Open Automation Documentation

Release 0.1

Rob Edwards, Matt Day, Russ Whitear

April 06, 2016

This guide was created to help with developing a custom Open Automation plugin for UCS Director. It is not an official Cisco guide however deep knowledge of the products, using official documentation and utilising resources availble to the authors this has been put together. Hopfully it will help on your journy and if you have feedback please get in touch or even make updates yourself and make a pull request.

The code is open source, and available on github.

The documentation for the site is organized into the following sections:

CHAPTER 1

Introduction

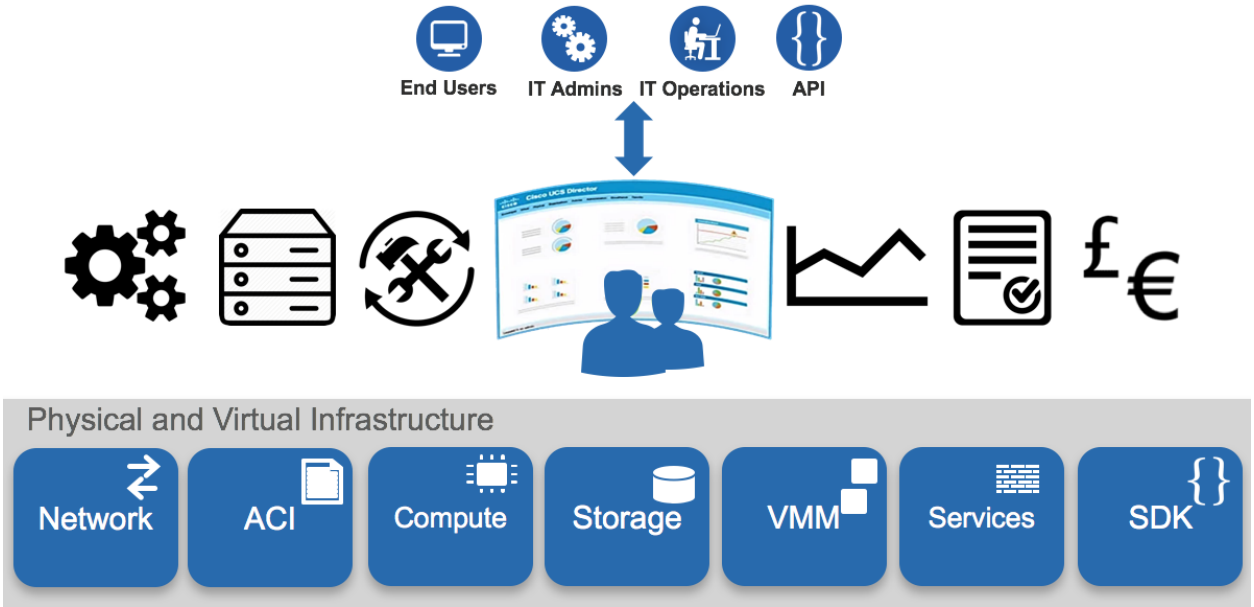
Note: These pages are being built up at the moment and not complete, please treat them as a work in progress!

1.1 What is UCS Director?

UCS Director is a tool to help you automate your data centre infrastructure (Cisco Website). It abstracts hardware and software into programmable tasks that are assembled together to provision infrastructure across computing, networking and storage resources that reside on multiple hypervisor. The busines value is;

- Simplified infrastrucitre provisioning and managment that takes munites rather than weeks
- Pyhsical, virtaul and multi-vendor management from a single place to accomodate heterogeneous data centres
- Increased IT agility allowing IT to have greater impact on the effectiveness of the business.

