LinQ Documentation

Release 1

Altronix

Mar 12, 2019

Contents

The Altronix SDK's are available in the following languages

- Cloud or LAN
- Secured with TLS
- Free Support
- Docs: *LinQJS SDK*

- Light weight
- Free Support

- Windows enviorments
- Secure with TLS
- Free Support
- Docs: *C# SDK*

CHAPTER 1

Contents

1.1 LinQJS SDK

1.1.1 Overview

Host Setup

Installation

` npm install --save @altronix/linq `

Import

```
const linq = require("@altronix/linq").linq;
```

Listen for events

```
linq.on("new", device => console.log("New device connected! [%s]", device.serial()))
    .on("alert", (device, alert) => console.log("Received alert from device!"))
    .on("error", error => console.log(error));
```

Start LinQJS

```
const config = {
  key: key,     // x509 TLS Private Key (optional)
  cert: cert,   // x509 TLS Public Certificate (optional)
```

(continues on next page)

(continued from previous page)

```
  tcps: 99443,  // Secure Port for incoming secure device connections
  tcp: 9980     // Plain text port for incoming non secure device connections
};
linq.listen(config).then(() => console.log("LinQ Ready!"))
```

Example

app.js

```
/**
 * Basic app to print alerts to console
 */

let config = require("./config"),
  linq = require("@altronix/linq").linq;

// Setup event callback handlers
linq
  .on("error", e => console.log(e))
  .on("new", d => console.log("[%s] new device", d.serial()))
  .on("alert", (device, alert, email) => {
    console.log("[%s] alert", device.serial());
  })
  .listen(config).then(() => console.log("LinQ Ready!"))
  .catch(e => console.log(e);
```

config.js

```
let fs = require("fs");
const env = process.env.NODE_ENV || "dev";

// Read a potential TLS key from the enviorment
let key;
try {
  key = fs.readFileSync(process.env.TLS_KEY || "../enviorments/unsafe-key.pem");
  console.log("TLS KEY FOUND");
} catch (e) {
  console.log("No TLS key found");
}

// Read a potential TLS cert from the enviorment
let cert;
try {
  cert = fs.readFileSync(
    process.env.TLS_CERT || "../enviorments/unsafe-cert.pem"
  );
  console.log("TLS CERT FOUND");
} catch (e) {
  console.log("No TLS cert found");
}

const dev = {
  key: key,
  cert: cert,
  tcp: parseInt(process.env.TCP_PORT) || 3380,
```

(continues on next page)

(continued from previous page)

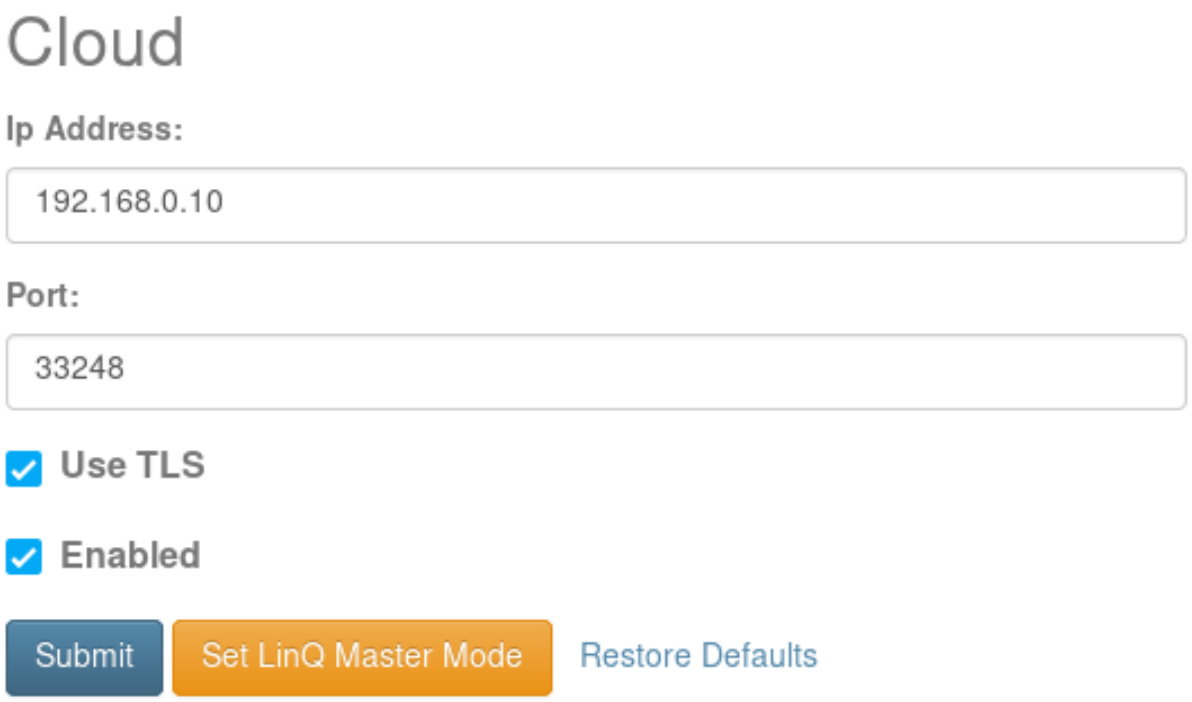
```
  tcps: parseInt(process.env.TCPS_PORT) || 33433
};

const test = {
  key: key,
  cert: cert,
  tcp: parseInt(process.env.TCP_PORT) || 3380,
  tcps: parseInt(process.env.TCPS_PORT) || 33433
};

const config = { dev, test };

module.exports = config[env];
```

Device Setup

Enable your LinQ product to connect to LinQJS

Connect your Altronix LinQ enabled product to your local area network. Using your browser, connect to your Altronix LinQ enabled product and navigate to Settings->Cloud->TCP/IP. Fill out the form to enable your product to connect to LinQJS.

(see form image below)

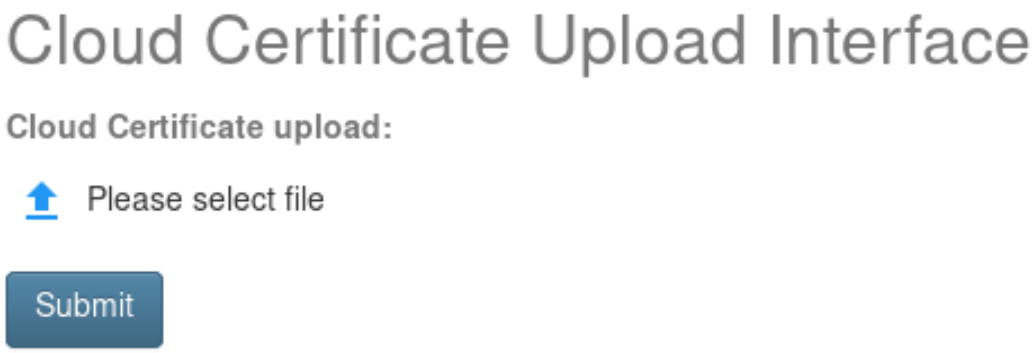


Field	Description
IP Address	IP address of the host machine running linqjs.
Port	Port number where the linqjs application is listening.
Use TLS	If the linqjs application is using tls, select the "Use TLS" property.
Enabled	To enable the device to establish a connection to the host machine running linqjs, select "Enabled" property.

Enable security

To ensure a secure connection between your device and the LinQJS host machine, you must setup a secure connection with TLS. The host machine has a public and private key pair. The device must be installed with the public key only.

(see form image below)



Field	Description
Cloud Upload	Upload the public key (.PEM), then click submit.

1.1.2 Tips and Tricks

Manage device timeouts

When a device connects to linqjs, the device contexts are cached indefinitely. If you are interested to know when a device is not likely connected anymore you can use the "lastSeen" property on the device context.

1.1.3 API Reference

linq.listen

```
linq.listen(config: number | LinQConfig): Promise
```

The linq.listen() method takes a port number parameter, or a configuration object. To listen with TLS termination for the embedded devices you must use the configuration object.

The linq.listen() method will create a socket for the ZMTP protocol and listen for incoming device connections. linq.listen() will return a promise when the socket is ready. This is typically the first function called to start the linqjs module.

- see also
  - *linq.on* to listen for incoming alerts
  - *linq.devices* to get an object of connected devices indexed by serial number
  - *linq.shutdown* to clean up and close LinQJS module

Parameters

Parameter	type	description
config	number | LinQConfig	Config object or Port number

LinQConfig configuration object

Prop	type	required	description
cert	string	Only when using TLS	A pem format certificate
key	string	Only when using TLS	A pem format key
tcp	number	true	A pem format port number for linq listener
tcps	number	Only when using TLS	A pem format port number for linq tls termination

Example: Listen with TLS termination

```
linq.listen({
   key = fs.readFileSync(process.env.TLS_KEY),
   cert = fs.readFileSync(process.env.TLS_CERT),
   tcp = 4590,
   tcps = 4591
});
```

Example: Listen unsecure

```
linq.listen(4590);
```

linq.shutdown

```
linq.shutdown(): void
```

The linq.shutdown() method will free the resources from linq.listen() and is required to close the application gracefully.

Parameters

None

linq.on

```
linq.on(alert_type, alert_callback): Linq
```

The linq.on() method will set up a callback for the nodejs event emitter. The following alerts and callback signatures are described below.

Parameters

Parameter	type	description
alert_type	string	Config object or Port number
alert_callback	function	callback function executed when alert of type is emitted

Alerts

Parameter	callback args
new	*Device*
alert	*Device*, *Alert*
timeout	*Device*
error	string

1. new
   - Emits when a new device is connected to linq
   - A *Device* context is passed to callers event handler
2. alert
   - Emits when a a device has generated an alert
   - A device context and the alert context is passed to the callers event handler
3. timeout
   - Emites when a device has not been heard from in a period of time specified.
4. error
   - Emits when linq throws and error
   - A string describing the error is passed to the callers event handler

Example Event Handler (New Device)

```
linq.on("new", (device) => {
  console.log(`A new device is connected to linq!`);
  console.log(`Serial Number: ${device.serial()}`);
});
```

Example Event Handler (Alert from Device)

```
linq.on("alert", (device, alert) => {
  console.log(`An alert from ${device.serial()}!`);
  console.log("who: %s\nwhat: %s\nwhen: %s\n", alert.who, alert.what, alert.when);
});
```

- see also
  - *Alert*

linq.devices

```
linq.devices: {[x:string]: Device }
```

The linq.device object contains a map of devices that are indexed by the serial number of the device. The device context provides methods to interact further with the device.

