967,1588007968,1588007969,1588007970,1588007971,1588007972,1588
007973,1588007974,1588007975,1588007976,1588007977,1588007978,158800797
9,1588007980,1588007981,1588007982,1588007983,1588007984,1588007985,158
8007986,1588007987,1588007988,1588007989,1588007990,1588007991,15880079
92,1588007993,1588007994,1588007995,1588007996,1588007997,1588007998,15
88007999,1588008000,1588008001,1588008002,1588008003,1588008004,1588008
005,1588008006,1588008007,1588008008,1588008009,1588008010,1588008011,1
588008012,1588008013,1588008014,1588008015,1588008016,1588008017,158800
8018,1588008019,1588008020,1588008021,1588008022,1588008023,1588008024,
1588008025,1588008026,1588008027,1588008028,1588008029,1588008030,15880
08031,1588008032,1588008033,1588008034,1588008035,1588008036,1588008037
,1588008038,1588008039,1588008040,1588008041,1588008042,1588008043,1588
008044,1588008045,1588008046,1588008047,1588008048,1588008049,158800805
0,1588008051,1588008052,1588008053,1588008054,1588008055,1588008056,158
8008057,1588008058,1588008059,1588008060,1588008061],

"thAction":[514,1063675496,1065353217,1065353217,1065353217,1065353217,
1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,10653
53217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217
,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065
353217,1065353217,1065353217,1065353217,1065353217,1065353217,106535321
7,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,106
5353217,1065353217,1065353217,1065353217,1065353217,1065353217,10653532
17,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,10
65353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353
217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1
065353217,1065353217,1065353217,1065353217,1065353217,1065353217,106535
3217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,
1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,10653
53217,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217
,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,1065
353217,1065353217,1065353217,1065353217,1065353217,1065353217,106535321
7,1065353217,1065353217,1065353217,1065353217,1065353217,1065353217,106
5353217,1065353217,1065353217,1065353217,1065353217,1065353217,10653532
17,1065353217,1065353217,1065353217,1065353217],"Result":"Success: 0
","ResultCode":0,"ResultName":"Success" }
```

Changing the index will allow you to query any of the time history data collected thus far in the measurement. Once the Meter has stored the current run, the Time History for that run will no longer be available. You will be most interested in the thLeq, the thMetrics and the thTime entries.

Return Array Values Definitions

thLeq:

Values in dB of the defined Leq for Time History

Exception values less than -99 are invalid (usually means that it is a run, stop or some other record.)

thMetrics:

Values in dB representing the selected metric (in this case 1, the first metric after the Leq). If Time History is configured with only LAFmax then this will be 1. See G Time History Options

Exception values less than -99 are invalid (usually means that it is a run, stop or some other record.)

thFlags:

See the Functional description below, Flags and Their Use

Exception: The value 0 represents no flag.

If Flags = `2147483648 = 0x80000000` then refer to the thAction otherwise ignore thAction

In the case 514 = 0x0202 => Run

`8192` => Time History Partial Record, means the record did not align with the time history period (if the time history period is one minute then this would appear for the first and last record of a run if you did not start and end them exactly on a one minute boundary)

Most all other values hold no meaning.

thActions:

See below description below, Actions

thTime:

Seconds since epoch:

`1588007943 =>` **GMT**: Monday, April 27, 2020 5:19:03 PM

Flags and Their Use

Functional descriptions of flags. The function shows the ones most used or seen during standard environmental monitoring.

```
var TIMEHIST_MARKER1 = (0x00000001);
var TIMEHIST_MARKER2 = (0x00000002);
var TIMEHIST_MARKER3 = (0x00000004);
var TIMEHIST_MARKER4 = (0x00000008);
var TIMEHIST_MARKER5 = (0x00000010);
var TIMEHIST_MARKER6 = (0x00000020);
var TIMEHIST_MARKER7 = (0x00000040);
var TIMEHIST_MARKER8 = (0x00000080);
var TIMEHIST_MARKER9 = (0x00000100);
var TIMEHIST_MARKER10 = (0x00000200);
var TIMEHIST_OVERLOAD = (0x00000400);
var TIMEHIST_OBA_OVLD = (0x00000800);
var TIMEHIST_EXCDED = (0x00001000);
var TIMEHIST_T2READY = (0x00002000);
var TIMEHIST_PARTIAL = (0x00002000);
var TIMEHIST_SR = (0x00004000);
var TIMEHIST_BACKERASE = (0x00008000);
var TIMEHIST_MANUAL = (0x00010000);
var TIMEHIST_SESSION_LOG = (0x80000000);
var TIMEHIST_MARKER_MASK = (TIMEHIST_MARKER1 | TIMEHIST_MARKER2 | TIMEHIST_MAR
KER3 |
 TIMEHIST_MARKER4 | TIMEHIST_MARKER5 | TIMEHIST_MARKER6 | TIMEHIST_MARKER7 |

TIMEHIST_MARKER8 | TIMEHIST_MARKER9 | TIMEHIST_MARKER10 | TIMEHIST_BACKERASE |
 TIMEHIST_MANUAL);

function transFlag(flag, noOBA)
{
        var strDetail = "";
        if (flag & TIMEHIST_OVERLOAD) {
                 strDetail = "Overload";
        }
        if ((flag & TIMEHIST_OBA_OVLD) && !noOBA) {
                 strDetail = "IDS_OBA_OVLD";
        }
        if (flag & TIMEHIST_EXCDED) {
                 strDetail = "Exceeded";
        }
        if (flag & TIMEHIST_T2READY) {
                 strDetail = "IDS_T2READY";
        }
        if (flag & TIMEHIST_PARTIAL) {
                 strDetail = "Partial Record ";
        }
        return strDetail;
}
```

Actions

thActions array contains values for every entry, though most are not used. If the thFlags contains 0x80000000, then the Action will have a meaning as described below.

```
function actionCause(action) {
        var retVal = false;
        for (var i = 0; i < 8; ++i) {
                 topElems[i].html("&nbsp;");
        }
        if (lastCount > 0) {
                 var act = (action & 0x00FF).toString();
                 var cause = (action & 0xFF00).toString();
                 var actVal = Sess.Actions[act]
                 if (actVal !== undefined) {
                  topElems[0].html("<img src='/images/firmware/" + actVal.i +
".png' class='hdrImages'/> &nbsp;" + actVal.n);
                         topElems[2].html(Sess.Causes[cause]);
                         retVal = true;
                 }
        }
        return retVal;
}
```

```
        var Actions = {};
        Actions["0"] = { n: "IDS_ERROR", i: "warning" };
                      //      // Err action performed
        Actions["1"] = { n: "IDS_STOP", i: "bmp_stop" };
                      //var PRM_ACTION_STOP = (0x0001);     // Stop action perf
ormed
        Actions["2"] = { n: "IDS_RUN", i: "bmp_run1" };
                      //var PRM_ACTION_RUN = (0x0002);     // Run
        Actions["4"] = { n: "IDS_PAUSE", i: "bmp_pause" };
                      //var PRM_ACTION_PAUSE = (0x0004);     // Pause
        Actions["8"] = { n: "IDS_RESUME", i: "bmp_runnext" };
                      //var PRM_ACTION_RESUME = (0x0008);     // Resume from Pa
use
        Actions["16"] = { n: "IDS_VOICE", i: "audio" };
                //var PRM_ACTION_VOICE = (0x0010);     // Voice Recording
        Actions["32"] = { n: "IDS_SOUND", i: "audio" };
                //var PRM_ACTION_AUDIO = (0x0020);     // Audio Recording {reserv
ed for future}
        Actions["64"] = { n: "IDS_CALCHECK", i: "bmp_cal" };
     //var PRM_ACTION_CAL = (0x0040);     // Calibration record, need deviation
 stored also! {reserved for future}
        Actions["128"] = { n: "IDS_RESET", i: "bmp_stopreset" };
                      //var PRM_ACTION_RESET = (0x0080);     // Reset {reserved
 for future}
```

```
        Actions["129"] = { n: "IDS_GPSTIMESYNC", i: "gps" };
     //var PRM_ACTION_GPS_SYNC = (0x0081);    // GPS Time Sync
        Actions["130"] = { n: "IDS_BACKERASE", i: "back_erase" };
              //var PRM_BACK_ERASE = (0x0082);    // Back Erase Session Log
        Actions["131"] = { n: "IDS_MARK", i: "x_mark" };
                      //var PRM_ACTION_MARKER = (0x0083);    // Marker set
        Actions["132"] = { n: "IDS_CALCHANGE", i: "bmp_cal" };
              //var PRM_ACTION_CALCHG = (0x0084);    // Calibration Change Per
formed
        Actions["133"] = { n: "IDS_PREAMP_DISCONNECT", i: "warning" };     //v
ar PRM_ACTION_PREAMPOFF = (0x0085);    // Preamp Removed =(show type of preamp
 removed);
        Actions["134"] = { n: "IDS_PREAMP_CONNECT", i: "warning" };
     //var PRM_ACTION_PREAMPON = (0x0086);    // Preamp Connected =(show type
of preamp connected);
        Actions["135"] = { n: "IDS_CREATED_AVERAGE", i: "bmp_CreateAvg" };
     //var PRM_ACTION_FILE_AVG = (0x0087);    // Create a new average or add t
o average in data explorer
        Actions["136"] = { n: "IDS_COMMS_WATCHDOG", i: "warning" };
     //var PRM_ACTION_WATCHDOG = (0x0088);    // Phoenix watchdog reset
        Actions["137"] = { n: "USB", i: "bmp_usb" };
                               //var PRM_ACTION_USB_FAULT = (0x0089);    // US
B controller fault detected
        Actions["138"] = { n: "Panic", i: "warning" };
                               //var PRM_ACTION_PANIC = (0x008a);    // Panic
restart detected
        Actions["139"] = { n: "IDS_HLD_CHG_FAULT", i: "warning" };
                                       //var PRM_ACTION_PANIC = (0x008a);    /
/ Panic restart detected
        Actions["140"] = { n: "IDS_HLD_NTP_SYNC", i: "ntpSync" };
                                       //var PRM_ACTION_NTP_SYNC            (0
x008C)
        Actions["141"] = { n: "IDS_HLD_TIMEADJUST", i: "clock" };
                                       //var PRM_ACTION_TIME_CHANGE
 (0x008D)

Sess.Actions = Actions;

        var Causes = [];
        Causes["256"] = "IDS_KEY";
         //var PRM_CAUSE_KEY                  = (0x0100);    // Action caused
by keyboard command            // SR Events Cause
        Causes["257"] = "IDS_EVENT";
         //var PRM_CAUSE_KEY                  = (0x0100);    // Action caused
by keyboard command            // SR Events Cause
        Causes["258"] = "IDS_HLD_SYNC_POS";
                 //var PRM_CAUSE_TIME_SYNC_POSITIVE   (0x0100)    // Action ca
used by keyboard command            // SR Events Cause
        Causes["512"] = "IDS_IO";
         //var PRM_CAUSE_IO                   = (0x0200);    // Action caused
by I/O command                 // SR Measurement
        Causes["513"] = "IDS_MEASUREMENT";
                 //var PRM_CAUSE_IO                   = (0x0200);    // Action
 caused by I/O command                 // SR Measurement
```

```
        Causes["514"] = "IDS_HLD_SYNC_NEG";
                 //var PRM_CAUSE_TIME_SYNC_NEGATIVE   (0x0200)    // Action ca
used by keyboard command            // SR Events Cause
        Causes["1024"] = "IDS_TIMER";
//var PRM_CAUSE_TIMER                  = (0x0400);    // Action caused by the ru
n or stable timer     // SR Markers
        Causes["1025"] = "IDS_TABMARKER";                     //var PRM_CAUSE_TIM
ER                 = (0x0401);    // Action caused by the run or stable timer
   // SR Markers