The following image summarises the functionality of UCS Director;



The main call out feature are;

- Tasks and workflows - more than 2200 out of the box tasks (about 2400 person days of effort), these can be used to configure workflows that combine usful tasks into a single goal.
- Bare Metal - Assist with the provision and configuration of bare metal (PXE)
- LIfecycle Management - manage day to day tasks of the entire infrastruitre from a single place
- Metrics - Collects some preformance related metrics (NOT a perfromance management tool)
- Reports - Generate reports on the infrastruture
- Showback/chargeback - Tracks the usage of infrastructure resources to showback usage for groups (or extracted for billing engines)

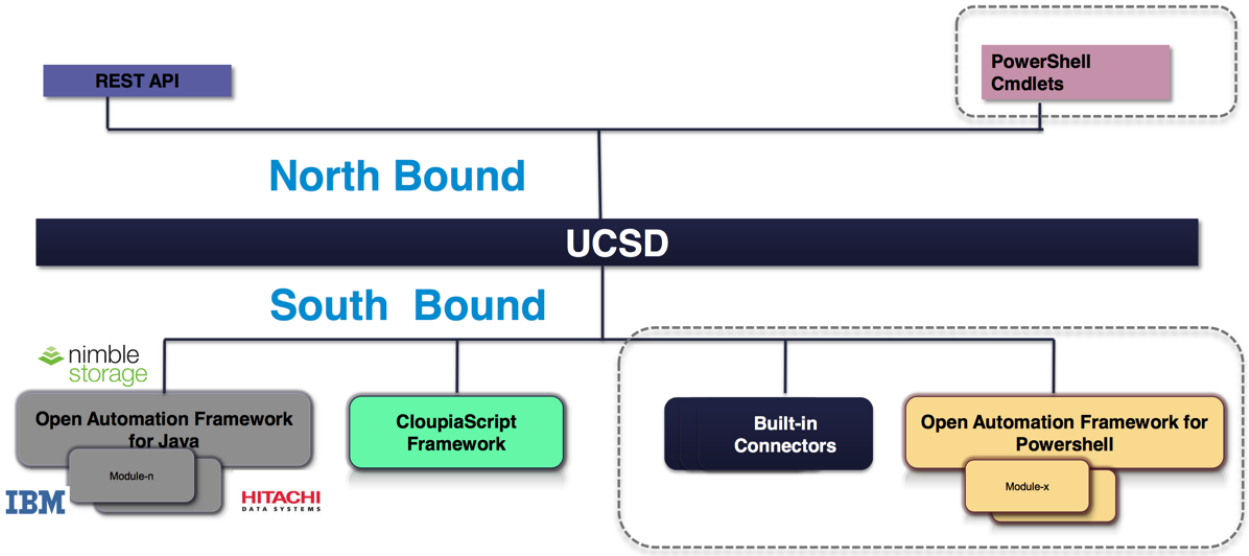
It is possible to interact with UCS Director in multiple ways;

- Southbound
- Northbound

Access Northbound, access to the UCSD functioanlity, can be via the web GUI, RestAPI or Powershell Cmdlets (uses the RestAPI).

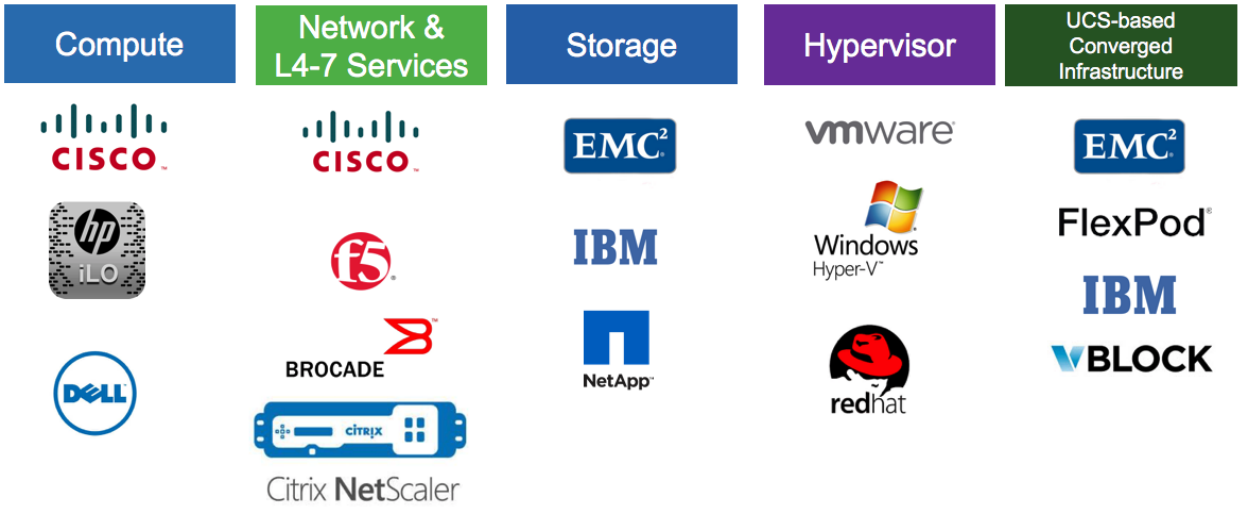
The Southbound access, communication to infrastruitre, can be achieved by one of these mechanisms;

