System Documentation

Overview

This details some systems information about the new scripts required extensions and notices of changes made to existing scripts. Documentation comments are included in each script. Example files and network designs are included separately. A separate user manual describes how to create templates and deploy networks in this system.

The additional scripts add functionality to create templates for networks. These templates specify a set of rules that will allow the construction of a larger scale network. The system allows for the specification of IP address ranges, interface connections and routing protocols, which will be applied to the device to correctly set up IP address allocation and routing.

Additional settings can be applied to specific routers through components which are specified in the template. These allow for specific changes to a subset of routers, within a certain container. E.g. "Every Model 1 in Enclosure Red".

The scripts utilise the existing architecture and data structures. Changes made to existing scripts are documented in 2.2 Edited Scripts.

System Architecture

The scripts can be categorised into two roles. The first role primrarily interfaces between apache and the network builder scripts, this provides a way to build network configurations as described in the user manual. in the templates folder, various network topologies are logically described and the devices they will be replicated to. Once the configurations are built, they are saved in the networks folder which contains a manifest of the devices to be uploaded to as well as teh actual config file to be pushed to the specified switches/routers.

The second role interfaces primarily between the upload scripts and net control. The upload scripts will request device details form netcontrol and the ssh connection to each of the routes/switches that configs are desired to be pushed to. The upload scripts will read the configs from the networks folder and push these configs to each device according to the manifest.

System Setup

Packages

- Apache2
- PHP 7.4
- PHP cURL
- PHP ssh2
- PHP pecl_http1.7.6 (if php version is less than 5.4)
- dialog

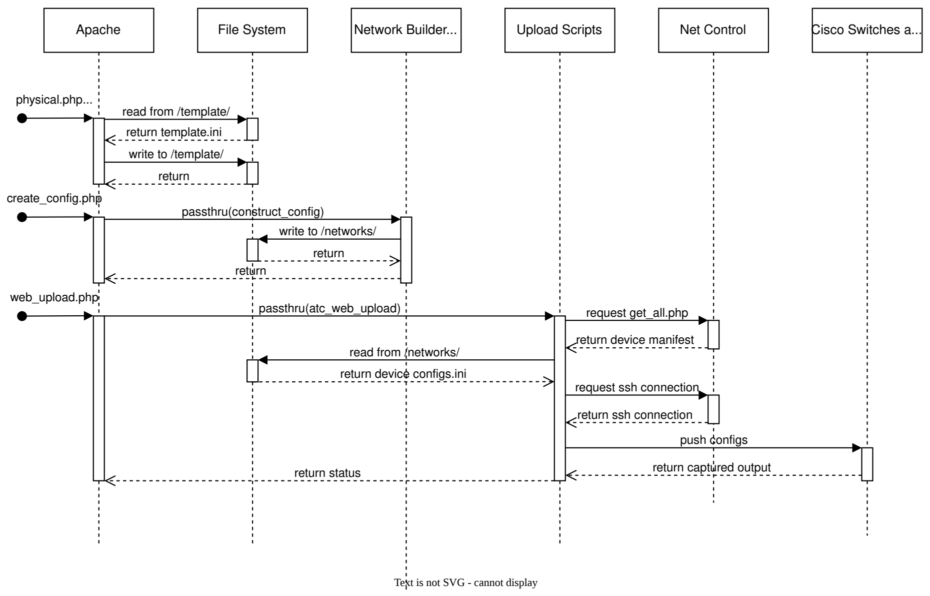

Figure 1: System Architecture

Directory Structure

- cisco_scripts
  - bin/
  - lib/
  - templates/
    - config/
    - replication/
    - physical/
    - details.ini
  - networks/
  - template_trash/
  - config.ini
- www/html/

Config.ini

An Environment Variable for the path to Config.ini is required under the name CISCO_PATH. Config.ini contains the file paths for the following

- bin_path: This is where the command line, and web backend scripts are stored.
- lib_path: php library scripts that support the php scripts.
- physical_path: path to the physical folder
- template_path: This is where all the templates are stored.

- network_path: This is where built networks are stored when constructed from the templates.
- template_trash: When deleting templates, this is the path that the files will be moved to, to be manually removed later.

Details.ini

Details.ini is a file that contains arrays by the name of Rooms, Racks, Kits, Routers, Switches that include the name of each Room, Rack, Kit, Router, and Switch. This is already included.

