# Automation

Distech-controls provides the xpressNetwork tool to manage a fleet of ECLYPSE controllers:

* Small number of devices
* From a Windows PC
* On a simple network

When these tools are insufficient, ECLYPSE also provides a REST API. This development kit (SDK) includes libraries and example scripts to manage many aspects of ECLYPSE through the API using the very popular Python scripting language.

We developed this capability primarily to solve for:

* Massive scale – thousands of controllers and millions of points
* Complex enterprise networks – No discovery
* Consistency – Set and verify consistent configuration
* Extensions – Custom points, configurations, and ECLYPSE extensions

# CLI

Most of the applications we interact with in Windows or MacOS have a graphical user interface (GUI). Many applications are web apps that present the user interface in a browser like Chrome, Edge, or Safari. GUIs are great for small bits of data like a weather forecast or text like a news story. But GUI and Web apps are not great for working with data. For viewing data, most of us prefer Excel.

The scripts in this SDK are CLI apps. Which means they are executed in a terminal application like MacOS terminal or Windows CMD or PowerShell.

There is no GUI. Instead, we run the scripts to collect and organize data. We manage the inputs and view the outputs using Excel.

# Python

Python is very easy to learn and a very popular starting point for new developers and non-developers. But it is also powerful and can accomplish real work like building management.

Because of its popularity with engineers, scientists, and businesses, Python’s PyPi repository is full of modules and examples for accessing databases, social media, devices, and business systems.

In this development kit, we use Python 3. Even if your operating system comes with a version of Python, we recommend installing the latest version using the link below.

You can download Python here:

https://www.python.org/

## Requests

Requests is a very popular Python module for interacting with web sites, especially those that return structured data like an API. More on that in a minute.

In the example modules and scripts, we use Requests to send and retrieve data from ECLYPSE using its API.

You can install Requests and other modules using Python’s package manager, pip. Pip is installed with Python if you used the link above. Type this command in a terminal to install Requests:

pip install requests

## Concurrent futures

Basic Python scripts execute one instruction at a time. In our example, that means one ECLYPSE at a time. We call this serial code. At a certain scale, serial code will be too slow.

Your computer is fast and it spends a lot of time waiting for slow things. Networks, for example, are very slow in comparison. Since we are talking to a lot of ECLYPSE, there’s really no reason to talk to one at a time and wait for a reply.

Code that doesn’t stop and wait is called asynchronous and Python has a couple of built in methods to accomplish it. AsyncIO is popular with developers but it’s an advanced topic and more than we need. Luckily there is a middle ground called concurrent futures. Don’t let the name scare you. It means that we are sending multiple requests at the same time and the responses will arrive later. If a computer can do two things at once, the job will finish in half the time.

# REST API

An API allows applications to communicate with each other. REST is a specific type of API that mimics HTTP, the same language that your Internet browser uses. Making it possible for HTTP servers, proxy servers, and browsers to participate in API communication. In our example, we use Python and the Requests module to communicate with the REST API on ECLYPSE.

Parts of the API are implemented in the included Python modules. For example, the module eclypse.py contains functions for:

* Enabling and Disabling interfaces
* Setting the timezone
* Changing the hostname in the web server

The full ECLYPSE API is documented here:  
  
<http://eclypseapi.distech-controls.com/publicAPI>

## Authentication:

ECLYPSE API supports HTTP Basic Authentication using any account with the API role assigned in ECLYPSE. This includes the default admin account.

In the example scripts, Basic Authentication is implemented using Requests Session object.

For more information, see the Requests documentation here:

<https://requests.readthedocs.io/en/latest/user/authentication/>

And ECLYPSE documentation here:

<http://eclypseapi.distech-controls.com/documentation/1/authentication>

## Certificates

ECLYPSE ships with a self-signed certificate for HTTPS. When you access the ECLYPSE UI on your network, your browser warns you that the certificate cannot be verified. Python, by default, does the same check and blocks HTTPS communication with a self-signed certificate.

By using these scripts, you accept that the IP address or hostname you put in the input file belongs to the device you installed on your private network. I have disabled the strict verification of certificates in each script to work with the self-signed certificate. The script will issue a warning every time it communicates with a device. I have provided an option to disable the warning at the top of every example:

SUPPRESS\_SSL\_WARNING = False

## ECLYPSE Versions

Hardware versions require specific firmware update files. The examples in this kit demonstrate how to identify the hardware model and include or exclude devices based on the model. It is not strictly enforced, so it is the users responsibility to send the right firmware to the device.