- see also
  - *Device*

Example

```
app.get('/api-v1/device/:sid', function(req, res, next) {
  let device = linq.device(req.params.sid);
  if (!device) {
    res.status(404).send("device not found");
  } else {
    res.locals._device = device;
    next();
  }
});
```

linq.timeout

```
linq.timeout(ms?: number): Linq
```

The linq.timeout() method will remove known devices from the list of connected devices if they have not been seen by the application in x amount of seconds.

For example: Remove all devices that have not been heard from in over 1 day, and log the remaining devices.

```
let devices = linq.timeout(86400).devices();
console.log(devices);
```

Parameter

Parameter	type	description
ms	number	Number of seconds

Alert

Device

The main device context to. One instance is created per each device connected to linqjs

Properties

property	type	description
uptime	unix	
lastSeen	unix	
data	any	
device.product	method	return product string of device
device.serial	method	return serial number of device
device.request	method	send a request to the device
device.batch	method	send multiple requests to the device
device.update	method	update the device
device.pending	method	return how many requests are pending to the device

device.serial

```
device.serial(): string;
```

The device.serial() method will return a serial number string of the device.

Parameters

None.

device.product

```
device.product(): string;
```

The device.product() method will return the product string of the device.

Parameters

None.

Valid product strings

string	product
"LINQ2"	https://www.altronix.com/products/LINQ2
"LINQ8PD"	https://www.altronix.com/products/LINQ8PD
"LINQ8ACM"	https://www.altronix.com/products/LINQ8ACM

device.request

```
device.request(path: string): Promise;
device.request(path: string, data: object): Promise;
device.request(request: RequestParams, optional_data?: object): Promise;
```

The device.request method will add a request to the queue and return a promise when the request is fullfilled by the device. The arguments match typically HTTP syntax and the API is the same. For a list of the API, see the HTTP API section of this document.

RequestParams

parameter	type	description
path	string	A URL representing the resource for the request
data?	any	Optional data if this is a post request
timeout?	number	Optional amount of milliseconds to wait for a response (default 8 seconds)
retry?	number	Optional number of times to try request before throwing error

Supported Errors

error	description
timeout	The device took to long to respond to the request
cancel	The caller destroyed all pending requests
device	The device sent an error

Example GET Request

```
device.request("/ATX/about")
  .then(resp => console.log(resp))
  .catch(error => console.log(error));
```

Example POST Request

```
device.request("/ATX/network/ip",{ip: "192.168.0.111"})
  .then(resp => console.log(resp))
  .catch(error => console.log(error));
```

Note: The underlying transport protocol being used is ZMTP (+TLS). The API just looks like HTTP

device.batch

```
device.batch(request: RequestParams[]): Promise;
```

The device.batch method will append multiple requests to the queue and return a promise when all requests are fullfilled by the device. For a list of the API, see the HTTP API section of this document.

Example Batch Request

Change the IP address of the device and save settings into ROM

```
device.batch([{
  path: "ATX/network/ip",
  data: {ip: "192.168.0.111"}
},{
  path: "ATX/exe/save",
  data: {save:1}
}).catch(e => console.log(e));
```

- see also
  - *RequestParams*
  - *Supported Errors*

device.update

```
device.update(update: DashboardPayload): Promise;
```

The device.update method will update the firmware of the device. Dashboard updates can be found at https://linq.altronix.com

Example Firmware Update

```
let result = false;

device.update(update)
  .then(resp => result = resp)
  .catch( err => result = err);

(function _log(){
  if (result) return;
  console.log(`${device.pending()} requests remaining`);
  setTimeout(_log, 1000);
})()
```

device.pending

```
device.pending(): number;
```

The device.pending method will return the number of requests in the queue waiting to be fullfilled by the device.

Parameters

None.

Example

```
if (device.pending() > 10) {
  // ruh-roh
}
```

1.2 C# SDK

1.2.1 Overview

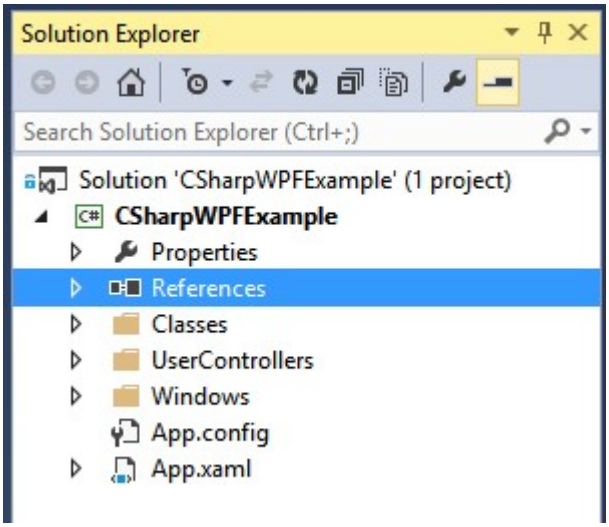
Dependencies

- Built for .NET 4.5.2 and up.
- Newtonsoft.Json

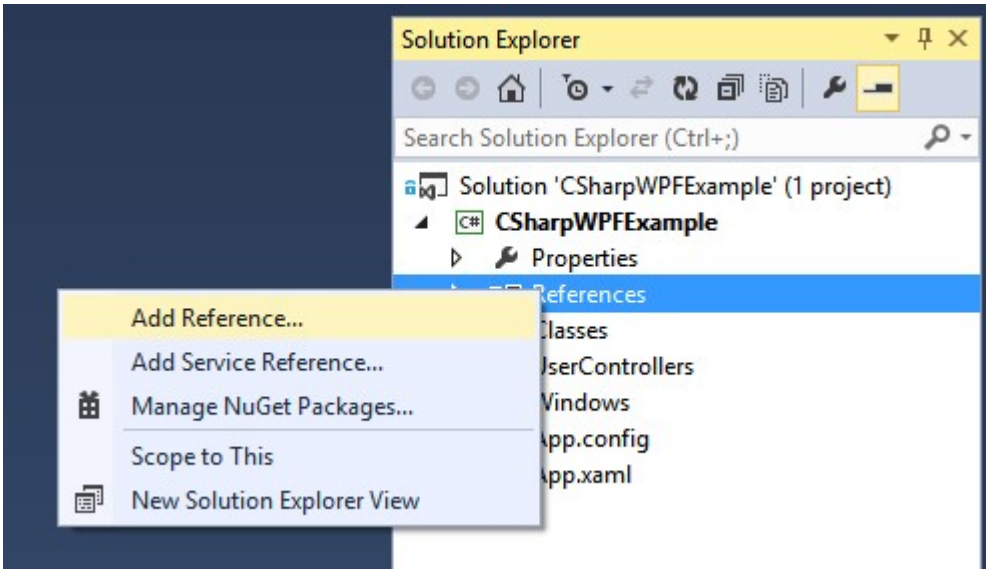
Installation

Once a project has been setup, to add C# SDK please follow the steps below:

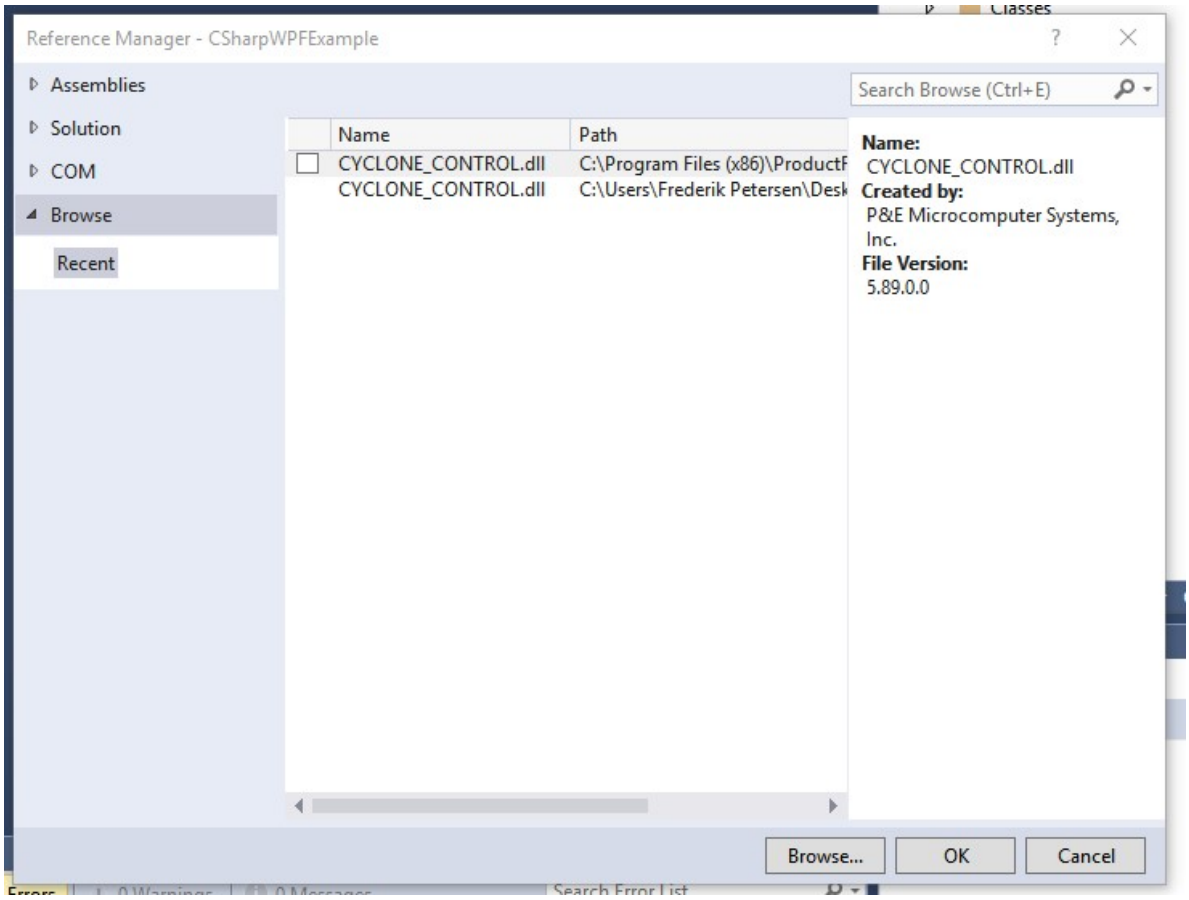
1. In the Solution Explorer, right-click the References section found under [Project name] -> References