File Permissions

The folders: templates, templates/config, templates/replication, physical and networks need to be owned by the apache group www-data. This is because the web interface is responsible for saving and overwriting files in these folders.

> NOTE: If you have done the above and are tyring to load an already existing config file such as physical.ini, it won't let you save it because you'll need to remember to change the permission of the file to www-data in this case as well.

Apache Setup

In order for the web interface to function correctly, values need to be added to the apache.conf file, in the directory container for /var/www. The default directory index needs to be changed to index.php and an apache environment variable named CISCO_PATH needs to be added to point to the directory of where config.ini is stored.

Additions

New Scripts

bin/construct_config: This handles the creation of network configurations based on a replication, template and physical wiring scheme. This will produce a set of config files for each device as well as an upload ini file for each room in the networks folder under a subfolder with the job name. bin/atc_web_upload: This is a modified version of atc_build_network which is designed to present information used for the web page and achieves the same result of uploading configuration files to devices specified in an upload ini file.

bin/check_topology: This script uses the physical template and replication scheme to validate that the connections are expected by running ipv6 pings across devices. The results are reported to the web interface. **bin/atc_ios_check**: This script will collect the name of the bin file in the device directory. It will then report this to the user to validate the ios.

lib/http_request: This is a wrapper class that encapsulates a subset of the functionality of pecl_http-1.7.6s HttpRequest class. This script uses cURL and is designed for backwards compatibility for the scripts when run post php version 5.3. See edited scripts. This script includes functionality for setting a url, setting Method, setting PostFields, sending the request and getting the result code and body. As well as enabling cookies (untested).

lib/node.php: This is a class that holds data about connections and interface settings for use in construct_config

lib/route_map_file.php: This contains some simple classes for holding ACL, PrefixList and RouteMap Statements. They work similarly to the data structures in router_config.php, but are used to not interfere with the existing complicated data structure for ACLs.

lib/upload_ini_generator.php: This is a simple script used to handle the associative array produced by construct_config and ensure it correctly saves the configs for each device creating the upload ini file that is used in the uploading scripts.

Edited Scripts

bin/atc_build_network, bin/atc_upload, bin/atc_upload_new, bin/atc_wipe_devices, bin/atc_check_devices, bin/clear_devices, lib/smartrack_functions, lib/esp_control: All these scripts used the deprecated HttpRequest class from pecl_http1.7.6 which is no longer compatible in php versions post 5.3. Thus a check of the php version at the start of the script is made to see if the php version is post 5.3, if it is the library lib/httpRequest is loaded. This allows for backwards compatibility.

routerconfig.php: edits to the classes were made to allow for their use in the construct config script. This mainly involved the addition of the function ConstructConfigString which parses the data each class holds to a string of configs that can be sent to the cisco devices. Additionally various fields and a few classes were added to the data structure to allow for further configurations, these include the classes: Neighbour, Syslog, NTP, VRFDefinition. No edits were made to the parsing of data in relation to collecting configs.

devicecontrol.php: added the reset3650Switch function which will issue the no system ignore startup-config switch all command. Also added setting of default value for the config-reg. The device model is now required for the atc_wipe_device script.

bin.program_device: commented out the GOEnable command as this was calling HandlePrompt which would do undebug all. Alos change teh copy run start to do copy run start, but either way we cannot guarantee we will be at enable mode in the end. So this is not good.

Template Files

Physical Files Physical Files are ini files, they specify each router by name as a Heading. Under each router is a mapping between what is denoted as a 'direction' and the physical interface name. This allows an abstraction from the actual names of the interface. Additionally it includes a list of connections these are used when verifying the physical topology's connectivity. An example physical file is provided as physical.ini.

You can specify subinterfaces to allow the use of multiple logical interfaces on the single physical one, however if done so the physical template must not also include the parent interface, as it will have mandatory configurations pushed to it that prohibit its use as a regular interface.

Direction The direction is used to allow for configurations to be mapped to each router in the same way regardless of the actual name using templates. For each router a direction should map to the "opposite" direction of an adjacent router. Every interface that has an equivalent connection on each router should be given the same direction

Loopbacks The construction of networks using "construct_config" will validate that each interface is actually present on the router. This uses the physical configuration file. Thus if requiring loopbacks to be configured through the template system, it is required that you add a new direction, interface pair for the loopback interfaces. You do not need to specify any connections for them.