Causes["1280"] = "IDS_HLD_PRM_CAUSE_SCHEDULE";//var PRM_CAUSE_SCHEDULE
     = (0x0500);    // Action caused by schedule
        Causes["2048"] = "IDS_POWER";
//var PRM_CAUSE_POWER                  = (0x0800);    // Stop due to power failu
re
        Causes["4096"] = "IDS_OUTOFMEMORY";                                   //v
ar PRM_CAUSE_MEMORY                 = (0x1000);    // Stop due to out-of-memory
        Causes["8192"] = "IDS_PREAMP_CONNECT";                 //var PRM_CAUSE_P
REAMP_CONNECT        = (0x2000);    // Stop due to preamp connect
        Causes["16384"] = "IDS_PREAMP_DISCONNECT";             //var PRM_CAUSE_P
REAMP_DISCONNECT     = (0x4000);    // Stop due to preamp disconnect
        Causes["32768"] = "IDS_STABLE";
//var PRM_CAUSE_STABLE                 = (0x8000);    // Stop on STABLE
        Causes["33024"] = "IDS_COMMS_WATCHDOG";                //var PRM_CAUSE_8
31_INT_ET            = (0x8100);    // Phoenix communications watchdog via 831
INTET
        Causes["33280"] = "IDS_COMMS_WATCHDOG";                //var PRM_CAUSE_A
NALOGMODEM           = (0x8200);    // Phoenix UsbHost watchdog via Analog Mode
m
        Causes["33536"] = "Other";
         //var PRM_CAUSE_COUNTER_WRAP         = (0x8300);    // uClinux tick c
ounter is a signed 32 bit value that will wrap around in less than 248 days
        Causes["33792"] = "Internal Fault";                                    /
/var PRM_CAUSE_INTERNAL_FAULT        = (0x8400);    // Internal fault
        Causes["34048"] = "IDS_HLD_METER_UPDATE";              //var PRM_CAUSE_M
ETER_UPDATE          = (0x8500);    // Meter update (FW, Options, etc.)
        Causes["34304"] = "IDS_HLD_CHG_FAULT_TEMP";                   //var PRM_
CAUSE_CHG_TEMP_FAULT       (0x8600)    // Charger Temperature fault
        Causes["34560"] = "IDS_HLD_CHG_FAULT_OVERV";         //var PRM_CAUSE_CHG
_OVER_V_FAULT      (0x8700)    // Charger Over Voltage fault
        Causes["34816"] = "IDS_HLD_CHG_FAULT_TYPE";
//var PRM_CAUSE_CHG_BATT_TYPE_FAULT  (0x8800)    // Charger Battery Type fault
        Causes["35072"] = "IDS_HLD_CHG_LOW_PWR";   //var PRM_CAUSE_CHG_NO_CURR
ENT_FAULT  (0x8900)    // Charger can't supply current fault

Sess.Causes = Causes;
```

G. Time History Options

The Time History Options are defined here in the order they would come, if enabled, in the SLMRecord from the translator.

Time History Options

ID: ID_TH_LEQ
Mask: 0x00000001
Name: "L<sub>eq</sub>"
Options: "TAG_TIME_HISTORY_Options"
Condition: function() { return (SystemProperties['TAG_MODEL'].indexOf('LxT') > -1) }

ID: ID_TH_LPEAK
Mask: 0x00000002
Name: "Lpeak"
Options: "TAG_TIME_HISTORY_Options"
Condition: function() { return (SystemProperties['TAG_MODEL'].indexOf('LxT') > -1) }

ID: ID_TH_SPL
Mask: 0x00000004
Name: "SPL"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_LMAX
Mask: 0x00000008
Name: "L<sub>max</sub>"
Options: "TAG_TIME_HISTORY_Options"
Condition: function() { return (SystemProperties['TAG_MODEL'].indexOf('LxT') > -1) }

ID: ID_TH_LMIN
Mask: 0x00000010
Name: "L<sub>min</sub>"
Options: "TAG_TIME_HISTORY_Options"
Condition: function() { return (SystemProperties['TAG_MODEL'].indexOf('LxT') > -1) }

ID: ID_TH_LTM5
Mask: 0x00000020
Name: "L<sub>AFTM5</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_LTWA1
Mask: 0x00000040
Name: "Ltwa1"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_LTWA2
Mask: 0x00000080

Name: "Ltwa2"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_LTWA3
Mask: 0x00000100
Name: "Ltwa3"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_LTWA4
Mask: 0x00000200
Name: "Ltwa4"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_ALEQ
Mask: 0x00000400
Name: "L<sub>A#eq</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_APEAK
Mask: 0x00000800
Name: "L<sub>Apeak</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_ASLOWSPL
Mask: 0x00001000
Name: "L<sub>AS</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_ASLOWMAX
Mask: 0x00002000
Name: "L<sub>ASmax</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_ASLOWMIN
Mask: 0x00004000
Name: "L<sub>ASmin</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_AFASTSPL
Mask: 0x00008000
Name: "L<sub>AF</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_AFASTMAX
Mask: 0x00010000
Name: "L<sub>AFmax</sub>"

Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_AFASTMIN
Mask: 0x00020000
Name: "L<sub>AFmin</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_AIMPLSPL
Mask: 0x00040000
Name: "L<sub>AI</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_AIMPLMAX
Mask: 0x00080000
Name: "L<sub>AImax</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_AIMPLMIN
Mask: 0x00100000
Name: "L<sub>AImin</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CLEQ
Mask: 0x00200000
Name: "L<sub>C#eq</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CPEAK
Mask: 0x00400000
Name: "L<sub>Cpeak</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CSLOWSPL
Mask: 0x00800000
Name: "L<sub>CS</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CSLOWMAX
Mask: 0x01000000
Name: "L<sub>CSmax</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CSLOWMIN
Mask: 0x02000000
Name: "L<sub>CSmin</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CFASTSPL
Mask: 0x04000000
Name: "L<sub>CF</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CFASTMAX
Mask: 0x08000000
Name: "L<sub>CFmax</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CFASTMIN
Mask: 0x10000000
Name: "L<sub>CFmin</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CIMPLSPL
Mask: 0x20000000
Name: "L<sub>CI</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CIMPLMAX
Mask: 0x40000000
Name: "L<sub>CImax</sub>"
Options: "TAG_TIME_HISTORY_Options"

ID: ID_TH_CIMPLMIN
Mask: 0x80000000
Name: "L<sub>CImin</sub>"
Options: "TAG_TIME_HISTORY_Options"

Time History Misc (Options 2)

ID: ID_TH_ZLEQ
Mask: 0x00000001
Name: "L<sub>Z#eq</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZPEAK
Mask: 0x00000002
Name: "L<sub>Zpeak</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZSLOWSPL
Mask: 0x00000004
Name: "L<sub>ZS</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZSLOWMAX
Mask: 0x00000008
Name: "L<sub>ZSmax</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZSLOWMIN
Mask: 0x00000010
Name: "L<sub>ZSmin</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZFASTSPL
Mask: 0x00000020
Name: "L<sub>ZF</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZFASTMAX
Mask: 0x00000040
Name: "L<sub>ZFmax</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZFASTMIN
Mask: 0x00000080
Name: "L<sub>ZFmin</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZIMPLSPL
Mask: 0x00000100
Name: "L<sub>ZI</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZIMPLMAX
Mask: 0x00000200
Name: "L<sub>ZImax</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ZIMPLMIN
Mask: 0x00000400
Name: "L<sub>ZImin</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_BATTERY
Mask: 0x00000800
Name: "Battery"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_EXTPOWER
Mask: 0x00001000

Name: "External Power"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_INTTEMP
Mask: 0x00002000
Name: "Internal Temp."
Options: "TAG_TIME_HISTORY_MISC"

Weather and Internal Units

The meter stores Time History Float Values with the specified units.

Metric	Internal Units	Data
Temperature	°C	Internal Temperature Vaisala Temperature (ADC1) Preamp Temperature
Speed	m/s	Vaisala Wind Vaisala Gust
Humidity	%RH	Preamp Humidity Vaisala Humidity (ADC2)
Direction	degrees	Vaisala Wind Direction Vaisala Gust Direction
Pressure	kPa	Vaisala Barometric Pressure (ADC3)

ID: ID_TH_2102TEMPERATURE
Mask: 0x00004000
Name: "Preamp Temp"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_2102HUMIDITY
Mask: 0x00008000
Name: "Preamp Humidity"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_WINDSPD
Mask: 0x00010000
Name: "Wind Speed"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_GUSTDIR
Mask: 0x00020000
Name: "Gust Direction"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ADC1AVG
Mask: 0x00040000

Name: "Temp Avg"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_ADC2AVG
Mask: 0x00080000
Name: "Humidity Avg"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_CMINUSA
Mask: 0x00100000
Name: "L<sub>C#eq</sub> - L<sub>A#eq</sub>"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBALEQ_UNUSED
Mask: 0x00200000
Name: ""
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBALMAX_UNUSED
Mask: 0x00400000
Name: ""
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBASPL_UNUSED
Mask: 0x00800000
Name: ""
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBASPL11
Mask: 0x01000000
Name: "OBA 1/1 SPL"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBALEQ11
Mask: 0x02000000
Name: "OBA 1/1 Leq"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBAMAX11
Mask: 0x04000000
Name: "OBA 1/1 max"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBAMIN11
Mask: 0x08000000
Name: "OBA 1/1 min"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBASPL13
Mask: 0x10000000
Name: "OBA 1/3 SPL"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBALEQ13
Mask: 0x20000000
Name: "OBA 1/3 Leq"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBAMAX13
Mask: 0x40000000
Name: "OBA 1/3 max"
Options: "TAG_TIME_HISTORY_MISC"

ID: ID_TH_OBAMIN13
Mask: 0x80000000
Name: "OBA 1/3 min"
Options: "TAG_TIME_HISTORY_MISC"

Time History Options 3

ID: ID_TH_GUSTSPEED
Mask: 0x00000001
Name: "Gust Speed"
Options: "TAG_TIME_HISTORY_Options3"
//64
ID: ID_TH_ADC1MAX
Mask: 0x00000002
Name: "Temp Max"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_ADC1MIN
Mask: 0x00000004
Name: "Temp Min"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_ADC2MAX
Mask: 0x00000008
Name: "Humidity Max"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_ADC2MIN
Mask: 0x00000010
Name: "Humidity Min"
Options: "TAG_TIME_HISTORY_Options3"

Weather and Internal Units

The meter stores Time History Float Values with the specified units.

Metric	Internal Units	Data
Temperature	°C	Internal Temperature Vaisala Temperature (ADC1) Preamp Temperature
Speed	m/s	Vaisala Wind Vaisala Gust
Humidity	%RH	Preamp Humidity Vaisala Humidity (ADC2)
Direction	degrees	Vaisala Wind Direction Vaisala Gust Direction
Pressure	kPa	Vaisala Barometric Pressure (ADC3)

ID: ID_TH_WEATHER1
Mask: 0x00000020
Name: "Pressure Avg"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_WEATHER2
Mask: 0x00000040
Name: "Pressure Max"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_WEATHER3
Mask: 0x00000080
Name: "Pressure Min"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_AVG_WIND_DIR
Mask: 0x00000100
Name: "Wind Dir."
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_RAIN_MAX_INTENSITY – Updated in FW version 4.5.2
Mask: 0x00000200
Name: "Rain Max Intensity"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_HAIL_MAX_INTENSITY – Updated in FW version 4.5.2
Mask: 0x00000400
Name: "Hail Max Intensity"
Options: "TAG_TIME_HISTORY_Options3"