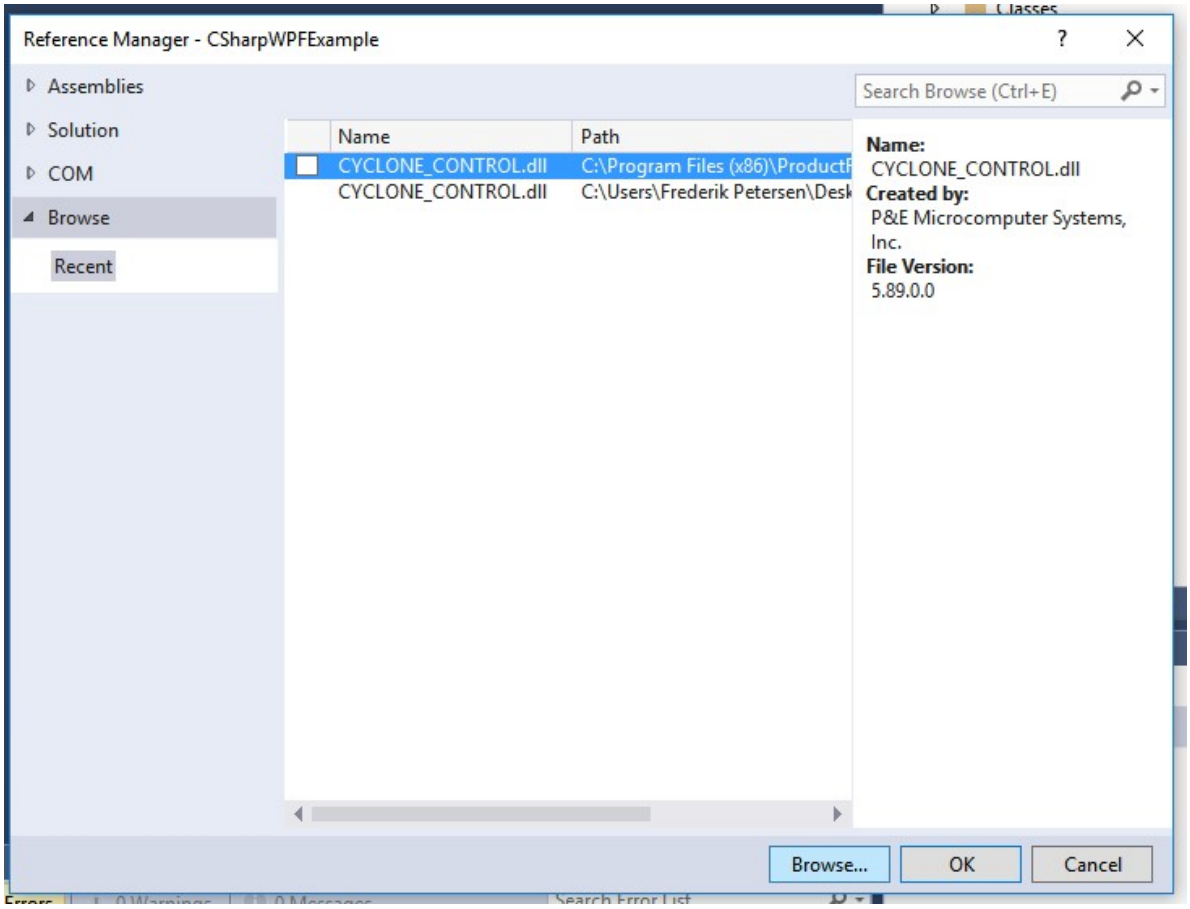
2. Click 'Add Reference…'



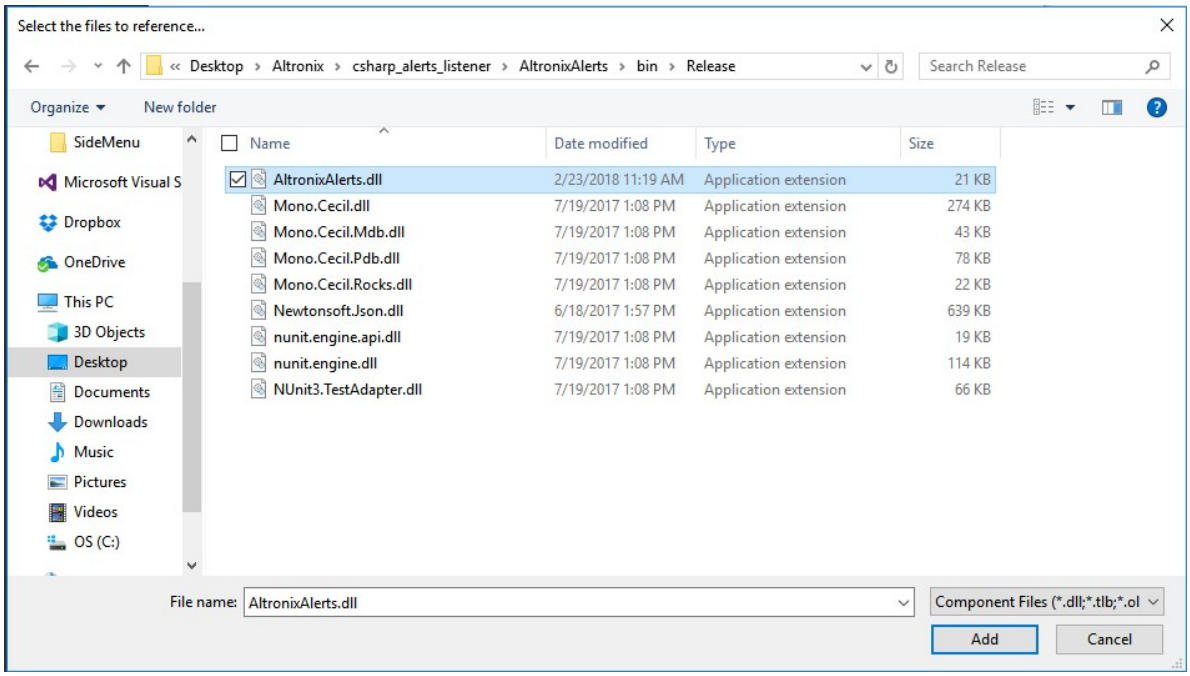
3. In the Reference Manager go to 'Browse' subsection



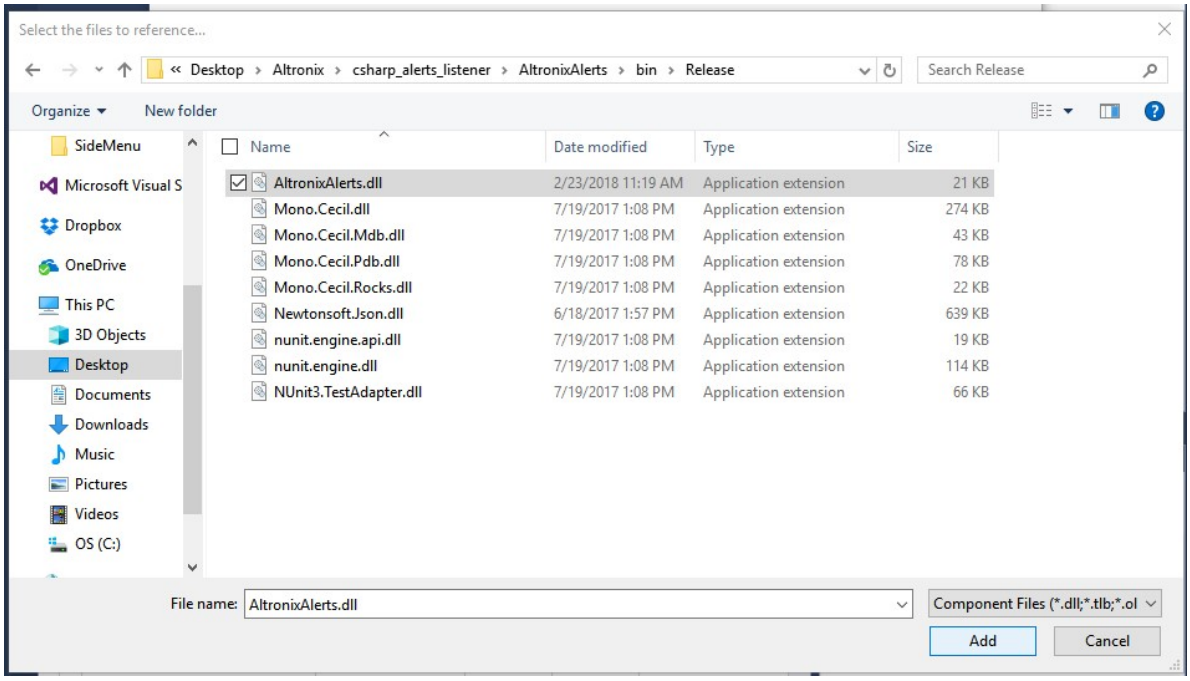
4. Click the 'Browse. . . 'button



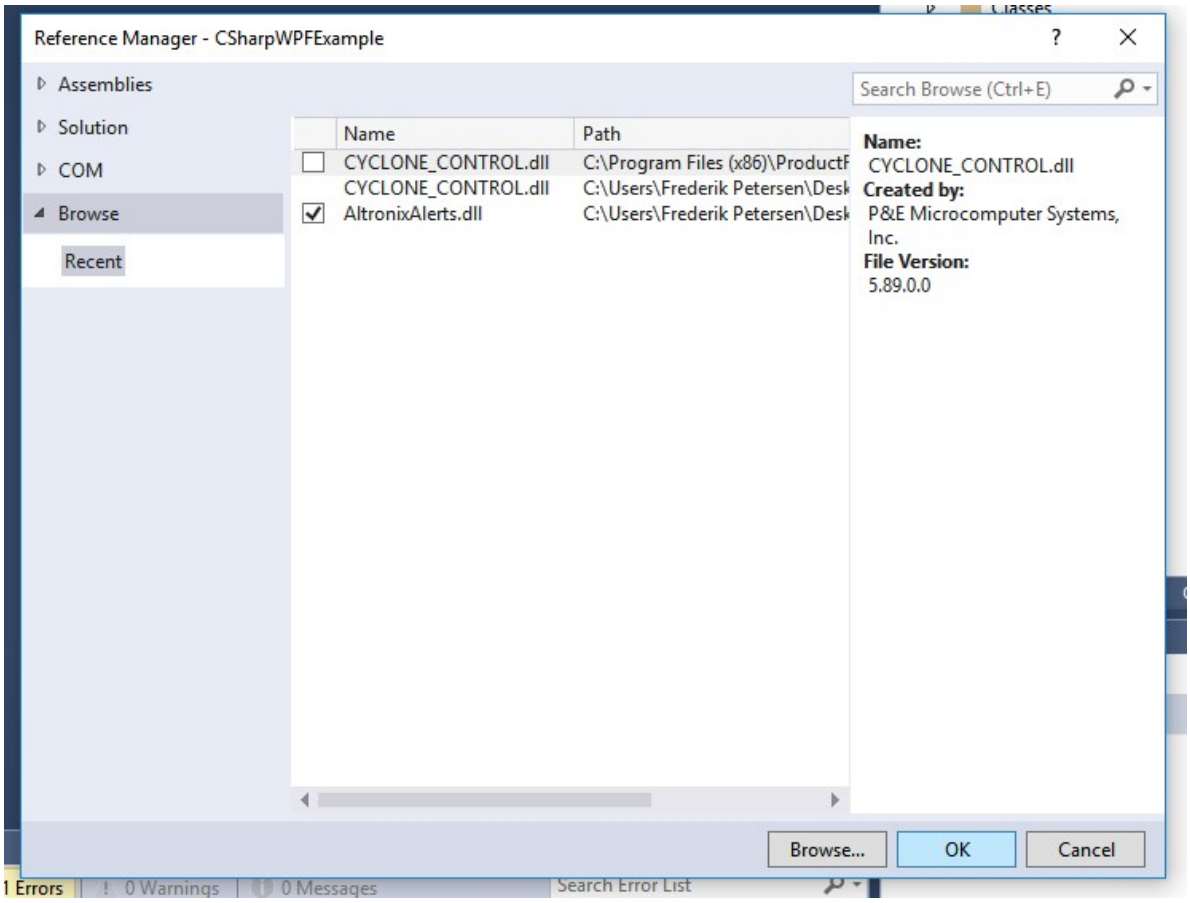
5. Locate the AltronixAlerts.dll file on your local computer