- Built-in Connectors
- Powershell
- CloupiaScript
- Open Automation



The built-in connectors are the devices that are supported out of the box. These suported devices have automation tasks prebuilt and tested by Cisco. The official list of infrastructure supported by UCS Director is here.

Supported vendors;



Powershell support is via a powershell agenet and allos interaction with the underlying infrastrutre using powersehll functionality.

The CloupaScript framework allows you to create custom task (based around JavaScript) to interact with manged devices or it could be expanded to interacte with thinsg that are not managed in UCS Director. These taks are quick ways to expand the capbilities of UCS Director and many examples can be found on the Cisco Communities site.

1.2 What is Open Automation?

While the use of CloudpiaScripts and the Powershell agent makes UCS Director a big flexibe tool its southbound capabiliites don't end there, it also has the concept of 'Open Automation'. This can be used to make 3rd party devices look like they are nativly supported by UCSD by allowing you to;

- develop reports and report actions
- inventory devices
- track changes made to the system through the module
- develop task that can be used for workflows
- develop and scheulde repeatable tasks
- set up new resource limits

Open Automation is in essance a Java SDK that can be used to create a connector/plug-in that can be imported into UCSD.

The official Cisco documentation (always improving) for this can be found here.

CHAPTER 2

Getting Started

To start will we will run though how to setup your development environment and build the example dummy plugin that is on the Cisco download site for UCS Director.

Note: Make sure that you pull the dummy example for the version of UCSD you are running!

The task that we will run thorugh with you are;

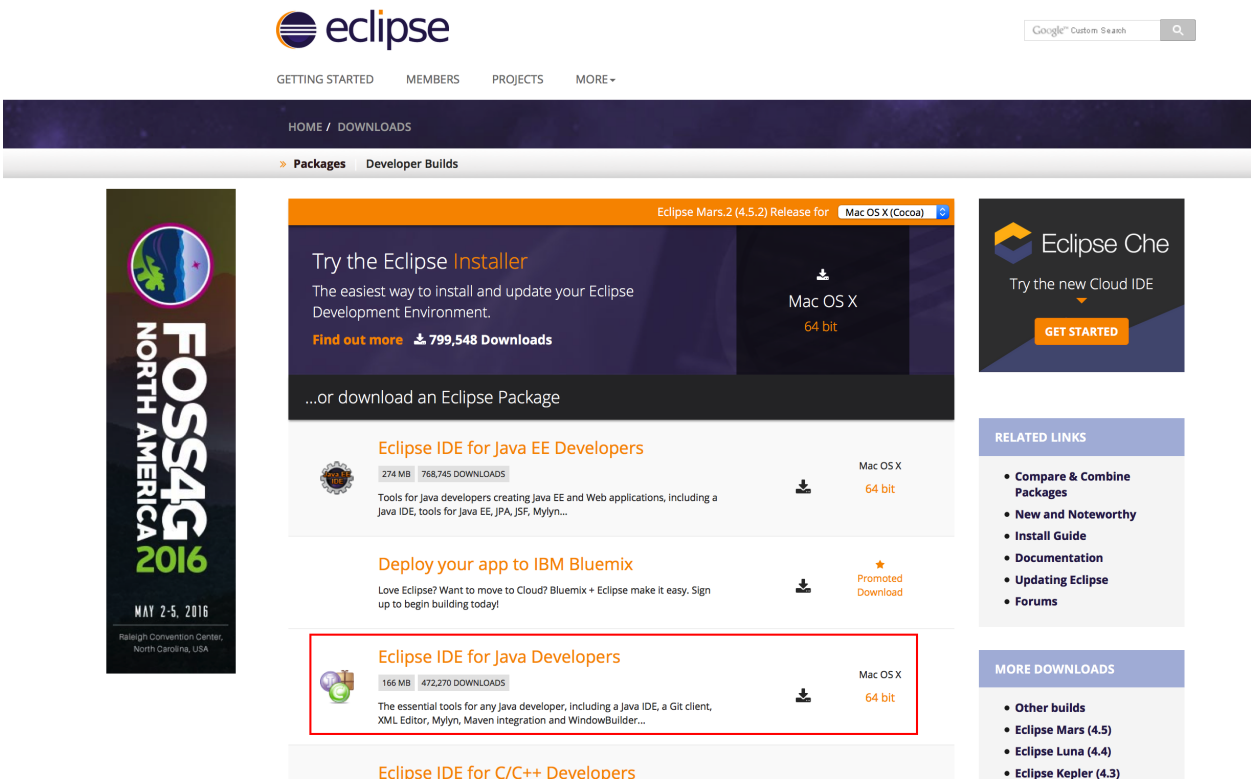
- Getting started and setting up Eclipse
- Importing dummy SDK into Eclipse
- Build Dummy SDK
- Import Dummy SDK & Enable
- Check Dummy SDK loaded (inluding logging info)