```
ID: ID_TH_DUR_FRAC
Mask: 0x00000800
Name: "Tms"
Options: "TAG_TIME_HISTORY_Options3"
//75
ID: ID_TH_WSLOWSPL
Mask: 0x00001000
Name: "L<sub>$S</sub>"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_WFASTSPL
Mask: 0x00002000
Name: "L<sub>$F</sub>"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_WIMPLSPL
Mask: 0x00004000
Name: "L<sub>$I</sub>"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_WIMPLMINUSLEQ
Mask: 0x00008000
Name: "L<sub>$Ieq</sub> - L<sub>$eq</sub>"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN1
Mask: 0x00010000
Name: "Ln Statistics"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN2
Mask: 0x00020000
Name: "Ln Statistics"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN3
Mask: 0x00040000
Name: "Ln Statistics"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN4
Mask: 0x00080000
Name: "Ln Statistics"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN5
Mask: 0x00100000
```

Name: "Ln Statistics"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN6
Mask: 0x00200000
Name: "Ln Statistics"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_LN_ENABLES
Mask: 0x003F0000 //(ID_TH_LN1 | ID_TH_LN2 | ID_TH_LN3 | ID_TH_LN4 | ID_TH_LN5 | ID_TH_LN6)
Name: "All Ln Stats"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_RUNNING_LEQ – Added in FW version 3.2.1
Mask: 0x00400000
Name: "Running Leq"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_RAIN_ACCUMULATION – Added in FW version 4.5.2
Mask: 0x00800000
Name: "Rain Accumulation"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_RAIN_DURATION – Added in FW version 4.5.2
Mask: 0x01000000
Name: "Rain Duration"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_HAIL_ACCUMULATION – Added in FW version 4.5.2
Mask: 0x02000000
Name: "Hail Accumulation"
Options: "TAG_TIME_HISTORY_Options3"

ID: ID_TH_HAIL_DURATION – Added in FW version 4.5.2
Mask: 0x04000000
Name: "Hail Duration"
Options: "TAG_TIME_HISTORY_Options3"

H. New Tags for Data – SoundAdvisor Only

These tags can be used with the func=getData or func=getDataMulti to retrieve many of the available metrics from the SoundAdvisor Model 831C meter.

Types

Use the values for the Types as defined here:

TypeInt = 1;
TypeUInt = 2;
TypeFloat = 3;
TypeFloatSeries = 4;
TypeFloatWithFlags = 5;
TypeString = 6;
TypeByteSeries = 7;
TypeUIntSeries = 8;
TypeEnum = 9;
TypeUIntWithFlags = 10;
TypeFloatWithFlagsSeries = 11;

Groups

GROUP_LIVE_SLM	= (200)
GROUP_CURRENT_SLM	= (201)
GROUP_OVERALL_SLM	= (202)
GROUP_MEASUREMENT_SLM	= (203)
GROUP_FILE_OVERALL_SLM	= (204)
GROUP_FILE_MEASUREMENT_SLM	= (205)
GROUP_LIVE_ONE_SECOND_SLM	= (206)
GROUP_TIME_HISTORY	= (207)
GROUP_FILE_TIME_HISTORY	= (208)
GROUP_SESSION_LOG	= (230)
GROUP_FILE_SESSION_LOG	= (231)
GROUP_EVENTS	= (240)
GROUP_FILE_EVENTS	= (241)

Examples

Example of the TAG_IO_ANYLEVEL_LEQ:

`http://localhost:2508/sdk?func=getDataNew&group=live&tagid=0x51454C41&type=5`

Example of getDataMulti for TAG_IO_ANYLEVEL_SPL, TAG_IO_OBA_SERIES_LIVE_1_3:

http://localhost:2508/sdk?func=getdatamulti&group=live&tagids=0x4C505341,0x334C424F&types=5,4&indices=0,0

Live

Tag Names	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_STABILITY_LEVEL	0x42415453	Int	0	10		
TAG_IO_ANYLEVEL_SPL	0x4C505341	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_SPL_1	0x314C5053	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_SPL_2	0x314C5053	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_SPL_3	0x314C5053	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_SPL_4	0x314C5053	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_LEQ	0x51454C41	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_LPEAK	0x4B504C41	FloatWithFlags	0	140		
TAG_IO_LIVE_1SEC_LEQ	0x4c53314c	FloatWithFlags	0	140		
TAG_IO_LIVE_1SEC_SPL	0x5353314c	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_LMAX	0x584D4C41	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_LMAX_TIME	0x54584C41	UIntWithFlags	0	140		
TAG_IO_ANYLEVEL_LMIN	0x4E4D4C41	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_LMIN_TIME	0x544E4C41	UIntWithFlags	0	140		
TAG_IO_ANYLEVEL_LPEAK_MAX	0x584D4B50	FloatWithFlags	0	140		
TAG_IO_ANYLEVEL_LPEAK_TIME	0x544D4B50	UIntWithFlags	0	140		
TAG_ACZ_LEQ	0x414C3031	FloatWithFlagsSeries				All tags named TAG_ACZ_* Return L_{AEQ}, L_{CEQ}, L_{ZEQ}, with flags in an orderd json array named value. (e.g. `"value":[ {"val":84.3162,"dt":0,"flags":0}, {"val":82.1306,"dt":0,"flags":0}, {"val":83.4263,"dt":0,"flags":0} ]`)
TAG_ACZ_PEAK	0x414C3032	FloatWithFlagsSeries				
TAG_ACZ_MAX_S	0x414C3033	FloatWithFlagsSeries				
TAG_ACZ_MAX_F	0x414C3034	FloatWithFlagsSeries				
TAG_ACZ_MAX_I	0x414C3035	FloatWithFlagsSeries				
TAG_ACZ_MIN_S	0x414C3036	FloatWithFlagsSeries				
TAG_ACZ_MIN_F	0x414C3037	FloatWithFlagsSeries				
TAG_ACZ_MIN_I	0x414C3038	FloatWithFlagsSeries				
TAG_ACZ_SPL_S	0x414C3039	FloatWithFlagsSeries				
TAG_ACZ_SPL_F	0x414C3130	FloatWithFlagsSeries				
TAG_ACZ_SPL_I	0x414C3131	FloatWithFlagsSeries				

TAG_IO_PROFILE_SERIES	0x4f525047	FloatSeries				
TAG_IO_OBA_VALUE_LEQ_1_1	0x514c3131	FloatWithFlags	0	10		
TAG_IO_OBA_SERIES_1_1	0x3141424F	FloatSeries	0	10		
TAG_IO_OBA_SERIES_MAX_1_1	0x3158424F	FloatSeries	0	10		
TAG_IO_OBA_SERIES_MIN_1_1	0x314E424F	FloatSeries	0	10		
TAG_IO_OBA_SERIES_1_3	0x3341424F	FloatSeries	0	10		
TAG_IO_OBA_SERIES_MAX_1_3	0x3358424F	FloatSeries	0	10		
TAG_IO_OBA_SERIES_MIN_1_3	0x334E424F	FloatSeries	0	10		

Measurement History

Tag Names (Group: Measurement)	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_NUM_HISTORY_RECORDS	0x5443484D	UInt	0	10		
TAG_IO_MEAS_HIST_INTERVAL_TIMESTAMP	0x54544E49	UInt				
TAG_IO_MEAS_HIST_INTERVAL_DURATION	0x44544E49	Float				Run time + Pause Time for that interval
TAG_IO_MEAS_HIST_INTERVAL_LEQ	0x51454C49	FloatSeries				

Measurement Record

Tag Names (Group: Measurement)	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_MEAS_HIST_LOAD_RECORD	0x524C484D	UInt				Set using “func=setactiveindex”
TAG_IO_MEAS_HIST_CUR_RECORD	0x5243484D	UInt				

The function call (**setactiveindex**) for this is different from other data calls. This will load the Measurement History Record into active memory for retrieval using the other Get Data calls on Measurement History.
The group 203 is for current measurement history data (as opposed to 205 for file data).
The idx is the measurement history record to load; note it is zero based.
http://127.0.0.1:2550/sdk?func=setactiveindex&group=203&tag=0x524C484D&idx=0

Events

Tag Names (Group: Events)	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_SPL_EXCEEDED	0x4458454c	TypeUInt	0	10		
TAG_IO_SPL_EXCEEDED_SERIES	0x5344584c	TypeUIntSeries	0	1		
TAG_IO_PEAK_EXCEEDED_SERIES	0x53445850	TypeUIntSeries	0	1		
TAG_IO_SPL_EXCEED_PERCENT	0x54435045	TypeFloat	0	10		
TAG_IO_SPL_EVENT_COUNT	0x56455053	TypeInt	0	10		
TAG_IO_SPL_EVENT_TIME	0x54455053	TypeFloat	0	10		
TAG_IO_PEAK_EVENT_COUNT	0x56454B50	TypeInt	0	10		
TAG_IO_PEAK_EVENT_TIME	0x54454B50	TypeFloat	0	10		
TAG_IO_SPL_EVENT_COUNT_SERIES	0x5343454c	TypeUIntSeries	0	1		
TAG_IO_SPL_EVENT_TIME_SERIES	0x5354454c	TypeFloatSeries	0	1		
TAG_IO_PEAK_EVENT_COUNT_SERIES	0x53434550	TypeUIntSeries	0	1		
TAG_IO_PEAK_EVENT_TIME_SERIES	0x53544550	TypeFloatSeries	0	1		
TAG_IO_SOUND_RECORDER_STATE	0x43455253	TypeUInt	0	1		

Event History

TAG_IO_EVENT_HIST_LOAD_RECORD - Tag ID: 0x524C4845
Type: TypeFloat
Used to tell the meter which Event History Record to prepare as the current record. Subsequent requests will respond with data that pertains to the supplied Event History Record number (recordNum). **group** represents either live event data (=240) or file event data (=241)