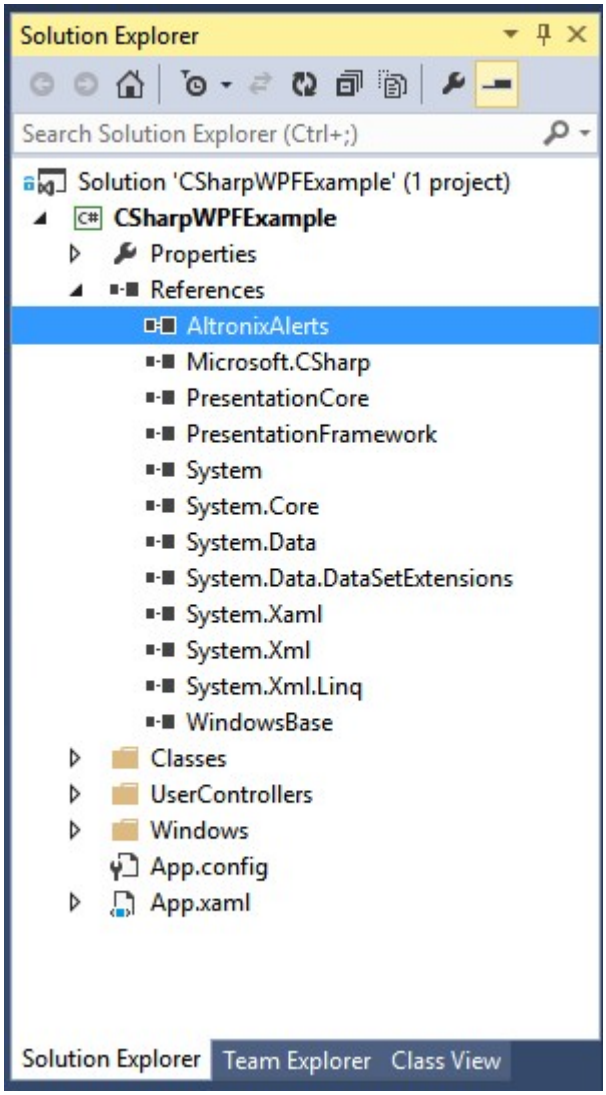
6. Once located, select the file and then click ‘Add’



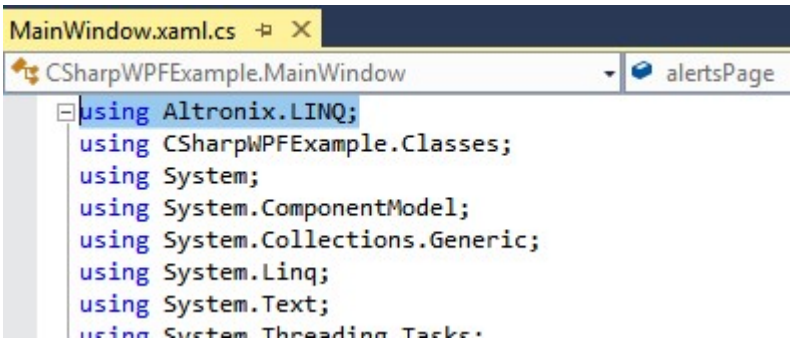
7. You will return to the Reference Manager where you can check to confirm that the library has been added and click 'OK'



8. Under References you should now see 'AltronixAlerts.dll' listed



9. To use the library add the following line to the top of a file: 'using Altronix.LINQ;'



Import

```
using Altronix.LINQ;
```

Setup for listening to devices

```
// User Callback that triggers OnAlerts events
public void UserCallback(List<Alert> list)
{
    list.ForEach(alert =>
    {
        Console.WriteLine(
          "New Alert:\n"
          + "Site ID: " + alert.siteid + "\n"
          + "Message: " + alert.message + "\n"
          + "Date: " + alert.date + "\n"
          + "Address: " + alert.ip + ":" + alert.port + "\n"
          );
    });
}

// User Callback that triggers OnError events
public void UserErrorCallback(Device device)
{
    switch (device.error)
    {
        case Error.SUCCESS:
            Console.WriteLine("Success connecting to " + device.ip + ":" + device.
↪port + "\n");
            break;
        case Error.CONNECTING:
            Console.WriteLine("Error connecting to " + device.ip + ":" + device.port␣
↪+ "\n");
            break;
        case Error.AUTHENTICATE:
            Console.WriteLine("Error authenticating on " + device.ip + ":" + device.
↪port + "\n");
            break;
        case Error.FORBIDDEN:
            Console.WriteLine("Error access forbidden on " + device.ip + ":" + device.
↪port + "\n");
            break;
    }
}

// Async function creates a listener for one device
public async Task ListenToNewDevice(Device device)
{
    // Create Listener Class instance
    Listener atx = new Listener();

    // Set user's callback for OnAlerts Event
    atx.OnAlerts(UserCallback);

    // Set user's callback for OnError Event
    atx.OnError(UserErrorCallback);

    // Start Listening on device
    atx.Listen(device);
}
```

(continues on next page)

(continued from previous page)

```
// Async function creates a listener for a list of devices
public async Task ListenToAllDevices(List<Device> devices)
{
    // Create Listener Class instance
    Listener atx = new Listener();

    // Set user's callback for OnAlerts Event
    atx.OnAlerts(UserCallback);

    // Set user's callback for OnError Event
    atx.OnError(UserErrorCallback);

    // Start Listening to device
    atx.ListenAll(devices);
}
```

Listen for events on one device

```
Device device = new Device("10.10.10.10", "80", false, "username", "password");

await ListenToNewDevice(device); // Creates Listener for one new device
```

Listen for events on list of devices

```
List<Device> devices = DeviceStorage.getDevices(); // Get or create list of devices␣
↪you want to listen to

await ListenToAllDevices(devices); // Creates Listeners for all devices in list
```

Example

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Altronix.LINQ;

namespace AlertsNugetTest
{
  public partial class Form1 : Form
  {
      public Form1()
      {
          InitializeComponent();
      }
```

(continues on next page)

(continued from previous page)

```
    private async void ListenToSingleDevice_Click(object sender, EventArgs e)
    {
        // Create instance of Altronix.LINQ.Device
        Device device = new Device("10.10.10.188",  // ip addresss
                                   "443",           // port number
                                   true,            // TLS/SSL = true
                                   "admin",         // username
                                   "admin");        // password

        // Create Altronix.LINQ.Listener Class instance
        Listener atx = new Listener();

        // Set user's callback for OnAlerts Event
        atx.OnAlerts(UserCallback);

        // Set user's callback for OnError Event
        atx.OnError(UserErrorCallback);

        // Start Listening on device
        await atx.Listen(device);

    }

    // User Callback that triggers OnAlerts events
    public void UserCallback(List<Alert> list)
    {
        list.ForEach(alert => {
            var str = "[ " + alert.date + " ]  [ " + alert.siteid + alert.ip + ":"␣
↪+ alert.port + " ] ===> " +
                      alert.message + "\n";
            // Print to console
            Console.WriteLine(str);
        });
    }

    public void UserErrorCallback(Device device)
    {
        switch (device.error)
        {
            case Error.CONNECTING:
                /* Handle this error here */
                break;
            case Error.AUTHENTICATE:
                /* Handle this error here */
                break;
            case Error.FORBIDDEN:
                /* Handle this error here */
                break;
        }
    }

    private void ListenToSeveralDevices2_Click(object sender, EventArgs e)
    {
        // Create List<Altronix.LINQ.Device> of devices
        List<Device> devices = new List<Device>
        {
            new Device(){ ip="10.10.10.183",port="80",ssl=false,username="admin",
↪password="admin"},
```

(continues on next page)

(continued from previous page)

```
            new Device(){ ip="10.10.10.185",port="443",ssl=true,username="admin",
↪password="admin"},
            new Device(){ ip="10.10.10.182",port="443",ssl=true,username="admin",
↪password="admin"},
            new Device(){ ip="10.10.10.186",port="443",ssl=true,username="admin",
↪password="123"},
            new Device(){ ip="10.10.10.188",port="443",ssl=true,username="admin",
↪password="admin"}
        };

        // Create Altronix.LINQ.Listener Class instance
        Listener atx = new Listener();

        // Set user's callback for OnAlerts Event
        atx.OnAlerts(UserCallback);

        // Set user's callback for OnError Event
        atx.OnError(UserErrorCallback);

        // Start Listening on list of devices
        atx.ListenAll(devices);

    }
  }

}
```

1.2.2 Tips and Tricks

Extend Classes

If you want to append to an SDK class, you can create a class to wrap around the SDK class. This class can have additional functions or Member Fields. This can be useful for handling UI. For example a member field and function could be added to handle hiding or showing alerts. You can name these extenstions anything but we recommend something like 'ExtendedAlert' or 'ExtendedDevice' to more easily identify them as extensions of other classes.

Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpWPFExample
{
    public class ExtendedAlert
    {
        public string SiteId { get; set; }
        public string name { get; set; }
        public string type { get; set; }
        public string time { get; set; }
```