If you are already using a Java IDE and happy with the process to build and import plugins skip ahead.

2.1 Setting up Environment

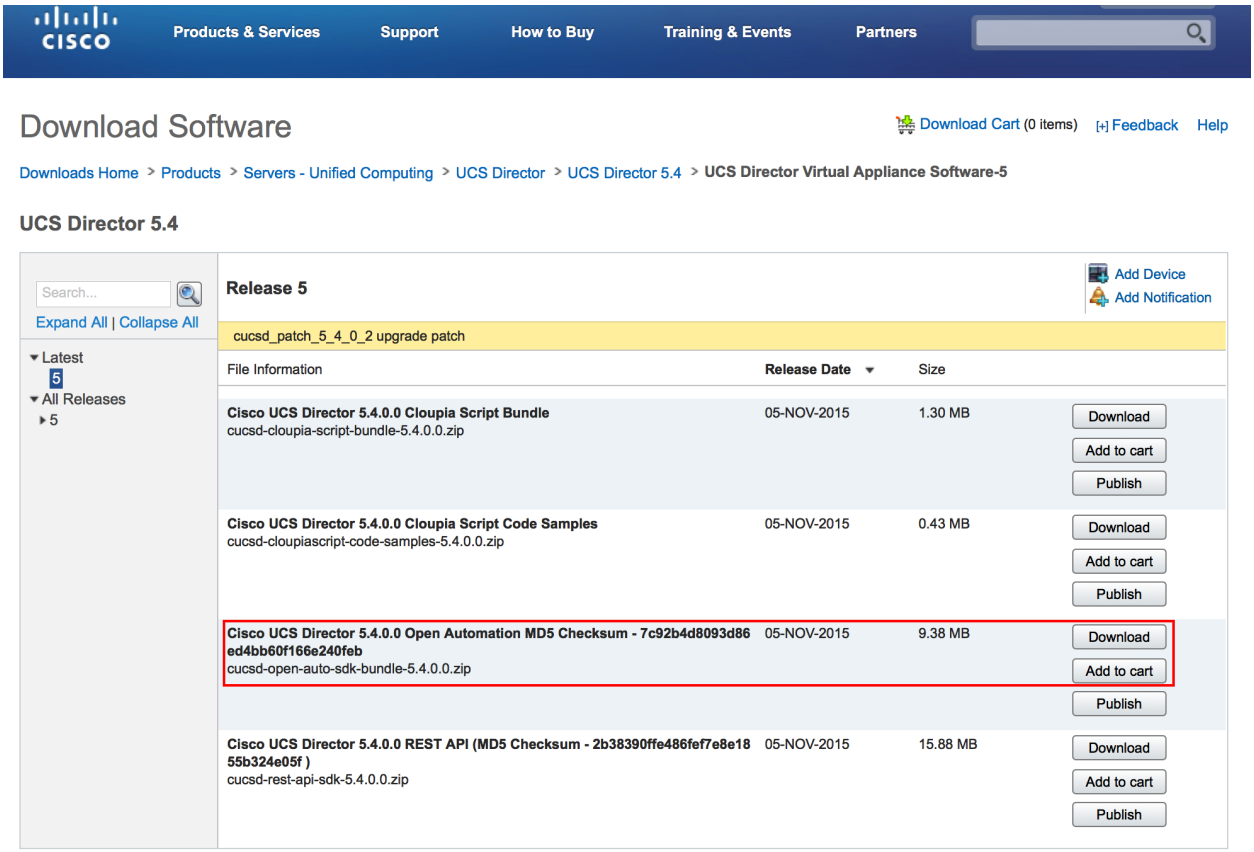
2.1.1 Downloads

The first think to do is download Eclipse for the operating system that you use. The one we used was 'Eclipse IDE for Java Developers'.

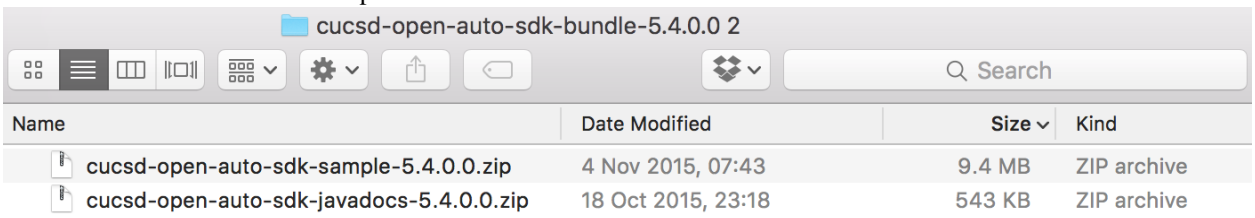


Once downloaded follow the instalation instruction, we won't go through these here.

Now you should download the UCS Director Dummy Plugin for your version from Cisco