Configuration File The configuration file describes a sub-topology. The sub-topology is a repeatable part of your large scale network that will be replicated to each device. The configuration is composed of models. Models are representation for a configuration of a router. Thus each model can be thought of as a single router, the connections between models can be thought of as being internal, while the connections between groups of models (sub-topology) can be thought of as being external. Please view the sample template file, configuration.ini

The key elements described in a configuration file are the models and how they map to the physical routers. The connections between models, and the routing protocols and their advertisements. The configuration file also allows the configuration of "components" which are specific configurations that only need to be configured on that device without regard for the configurations of its neighbours.

Replication File The replication file describes what devices will be included in the upload. It lists the rooms, enclosures, kits and routers to be uploaded to. The order of the list is important, it describes what room/rack/kit is connected to another. If the connection differs from one room to another or from one rack/kit to another you can specify a heading based on the parent container and describe a unique order for that container. Please view the sample replication file: replication.

VRFS The first lines of the replication file set what template is to be used. When wanting a standard single network ensure you set the "VRF[Global] = template_name." Global is a keyword that will ensure the network settings are deployed on the global process. Additional templates can be deployed to the same set of routers by including additional VRF names, and template file names.

Note: There is a maximum of 10 vrfs suppoerted simultaneously.

Networks Folder The Networks Folder includes instances of configurations that are to be uploaded to the devices. A folder is made on execution of "contruct_config" by the job name. Inside the folder there will be an upload ini file for each room that is to be uploaded. These are in the format as required by "atc_build_network". Included is a config folder which contains the text files for the configuration of each device included in the replication.ini file used.

Expanding The Program

Adding Additional Configurations

When wanting to add new configurations there are 4 areas to script.

1. You need to decide whether this configuration requires an awareness of the sub-topology and its neighbours Also whether you want to be able to restrict the configuration to particular Models, or Containers.
2. Add or modify a relevant data structure in routerconfig.php.

- Here you will need to hold all the data that you want to be able to set in the configuration. You also need to make a ConstructConfigString function to convert the data to the formatted configuration for the cisco devices.
- Then ensure that the RouterConfig Class can store your new data structure, and add the lines in its ConstructConfigString to call ConstructConfigString for each of your data structures

3. Add the configuration in construct_config

- Add the configuration to construct_config. Your answers in 1 will decide how best to implement it. If the configuration requires complete understanding of the entire sub topology, have a look at how Interfaces are created in ConstructComponents. If the configuration is only router specific, or you want to be able to exclude the configuration based on the room etc. It may fall in the category of components, so you can see how syslog is configured for an example.
- Go to BuildRouter, here is where you will attach the configuration to the actual router instance added in RouterConfig. If your configuration uses

abstracted names such as interface directions, you will need to translate them to the actual physical name.

> NOTE: The use of VRFS modifies the physical name so to make sure you get the correct one use the “links_to_name” array in the Router Class to get the correct physical name for the direction.

4. Modify the Web Interface

- Here you will need to modify the build_template.js file to include interactive elements for creating your configuration. If it is a new routing protocol or component simply check what the select input is for the component/routing protocol in CreateComponentDetails/CreateRoutingDetails, and put your inputs in the if statement.
- You then need to go into the save_template.php file and ensure the configurations are saved in a way that will suit the construct_config script.

Configuring Switches

Adding Switches should not be too difficult but may require a lot of alterations. First there would need to be a new loop in the ConstructConfigs function to loop through each switch similar to how the router loop works. This then can take in similar parameters and can be built up in a similar way to how the routers are built.

The file switchconfig.php will need to be edited in a similar way to the routerconfig.php adding ConstructConfig to each data structure to produce the configurations for the cisco devices.

Upload_ini_generator shouldn’t require modification as long as the array is structured in the same way, just with added switch_keys adjacent to the router_keys.

The templates may be the most difficult to modify to include switches. So it may be more feasible to have separate switch templates. This is because you need to determine whether a model is applied to a router or to a switch, and prevent the user from applying a model to both a router and a switch.

Regardless whether a single template or two templates are required to implement switch configuration, the connection between router and switch will need to be considered. The routers usually will check the configuration of connected links to determine their own ip address. However this is not to be the case with the connection to a switch. Additionally there needs to be configuration of subinterfaces in the case of use with VLANs. But this also needs to not interfere with subinterfaces created due to VRFs being deployed.