(continues on next page)

(continued from previous page)

```
        public string user { get; set; }
        public string message { get; set; }
        public string detailedViewVisibility { get; set; }
        public string visibility { get; set; }

        public ExtendedAlert() { }

        public ExtendedAlert(String SiteId, String name, String type, String time,␣
↪String user, String message)
        {
            this.SiteId = SiteId;
            this.name = name;
            this.type = type;
            this.time = time;
            this.user = user;
            this.message = message;
            this.detailedViewVisibility = "Collapsed";
            this.visibility = "Visible";
        }

        // Additional function for hidding a UI popup
        public void ToggleDetailedViewVisibility()
        {
            if (this.detailedViewVisibility == "Collapsed")
            {
                this.detailedViewVisibility = "Visible";
            }
            else
            {
                this.detailedViewVisibility = "Collapsed";
            }
        }

        // Additional function for toggle visibility of alert
        public void ToggleVisibility()
        {
            if (this.visibility == "Collapsed")
            {
                this.visibility = "Visible";
            }
            else
            {
                this.visibility = "Collapsed";
            }

            this.detailedViewVisibility = "Collapsed";
        }
    }
}
```

1.2.3 API Reference

Alert

Class that represents a device Alert

Constructor(s)

constructor(s)	description
Alert()	Creates an instance of Alert

Member Fields

property	type
user	string
date	string
message	string
ip	string
port	string
siteid	string

Device

Class that represents an Altronix LINQ device

Constructor(s)

constructor(s)	description
Device()	Creates an empty instance of Device
Device(string Nickname, string Ip, string Port, bool Ssl, string Username, string Password)	Creates instance of Device with initial member fields
Device(string Ip, string Port, bool Ssl, string Username, string Password)	Creates instance of Device with initial member fields but no username

Member Fields

property	type
ip	string
port	string
ssl	bool
username	string
password	string
nickname	string
error	uint
authority	string

Functions

property	description
device.setAuthority	Sets the authority class variable
device.ToString	returns error in string form
device.GetHashCode	
device.Equals	compares two devices and returns true if they are equal
device.CompareTo	compares authority of device against another device

device.setAuthority

```
void device.setAuthority(String authority)
```

Assigns a string value to device.authority

Params

parameter	type	description
authority	String	A string in the format <ip>:<port> used to identify a device

Output

None

Examples

```
string deviceId = "10.10.10.10:80"
Device device = new Device();
device.setAuthority(deviceId);

Console.WriteLine(device.authority);
```

device.ToString

Creates status string of the device in the format "Authority: <authority> Username: <username> Error: <error_code>";

```
string device.ToString()
```

Params

None

Output

type	description
string	String formatted status of device “Authority: <authority> Username: <username> Error: <error_code>”

Examples

```
Device device = new Device("10.10.10.10", "80", false, "username", "password");
Console.WriteLine(device.ToString()); // logs "Authority: 10.10.10.10:80 Username:␣
↪username Error: 0"
```

device.GetHashCode

```
int device.GetHashCode()
```

Serves as the device hash function.

Params

None

Output

type	description
int32	A hash code for the current object

Examples

```
Device device = new Device("10.10.10.10", "80", false, "username", "password");

int deviceHashCode = device.GetHashCode();

Console.WriteLine(deviceHashCode)
```

device.Equals

```
bool device.Equals(Device device)
```

Compares the current device instance with another device by authority (<IP>:<port>) and returns true if the authority of the two instances are equal and false if not

Params

parameter	type	description
device	Device	An instance of the LinQ class Device

Output

type	description
bool	true if authority is a String and its value is the same as this instance; otherwise, false. If authority is null, the method returns false

Examples

```
// device_one.authority = "10.10.10.10:80"
// device_two.authority = "10.10.10.10:81"
Device device_one = new Device("10.10.10.10", "80", false, "username", "password");
Device device_two = new Device("10.10.10.10", "81", false, "username", "password");

int equalsResult = device_one.Equals(device_two);

if (equalsResult) {
  Console.WriteLine("device_one equals device_two")
} else {
  Console.WriteLine("device_one does not equal device_two")
}
```

device.CompareTo

```
int device.CompareTo(Device device)
```

Compares the current device instance with another device by authority (<IP>:<port>) and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other device.

Params

parameter	type	description
device	Device	An instance of the LinQ class Device

Output

type	description
int	A value that indicates the relative order of the devices being compared. if < 0, this instance precedes the device, if = 0, the instance is at the same position as the device, if > 0 instance follows the device in order

Examples

```
// device_one.authority = "10.10.10.10:80"
// device_two.authority = "10.10.10.10:81"
Device device_one = new Device("10.10.10.10", "80", false, "username", "password");
Device device_two = new Device("10.10.10.10", "81", false, "username", "password");

int compareResult = device_one.CompareTo(device_two);

if (compareResult < 0) {
  Console.WriteLine("device_one precedes device_two")
} else if (compareResult == 0) {
  Console.WriteLine("device_one is at same position as device_two")
} else {
  Console.WriteLine("device_one follows device_two")
}
```

Devices

Class that represents a list of Altronix LINQ devices

Constructor(s)

constructor(s)	description
Devices()	Creates an empty instance with an empty list of devices
Devices(List<Device> devices)	Creates an instance with an initial list of devices

Member Fields

property	type
devices	List<Device>

Functions

property	description
devices.insertList	
devices.addDevice	
devices.removeDevice	
devices.getDevice	
devices.getDevices	
devices.updateDevice	
devices.deviceAlreadyExists	
devices.count	

devices.insertList

```
void devices.insertList(List<Device> devices)
```

Override the current list of devices with a new list of devices

Params

parameter	type	description
devices	List<Device>	A list of LinQ Device class instances

Output

None

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

devices.addDevice(device);
Console.WriteLine(devices.count()); // 1

List<Device> newDeviceList = {
  new Device("10.10.10.10", "81", false, "username", "password"),
  new Device("10.10.10.10", "82", false, "username", "password"),
  new Device("10.10.10.10", "83", false, "username", "password"),
  new Device("10.10.10.10", "84", false, "username", "password")
}

devices.insertList(newDeviceList);
Console.WriteLine(devices.count()); // 4
```

devices.addDevice

```
Boolean devices.addDevice(Device device)
```

Adds a new device to devices. Returns true if device was successfully added

Params

parameter	type	description
device	Device	An instance of the LinQ class Device

Output

type	description
Boolean	Returns true if device was successfully added, otherwise false

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

Console.WriteLine(devices.count()); // 0
devices.addDevice(device);
Console.WriteLine(devices.count()); // 1
```

devices.removeDevice

```
Boolean devices.removeDevice(String deviceId)
```

Remove a device from the list of devices

Params

parameter	type	description
deviceId	String	Id string used to look up a specific device

Output

type	description
Boolean	Returns true if device was successfully deleted, otherwise false

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

devices.addDevice(device);
Console.WriteLine(devices.count()); // 1
devices.deleteDevice(device.authority);
Console.WriteLine(devices.count()); // 0
```

devices.getDevice

```
Device devices.getDevice(String deviceId)
```

If device is in devices, an instance of it will be returned

Params

parameter	type	description
deviceId	String	Id string used to look up a specific device

Output

type	description
Device	If device by same authority is in the list, its Device instance will be returned, else null

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

Device newDeviceInstance = devices.getDevice(device.authority);
Console.WriteLine(newDeviceInstance.ToString());
```

devices.getDevices

```
List<Device> devices.getDevices()
```

Returns a list of all devices contained in instance

Params

None

Output

type	description
List<Device>	A list of devices of class Device

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

devices.addDevice(device);

List<Device> listOfDevices = devices.getDevices();
```

devices.updateDevice

```
Boolean devices.updateDevice(String deviceId, Device device)
```

Update device in list by authority id

Params

parameter	type	description
deviceId	String	Id string used to look up a specific device
device	Device	Device instance to replace instance of deviceId

Output

type	description
Boolean	Returns true if device was successfully updated, otherwise false including if device is not in list

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

Console.WriteLine(device.ToString());

devices.addDevice(device);
devices.updateDevice(device.authority, new Device("10.10.10.10", "80", false,
↪"username2", "password2"));

Device updatedDevice = devices.getDevice(device.authority);

Console.WriteLine(updatedDevice.ToString());
```

devices.deviceAlreadyExists

```
Boolean devices.deviceAlreadyExists(String deviceId)
```

Returns true if a device with the same authority string already exists in the list of devices

Params

parameter	type	description
deviceId	String	Id string used to look up a specific device

Output

type	description
Boolean	Returns true if a device with the same authority string already exists in the list of devices

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

if (devices.deviceAlreadyExists(device.authority)) {
  Console.WriteLine("device already in list")
} else {
  Console.WriteLine("device not in list") // This will print
}

devices.addDevice(device);

if (devices.deviceAlreadyExists(device.authority)) {
  Console.WriteLine("device already in list") // This will print
} else {
  Console.WriteLine("device not in list")
}
```

devices.count

```
int devices.count()
```

Returns the number of devices that the instance of Devices contains

Params

None

Output

type	description
int	number of devices that the instance of Devices contains

Examples

```
Devices devices = new Devices();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

Console.WriteLine(devices.count()); // 0
devices.addDevice(device);
Console.WriteLine(devices.count()); // 1
```

DeviceStorage

Class that represents local storage of connected devices

Constructor(s)

constructor(s)	description
DeviceStorage()	Creates an instance DeviceStorage

Functions

property	description
deviceStorage.activateSetupState	
deviceStorage.getDeviceId	
deviceStorage.saveDevices	
deviceStorage.getDevices	
deviceStorage.addDevice	
deviceStorage.getDevice	
deviceStorage.deleteDevice	
deviceStorage.updateDevice	

deviceStorage.activateSetupState

```
void deviceStorage.activateSetupState()
```

Resets setupState to 'true'. Calling DeviceStorage.getDevices() will read devices from an IsolatedStorageFile: TrackedDevices.txt instead of from Class Devices and reset setupState = false

Params

None

Output

None

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();

deviceStorage.activateSetupState();
//Calling DeviceStorage.getDevices() will read devices from an IsolatedStorageFile
```

deviceStorage.getDeviceId

```
String deviceStorage.getDeviceId(Device device)
```

Returns the id of the device

Params

parameter	type	description
device	Device	An instance of the LinQ class Device

Output

type	description
String	String ID of device (its authority)

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

String deviceId = deviceStorage.getDeviceId(device);
Console.WriteLine(deviceId);
```

deviceStorage.saveDevices