Distech recently released version 2 of the ECLYPSE firmware. The new firmware supports a new v2 API which is not backward compatible with v1. Most of the examples in this kit have been updated to support both versions. In fact, we demonstrate how to determine which API is supported and make the right call per device. Examples that support both API versions include an optional parameter –apiversion if the user wishes to specify a version.

# Example Scripts and Libraries

How to install this package:

1. Unzip the package

2. Install dependencies

>`pip install -r requirements.txt`

3. Create an input csv file based on the provided example

>`python eclypse\_firmware\_upgrade.py host\_list.csv eclypse\_firmware.zip firmware\_version`

# Upgrading firmware

Upgrade ELCYPSE firmware on one or more ECLYPSE S1000.

Process Summary:

1. Obtain the latest firmware using Distech’s software center.
2. Upload the firmware file to each ECY.

Note: By default the upgrade script will skip ECLYPSE running the specified or newer firmware.

Obtain the latest ECY firmware from Distech Controls. Download Distech’s Software Center here (login required):

<https://docs.distech-controls.com/bundle/SoftwareCenter/page/en-US/811622283.html>

Note: With the introduction of new APEX hardware and BI firmware, it’s important to get the right firmware file:

|  |  |
| --- | --- |
| Device Model | Example |
| S1000 | ECYSeries\_v1.19.22339.853.zip |
| APEX Hybrid | ECYAPEX\_v1.18.22209.893.zip |
| APEX Building Intelligence | 2.2.0+23305.1 |

Note: BI firmware is now modular. A firmware update consists of at least 3 files including the os, framework, and ui packages. The example in this kit assumes the bundle available in Distech’s software center. The bundle is a zip file containing most of the available packages. ECLYPSE will install or upgrade any packages included in the zip.

Note: BI packages have dependencies on the version of other packages, especially the os and framework. We highly recommend that you upgrade everything that is installed on ECLYPSE to the same version. A version mismatch may prevent an upgrade.

To execute the upgrade script:

1. Copy the ECLYPSE firmware zip file or bundle to the same directory as the upgrade script.
2. Create a host list for the upgrade.
3. Execute the script:

Usage:

python eclypse\_firmware\_upgrade.py [-h] host\_file update\_file update\_version

python eclypse\_bi\_upgrade.py [-h] host\_file update\_bundle

# Upload GFX

Process Summary:

1. Create a common GFX file using the EC-gfx emulator.
2. Pre-compile the GFX instructions creating a zip file.
3. On each ECLYPSE controller, stop the GFX engine.
4. Upload the pre-compiled GFX zip file
5. Restart the GFX engine.

To execute the GFX upload script:

1. Copy the compiled GFX zip file to the same directory as the upgrade script.
2. Create a host list for the upgrade.
3. Execute the script:

Usage:

python eclypse\_gfx\_upload.py [-h] host\_file gfx\_file

# Managing Users

Options

1. Create or modify user
2. Delete a user
3. Create a report of existing users (no passwords)
4. Verify user and password, report missing user or wrong password

Usage:

Add a user:

python eclypse\_user\_add.py [-h] username password host\_file –apiversion[1/2]

Delete a user:

python eclypse\_user\_delete.py [-h] username host\_file –apiversion[1/2]

Set password:

python eclypse\_user\_set\_password.py [-h] username password host\_file –apiversion[1/2]

Create a consolidated report of all users:

python eclypse\_users.py [-h] host\_file –apiversion[1/2]

# Backups

Manage ECLYPSE backups.

Note: Downloaded backups are saved in the current directory with the date and hostname in the filename.

Options:

* Create a full backup
* Download latest backup
* List backups

Using the examples:

Create a full backup:

python eclypse\_backup\_create.py [-h] host\_file –apiversion=[1/2]

Download a copy of the newest backup:

python eclypse\_backup\_download.py [-h] host\_file –apiversion=[1/2]

Create a list of available backups on multiple ECLYPSE:

python eclypse\_backup\_download.py [-h] host\_file –apiversion=[1/2]