```
$.getJSON("/sdk?func=get&f=128&p1=" + group + "&vals=0x524C4845," + recordNum + "&format=tag:u4,ack:u4", function (data) {
        if (data.Response.ack == CTL_INVALID) {
                //Handle bad response.
        }
        //request updated data based on the new Event History Record
}
```

Alternate:

You may also use the function call (**setactiveindex**) for this. This will load the Event History Record into active memory for retrieval using the other Get Data calls on Event History.
The group 240 is for current event history data (as opposed to 241 for file data).
The idx is the event history record to load; note it is zero based.
http://127.0.0.1:2550/sdk?func=setactiveindex&group=240&tag=0x524C484D&idx=0

Tag Names	Tag ID	Type	Notes
TAG_IO_DYNAMIC_TRIGGER_LEVEL	0x4C445645	TypeFloat	
TAG_IO_DYNAMIC_BACKGROUND_LEVEL	0x42445645	TypeFloat	
TAG_IO_DYNAMIC_TRIGGERED	0x54445645	TypeFloat	
TAG_IO_EVENT_LEQ	0x51454C45	TypeFloat	
TAG_IO_EVENT_LPEAK	0x4B504C45	TypeFloat	
TAG_IO_EVENT_LMAX	0x584D4C45	TypeFloat	
TAG_IO_EVENT_SEL	0x4C455345	TypeFloat	
TAG_IO_EVENT_SE	0x30455345	TypeFloat	
TAG_IO_EVENT_STATE	0x54535645	TypeUInt	PRETRIG:0-1, READY:2, TRIGGER:3, VALID:4, CONTINUE:5
TAG_IO_EVENT_START_TIME	0x4D545345	TypeUInt	
TAG_IO_EVENT_RUN_TIME	0x4D545245	TypeFloat	
TAG_IO_EVENT_MAX_TIME	0x4D545845	TypeUInt	
TAG_IO_EVENT_MIN_DURATION	0x444D5645	TypeUInt	
TAG_IO_EVENT_HIST_CUR_RECORD	0x52434845	TypeUInt	
TAG_IO_EVENT_OBA_1_1_LEQ_SERIES	0x4C315645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_OBA_1_1_MAX_SERIES	0x4D315645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_OBA_1_1_SEL_SERIES	0x53315645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_OBA_1_3_LEQ_SERIES	0x4C335645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_OBA_1_3_MAX_SERIES	0x4D335645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_OBA_1_3_SEL_SERIES	0x53335645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_TIME_HIST_SERIES	0x53545645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_TIME_HIST_LEQ_SERIES	0x4C545645	TypeFloatSeries	Call using "get" function.
TAG_IO_EVENT_TIME_HIST_NUM_RECORDS	0x4D554E45	TypeUInt	

TAG_IO_EVENT_TIME_HIST_TIME_SERIES	0x54545645	TypeFloatSeries	Call using “get” function.
TAG_IO_EVENT_TH_OBA_BY_TIME	0x544F5645	TypeFloatSeries	Call using “get” function.
TAG_IO_EVENT_TH_OBA_BY_FREQ	0x464F5645	TypeFloatSeries	Call using “get” function.
TAG_IO_NUM_EVENT_RECORDS	0x54435645	TypeUInt	

Examples on how to call with get:

1. /sdk?func=get&f=128&p1=240&vals=1398036037,0,0,120&format=t:u4,lt:u4,tag:u4,len:u4,value:af120,tag1:u4,len1:u4,time:ai120
2. /sdk?func=get&f=128&p1=240&vals=0x54545645,7,0,120&format=tag:u4,len:u4,time:ai120

Time History

Several of these TAGs need to be called with “*func=get*” in order to provide the correct parameters to the SLM. If the *getDataNew* or *getDataMulti* are called then the SLM may crash.

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_TIME_HIST_NUM_RECORDS	0x54434854	TypeInt	0	10		
TAG_IO_TIME_HIST_AVAILABLE_METRICS	0x494D4854	TypeUIntSeries	0	10		
TAG_IO_TIME_HIST_SERIES	0x30534854	TypeFloatSeries	0	10	0	
TAG_IO_TIME_HIST_LEQ	0x31534854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_METRIC	0x32534854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_FLAGS	0x33534854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_TIMESTAMP	0x34534854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_ACTION	0x35534854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_RECORD	0x36534854	TypeUIntSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_1_SERIES	0x53314854	TypeFloatSeries	0	10		Call using “get” function. Ex 1
TAG_IO_TIME_HIST_OBA_1_1	0x4C314854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_1_MAX	0x58314854	TypeFloatSeries	0	10		

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_TIME_HIST_OBA_1_1_MIN	0x4E314854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_SERIES	0x53334854	TypeFloatSeries	0	10	0	Call using “get” function. Ex 2
TAG_IO_TIME_HIST_OBA_1_3	0x4C334854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_MAX	0x58334854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_MIN	0x4E334854	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_1_BY_TIME_SER	0x41315442	TypeFloatSeries	0	10		Call using “get” function. Ex 3
TAG_IO_TIME_HIST_OBA_1_1_LEQ_BY_TIME	0x4C315442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_1_SPL_BY_TIME	0x53315442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_1_MAX_BY_TIME	0x58315442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_1_MIN_BY_TIME	0x4E315442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_BY_TIME_SER	0x41335442	TypeFloatSeries	0	10		Call using “get” function. Ex 4
TAG_IO_TIME_HIST_OBA_1_3_LEQ_BY_TIME	0x4C335442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_SPL_BY_TIME	0x53335442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_MAX_BY_TIME	0x58335442	TypeFloatSeries	0	10		
TAG_IO_TIME_HIST_OBA_1_3_MIN_BY_TIME	0x4E335442	TypeFloatSeries	0	10		

Examples on how to call with get:

1. /sdk?func=get&f=128&p1=207&vals=0x53314854,50&format=tag:x4,sz:i4,t1:x4,s1:i4,series0:af12,t2:x4,s2:i4,series1:af12,t3:x4,s3:i4,series2:af12
2. /sdk?func=get&f=128&p1=207&vals=0x53334854,50&format=tag:x4,sz:i4,t1:x4,s1:i4,series0:af12,t2:x4,s2:i4,series1:af12,t3:x4,s3:i4,series2:af12
3. /sdk?func=get&f=128&p1=207&vals=1093751874,7,0,77&format=tag:x4,sz:i4,series0:af77,sz:i4,series1:af77,sz:i4,series2:af77,flagtag:x4,flagsz:i4,thFlags:ai77,timetag:x4,timesz:i4,thTime:ai77,acttag:x4,actsz:i4,thAction:ai77
4. /sdk?func=get&f=128&p1=207&vals=1093882946,22,0,18&format=tag:x4,sizeOfSeries0:i4,series0:af18, sizeOfSeries1:i4,series1:af18,sz:i4,series2:af18,flagtag:x4,flagsize:i4,thFlags:ai18,timetag:x4,timesz:i4,thTime:ai18,actiontag:x4,actionsz:i4,thAction:ai18

Session Log

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_SLOG_NUM_RECORDS	0x534C3031	TypeInt	0	9999		

Sound Recs

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_NUM_SOUND_RECORDINGS	0x54435253	TypeInt	0	9999		

Spectral Ln

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_SPECTRAL_LN_SERIES_0	0x4E4C5347	TypeFloatSeries	0	1	0	These 6 values use the same tag and are distinguished by nDx
TAG_IO_SPECTRAL_LN_SERIES_1	0x4E4C5347	TypeFloatSeries	0	1	1	
TAG_IO_SPECTRAL_LN_SERIES_2	0x4E4C5347	TypeFloatSeries	0	1	2	
TAG_IO_SPECTRAL_LN_SERIES_3	0x4E4C5347	TypeFloatSeries	0	1	3	
TAG_IO_SPECTRAL_LN_SERIES_4	0x4E4C5347	TypeFloatSeries	0	1	4	
TAG_IO_SPECTRAL_LN_SERIES_5	0x4E4C5347	TypeFloatSeries	0	1	5	
TAG_IO_SPECTRAL_LN_1	0x314e4c53	TypeFloat	0	140		
TAG_IO_SPECTRAL_LN_2	0x324e4c53	TypeFloat	0	140		
TAG_IO_SPECTRAL_LN_3	0x334e4c53	TypeFloat	0	140		
TAG_IO_SPECTRAL_LN_4	0x344e4c53	TypeFloat	0	140		
TAG_IO_SPECTRAL_LN_5	0x354e4c53	TypeFloat	0	140		
TAG_IO_SPECTRAL_LN_6	0x364e4c53	TypeFloat	0	140		

Ln Percentiles

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_TAKT_MAXIMAL_5_SEC	0x544B4154	TypeFloat	0	10		
TAG_IO_LN_95	0x35394E4C	TypeFloat	0	10		
TAG_IO_LN_LEVEL	0x564C4E4C	TypeFloat	0	10		
TAG_IO_LN_PERCENT	0x43504E4C	TypeFloat	0	10		
TAG_IO_LN_LEVEL_SERIES	0x534C4E4C	TypeFloatSeries	0	200		

Overload tags

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_OVERLOAD_COUNT	0x434C564F	TypeUInt	0	999999		
TAG_IO_OVERLOAD_TIME	0x544C564F	TypeFloat	0	999999		
TAG_IO_OVERLOAD_PERCENT	0x504C564F	TypeFloat	0	100		
TAG_IO_OVERLOAD_OBA_COUNT	0x434F564F	TypeUInt	0	999999		
TAG_IO_OVERLOAD_OBA_TIME	0x544F564F	TypeFloat	0	999999		
TAG_IO_OVERLOAD_OBA_PERCENT	0x504F564F	TypeFloat	0	100		

Community Noise Tags

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_LEQA	0x4151454C	TypeFloat	0	140		
TAG_IO_LEQC	0x4351454C	TypeFloat	0	140		
TAG_IO_C_MINUS_A	0x414E4D43	TypeFloat	0	140		
TAG_IO_LEQ_IMPULSE	0x4951454C	TypeFloat	0	140		
TAG_IO_LEQ_LINEAR	0x4C51454C	TypeFloat	0	140		
TAG_IO_IMPULSIVITY	0x56534D49	TypeFloat	0	140		

Preamp

Tag Names	Tag ID	Type	Min	Max	Perc	Notes
TAG_IO_EA_STATUS	0x54534145	TypeUInt	0	10		
TAG_IO_PREAMP_VOLTS	0x564d5250	TypeFloat	0	10		
TAG_IO_PREAMP_ICP_STATUS	0x53495250	TypeInt	0	2		
TAG_IO_ENV_PREAMP_TEMPERATURE	0x54564E45	TypeFloat	0	10		
TAG_IO_ENV_PREAMP_HUMIDITY	0x48564E45	TypeFloat	0	10		
TAG_IO_ENV_PREAMP_DEW_POINT	0x44564E45	TypeFloat	0	10		
TAG_IO_ENV_PREAMP_MODEL	0x4d564E45	TypeUInt	0	10		
TAG_IO_ENV_PREAMP_SN	0x53564E45	TypeUInt	0	10		
TAG_IO_ENV_PREAMP_FW_VERSION	0x56564E45	TypeFloat	0	99	3	Only use if attached preamp has queryable firmware.
TAG_OVERLOAD_IN_STATE	0x444C564F	TypeInt	0	10		

Logic IO

Tag Names	Tag ID	Type	Min	Max	Notes
TAG_LOGIC_IN_STATE	0x4943474C	TypeInt	0	10	
TAG_LOGIC_OUT_STATE	0x4F43474C	TypeInt	0	10	

INT-ET Specific Tags

Tag Names	Tag ID	Type	Min	Max	Notes
TAG_MAINS_OK	0x4B4F4E4D	TypeInt	0	10	
TAG_831_INT_STATE	0x53544E49	TypeInt	0	10	
TAG_IO_COMM_WATCHDOG_COUNT	0x43445743	TypeInt	-999	999	
TAG_IO_SPL_EXCEED_PERCENT	0x54435045	TypeFloatWithFlags	0	10	

GPS Tags

Tag Names	Tag ID	Type	Min	Max	Notes
TAG_IO_GPS_LAT_DEGREE	0x31535047	TypeInt	0	10	
TAG_IO_GPS_LAT_MINUTE	0x32535047	TypeFloat	0	10	
TAG_IO_GPS_LON_DEGREE	0x33535047	TypeInt	0	10	
TAG_IO_GPS_LON_MINUTE	0x34535047	TypeFloat	0	10	
TAG_IO_GPS_ELEVATION	0x35535047	TypeFloat	0	10	
TAG_IO_GPS_TIME	0x36535047	TypeUInt	0	10	
TAG_IO_GPS_STATUS	0x37535047	TypeUInt	0	10	
TAG_IO_GPS_SATELLITES	0x38535047	TypeInt	0	10	

Weather data

Tag Names	Tag ID	Type	Min	Max	Notes
TAG_IO_RAIN_RATE	0x30525257	TypeFloat	0	100	
TAG_IO_RAIN_RATE_MAX	0x58525257	TypeFloat	0	100	
TAG_IO_RAIN_ACCUMULATION	0x30415257	TypeFloat	0	100	
TAG_IO_RAIN_DURATION	0x30445257	TypeFloat	0	100	
TAG_IO_HAIL_RATE	0x30524857	TypeFloat	0	100	
TAG_IO_HAIL_RATE_MAX	0x58524857	TypeFloat	0	100	
TAG_IO_HAIL_ACCUMULATION	0x30414857	TypeFloat	0	100	
TAG_IO_HAIL_DURATION	0x30444857	TypeFloat	0	100	
TAG_IO_WIND_SPEED	0x30535757	TypeFloat	0	100	
TAG_IO_WIND_SPEED_AVERAGE	0x41535757	TypeFloat	0	100	
TAG_IO_WIND_SPEED_MIN	0x4E4D5357	TypeFloat	0	100	
TAG_IO_WIND_DIRECTION	0x30445757	TypeFloat	0	100	
TAG_IO_WIND_COMPASS	0x30445757	TypeInt	0	100	
TAG_IO_WIND_GUST	0x30475757	TypeFloat	0	100	
TAG_IO_WIND_GUST_DIRECTION	0x44475757	TypeFloat	0	100	

Tag Names	Tag ID	Type	Min	Max	Notes
TAG_IO_WIND_GUST_COMPASS	0x44475757	TypeInt	0	100	
TAG_IO_WINDY_PERCENT	0x54435057	TypeFloat	0	100	
TAG_IO_WIND_DIR_STATS	0x53445757	TypeUIntSeries	0	100	
TAG_IO_TEMPERATURE	0x504D5457	TypeFloat	0	100	
TAG_IO_TEMPERATURE_MAX	0x584D5457	TypeFloat	0	100	
TAG_IO_TEMPERATURE_MIN	0x4E4D5457	TypeFloat	0	100	
TAG_IO_HUMIDITY	0x4D554857	TypeFloat	0	100	
TAG_IO_HUMIDITY_MAX	0x584D4857	TypeFloat	0	100	
TAG_IO_HUMIDITY_MIN	0x4E4D4857	TypeFloat	0	100	
TAG_IO_PRESSURE	0x53525057	TypeFloat	0	100	
TAG_IO_PRESSURE_MAX	0x58525057	TypeFloat	0	100	
TAG_IO_PRESSURE_MIN	0x4E525057	TypeFloat	0	100	

Dose / Exposure

Tag Names	Tag ID	Type	Min	Max	Ndx	Notes
TAG_IO_DOSE_NAME	0x4E534F44	TypeString	0	255	0 or 1	0: Gets name for Dose1, 1: Gets name for Dose2
TAG_IO_TWA_LABLE	0x4C415754	TypeString	0	255		
TAG_IO_TWA	0x58415754	TypeString	0	255		
TAG_IO_TWA_PROJECTED	0x50415754	TypeString	0	255		
TAG_IO_LEPD	0x4450454C	TypeString	0	255		
TAG_IO_DOSE	0x45534F44	TypeString	0	255		
TAG_IO_DOSE_PROJECTED	0x50534F44	TypeString	0	255		

SEL

Tag Names (Group: Time History)	Tag ID	Type	Min	Max	Notes
TAG_IO_LWDE	0x4544574C	TypeFloat	0	140	
TAG_IO_E_H	0x48455845	TypeFloat	0	140	
TAG_IO_E8_H	0x48383045	TypeFloat	0	140	
TAG_IO_E40_H	0x48303445	TypeFloat	0	140	
TAG_IO_E_S	0x53455845	TypeFloat	0	140	
TAG_IO_E8_S	0x53383045	TypeFloat	0	140	
TAG_IO_E40_S	0x53303445	TypeFloat	0	140	

Daily Noise Levels (Community Noise)

Tag Names (Group: Time History)	Tag ID	Type	Min	Max	Notes
TAG_IO_LDN	0x584E444C	TypeFloat	0	140	
TAG_IO_LDN_DAY	0x444E444C				
TAG_IO_LDN_NIGHT	0x4E4E444C				
TAG_IO_LDEN	0x4E45444C				
TAG_IO_LDEN_DAY	0x444E454C				
TAG_IO_LDEN_EVENING	0x454E454C				
TAG_IO_LDEN_NIGHT	0x4E4E454C				

Power page

Tag Names (Group: Time History)	Tag ID	Type	Min	Max	Perc	Notes
TAG_IO_BATTERY_VOLTS	0x56544142	TypeFloat	0	100	1	
TAG_IO_BATTERY_TIME	0x54544142	TypeFloat	0	100	1	
TAG_IO_BATTERY_PERCENT	0x50544142	TypeFloat	0	100	1	
TAG_IO_EXTERNAL_VOLTS	0x56545845	TypeFloat	0	100	1	
TAG_IO_INPUT_VOLTS	0x56504E49	TypeFloat	0	100	1	
TAG_IO_POWER_STATUS	0x54535750	TypeInt	-5	100		
TAG_IO_CHARGE_STATE	0x43484753	TypeInt	0	8		
TAG_IO_TOTAL_FLASH	0x544D454D	TypeUInt	0	4294967295		
TAG_IO_FREE_FLASH	0x464D454D	TypeUInt	0	4294967295		
TAG_IO_NUM_FILES	0x464D554E	TypeUInt	0	100000		
TAG_IO_START_TIME	0x54525453	TypeUInt	0			
TAG_IO_END_TIME	0x54444E45	TypeUInt	0			
TAG_IO_RUN_TIME	0x544E5552	TypeFloat	0			
TAG_IO_RUN_TIMER	0x524E5552	TypeFloat	0			
TAG_IO_PAUSE_TIME	0x54534150	TypeFloat	0			
TAG_IO_DURATION	0x54525544	TypeFloat	0			
TAG_IO_MEAS_DURATION	0x54525544	TypeFloat	0			
TAG_IO_CALIBRATION_SPL	0x4c4c4143	TypeFloat	0	280	1	
TAG_IO_CALIBRATION_DELTA	0x444c4143	TypeFloat	0	280	1	
TAG_IO_SYS_NOISE_FLOOR	0x30464E53	TypeFloat	0	140		
TAG_IO_SYS_NOISE_FLOOR_A	0x41464E53	TypeFloat	0	140		
TAG_IO_SYS_NOISE_FLOOR_C	0x43464E53	TypeFloat	0	140		
TAG_IO_SYS_NOISE_FLOOR_Z	0x5A464E53	TypeFloat	0	140		
TAG_IO_SYS_UNDER_RANGE	0x30444E55	TypeFloat	0	140		
TAG_IO_SYS_UNDER_RANGE_A	0x41444E55	TypeFloat	0	140		
TAG_IO_SYS_UNDER_RANGE_C	0x43444E55	TypeFloat	0	140		
TAG_IO_SYS_UNDER_RANGE_Z	0x5A444E55	TypeFloat	0	140		

Tag Names (Group: Time History)	Tag ID	Type	Min	Max	Perc	Notes
TAG_IO_SYS_PEAK_OVERLOAD	0x444C4f50	TypeFloat	0	140		
TAG_IO_SYS_TEMP	0x54454D50	TypeFloat				
TAG_IO_LOAD_MEAS_HIST_RECORD	0x524C484D	TypeUInt				
TAG_IO_SPL_EVENT_TIME	0x54455053	TypeFloat	0	90000000		
TAG_IO_REM_LEQ_TARGET_TIME	0x54544C52	TypeUInt				Remaining Leq target time
TAG_IO_REM_LEQ	0x51454C52	TypeFloat				
TAG_IO_ANYLEVEL_LEQ_LIVE	0x51454C41	TypeFloat				

I. Examples of Using the get Function

Here are several examples in JavaScript of TAGs that need to use the **get** function.