```
void deviceStorage.saveDevices()
```

Stores devices in an IsolatedStorageFile: TrackedDevices.txt

Params

None

Output

None

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

deviceStorage.addDevice(device);

deviceStorage.saveDevices();
```

deviceStorage.getDevices

```
List<Device> deviceStorage.getDevices()
```

Returns a list of all devices contained in instance

Params

None

Output

type	description
List<Device>	A list of devices of class Device

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

deviceStorage.addDevice(device);

List<Device> listOfDevices = deviceStorage.getDevices();
```

deviceStorage.addDevice

```
Boolean deviceStorage.addDevice(Device device)

Adds a new device to deviceStorage. Returns true if device was successfully added
```

Params

parameter	type	description
device	Device	An instance of the LinQ class Device

Output

type	description
Boolean	Returns true if device was successfully added, otherwise false

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

deviceStorage.addDevice(device);
```

deviceStorage.getDevice

```
Device deviceStorage.getDevice(String deviceId)
```

If device is in devices, an instance of it will be returned

Params

parameter	type	description
deviceId	deviceId	Id string used to look up a specific device

Output

type	description
Device	If device by same authority is in the list, its Device instance will be returned, else null

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

Device newDeviceInstance = deviceStorage.getDevice(device.authority);
Console.WriteLine(newDeviceInstance.ToString());
```

deviceStorage.deleteDevice

```
Boolean deviceStorage.deleteDevice(String deviceId)
```

Remove a device from the list of devices

Params

parameter	type	description
deviceId	deviceId	Id string used to look up a specific device

Output

type	description
Boolean	Returns true if device was successfully deleted, otherwise false

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

deviceStorage.deleteDevice(device.authority);
```

deviceStorage.updateDevice

```
Boolean deviceStorage.updateDevice(String deviceId, Device updateThisDevice)
```

Update device in list by authority id

Params

parameter	type	description
deviceId	deviceId	Id string used to look up a specific device

Output

type	description
Boolean	Returns true if device was successfully updated, otherwise false including if device is not in list

Examples

```
DeviceStorage deviceStorage = new DeviceStorage();
Device device = new Device("10.10.10.10", "80", false, "username", "password");

Console.WriteLine(device.ToString());

deviceStorage.addDevice(device);
deviceStorage.updateDevice(device.authority, new Device("10.10.10.10", "80", false,
↪"username2", "password2"));
```

(continues on next page)

(continued from previous page)

```
Device updatedDevice = devices.getDevice(device.authority);

Console.WriteLine(updatedDevice.ToString());
```

Error

A class of constants used to set the current state of a device

Error Codes

name	value	description
SUCCESS	0	Successfully connected to device
CONNECTING	1	Connecting to device
AUTHENTICATE	2	Authenticating connection
FORBIDDEN	3	Connection forbidden/rejected

Example

```
public void UserErrorCallback(Device device)
   {
       switch (device.error)
       {
           case Error.SUCCESS:
               Console.WriteLine("Successfully connected")
               break;
           case Error.CONNECTING:
               Console.WriteLine("Error connecting to " + device.ip + ":" + device.
↪port + "\n");
               break;
           case Error.AUTHENTICATE:
               Console.WriteLine("Error authenticating on " + device.ip + ":" +␣
↪device.port + "\n");
               break;
           case Error.FORBIDDEN:
               Console.WriteLine("Error access forbidden on " + device.ip + ":" +␣
↪device.port + "\n");
               break;
       }
   }
```

Listener

Constructor(s)

constructor(s)	description
Listener()	Creates an instance of the Listener class

Member Fields

property	type
AlertsCallback(List<Alert> alertList)	Callback function
ErrorCallback(Device device);	Callback function

Functions

property	description
atx.Listen	Start a background thread that listens for Alerts from a Device
atx.ListenAll	Start a background thread that listens for Alerts for each Device in AltronixDeviceList
atx.OnAlerts	Sets the user callback to trigger on Alerts events
atx.OnError	Sets the user callback to trigger on Errors events

atx.Listen

```
void Listen(Device device)
```

Start a background thread that listens for Alerts from a Device

Params

parameter	type	description
device	Device	An instance of Class Device

Output

None

Examples

```
// Create instance of Altronix.LINQ.Device
Device device = new Device("10.10.10.188",  // ip addresss
                           "443",           // port number
                           true,            // TLS/SSL = true
                           "admin",         // username
                           "admin");        // password

// Create Altronix.LINQ.Listener Class instance
Listener atx = new Listener();

// Add listeners here, such as UserError and OnAlerts

// Start Listening on device
await atx.Listen(device);
```

atx.ListenAll

```
void ListenAll(List<Device> device_list)
```

Start a background thread that listens for Alerts for each Device in AltronixDeviceList

Params

parameter	type	description
device_list	List<Device>	An instance of List<Device> device_list

Output

None

Examples

```
// Create List<Altronix.LINQ.Device> of devices
List<Device> devices = new List<Device>
{
    new Device(){ ip="10.10.10.183",port="80",ssl=false,username="admin",password=
↪"admin"},
    new Device(){ ip="10.10.10.185",port="443",ssl=true,username="admin",password=
↪"admin"},
    new Device(){ ip="10.10.10.182",port="443",ssl=true,username="admin",password=
↪"admin"},
    new Device(){ ip="10.10.10.186",port="443",ssl=true,username="admin",password="123
↪"},
    new Device(){ ip="10.10.10.188",port="443",ssl=true,username="admin",password=
↪"admin"}
};

// Create Altronix.LINQ.Listener Class instance
Listener atx = new Listener();

// Add listeners here, such as UserError and OnAlerts

// Start Listening on list of devices
atx.ListenAll(devices);
```

atx.OnAlerts

```
void OnAlerts(Listener.AlertsCallback callback)
```

Sets the user callback to trigger on Alerts events

Params

parameter	type	description
callback	Listener.AlertsCallback	A method with 'void AlertsCallback(List<Alert> alertList)' signature

Output

None

Examples

```
// Create Listener Class instance
Listener atx = new Listener();

// Set user's callback for OnAlerts Event
atx.OnAlerts(UserCallback);

// Sample User Callback that triggers OnAlerts events
public void UserCallback(List<Alert> list)
{
    list.ForEach(alert => {
        var str = "[ " + alert.date + " ]  [ " + alert.siteid + alert.ip + ":" +␣
↪alert.port + " ] ===> " +
                    alert.message + "\n";
        // Print to console
        Console.WriteLine(str);
    });
}
```

atx.OnError

```
void OnError(Listener.ErrorCallback callback_error)
```

Sets the user callback to trigger on Errors events

Params

parameter	type	description
callback	Listener.ErrorCallback	A method with 'void ErrorCallback(Device device)' signature

Output

None

Examples

```
// Create Altronix.LINQ.Listener Class instance
Listener atx = new Listener();

// Set user's callback for OnError Event
atx.OnError(UserErrorCallback);

// Sample User Callback that triggers OnError events
public void UserErrorCallback(Device device)
{
    switch (device.error)
    {
        case Error.CONNECTING:
            /* Handle this error here */
            break;
        case Error.AUTHENTICATE:
            /* Handle this error here */
            break;
        case Error.FORBIDDEN:
            /* Handle this error here */
            break;
    }
}
```

1.3 HTTP API

1.3.1 Overview

The API is divided into two parts.

- Common API contain RESTful endpoints common to all Altronix LinQ enabled products
- Product API are unique RESTful endpoints per each product. Product API's contain all of the common API plus the additional product specific API

Supported HTTPs Requests

- GET: Retrieve a resource or collection of resources
- POST: Edit or Create a resource or collection of resources
- PUT: Same as POST. (–info) Note that resources are static and therefore PUT and POST are interpretted as the same)
- DELETE: Remove or zero a resource

Supported HTTPs Responses

- 200 (OK) The request was successful, new state was accepted
- 400 (Bad Request) The request was not successful. The server rejected the request because of one of the following reasons
  - The request had bad format

- 403 (Unauthorized) The credentials supplied with the request were not valid. Or the credentials supplied with the request are valid, but the user does not have access to the requested resource.
- 404 (Not found) The resource endpoint does not exist
- 500 (Internal Server Error) The server cannot process the request at this time.

Supported Authentication Scheme

All Altronix LinQ enabled products support Basic Authentication specification. Therefore the server expects the 'Authorization' header to be provided with each request requiring credentials.

```
| GET /ATX/userManagement/users\r\n
| Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l\r\n
```

1.3.2 Common

The following documentation outlines the Altronix API common to all Altronix LinQ products.

Saving settings to ROM

POST /ATX/exe/save

Save current state into persistent memory of the device. Without saving data, most of the state of the database will not persist across a reboot.

Example request:

```
POST /ATX/exe/save HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
Content-Type: application/json

{
   "save":1
}
```

Request JSON Object

- **save** (*number*) – Save profile index. (Currently only supports one profile)

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "error":200
}
```

Alert types

GET /ATX/alerts/types

Returns a list of all the supported alert types for the product. Different products may support different alert

types. For a detailed list of each alert type per product please reference the product specific api notes for your specific product. Below describes the API format in general form for all products.

Example request:

```
GET /ATX/alerts/types HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "types": {
    "acFail": {
      "name": "acFail",
      "mesg": "AC input failure",
      "timer": 0
    },
    "acFailR": {
      "name": "acFailR",
      "mesg": "AC input return",
      "timer": 0
    },
    "battFail": {
      "name": "battFail",
      "mesg": "Battery fail",
      "timer": 0
    },
    "battFailR": {
      "name": "battFailR",
      "mesg": "Battery return",
      "timer": 0
    },
    "sysUp": {
      "name": "sysUp",
      "mesg": "System Startup",
      "timer": 0
    },
    "...continued": {
      "...": ""
    }
  }
}
```

Response JSON Object

- **name** (*string*) – The name of alert. Is a KEY property to map alerts with a human readable message
- **mesg** (*string*) – The message describing the alert.
- **timer** (*number*) – A number in seconds used to filter alerts. When an alert condition only persists for less than x seconds, the alert will be filtered

Number of alerts since last power up

GET /ATX/alerts/count
Returns the number of alerts since last power up. The number is incremented per each alert as they occur.

Example request:

```
GET /ATX/alerts/count HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "count":25
}
```

Response JSON Object

- **count** (*number*) – The number of alert events since last power up

Most recent alerts

GET /ATX/alerts/recent
Return a minimal set of the most recent alerts (Max 5)

Example request:

```
GET /ATX/alerts/recent HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "recent": [{
     "who": "system",
     "what": "Power Supply ID",
     "when": 1520445079,
     "name": "ihigh",
     "mesg": "Over current",
     "siteId": "Demo Laptop"
   }, {
     "who": "system",
     "what": "Power Supply ID",
     "when": 1520445079,
     "name": "ihigh",
     "mesg": "Over current",
     "siteId": "Demo Laptop"
   }, {
     "who": "admin",
```

(continues on next page)

(continued from previous page)