Concerns for Merging

The following concerns should be considered if these new scripts are moved to replace or merge with the existing php cisco_scripts.

- httpRequest is deprecated so the new scripts use a wrapper of cURL in a class of the same name, the function enableCookies in esp_control is untested and may not work as expected.
- The file routerconfig.php has been modified, while no changes were made to the parsing from collected configurations into the data structure, the data structures themselves have had new fields added. Thus tests should be made that the marking of exams are consistent across the old version of routeconfig.php and the new version.
- There is a possible conflict with the frame relay interface DLCI command. When parsed as per ParseNetInterface(), it sets mapIP to be InverseArp, However the ConstructConfigString in NetInterface class will assume mapIP is the next hop ip to map the DLCI to the next hop, this only occurs if the mapDLCI is set. It seems that the passing of EIGRP and OSPF will set the mask to the wildcard straight from show run. This will cause erroneous output if you use ConstructConfigString to reassemble the configuration. Currently no scripts do this.
- program_atc_device as been changed which might cause issues.

Reasonings

Directions

Connected Interfaces should not have the same direction number. This arises due to ambiguity for connections across sub-topologies. This would result in a connection such as: (First Replication) Model 1 Interface 1 Connected to (Second Replication) Model 1 Interface 1. The issue arises in the external connections because Interface 1 becomes both an input of the previous replication and an output to the next replication. As the construction of configs is agnostic to the replication the program may result in an erroneous determination that interface 1 is connected to itself. Thus Ip address assignment will be incorrect.

Interface Naming

Interface names can be shorthand or full named, as ling as the cisco ios command line would recognise the name, it should work. However, the check_topology script and construct_configs script handle names slightly differently.

Check Topology Interface Naming Check topology will always shorthand the interface names, by taking the first letter and the code, and contracting them. This is becuase the various tests run all report different interface schemas such as ipv6 interface brief, and show cdp summary. Thus the reliable way to match interfaces is by shortening them in each case.

Construct Topology Interface Naming This will not shortened the interface name, instead configuring the interface as specified, so Serial0/1/0 will be exuctued on the Cisco CLI as Serial0/1/0, or s0/1/0 will be executed as s0/1/0.

However some function will still make use of the trend that the first letter and code should uniquley identify the interface such as for the encpasulation process.

Sub-Topology Limit

The sub-topology chosen when creating a template must be limited to a single kit. This is because it simplifies the process of creating the network by the program in a way that it can also easily interface with the existing organisation of the devices. Furthermore as the devices are already organised and connected in this way the possibility for creating connections differently would likely require a physical change anyway. Given further development this connection between sub-topology and kit could be removed.

Loading the page

In order to get the page to give feedback while uploading the configurations or checking hte topology, each script is modified to flush its output. The output of these scripts are an array that can hold information or a percentage of the completed progress as a serialised array. The php handling script that is present in the web directory will receive this data and flush its own output after processing the data. (Usually into scripts that will update the information or loading bar).

Receiving Reports From Devices

Both the physical checking script and the ios check script require Reports based on the configuration of the devices. To achieve this the parent process will create a socket pair for each child process. When the child captures outputs of commands it will write to its socket for the parent to later receive the data.

No child Script in ios_check and topology_check

The issue with using sockets is that the usual format of a child script that handles the device control console for actually executing the commands is not possible as the socket resource cant be serialised through the passthru command. Thus the ssh connection is made directly in the main atc scripts

Physical Checking Reporting Process

The physical checking reporting process is a bit more complicated than the ios check, as it requires the device and its neighbours to be in a particular state before moving to the next portion. This is achieved by making the children script wait on a socket read for the go ahead form the parent. Thus the device will collect ping data once all its neighbours have completed uploading. Once completed the ssh connection will be terminated. The devices must be manually wiped, using the wipe menu option.

Removing Invalid ACL/Prefix/Routemaps

Sometimes when configuring a list it might require that a statement referening to a link that only exists on some models of each instances sub-topology. This would be the case if applying an acl to a link to connected to a previous sub-toopolgy, which would be a link that doesn;t exist on the first instance of a sub-topology. In this case, the acl and any other lists that depend on it are removed to prevent undefined behviour.