CCO. The file that you are looking for is *cucsd-open-automation-sdk-bundle-<ver>.zip*.

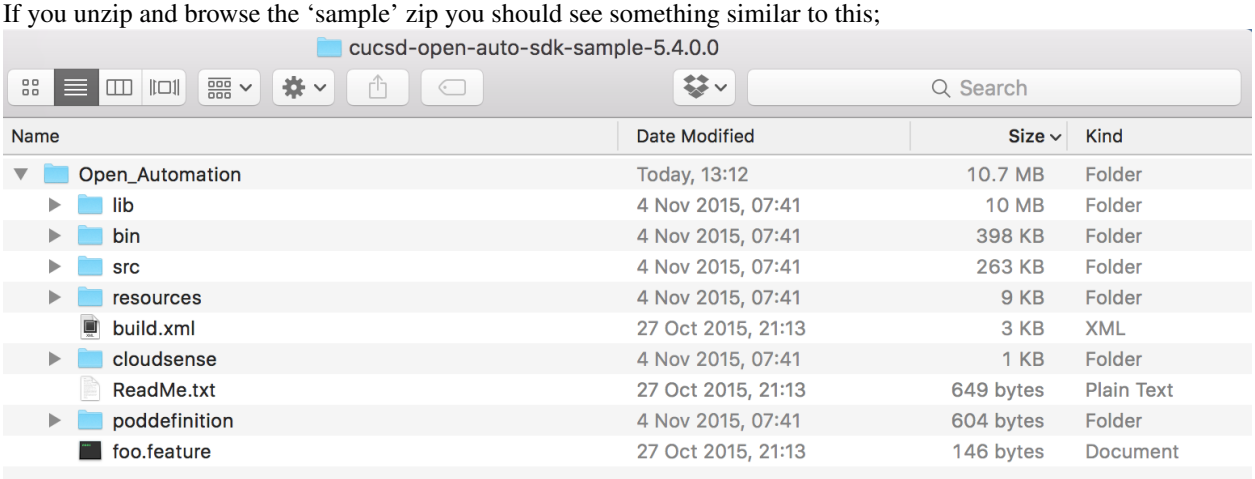


The download will contain two zip's



The first is the sample, or dummy, open automation code while the second is the Javadocs that we will use to enhance Eclipse.

If you unzip and browse the 'sample' zip you should see something similar to this;



2.1.2 Eclipse

As we ae using GitHub to share and collaborate on the examples we create the Eclipse EGit plugin was added to our environment. This is optional.