```
      "what": "Power Supply ID",
      "when": 1520445407,
      "name": "psOff",
      "mesg": "Power supply off",
      "siteId": "Demo Laptop"
    }, {
      "who": "admin",
      "what": "Power Supply ID",
      "when": 1520445413,
      "name": "psOn",
      "mesg": "Power supply on",
      "siteId": "Demo Laptop"
    }]
}
```

Response JSON Object

- **who** (*string*) – The name of the user who caused the reason for the alert
- **what** (*string*) – The name of the port or power supply that is cause for alert
- **when** (*number*) – The unix time stamp of when the alert was generated
- **name** (*string*) – The alert [key] property used to map alert types to a human readable message
- **mesg** (*string*) – A human readable message describing the alert
- **siteId** (*string*) – The location of where the device that caused the alert

Reading entire event history

GET /ATX/alerts/all

Return the entire log history from the peristant storage of the device. Note the JSON properties are a minimal set of alert meta data. For additinal data referencing the alert you can map the alert name with the alert type to reference relavent alert meta-data.

Example request:

```
GET /ATX/alerts/all HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "log": [
     { "who": "system", "what": "log reset", "when": 0 },
     { "who": "admin", "what": "Power Supply ID", "when": 1519057709, "name":
↪"psOff" },
     { "who": "admin", "what": "Power Supply ID", "when": 1519057713, "name":
↪"psOn" },
     { "who": "admin", "what": "Power Supply ID", "when": 1519826227, "name":
↪"psOff" },
```

(continues on next page)

(continued from previous page)


```
    { "who": "admin", "what": "Power Supply ID", "when": 1519826485, "name":
↪"psOn" },
    { "who": "system", "what": "test", "when": 1519840284, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519840292, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519840356, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519846949, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519846963, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519846967, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519846972, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519847044, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519847047, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519847067, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519847071, "name": "tmr" },
    { "who": "system", "what": "test", "when": 1519847075, "name": "battFail" },
    { "who": "system", "what": "test", "when": 1519847080, "name": "ilowR" },
    { "who": "system", "what": "test", "when": 1519847152, "name": "psOn" },
    { "who": "system", "what": "test", "when": 1519847160, "name": "psOff" },
    { "who": "system", "what": "test", "when": 1519847174, "name": "rOn" },
    { "who": "admin", "what": "Power Supply ID", "when": 1519847194, "name":
↪"psOff" },
    { "who": "admin", "what": "Power Supply ID", "when": 1519847247, "name":
↪"psOn" },
    { "who": "admin", "what": "Power Supply ID", "when": 1520445407, "name":
↪"psOff" },
    { "who": "admin", "what": "Power Supply ID", "when": 1520445413, "name":
↪"psOn" }
  ]
}
```


Response JSON Object

- **who** (*string*) – The name of the user who caused the reason for the alert
- **what** (*string*) – The name of the port or power supply that is cause for alert
- **when** (*number*) – The unix time stamp of when the alert was generated
- **name** (*string*) – The alert [key] property used to map alert types to a human readable message

User Management

GET /ATX/userManagement/users

Returns an array of users permissioned to access the device at various restriction levels

Example request:

```
GET /ATX/userManagement/users HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json
```

(continues on next page)

(continued from previous page)

```
{
   "users": ["admin", "thomas"]
}
```

GET /ATX/userManagement/users/(string: *user_id*)
Returned detailed meta-data for a specific user

Example request:

```
GET /ATX/userManagement/users/thomas HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "thomas": {
     "user": "thomas",
     "fail": 0,
     "enable": 1,
     "realms": ["/ATX/userManagement/users/thomas/pass", "1:/ATX/userManagement/
↪users", "3:/ATX/exe/save"]
   }
}
```

Response JSON Object

- **user** (*string*) – The name of the user in reference
- **fail** (*string*) – The number of failed login attempts
- **enable** (*number*) – 1 - user is enabled, 0 - user is not enabled
- **realms** (*object*) – An array of URL endpoints that this user is allowed to have access too.

POST /ATX/userManagement/users
Create an additional user

Example request:

```
POST /ATX/userManagement/users HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
Content-Type: application/json

{
   "user":"new user name",
   "pass":"new user password",
   "realms":["/"]
}
```

Request JSON Object

- **user** (*string*) – The name of the new user

- **pass** (*string*) – The password of the new user
- **realms** (*number*) – An array of realms this new user shall have access too

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "error":200
}
```

1.3.3 Products

The following documentation outlines the Altronix API specific to the Altronix LinQ product. Product API's include all of the LinQ HTTP API (common) resources plus the additional product specific API's.

LinQ2

Alert types

GET /ATX/alerts/types
Returns a list of all the supported alert types for the LinQ2 product.

Example request:

```
GET /ATX/alerts/types HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "types": {
     "tmr": { "name": "tmr", "mesg": "Timer event trigger", "timer": 0 },
     "acFail": { "name": "acFail", "mesg": "AC input failure", "timer": 0 },
     "acFailR": { "name": "acFailR", "mesg": "AC input return", "timer": 0 },
     "battFail": { "name": "battFail", "mesg": "Battery fail", "timer": 0 },
     "battFailR": { "name": "battFailR", "mesg": "Battery return", "timer": 0 },
     "battServ": { "name": "battServ", "mesg": "Service Battery", "timer": 0 },
     "battServR": { "name": "battServR", "mesg": "Battery serviced", "timer": 0 },
     "psOn": { "name": "psOn", "mesg": "Power supply on", "timer": 0 },
     "psOff": { "name": "psOff", "mesg": "Power supply off", "timer": 0 },
     "ilow": { "name": "ilow", "mesg": "Under current", "timer": 0 },
     "ilowR": { "name": "ilowR", "mesg": "Under current clear", "timer": 0 },
     "ihigh": { "name": "ihigh", "mesg": "Over current", "timer": 0 },
     "ihighR": { "name": "ihighR", "mesg": "Over current clear", "timer": 0 },
     "rOn": { "name": "rOn", "mesg": "Relay engaged", "timer": 0 },
     "rOff": { "name": "rOff", "mesg": "Relay dis-engaged", "timer": 0 },
     "input": { "name": "input", "mesg": "Input trigger", "timer": 0 },
     "test": { "name": "test", "mesg": "this is an example alert", "timer": 0 },
```

(continues on next page)

(continued from previous page)

```
      "sysUp": { "name": "sysUp", "mesg": "System Startup", "timer": 0 }
    }
}
```

Response JSON Object

- **tmr** (*string*) – The timer alert is recorded when a timer event expires
- **acFail** (*string*) – The acFail alert is recorded when the AC failes below the eFlow specification AC input requirements
- **acFailR** (*string*) – The acFailR alert is recorded when the AC recovers from an AC fail condition
- **battFail** (*string*) – The battFail alert is recorded when the eFlow battery is presently installed but is reporting under voltage
- **battFailR** (*string*) – The battFailR alert is recorded when the eFlow battery is presently installed and has recovered from an under voltage condition
- **psOn** (*string*) – The psOn alert is recorded when a LinQ administrator turns on an eFlow Power Supply
- **psOff** (*string*) – The psOff alert is recorded when a LinQ administrator turns off an eFlow Power Supply
- **iLow** (*string*) – The iLow alert is recorded when a LinQ administrator has configured minimum load requirments, and the load to the eFlow Power Supply has dropped below limit the under current limit
- **iLowR** (*string*) – The iLow alert is recorded when the eFlow Power Supply recoveres from an under current condition
- **iHigh** (*string*) – The iHigh alert is recorded when a LinQ administrator has configured maximum load requirments, and the load to the eFlow Power Supply has exceeded the over current limit
- **iHighR** (*string*) – The iHighR alert is recorded when the eFlow Power Supply recoveres from an over current condition
- **rOn** (*string*) – The rOn alert is recorded when a LinQ administrator turns on a relay
- **rOff** (*string*) – The rOff alert is recorded when a LinQ administrator turns off a relay
- **input** (*string*) – The input alert is recorded when a LinQ2 module input trigger is engaged
- **test** (*string*) – The test alert is recorded when a LinQ administrator activates a test alert. A test alert is used to simulate any of the other alert types and can be helpful when diagnosing alert notification configuration
- **sysUp** (*string*) – The sysUp alert is recorded when the LinQ2 module first powers on (typically after a reboot or a power cycle event)
- **name** (*string*) – The name of alert. Is a KEY property to map alerts with a human readable message
- **mesg** (*string*) – The message describing the alert.
- **timer** (*number*) – A number in seconds used to filter alerts. When an alert condition only persists for less than x seconds, the alert will be filtered

Device IO summary

GET /ATX/hardware

Returns a summary of the IO state of all hardware inputs, relays, and power supplies. For detailed information on JSON object properties, refer to the more explicit non-summary api, (IE: /ATX/hardware/eflows/eflow0)

Example request:

```
GET /ATX/hardware HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "hardware": {
     "exe": { "save": "application" },
     "eflows": {
       "eflow0": {
         "id": "Power Supply ID",
         "type": "",
         "psPresence": 1,
         "ps": 1,
         "pulse": -15,
         "status": "on",
         "ac": 0,
         "batt": 0,
         "vCal": 13863,
         "iCal": 4266,
         "v": 8000,
         "i": 13000,
         "vHigh": 28000,
         "iHigh": 5000,
         "iLow": 0,
         "service": 0,
         "reminder": 0,
         "battPresence": 1,
         "installation": 0,
         "timers": 500
       },
       "eflow1": {
         "id": "Power Supply ID",
         "type": "",
         "psPresence": 1,
         "ps": 1,
         "pulse": -15,
         "status": "on",
         "ac": 0,
         "batt": 0,
         "vCal": 13863,
         "iCal": 4266,
         "v": 7000,
         "i": 9000,
         "vHigh": 28000,
         "iHigh": 5000,
```

(continues on next page)

(continued from previous page)


```
          "iLow": 0,
          "service": 0,
          "reminder": 0,
          "battPresence": 1,
          "installation": 0,
          "timers": 500
        },
        "mask": ""
      },
      "relays": {
        "relay0": { "id": "Relay ID", "status": 0, "pulse": -15 },
        "relay1": { "id": "Relay ID", "status": 0, "pulse": -15 },
        "mask": ""
      },
      "inputs": {
        "input0": { "id": "Input ID", "status": 0, "test": 0, "cmds": [] },
        "input1": { "id": "Input ID", "status": 0, "test": 0, "cmds": [] },
        "input2": { "id": "Input ID", "status": 0, "test": 0, "cmds": [] }
      },
      "temp": 0,
      "temp": 12205
    }
}
```