```
function getTHByTime() {
     var num = Math.min(lastCount, 120);
     var group = 207; //time

     if (inFileDsp) group = 208; //filetime
     var tag = 0x41315442; // TAG_IO_TIME_HIST_OBA_1_1_BY_TIME_SER
     if (onThird) {
          tag = 0x41335442; //TAG_IO_TIME_HIST_OBA_1_3_BY_TIME_SER
     }
     var getString = "/sdk?func=get&f=128&p1=" + group + "&vals=" + tag + "," + freqIndex + "," + start + "," + num +
 "&format=tag:x4,";//tag:x4,sz:i4,series0:af" + num + ",sz:i4,series1:af" + num + ",sz:i4,series2:af" + num;
     var prop = MeasurementProperties.TAG_TIME_HISTORY_MISC;
     var seriesCnt = 0;
     if (onFull) {
          if (prop & THOptions.ID_TH_OBALEQ11.mask) {
               getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
          } else if (prop & THOptions.ID_TH_OBASPL11.mask) {
               getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
          }
          if (prop & THOptions.ID_TH_OBAMAX11.mask) {
               if (seriesCnt > 0) getString += ",";
               getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
          }
          if (prop & THOptions.ID_TH_OBAMIN11.mask) {
               if (seriesCnt > 0) getString += ",";
               getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
          }
     } else {
          if (prop & THOptions.ID_TH_OBALEQ13.mask) {
               getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
          } else if (prop & THOptions.ID_TH_OBASPL13.mask) {
            getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
        }
        if (prop & THOptions.ID_TH_OBAMAX13.mask) {
            if (seriesCnt > 0) getString += ",";
            getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
        }
        if (prop & THOptions.ID_TH_OBAMIN13.mask) {
            if (seriesCnt > 0) getString += ",";
            getString += "sz:i4,series" + seriesCnt++ + ":af" + num;
        }
    }
    getString += ",flagtag:x4,flagsz:i4,thFlags:ai" + num + ",timetag:x4,timesz:i4,thTime:ai" + num + ",acttag:x4,ac
tsz:i4,thAction:ai" + num;
    if (iTms > 0) getString += ",tmstag:x4,tmssz:i4,thTms:af" + num;
    $.getJSON(getString)
    .done(function (dataVal) {
        specData = [];
        if (dataVal["ResultCode"] == 0 && dataVal.Response["series0"] != undefined)
        {
            var cnt = 0;
            if (onFull)
            {
                if (prop & THOptions.ID_TH_OBALEQ11.mask) {

    specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), fill: 1, fillColor: mainClr, color: m
ainClr });
                } else if (prop & THOptions.ID_TH_OBASPL11.mask) {

    specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), fill: 1, fillColor: mainClr, color: m
ainClr });
                }
                if (prop & THOptions.ID_TH_OBAMAX11.mask) {
                    specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), color: maxClr });
                }
                if (prop & THOptions.ID_TH_OBAMIN11.mask) {
                    specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), color: minClr });
                    }
               } else {
                    if (prop & THOptions.ID_TH_OBALEQ13.mask) {

     specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), fill: 1, fillColor: mainClr, color: m
ainClr });
                    } else if (prop & THOptions.ID_TH_OBASPL13.mask) {

     specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), fill: 1, fillColor: mainClr, color: m
ainClr });
                    }
                    if (prop & THOptions.ID_TH_OBAMAX13.mask) {
                         specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), color: maxClr });
                    }
                    if (prop & THOptions.ID_TH_OBAMIN13.mask) {
                         specData.push({ data: parseOvlOBAData(dataVal.Response["series" + cnt++]), color: minClr });
                    }
               }
               //get flag, action and record data.
               getFlags(dataVal.Response);

          }
          else if (dataVal["ResultCode"] == 0) {
               specData = [];
               specData.push({ data: liveGraph.getRandom(size), fill: 1, fillColor: mainClr, color: mainClr });
               flags = [];
               actions = [];
               timestamp = [];
               tmsStamp = [];
          }

     })
     .fail(function (e) {
          console.log(e);
     })
     .always(function (e) {
            //console.log("here");
            lockedGetData = 0;
            updateTable();
      });
}
```

```
var tag = DataTags["TAG_IO_EVENT_TH_OBA_BY_FREQ"].ID;

var getString = "/sdk?func=get&f=128&p1=" + group + "&vals=" + tag + "," + 0 + "," + recNum + "," + size +
"&format=tag:u4,len:u4,value:af" + size;

$.getJSON(getString)
.done(function (data) {
    OneOctaveData = parseOvlOBAData(data.Response.value);
    ovlFOMax = OneOctaveMaxData = [];

    ovlFOMin = OneOctaveMinData = [];

    // TAG_IO_EVENT_TIME_HIST_TIME_SERIES
    tag = DataTags["TAG_IO_EVENT_TIME_HIST_TIME_SERIES"].ID
    getString = "/sdk?func=get&f=128&p1=" + group + "&vals=" + tag + ",0," + recNum +
",1&format=tag:u4,len:u4,time:ai1";
    $.getJSON(getString)
    .done(function (dataVal) {
        if (dataVal["ResultCode"] == 0 && dataVal.Response["time"] != undefined)
        {
            timestamp = dataVal.Response["time"];
        }
        // Process the data
    });
});
```

J. AudCal Usage

Set the Meter to AudCal Mode

First off, you must have the AudCal Option installed on the meter. The Measurement Properties' Instrument Mode must be set to the AudCal Mode. See the section on setting Properties (Get and Set Properties) and the associated note for Latching Properties for more information. The meter must be stopped and reset then you may send the property like:

```
$.getJSON("/sdk?func=setproperty&tagid=0x494D3031&type=integer&value=3&index=0")
```

And to Latch the settings:

```
$.getJSON('/sdk?func=LATCHSETTINGS&data=2&marksettingschanged=true')
```

Command to Prepare Meter

Further, in order for data to be valid for a given test, the meter must be set to the corresponding view. This ensures that the data collected is available and valid. The **CMD_AUDCAL_MEAS_SET** command will set the meter to a specified **p1=**page.

```
const CMD_AUDCAL_MEAS_SET = 810;
```

Valid values for p1 to set up the meter for each test.

ID:	Value	Test Types
PRM_AUDCAL_FREQUENCY_ACCURACY	0	Broadband, Cross Talk, On/Off Ratio, Frequency and Speech Tests
PRM_AUDCAL_THD	1	Harmonic Distortion Test
PRM_AUDCAL_LINEARITY	2	Linearity Test

PRM_AUDCAL_FLATNESS	3	Narrow Band Noise and Hearing Level Tests
PRM_AUDCAL_FM	4	Frequency Modulation Test
PRM_AUDCAL_PULSE_CAL	5	Pulse Test

Prepare Meter for Test for Linearity:

```
$.getJSON("/sdk?func=cmd&id=" + CMD_AUDCAL_MEAS_SET + "&p1=" + PRM_AUDCAL_LINEARITY + "&p2=1");
```

Response:

{ "Result" : "Success: 0 ","ResultCode":0,"ResultName":"Success" }

Abort Test

Optional: You may send the following command to abort the current test.

```
$.getJSON("/sdk?func=cmd&id=" + CMD_AUDCAL_MEAS_SET + "&p1=-1&p2=1");
```

Locking a Frequency

Two of the tests, Cross Talk and Linearity, require that values are based off of a specified frequency. You use the **CMD_AUDCAL_LOCK_FREQ** command to tell the meter to which frequency bin to lock.

```
const CMD_AUDCAL_LOCK_FREQ = 811;
```

Format:

```
"/sdk?func=cmd&id=" + CMD_AUDCAL_LOCK_FREQ + "&p1=0&p2=0&vals=0x" + hex;
```

Params:

p1=0,

p2=0,

vals=hexadecimal representation of frequency in float form (e.g., 1000.0 = 0x447A0000)

Example:

```
$.getJSON("/sdk?func=cmd&id=" + CMD_AUDCAL_LOCK_FREQ + "&p1=0&p2=0&vals=0x447A0000";
```

Helper Function – numToFloat32Hex()

```
function numToFloat32Hex(val, littleEndian) {
        if (isNaN(val)) return false;
        var buf = new ArrayBuffer(4);
        var dv = new DataView(buf);
        dv.setFloat32(0, val, true);
        return ("0000000" + dv.getUint32(0, !(littleEndian || false)).toString(16)).slice(-8).toUpperCase();
}
```

Obtaining Data for Tests

In order to capture the data for taking a measurement, you will need to setup a

Retrieve File Data (831C) using getData and getDataMulti

In order to obtain data from files stored on the meter you can use the same commands as getData and getDataMulti if you load the file into the meters cached memory. Follow these step to get the Measurement data from a file. You can review the Error! Reference source not found. to obtain Time History, Event History, etc., in a similar manner. See **New Tags for Data – SoundAdvisor Only** for more details about the available tags.

6. Open the file with the exact filename (fn=21012600.LD0.s in this case)
    a. /sdk?func=filecmd&cmd=7&fs=0&fn=21012600.LD0.s

Then you will make the same calls as you did before except with the File Group as defined in the Error! Reference source not found. section.

Measurement = 203

FileMeasurement = 205

7. Query number of measurement history records
    a. http://ld831c.pcb.com/sdk?func=getDataNew&Group=FileMeasurement&tagid=0x5443484D&type=2
8. Select the desired measurement history record that is equal to the number returned by the previous query minus 1.
    a. http://ld831c.pcb.com/sdk?func=setactiveindex&group=205&tag=0x524C484D&idx=n
9. Query LAeq, L10 and L90 using
    a. http://ld831c.pcb.com/sdk?func=getDataMulti&Group=FileMeasurement&tagids=0x414C3031,0x564C4E4C,0x564C4E4C&types=5,3,3&indices=0,1,5
10. Close the file when you are done.
    a. /sdk?func=filecmd&cmd=9

Web Socket section in Using HttpLD (SDK). That section will help you be able to use the Data tags found in Appendix I – Tags for AudCal. The Tags are broken up by Test and some duplication of Tags in these lists will be found in order to ensure the each test has the needed set of data.

Data in dB does not have the corrections or the RETSPL's applied.

Distortion Test:

TAG_IO_FFT_MAX is an array of 2 values which are the frequency and Sound Pressure Level (SPL) of the highest level. TAG_IO_FFT_MAX[0] = Frequency and TAG_IO_FFT_MAX[1]=SPL. Several of the Test use this and refer to this definition.

TAG_IO_FFT_FREQ_STABILITY is a stability indicator where 1 means stable and 0 not stable.

These are the values used to in the test:

TAG_IO_FFT_THD_SERIES, TAG_IO_FFT_HARMONIC_SERIES, TAG_IO_AUD_THD_SERIES

Frequency Test:

TAG_IO_FFT_MAX, TAG_IO_FFT_FREQ_STABILITY are previously defined.

Hearing Level Test:

TAG_IO_FFT_MAX, TAG_IO_FFT_FREQ_STABILITY are previously defined.

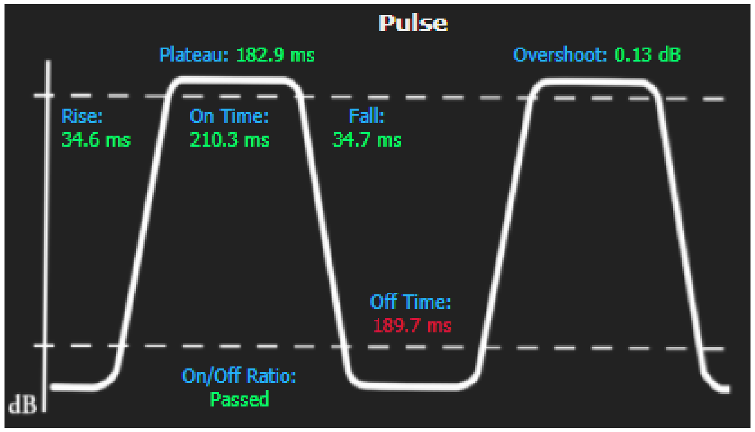
Linearity Test:

TAG_IO_FFT_MAX, TAG_IO_AUD_FREQ_LEVEL_STABILITY (Level stability) are previously defined.

Pulse Test:

The tags here are basically as their names represent and match the standard's description of Rise, Fall, On Time, Off Time, Plateau, and Overshoot.

TAG_IO_AUD_PULSE_RISE, TAG_IO_AUD_PULSE_FALL, TAG_IO_AUD_PULSE_ON, TAG_IO_AUD_PULSE_OFF, TAG_IO_AUD_PULSE_PLATEAU, TAG_IO_AUD_PULSE_OVERSHOOT, TAG_IO_AUD_HIGHPASS_SPL



Cross Talk and On/Off Ratio:

TAG_IO_FFT_MAX, TAG_IO_AUD_FREQ_LEVEL_STABILITY are previously defined similarly.

Frequency Modulation Test:

TAG_IO_FFT_MAX, TAG_IO_AUD_FREQ_LEVEL_STABILITY are previously defined. TAG_IO_AUD_FM_MOD_RATE, TAG_IO_AUD_FM_CARRIER_FREQ, TAG_IO_AUD_FM_MAX, TAG_IO_AUD_FM_MIN, TAG_IO_AUD_HIGHPASS_SPL are self explanatory.

Narrowband Noise Level Test:

TAG_IO_FFT_MAX, TAG_IO_FFT_FREQ_STABILITY are previously defined. TAG_ACZ_SPL_S, TAG_ACZ_SPL_F are Slow SPL and Fast SPL values respectively. Use the Slow SPL for the test and wait for the value to become stable before taking the reading.

Broadband Noise Test:

This test uses the following sdk call:

```
$.getJSON("/sdk?func=getDataNew&tag=CEPF&group=Live&type=4")
```

Response: (removed most of the floats from this example response as there are 1601)

```
{ "Data": [3.191406e+01,4.103756e+01,3.742173e+01,2.619611e+01,2.156787e+01,3.087417e+01,3.401059e+01,
. . .
1601 total float values
. . .
,2.985188e+01,4.058721e+01,4.501533e+01,4.191797e+01,3.491101e+01], "Result" : "Success: 0
","ResultCode":0,"ResultName":"Success" }
```

An average is performed on each of the bins used in the calculation over 64 requests to obtain the 191 data points based on the third octave bins for this test. The 1601 values are in dB so you must convert to power in order to compute the average.

Speech Test:

TAG_IO_ANYLEVEL_SPL is self explanatory.

TAG_IO_OBA_SERIES_LIVE_1_3 is an array of dB floats for third octave.

TAG_IO_FFT_MAX is previously defined.

Booth Test

TAG_IO_OBA_SERIES_LIVE_1_3 is the array of third octave dB levels that are used in this test.

Live OBA data consists of 36 bins. Frequencies of interest here map as follows:

Freq:	125	250	500	800	1000	1600	2000	3150	4000	6300	8000
Index:	13	16	19	21	22	24	25	27	28	30	31

13, 16, 19, 21, 22, 24, 25, 27, 28, 30 and 31 are the indices that correlate to the frequencies defined by the standard. Each value is converted to Power and summed with each iteration of the loop. Then averaged at the end and compared to the MPANL or OSHA values in the standard (S3.1).

K. Tags for AudCal

Tag Names	Tag ID	Type	nDx	Notes
TAG_IO_FFT_CURSOR_POSITION	0x43544646	TypeInt		Was TAG_IO_FFT_MAX_TRACK_CURSOR
TAG_IO_AUD_FFT_MAX	0x4d544646	TypeFloatSeries		
TAG_IO_FFT_GRAPH	0x47544646	TypeFloatSeries		
TAG_IO_FFT_GRAPH_SPAN	0x53474646	TypeFloatSeries		
TAG_IO_FFT_CURSOR_SPAN	0x53434646	TypeFloatSeries		
TAG_IO_FFT_THD_SERIES	0x50544646	TypeFloatSeries		
TAG_IO_FFT_HARMONIC_SERIES	0x48544646	TypeFloatSeries		
TAG_IO_FFT_HARMONIC_SERIES_MAX	0x48544646	TypeFloatSeries		
TAG_IO_AUD_THD_SERIES	0x44485441	TypeFloatSeries		
TAG_IO_FFT_THD_PLUS_NOISE	0x4E445446	TypeFloat		
TAG_IO_AUD_FM_MOD_RATE	0x524D4D46	TypeFloat		
TAG_IO_AUD_FM_CARRIER_FREQ	0x46434D46	TypeFloat		
TAG_IO_AUD_FM_MAX	0x584D4D46	TypeFloat		
TAG_IO_AUD_FM_MIN	0x4E4D4D46	TypeFloat		
TAG_IO_AUD_PULSE_RISE	0x45535250	TypeFloat		
TAG_IO_AUD_PULSE_FALL	0x4C414650	TypeFloat		
TAG_IO_AUD_PULSE_ON	0x4E4C5550	TypeFloat		
TAG_IO_AUD_PULSE_OFF	0x464C5550	TypeFloat		
TAG_IO_AUD_PULSE_ON_OFF_RATIO	0x524C5550	TypeFloat		
TAG_IO_AUD_PULSE_PLATEAU	0x54414C50	TypeFloat		
TAG_IO_AUD_PULSE_OVERSHOOT	0x534F5550	TypeFloat		
TAG_IO_AUD_NB_NOISE_LEVEL	0x4C4E424E	TypeFloatSeries		
TAG_IO_AUD_HIGHPASS_SPL	0x4C505348	TypeFloat		
TAG_IO_AUD_MEASUREMENT_GET	0x41444D47	TypeInt		
TAG_ACZ_SPL_S	0x414C3039	FloatWithFlagsSeries		
TAG_ACZ_SPL_F	0x414C3130	FloatWithFlagsSeries		
TAG_ACZ_SPL_I	0x414C3131	FloatWithFlagsSeries		
TAG_IO_ANYLEVEL_SPL	0x4C505341	TypeFloatWithFlags		
TAG_IO_OBA_SERIES_LIVE_1_3	0x334C424F	TypeFloatSeries		

Tag Names	Tag ID	Type	nDx	Notes
TAG_IO_SYS_NOISE_FLOOR	0x30464E53	TypeFloat		
TAG_IO_SYS_NOISE_FLOOR_A	0x41464E53	TypeFloat		
TAG_IO_SYS_NOISE_FLOOR_C	0x43464E53	TypeFloat		
TAG_IO_SYS_NOISE_FLOOR_Z	0x5A464E53	TypeFloat		
TAG_IO_SYS_UNDER_RANGE	0x30444E55	TypeFloat		
TAG_IO_SYS_UNDER_RANGE_A	0x41444E55	TypeFloat		
TAG_IO_SYS_UNDER_RANGE_C	0x43444E55	TypeFloat		
TAG_IO_SYS_UNDER_RANGE_Z	0x5A444E55	TypeFloat		
TAG_IO_SYS_PEAK_OVERLOAD	0x444C4f50	TypeFloat		

L. FFT With SoundAdvisor Model 831C (Ver 4.6.0)

Data Tags

Refer to Appendix H New Tags for Data – SoundAdvisor Only for explanation of columns.

Tag Names	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_FFT_CURSOR_POSITION	0x43544646	TypeInt				
TAG_IO_FFT_MAX	0x58544646	TypeFloatSeries				
TAG_IO_AUD_FFT_MAX	0x4d544646	TypeFloatSeries				Used for getting FFT data in AudCal
TAG_IO_FFT_GRAPH	0x47544646	TypeFloatSeries				
TAG_IO_FFT_GRAPH_SPAN	0x53474646	TypeFloatSeries	0	20000		
TAG_IO_FFT_CURSOR_SPAN	0x53434646	TypeFloatSeries				
TAG_IO_FFT_THD_SERIES	0x50544646	TypeFloatSeries				
TAG_IO_FFT_HARMONIC_SERIES	0x48544646	TypeFloatSeries				
TAG_IO_FFT_HARMONIC_SERIES_MAX	0x48544646	TypeFloatSeries				
TAG_IO_AUD_THD_SERIES	0x44485441	TypeFloatSeries				Used for getting THD data in AudCal
TAG_IO_FFT_THD_PLUS_NOISE	0x4E445446	TypeFloat				
TAG_IO_FFT_MAX	0x58544646	TypeFloatSeries				
TAG_IO_FFT_GRAPH	0x47544646	TypeFloatSeries				
TAG_IO_FFT_FLAGS	0x474C4646	TypeUInt				
TAG_IO_FFT_TIME_STAMP	0x53544646	TypeUIntSeries				
TAG_IO_FFT_DURATION	0x52554446	TypeFloat				
TAG_IO_FFT_COUNT	0x544E4346	TypeUInt				
TAG_IO_FFT_OVERLOAD_COUNT	0x434C4F46	TypeUInt				
TAG_IO_FFT_OVERLOAD_TIME	0x544C4F46	TypeFloat				
TAG_IO_FFT_OVERLOAD_PERCENT	0x504C4F46	TypeFloat				
TAG_IO_FFT_LEQ	0x51454C46	TypeFloat	-99			
TAG_IO_FFT_LPEAK	0x4B455046	TypeFloat	-99			

Tag Names	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_FFT_NUM_LINES	0x4C4E4646	TypeUInt				
TAG_IO_FFT_LIVE_MAX_LEVEL_AND_INDEX	0x58544646	TypeFloatSeries				
TAG_IO_FFT_AVG_MAX_LEVEL_AND_INDEX	0x41584D46	TypeFloatSeries				
TAG_IO_FFT_MAX_MAX_LEVEL_AND_INDEX	0x4D584D46	TypeFloatSeries				
TAG_IO_FFT_ARCHIVE_NAME	0x4E414646	TypeUInt				
TAG_IO_FFT_HIST_NUM_RECORDS	0x524E4846	TypeInt				
TAG_IO_FFT_HIST_LOAD_RECORD	0x444C4846	TypeInt				
TAG_IO_FFT_HIST_CUR_RECORD	0x52434846	TypeInt				
TAG_IO_FFT_HIST_CLOSE	0x43484646	TypeInt				
TAG_IO_FFT_GRAPH_AVG	0x47544646	TypeFloatSeries				
TAG_IO_FFT_GRAPH_MAX	0x47544646	TypeFloatSeries				
TAG_IO_FFT_SPECTRUM	0x43455046	TypeFloatSeries			0 or 1	0 = Average, 1 = Max
TAG_IO_FFT_SPECTRUM_AVG	0x43455046	TypeFloatSeries			0	Same as Spectrum with nDx 0
TAG_IO_FFT_SPECTRUM_MAX	0x43455046	TypeFloatSeries			1	Same as Spectrum with nDx 1

Tonality Info Tags (ISO 1996-02)

Tag Names	Tag ID	Type	Min	Max	nDx	Notes
TAG_IO_FFT_CURSOR_POSITION	0x43544646	TypeInt				
TAG_IO_FFT_MAX	0x58544646	TypeFloatSeries				
TAG_IO_AUD_FFT_MAX	0x4d544646	TypeFloatSeries				
TAG_IO_FFT_GRAPH	0x47544646	TypeFloatSeries				
TAG_IO_FFT_GRAPH_SPAN	0x53474646	TypeFloatSeries				
TAG_IO_FFT_CURSOR_SPAN	0x53434646	TypeFloatSeries				

M. Command Line Export from G4

Command-line commands may be used to export audio and data in a variety of ways. Data may be exported to Xlsx, XML, or JSON files. Command-line commands are executed as in the following example:

```
G4.exe -exportAudio all
```

The following table details how command-line parameters are used.

Input File Type	Output File Type	Command	Shortcut Command	Output Tags	Log File	Notes
.ldbin File	Audio File(s) (.wav or .ogg files)	-exportAudio	-ea	all, vme, vm, ve, me, v, m, e *(must use one, and only one, of these tags)*	AudioExportLog.txt* CommandLineLog.txt*	Tag specifying type of audio to export is required. Parameter can be "**all**" or any combination of "**v**", "**m**", "**e**" (for Voice, Measurement or Event). Must use "**-inputFile**" or "**-if**" to specify input file name and/or folder. If a folder is provided as the path without a file name, then all files in that folder will be processed. If a file name is provided without a path, the default input folder specified in G4 Options will be used and the file will be expected to be found there. May use optional "**-destFolder**" or "**-df**" to specify output folder. If the tag is not included, the default output folder specified in G4 Options is used. The output file will have the same name as the input file with the appropriate extension for the file type. Example: **-exportAudio All -if "c:\temp\LDBIN Files\Airplane.ldbin" -df "c:\temp\Audio Files"** Example: **-ex ve -if "c:\temp\LDBIN Files" -df "c:\temp\Audio Files"**
.ldbin File	XLSX File	-exportXlsx	-ex	*none*	FileExportLog.txt* CommandLineLog.txt*	Must use "**-inputFile**" or "**-if**" to specify input file name and/or folder. If a folder is provided as the path without a file name, then all files in that folder will be processed. If a file name is provided without a path, the default input folder specified in G4 Options will be used and the file will be expected to be found there. May use optional "**-destFolder**" or "**-df**" to specify output folder. If the tag is not included, the default output folder specified in G4 Options is used. The output file will have the same name as the input file with the appropriate extension for the file type. The "**xml**", "**json**", "**xlsx**" and "**ldbin**" tags specify the type of output file that will be created: XML, JSON, or XLSX (Excel spreadsheet) or LDBIN. LDBIN to XLSX Example: **-exportXlsx -inputFile "c:\temp\LDBIN Files" -destFolder "c:\temp\XLSX Files"** LDBIN to XML Example: **-exportXml xml -if "c:\temp\LDBIN Files\Airplane.ldbin" -destFolder "c:\temp\XML Files"** LDBIN to JSON Example: **-el json -if "c:\temp\LDBIN Files\ Airplane.ldbin" -df "c:\temp\JSON Files"** Archive to XML Example: **-exportUsb xml -if "c:\temp\LDBIN Files\Airplane.s" -df "c:\temp\XML Files"** Archive to LDBIN Example: **-eu ldbin -if "c:\temp\LDBIN Files\Airplane.s" -df "c:\temp\LDBIN Files"** Archive to JSON Example: **-eu json -if "c:\temp\LDBIN Files\Airplane.s" -df "c:\temp\JSON Files"** Archive to XLSX Example: **-eu xlsx -if "c:\temp\LDBIN Files" -df "c:\temp\XLSX Files"** **Note:** For "**-exportUsb**" or "**-eu**" commands, an LDBIN file will always be created in the output folder. The "**ldbin**" tag is used when no other files should be created.
.ldbin File	XML File	-exportXml	-el	xml *or* json *(must use one, and only one, of these tags)*	FileExportLog.txt* CommandLineLog.txt*	
.ldbin File	JSON File					
Archive Files *Folders with ".s", ".r", or ".f" extension, which are usually stored by the meter to USB flash drive*	LDBIN, XML, JSON, XLSX File	-exportUsb	-eu	xml *or* ldbin *or* json *or* xlsx *(must use one, and only one, of these tags)*	FileExportLog.txt* CommandLineLog.txt*	

* For all of the above commands, the log file will be created in the user's "**Documents\PCB Piezotronics\G4\Log Files**" folder.

G4 Options, Command Line Options Tab:

1. set pattern to define the output file name
2. set folder where input file will be found if path is not specified in "-inputFile" or "-if"
3. set output folder if "-destFolder" or "-df" are not present

Results log file will be written to the **User**'s folders:
Documents\PCB Piezotronics\G4\Logs\AudioExportLog.txt (for audio exports)
Documents\PCB Piezotronics\G4\Logs\XlsxExportLog.txt (for XLSX and XML file exports)

N. Tonality Analysis

To use the tonality analysis functionality provided by LDFirmwareUtilities.dll in a managed solution you must reference it as usual and use or import the namespace “LarsonDavis.UnmanagedUtils” in the source code file(s) it will be used in, or you can use the fully qualified class name “LarsonDavis.UnmanagedUtils.Tonality”.

The Tonality class contains the functions described below as well as three constants that can be used to verify data before calling the Tonality functions, MAX_FFT_LINES (6401), MAX_FREQ_SPAN (20000), MIN_FREQ_SPAN (100).

ISO 1996-2 Standard

The Tonality class provides two static functions related to initializing the utility and analyzing the data for tones according to ISO 1996-2.

```
static void Tonality.Init1996_2(
        float windowEffectiveBw,
        float toneSeekDelta,
        float regressionMult);
```

```
static IList<Tone1996_2>^ Tonality.GetTonality1996_2(
        array<float>^ fftData,
        int freqSpan);
```

To initialize the utility call Init1996_2, then all subsequent calls to GetTonality1996_2 will use these parameters until Init1996_2 is called again.

To analyze an FFT spectra call GetTonality1996_2 with an array of floats containing the FFT spectra in Decibels (Db) to analyze. The spectra cannot be larger than 6400 lines + DC bin (6401 lines in total) and freqSpan must be between 100 and 20000 inclusive, freqSpan represents Hertz. In addition to the tones detected that meet the ISO 1996-2 standard addition possible tones will be identified and marked with statis indicator noting if they meet the ISO standard or not.

All calls to GetTonality1996_2 must be completed before subsequent calls to Init1996_2 as this may have unexpected results.

The detected tones are returned in an IList<Tone1996_2>, the element type is defined below.

```
public struct Tone1996_2
{
    // Metrics
    float      ToneFreq;      // Fc
    float      ToneLevel;     // Lpt
    float      MaskingNoise;  // Lpn
    float      CriticalBw;    // Bcrit
    float      Audibility;    // Lta
    float      Adjustment;    // Kt

    // Analysis bandwidths
    float      Bw3db;         // 3-dB bandwidth that was used to detect tones
    float      BwEff;         // Effective analysis bandwidth of the window used

    // Status
    bool       b3DbBwOk;
    bool       bEffBwOk;
    bool       bRegLineValid;

    // Regression line info
    int        RegLineStart;
    int        RegLineEnd;
    float      Slope;
    float      Intercept;

    // Data-point indexes
    int        ToneIndex;     // Offset of Fc in the data array
    int        CritBandStart; // Index of the beginning of the critical band
    int        CritBandEnd;   // Index of the end of the critical band
};
```

ISO 20065

In addition to using the updated standard and only allowing standard usage, improvements were also made to support multi-threading making it possible to process a data set more quickly.

The Tonality class provides two functions to analyze the data but does not require a separate setup step as there are no configurable parameters for setup. Instead the two analysis functions allow data to be provided in either Decibels (Db) or in Volts Squared (Vsq). FFT files generated by an 831C store their data in Volts Squared but are converted to Db during typical processing.

```
static IList<Tone20065>^ Tonality.GetTonality20065Db(
        array<float>^ spectraData,
        uint32_t freqSpan);
```

```
static IList<Tone20065>^ Tonality.GetTonality20065Vsq(
        array<float>^ spectraData,
        uint32_t freqSpan);
```

Each call is thread-safe, meaning that multiple threads and all call these functions at the same time and it will not affect the results. As with ISO 1996-2, spectraData cannot be larger than 6401 (6400 + DC bin), and freqSpan must be between 100 and 20000 (Hz) inclusive.

The detected tones are returned in an IList<Tone20065>, the element type is defined below along with contained structs.

```
public struct Tone20065
{
   float      Frequency;     //          - Tone Frequency
   float      Level;         // LT       - Tone level
   float      Audibility;    // Delta_L  - Tone audibility
   float      Uncertainty;   //          - Uncertainty
   float      MaskingNoise;  // Ls       - Mean narrow-band masking noise
   float      LevelCB;      // LG       - Masking noise adjusted for CB
   ToneBounds CB;            // CB start/end frequency
   ToneBounds CbNoise;       // Noise level at CB start/end
   Uint       Consolidated;  // Effective bool, true/false
};

public value struct ToneBounds
{
   float Start;
   float End;
};
```

O. HVM200 Properties

The list of properties for the HVM200 work similar to the 831. See Section Get and Set Properties.

Measurement Properties

Name	TagId	Index	Type	Precision
TAG_REPORT_HEADER_0_6	6	0	TypeString	0
TAG_REPORT_HEADER_1_7	7	0	TypeString	0
TAG_REPORT_HEADER_2_8	8	0	TypeString	0
TAG_OPERATING_MODE_20	20	0	TypeInt	0
TAG_DETECTOR_RATE_21	21	0	TypeInt	0
TAG_GAIN_X_22	22	0	TypeInt	0
TAG_GAIN_Y_23	23	0	TypeInt	0
TAG_GAIN_Z_24	24	0	TypeInt	0
TAG_STORE_TIME_28	28	0	TypeInt	0
TAG_ACCELEROMETER_30	30	0	TypeInt	0
TAG_DISPLAY_UNIT_31	31	0	TypeInt	0
TAG_INTEGRATION_32	32	0	TypeInt	0
TAG_SUM_FACTOR_KX_34	34	0	TypeFloat	4
TAG_SUM_FACTOR_KY_35	35	0	TypeFloat	4
TAG_SUM_FACTOR_KZ_36	36	0	TypeFloat	4
TAG_WEIGHTING_X_37	37	0	TypeInt	0
TAG_WEIGHTING_Y_38	38	0	TypeInt	0
TAG_WEIGHTING_Z_39	39	0	TypeInt	0
TAG_AC_DC_OUTPUT_X_40	40	0	TypeInt	0
TAG_AC_DC_OUTPUT_Y_41	41	0	TypeInt	0
TAG_AC_DC_OUTPUT_Z_42	42	0	TypeInt	0
TAG_AUTO_STORE_43	43	0	TypeInt	0
TAG_HISTORY_VALUE_44	44	0	TypeInt	0
TAG_SETUP_OR_FILE_NAME_45	45	0	TypeString	0
TAG_SENSITIVITY_X_46	46	0	TypeFloat	8
TAG_SENSITIVITY_Y_47	47	0	TypeFloat	8
TAG_SENSITIVITY_Z_48	48	0	TypeFloat	8
TAG_PRINT_HISTORY_50	50	0	TypeInt	0
TAG_DB_REFERENCE_51	51	0	TypeInt	0
TAG_HAND_ARM_EXPOSURE_RE_52	52	0	TypeInt	0
TAG_STORE_RAW_53	53	0	TypeInt	0
TAG_FULL_OCTAVE_54	54	0	TypeInt	0
TAG_THIRD_OCTAVE_55	55	0	TypeInt	0
TAG_START_TIME_56	56	0	TypeInt	0
TAG_DURATION_57	57	0	TypeInt	0
TAG_DELAY_58	58	0	TypeInt	0
TAG_BASENAME_61	61	0	TypeString	0

TAG_START_DATE_62	62	0	TypeInt	0
TAG_END_DATE_63	63	0	TypeInt	0
TAG_ENABLE_SCHED_64	64	0	TypeInt	0

Example of Measurement Properties:

```
1.  TAG_ACCELEROMETER_30: 1
2.  TAG_ACCEL_SN: "16389"
3.  TAG_AC_DC_OUTPUT_X_40: 0
4.  TAG_AC_DC_OUTPUT_Y_41: 0
5.  TAG_AC_DC_OUTPUT_Z_42: 0
6.  TAG_AUTO_STORE_43: 0
7.  TAG_BASENAME_61: "HVMD"
8.  TAG_DB_REFERENCE_51: 1
9.  TAG_DELAY_58: 0
10. TAG_DETECTOR_RATE_21: 1
11. TAG_DISPLAY_UNIT_31: 0
12. TAG_DURATION_57: 0
13. TAG_ENABLE_SCHED_64: 0
14. TAG_END_DATE_63: 0
15. TAG_EXPOSUREACTION_66: 2.5
16. TAG_EXPOSURELIMIT_65: 5
17. TAG_FULL_OCTAVE_54: 1
18. TAG_GAIN_X_22: 0
19. TAG_GAIN_Y_23: 0
20. TAG_GAIN_Z_24: 0
21. TAG_HAND_ARM_EXPOSURE_RE_52: 0
22. TAG_HISTORY_VALUE_44: 0
23. TAG_IDLE_SHUTODWN_67: 0
24. TAG_INTEGRATION_32: 0
25. TAG_OPERATING_MODE_20: 1
26. TAG_PRINT_HISTORY_50: 0
27. TAG_REPORT_HEADER_0_6: "NA"
28. TAG_REPORT_HEADER_1_7: "NA"
29. TAG_REPORT_HEADER_2_8: "NA"
30. TAG_SENSITIVITY_X_46: 95.7106
31. TAG_SENSITIVITY_Y_47: 95.5958
32. TAG_SENSITIVITY_Z_48: 96.8078
33. TAG_SETUP_OR_FILE_NAME_45: "HandArm"
34. TAG_START_DATE_62: 0
35. TAG_START_TIME_56: 0
36. TAG_STORE_RAW_53: 0
37. TAG_STORE_TIME_28: 0
38. TAG_SUM_FACTOR_KX_34: 1
39. TAG_SUM_FACTOR_KY_35: 1
40. TAG_SUM_FACTOR_KZ_36: 1
41. TAG_THIRD_OCTAVE_55: 1
42. TAG_WEIGHTING_X_37: 4
43. TAG_WEIGHTING_Y_38: 4
44. TAG_WEIGHTING_Z_39: 4
```

System Properties

Name	TagId	Index	Type	Precision
TAG_MASK_OPTION	59	0	TypeUInt	0
TAG_OPTION_FLAGS	60	0	TypeUInt	0
TAG_MODEL	TAG_MODEL	0	TypeString	0

Example of System Properties