2.1.3 Setup Project

Now that we have an IDE and the sample open automation code downloded we need to import it into an Eclipse workspace.

import project java versions - JRE 1.8 Set Java compiler to JRE 1.8 java docs

2.2 Build Project

2.3 Import to UCS Dirctor

2.3.1 Import plugin

2.3.2 Enable plugin

2.3.3 Debug import

Check Dummy SDK loaded (inluding logging info)

2.4 Vaidation

Check it works

CHAPTER 3

Architecture

Explain the arhitecture of the plugin

- Account
- Tasks
- Reports
- Cloudsense
- Pod Defintions
- Resources
- etc.

CHAPTER 4

Cookbook

description of code and example snippets

CHAPTER 5

Account

CHAPTER 6

Tasks

To create tasks and make them availbe you will need to create two classes;

- <name>Config.java
- <name>Task.java

The jdo.files will also need to be created/updated.

You will also need to add the task classes to the main module

6.1 Config.java

```
@PersistenceCapable(detachable = "true", table = "spark_dummy")
public class SparkDummyConfig implements TaskConfigIf {

    public static final String displayLabel = "Spark Get Inventory";
    @Persistent
    private long configEntryId;
    @Persistent
    private long actionId;

    @FormField(label = "Spark Array IP", help = "spark Array IP Address", mandatory = true)
    @UserInputField(type = WorkflowInputFieldTypeDeclaration.IPADDRESS)
    @Persistent
    private String  ipAddress = "";

    @FormField(label = "Spark Username", help = "spark username", mandatory = true, type = FormFieldD
    @UserInputField(type = SparkConstants.GENERIC_TEXT_INPUT)
    @Persistent
    private String  username;

    @FormField(label = "Password", help = "Password", mandatory = true, type = FormFieldDefinition.FI
    @UserInputField(type = SparkConstants.PASSWORD)
    @Persistent
    private String  password;
```

6.2 Data.java

```
public class SparkDummyTask extends AbstractTask {

    private static Logger logger = Logger.getLogger( SparkDummyTask.class );

    @Override
    public void executeCustomAction(CustomActionTriggerContext context,
                    CustomActionLogger actionLogger) throws Exception {
            SparkDummyConfig config = (SparkDummyConfig) context.loadConfigObject();

            String username = config.getUsername();
            String password = config.getPassword();
            String ipAddress = config.getIpAddress();

            String arrayName = "";


    /* Code */
    }

    @Override
    public TaskConfigIf getTaskConfigImplementation() {
            return new SparkDummyConfig();
    }

    @Override
    public String getTaskName() {
            return SparkDummyConfig.displayLabel;
    }

    @Override
    public TaskOutputDefinition[] getTaskOutputDefinitions() {
            return null;
    }
}
```

6.3 jdo.files

```
//Each package with config classes should have a file called jdo.files
//Each config class should be listed in the format below
//This file informs the build file which classes need to go through JDO enhancement
+SparkDummyConfig
```

6.4 Main Module

```
@Override
    public AbstractTask[] getTasks() {
            AbstractTask task1   = new SparkDummyTask();
            AbstractTask task2   = new SparkMessageCreateTask();

            AbstractTask[] tasks = new AbstractTask[2];
            tasks[0]  = task1;
```

```
            tasks[1]  = task2;

            return tasks;
    }
```

CHAPTER 7

Reports

CHAPTER 8

Example Plugins

LIst of examples that have been created by the community (or other vendors). These are great ways to learn how others have build plugins (best way to learn in my eyes)

Plugin	Description	Creator	Location
Nimble		Community	https://github.com/rwhitear42/nimble
Nimble		Community	https://github.com/rwhitear42/slimcea
Infoblox		Community	https://github.com/rwhitear42/infoblox
3Par		Community	https://github.com/CiscoUKIDCDev/HP3ParPlugin
Spark		Community	https://github.com/CiscoUKIDCDev/UCSD-OA-Spark
Pure		Pure Storage	https://github.com/purestorage/ucs-director-plugin

Other comersial plugins are availbe however these will not be tracked here.

CHAPTER 9

Why and who

Quick description of why and who has been building up these pages