Power supplies

GET /ATX/hardware/eflows/(string: *index*)
Returns the state of the Eflow Power Supply at the respective index

Example request:

```
GET /ATX/hardware/eflows/eflow0 HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:


```
HTTP/1.0 200 OK
Content-Type: application/json

{
  "eflow0": {
    "id": "Power Supply ID",
    "type": "",
    "psPresence": 1,
    "ps": 1,
    "pulse": -15,
    "status": "on",
    "ac": 0,
    "batt": 0,
    "vCal": 13863,
    "iCal": 4266,
    "v": 8000,
    "i": 13000,
    "vHigh": 28000,
```


(continues on next page)

(continued from previous page)

```
    "iHigh": 5000,
    "iLow": 0,
    "service": 0,
    "reminder": 0,
    "battPresence": 1,
    "installation": 0
  }
}
```

Response JSON Object

- **id** (*string*) – The name of the resource (IE a relay or Eflow Power Supply can be named and are tagged with this name should an alert occur from this resource).
- **type** (*string*) – The eflow type used to scale the voltage and current. IE: eFlow3NB
- **psPresense** (*string*) – A boolean indicator to inform the system if the Eflow Power Supply is installed or not installed (1-installed, 0-not installed)
- **ps** (*string*) – A boolean indicator to indicate if the Eflow Power Supply is on or off (1-on, 0-off)
- **pulse** (*number*) – number of seconds to pulse the Eflow Power Supply, should a pulse command be executed. (<0 pulse off, >0 pulse on)
- **status** (*string*) – Human readable string indicating the status of the Eflow Power Supply. (IE: "on"/"off")
- **ac** (*number*) – Boolean indicator for AC status. (0 - AC fail, 1 - AC OK)
- **batt** (*number*) – Boolean indicator for Battery status. (0 - Battery fail, 1 - Battery OK)
- **vCal** (*number*) – TODO - calibration
- **iCal** (*number*) – TODO - calibration
- **v** (*number*) – Voltage level in (mV)
- **i** (*number*) – Current level in (mA)
- **vHigh** (*number*) – Over voltage threshold settings (mV)
- **iHigh** (*number*) – Over Current threshold setting (mA)
- **iLow** (*number*) – Under Current threshold setting (mA)
- **service** (*number*) – Unix date code when battery service is expected
- **reminder** (*number*) – Should battery serice reach expirey, an alert will be generated every [reminder] intervals (days)
- **battPresence** (*number*) – Boolean indicator representing the presence of a battery per this Eflow Power Supply. (0 - no battery installed, 1 - battery installed)
- **installation** (*number*) – Unix date code when the battery was installed, set by the administrator

POST /ATX/hardware/eflows/(**string:** *index*)**/status**

Change the state of the Eflow Power Supply output

Example request:

```
POST /ATX/hardware/eflows/eflow0/status HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
Content-Type: application/json

{
   "status":0
}
```

Request JSON Object

- **status** (*string*) – If the supplied property is a number, the Eflow Power Supply will either turn on (status=1), or turn off (status=0)
- **status** – If the supplied property is a string, the Eflow Power Supply will either Pulse (status='p') or toggle (status='t'), Where a "pulse" command will either pulse on for x seconds, or pulse off for x seconds, and where seconds is determined by the "pulse" property of the same eflow object.

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "error":200
}
```

Inputs

GET /ATX/hardware/inputs/(string: *index*)
Returns the state of the linq2 input at the respective index

Example request:

```
GET /ATX/hardware/inputs/input0 HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "input0": {
     "id": "Input ID",
     "status": 0,
     "test": 0,
     "cmds": [{
       "url": "/ATX/hardware/eflows",
       "data": {
         "mask": "px"
       }
     }]
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Response JSON Object

- **id** (*string*) – The name of the input resource. If an input event is triggered and the alert per this input is not masked, an alert will be recorded and the alerts 'what' property will contain this ID field
- **status** (*number*) – The current state of the input. (1 - active, 0 - in-active). Note that the input triggers on the LinQ2 module are active low
- **cmds** (*object*) – An array of commands to execute should the input go active. These commands can be any API command as supported by the device API. (Linq Common API + LinQ2 Product API)
- **url** (*string*) – When the commands array is executed by the input trigger, the URL is the resource location per the API.
- **data** (*object*) – When the commands array is executed by the input trigger, the data property contains POST data included with the URL.

Relays

GET /ATX/hardware/relays/(string: *index*)
Returns the state of the linq2 relays at the respective index

Example request:

```
GET /ATX/hardware/relays/relay0 HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "relay1": {
       "id": "Relay ID",
       "status": 0,
       "pulse": -15
   }
}
```

Response JSON Object

- **id** (*string*) – The name of the relay resource. If an relay event is triggered and the alert per this relay is not masked, an alert will be recorded and the alerts 'what' property will contain this ID field
- **status** (*number*) – The current state of the relay. (1 - active, 0 - in-active).
- **pulse** (*number*) – number of seconds to pulse the power supply, should a pulse command be executed. (<0 pulse off, >0 pulse on)

POST /ATX/hardware/relay/(string: *index*)/status

Change the state of the Relay

Example request:

```
POST /ATX/hardware/relay/relay0/status HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
Content-Type: application/json

{
   "status":0
}
```

Request JSON Object

- **status** (*string*) – If the supplied property is a number, the relay will either turn on (status=1), or turn off (status=0)
- **status** – If the supplied property is a string, the relay will either Pulse (status='p') or toggle (status='t'), Where a "pulse" command will either pulse on for x seconds, or pulse off for x seconds, and where seconds is determined by the "pulse" property of the same relay object.

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
   "error":200
}
```

LinQ8

LinQ8

Netway / Ebridge

Device IO summary

GET /ATX/hardware

Returns a summary of the IO state of all PoE (power over ethernet) ports. For detailed information on JSON object properties, refer to the more explicit non-summary api, (IE: /ATX/hardware/chs/ch0)

Example request:

```
GET /ATX/hardware HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
 "_sig": "",
 "board": {
     "about": {
         "version": "2.00.555",
         "type": "NETWAY4EWP"
     },
     "chs": {
         "mask": "",
         "ch0": {
             "data": {
                 "v": 185,
                 "i": 0,
                 "class": 0,
                 "state": 0,
                 "mode": 1,
                 "port_en": 1,
                 "port_num": 0,
                 "port_name": "Port ID 1",
                 "pcycl_dur": 5000
             }
         },
         "ch1": {
             "data": {
                 "v": 0,
                 "i": 0,
                 "class": 0,
                 "state": 0,
                 "mode": 1,
                 "port_en": 1,
                 "port_num": 1,
                 "port_name": "Port ID 2",
                 "pcycl_dur": 5000
             }
         },
         "ch2": {
             "data": {
                 "v": 0,
                 "i": 0,
                 "class": 0,
                 "state": 0,
                 "mode": 1,
                 "port_en": 1,
                 "port_num": 2,
                 "port_name": "Port ID 3",
                 "pcycl_dur": 5000
             }
         },
         "ch3": {
             "data": {
                 "v": 0,
                 "i": 0,
                 "class": 0,
                 "state": 0,
```

(continues on next page)

(continued from previous page)

```
                    "mode": 1,
                    "port_en": 1,
                    "port_num": 3,
                    "port_name": "Port ID 4",
                    "pcycl_dur": 5000
                }
            }
        },
        "status": {
            "temp": 36,
            "alert_cnt": 2
        },
        "cfg": {
            "id": 0,
            "temp_high": 50
        },
        "wdt": {
            "wdt_cnt": 2
        },
        "reboot": {
            "cnt": 2
        },
        "exe": {
            "save": "application"
        },
        "unix": 1475285335
    }
}
```

PoE Output channels

GET /ATX/hardware/chs/(string: *index*)

Returns the state of the PoE Channel at the respective index.

Example request:

```
GET /ATX/hardware/chs/ch2 HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
```

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
    "_sig": "",
    "ch2": {
        "data": {
            "v": 185,
            "i": 0,
            "class": 0,
            "state": 0,
            "mode": 1,
            "port_en": 1,
```

(continues on next page)

(continued from previous page)

```
            "port_num": 2,
            "port_name": "Port ID 3",
            "pcycl_dur": 5000
        }
    }
}
```

Response JSON Object

- **v** (*integer*) – PoE Channel voltage value in millivolts.
- **i** (*integer*) – PoE Channel current value in milliamps.
- **class** (*integer*) – PoE Channel Power Level Class.
- **state** (*integer*) – A boolean value to indicate if the PoE Channel is active or no-active (1:active, 0:no-active). The PoE is considered 'active' when is delivering power to a PD (PoE Powered Device). 'no-active' means no power is being delivered.
- **mode** (*integer*) – Read only boolean value to indicate if the PoE Channel status (1-enable, 0-disable).
- **port_en** (*integer*) – A Read/Write boolean value to enable or disable the PoE Channel (1-enable, 0-disable).
- **port_num** (*integer*) – PoE Channel Index.
- **port_name** (*string*) – The name of the resource (IE a PoE Channel can be named and are tagged with this name should an alert occur from this resource).
- **pcycl_dur** (*integer*) – number of seconds to pulse the PoE Channel Output, should a pulse command be executed. (<0 pulse off, >0 pulse on)

POST /ATX/hardware/chs/(string: *index*)/data

Toggle the state of the PoE Channel

Example request:

```
POST /ATX/hardware/chs/ch2/data HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
Content-Type: application/json

{
   "port_en":"t"
}
```

Request JSON Object

- **port_en** (*string*) – If the supplied property is a string "t" , the PoE Channel will toggle between the "on/off" states.

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
```

(continues on next page)

(continued from previous page)

```
    "_sig": "",
    "error": 0
}
```

POST /ATX/hardware/chs/mask

Pulse PoE Channel Power Output.

Example request:

```
POST /ATX/hardware/chs/mask HTTP/1.0
Host: 192.168.168.168
Accept: application/json, text/javascript
Content-Type: application/json

{
    "mask":"xpxx"
}
```

Request JSON Object

- **mask** (*string*) – This property follows the pattern "aaaa" with characters from left to right meaning ch1 to ch4. An 'a' = "p" indicates to pulse that specific channel and 'a'="x" indicates no action on the specific port. E.g to pulse PoE Channel 2 the corresponding string will be equal to "xpxx".

Example response:

```
HTTP/1.0 200 OK
Content-Type: application/json

{
    "_sig": "",
    "error": 0
}
```

1.4 ZMTP API

1.4.1 Overview

HTTP Routing Table

/ATX