```
1. TAG_MASK_OPTION: 7
2. TAG_MODEL: "HVM200"
3. TAG_OPTION_FLAGS: 4294967295
```

P. 730/721/821 Properties

The list of properties for the 730, 721, and 821. See Section Get and Set Properties with JSON.

Measurement Properties

Measurement Properties Name	Type	730	721/821
TAG_SPL_FREQ_WEIGHT	Int	Yes	Yes
TAG_SPL_TIME_WEIGHT	Int	Yes	Yes
TAG_PEAK_FREQ_WEIGHT	Int	Yes	Yes
TAG_DOS1_ENABLED	Int	Yes	Yes
TAG_DOS1_CONFIG_SELECT	Int	Yes	Yes
TAG_DOS1_MODE	Int	Yes	Yes
TAG_DOS1_TITLE	String	Yes	Yes
TAG_DOS1_FREQ_WEIGHT	Int	Yes	Yes
TAG_DOS1_TIME_WEIGHT	Int	Yes	Yes
TAG_DOS1_PEAK_WEIGHT	Int	Yes	Yes
TAG_DOS1_EXCH_RATE	Int	Yes	Yes
TAG_DOS1_THRES_ENABLE	Int	Yes	Yes
TAG_DOS1_THRES_LEVEL	Float	Yes	Yes
TAG_DOS1_CRIT_LEVEL	Float	Yes	Yes
TAG_DOS1_CRIT_TIME	Float	Yes	Yes
TAG_DOS1_SHIFT_TIME	Float	Yes	Yes
TAG_DOS2_ENABLED	Int	Yes	Yes
TAG_DOS2_CONFIG_SELECT	Int	Yes	Yes
TAG_DOS2_MODE	Int	Yes	Yes
TAG_DOS2_TITLE	String	Yes	Yes
TAG_DOS2_FREQ_WEIGHT	Int	Yes	Yes
TAG_DOS2_TIME_WEIGHT	Int	Yes	Yes
TAG_DOS2_PEAK_WEIGHT	Int	Yes	Yes
TAG_DOS2_EXCH_RATE	Int	Yes	Yes
TAG_DOS2_THRES_ENABLE	Int	Yes	Yes
TAG_DOS2_THRES_LEVEL	Float	Yes	Yes
TAG_DOS2_CRIT_LEVEL	Float	Yes	Yes
TAG_DOS2_CRIT_TIME	Float	Yes	Yes

Measurement Properties Name	Type	730	721/821
TAG_DOS2_SHIFT_TIME	Float	Yes	Yes
TAG_DOS3_ENABLED	Int	Yes	Yes
TAG_DOS3_CONFIG_SELECT	Int	Yes	Yes
TAG_DOS3_MODE	Int	Yes	Yes
TAG_DOS3_TITLE	String	Yes	Yes
TAG_DOS3_FREQ_WEIGHT	Int	Yes	Yes
TAG_DOS3_TIME_WEIGHT	Int	Yes	Yes
TAG_DOS3_PEAK_WEIGHT	Int	Yes	Yes
TAG_DOS3_EXCH_RATE	Int	Yes	Yes
TAG_DOS3_THRES_ENABLE	Int	Yes	Yes
TAG_DOS3_THRES_LEVEL	Float	Yes	Yes
TAG_DOS3_CRIT_LEVEL	Float	Yes	Yes
TAG_DOS3_CRIT_TIME	Float	Yes	Yes
TAG_DOS3_SHIFT_TIME	Float	Yes	Yes
TAG_DOS4_ENABLED	Int	Yes	Yes
TAG_DOS4_CONFIG_SELECT	Int	Yes	Yes
TAG_DOS4_MODE	Int	Yes	Yes
TAG_DOS4_TITLE	String	Yes	Yes
TAG_DOS4_FREQ_WEIGHT	Int	Yes	Yes
TAG_DOS4_TIME_WEIGHT	Int	Yes	Yes
TAG_DOS4_PEAK_WEIGHT	Int	Yes	Yes
TAG_DOS4_EXCH_RATE	Int	Yes	Yes
TAG_DOS4_THRES_ENABLE	Int	Yes	Yes
TAG_DOS4_THRES_LEVEL	Float	Yes	Yes
TAG_DOS4_CRIT_LEVEL	Float	Yes	Yes
TAG_DOS4_CRIT_TIME	Float	Yes	Yes
TAG_DOS4_SHIFT_TIME	Float	Yes	Yes
TAG_ALARM1_ENABLE	Int	Yes	Yes

Measurement Properties Name	Type	730	721/821
TAG_ALARM1_LED_INDICATOR	Int	Yes	Yes
TAG_ALARM1_SOURCE	Int	Yes	Yes
TAG_ALARM1_ACTION_LEVEL	Float	Yes	Yes
TAG_ALARM1_LIMIT_LEVEL	Float	Yes	Yes
TAG_ALARM2_ENABLE	Int	Yes	Yes
TAG_ALARM2_LED_INDICATOR	Int	Yes	Yes
TAG_ALARM2_SOURCE	Int	Yes	Yes
TAG_ALARM2_ACTION_LEVEL	Float	Yes	Yes
TAG_ALARM2_LIMIT_LEVEL	Float	Yes	Yes
TAG_TIME_HIST_ENABLE	Int	Yes	Yes
TAG_TIME_HIST_PERIOD	Int	Yes	Yes
TAG_TIME_HIST_METRIC	Int	Yes	Yes
TAG_OBA_ENABLE	Int	Yes	Yes
TAG_OBA_FREQ_WEIGHT	Int	Yes	Yes
TAG_EVENT_SR_ENABLE	Int	Yes	N/A
TAG_EVENT_SR_TRIG_SRC	Int	Yes	N/A
TAG_EVENT_SR_TRIG_LVL	Float	Yes	N/A
TAG_EVENT_SR_MIN_INTERVAL	Int	Yes	N/A
TAG_EVENT_SR_PERIOD	Int	Yes	N/A
TAG_EVENT_SR_PRE_PERIOD	Int	Yes	N/A
TAG_SPL1_LVL	Float	Yes	Yes
TAG_SPL2_LVL	Float	Yes	Yes
TAG_PEAK1_LVL	Float	Yes	Yes
TAG_PEAK2_LVL	Float	Yes	Yes
TAG_PEAK3_LVL	Float	Yes	Yes
TAG_AUTO_CAL_ENABLE	Int	Yes	Yes
TAG_TIMER_MODE	Int	Yes	Yes
TAG_TIMER_START_DATE	String	Yes	Yes
TAG_TIMER_STOP_DATE	String	Yes	Yes
TAG_TIMER1_START	String	Yes	Yes
TAG_TIMER1_STOP	String	Yes	Yes
TAG_TIMER2_ENABLE	Int	Yes	Yes
TAG_TIMER2_START	String	Yes	Yes
TAG_TIMER2_STOP	String	Yes	Yes
TAG_TIMER3_ENABLE	Int	Yes	Yes
TAG_TIMER3_START	String	Yes	Yes
TAG_TIMER3_STOP	String	Yes	Yes

Measurement Properties Name	Type	730	721/821
TAG_TIMED_STOP_DURATION	String	Yes	Yes
TAG_CONT_MODE_INTERVAL	Int	Yes	Yes
TAG_DAY_TIME	String	N/A	Yes
TAG_EVENING_TIME	String	N/A	Yes
TAG_NIGHT_TIME	String	N/A	Yes
TAG_EVENING_PENALTY	Float	N/A	Yes
TAG_NIGHT_PENALTY	Float	N/A	Yes
TAG_LN_ENABLE	Int	N/A	Yes
TAG_LN_PERC_1	Float	N/A	Yes
TAG_LN_PERC_2	Float	N/A	Yes
TAG_LN_PERC_3	Float	N/A	Yes
TAG_LN_PERC_4	Float	N/A	Yes
TAG_LN_PERC_5	Float	N/A	Yes
TAG_LN_PERC_6	Float	N/A	Yes
TAG_LN_FREQ_WT	Int	N/A	Yes
TAG_LN_TIME_WT	Int	N/A	Yes
TAG_MEAS_HIST_ENABLE	Int	N/A	Yes
TAG_MEAS_HIST_INTERVAL	Int	N/A	Yes
TAG_20DB_GAIN_ENABLE	Int	N/A	Yes
TAG_TIME_HIST_OBA_ENABLE	Int	N/A	Yes
TAG_MARKER_1_NAME	String	N/A	Yes
TAG_MARKER_2_NAME	String	N/A	Yes
TAG_MARKER_3_NAME	String	N/A	Yes
TAG_MARKER_4_NAME	String	N/A	Yes
TAG_MARKER_5_NAME	String	N/A	Yes
TAG_MARKER_6_NAME	String	N/A	Yes
TAG_MARKER_7_NAME	String	N/A	Yes
TAG_MARKER_8_NAME	String	N/A	Yes
TAG_MARKER_9_NAME	String	N/A	Yes
TAG_MARKER_10_NAME	String	N/A	Yes
TAG_TIME_HIST_PWR_ENABLE	Int	N/A	Yes
TAG_OBA_TIME_WT	Int	N/A	Yes
TAG_TIME_HIST_MIN_MAX	Int	N/A	Yes
TAG_TIME_HIST_FLAGS_0	Int	N/A	Yes
TAG_TIME_HIST_FLAGS_1	Int	N/A	Yes
TAG_TIME_HIST_FLAGS_2	Int	N/A	Yes
TAG_TIME_HIST_FLAGS_2	Int	N/A	Yes

System Properties

System Properties Name	Type	730	721/821
TAG_DEVICE_NAME	String	Yes	Yes
TAG_DATE_FORMAT	Int	Yes	Yes
TAG_NOISE_LEVEL	Float	Yes	Yes
TAG_DB_REF	Float	Yes	Yes
TAG_LANGUAGE	Int	Yes	Yes
TAG_DECIMAL_SEPARATOR	Int	Yes	Yes
TAG_AUTO_OFF_TIME	Int	Yes	Yes
TAG_BLE_AUTH_KEY	String	Yes	Yes
TAG_TIME	String	Yes	Yes
TAG_DATE	String	Yes	Yes
TAG_CAL_LEVEL	Float	Yes	Yes
TAG_SENSITIVITY	Float	Yes	Yes
TAG_EPOCH_TIME	UInt	Yes	Yes
TAG_ADMIN_PASS	String	Yes	Yes
TAG_ADMIN_FLAGS	Int	Yes	Yes
TAG_MODEL_INDEX	Int	Yes	Yes
TAG_OPTIONS	Int	Yes	Yes
TAG_SERIAL	String	Yes	Yes
TAG_MODEL	String	Yes	Yes
TAG_AUTO_STORE_ENABLE	Int	N/A	Yes
TAG_BACKLIGHT	Int	N/A	Yes
TAG_UI_METRIC_1	Int	N/A	Yes
TAG_UI_METRIC_2	Int	N/A	Yes
TAG_UI_METRIC_3	Int	N/A	Yes
TAG_UI_METRIC_4	Int	N/A	Yes
TAG_UI_METRIC_5	Int	N/A	Yes
TAG_UI_METRIC_6	Int	N/A	Yes
TAG_UI_METRIC_7	Int	N/A	Yes

System Properties Name	Type	730	721/821
TAG_UI_METRIC_8	Int	N/A	Yes
TAG_UI_METRIC_9	Int	N/A	Yes
TAG_UI_METRIC_10	Int	N/A	Yes
TAG_UI_METRIC_11	Int	N/A	Yes
TAG_UI_METRIC_12	Int	N/A	Yes
TAG_UI_METRIC_13	Int	N/A	Yes
TAG_LED_LIGHT_ENABLE	Int	N/A	Yes
TAG_GRAPH_UP_BOUND	Int	N/A	Yes
TAG_GRAPH_LOW_BOUND	Int	N/A	Yes
TAG_OBA_GRAPH_UP_BOUND	Int	N/A	Yes
TAG_OBA_GRAPH_LOW_BOUND	Int	N/A	Yes
TAG_CORRECTION_SELECT	Int	N/A	Yes
TAG_PREAMP_SERIAL	String	N/A	Yes
TAG_MIC_SELECT	Int	N/A	Yes
TAG_MIC_SERIAL	String	N/A	Yes
TAG_MIC_MODEL	String	N/A	Yes
TAG_MIC_SENSITIVITY_DB	Float	N/A	Yes
TAG_MIC_SENSITIVITY_MV	Float	N/A	Yes
TAG_DEVICE_DESCRIPTION	String	N/A	Yes
TAG_AUTO_SLEEP_TIME	Int	N/A	Yes
TAG_TH_METRIC_1	Int	N/A	Yes
TAG_TH_METRIC_2	Int	N/A	Yes
TAG_TH_GRAPH_UP_BOUND	Int	N/A	Yes
TAG_TH_GRAPH_LOW_BOUND	Int	N/A	Yes
TAG_DC_OUT_ENABLE	Int	N/A	Yes
TAG_DAC_OUT_ADJUST	Int	N/A	Yes
TAG_EXT_SHUTOFF_VOLT	Float	N/A	Yes

Index