**01\_connected**

List of devices that are mounted and connected in ODL. With a clean ODL there is just one item listed which is not a remote device.

**01\_not\_connected**

List of devices that are mounted in ODL but not connected to ODL. With a clean ODL the list is empty. If a network device is removed from the network then it is listed here.

**01\_mount\_device**

All the devices in the settings are mounted in ODL. Connection will be attempted. Freshly mounted devices will initially be listed as not connected but should be listed as connected after a few seconds.

Recommendation: run script 01\_connected repeatedly until the mounted devices appear in that list.

**01\_dismount\_device**

Reciprocal of the script for mounting devices. Dismounts all the devices listed in the settings.

At the time of writing, ODL does not support dismounting devices and returns an error code when attempted. Reported as bug 2184, see <https://bugs.opendaylight.org/show_bug.cgi?id=2184>, which has status ‘resolved’.

**02\_interface*\_*names**

The settings contain the address of one management interface per network device. Note that the name of the interface is not present in the settings. The name can be discovered using this script. The names of all other interfaces will also be listed. This is done per network device, subject to a device being mounted and connected, of course.

The interface names are extremely useful because they are the input parameter to many functions, such as configuring an interface.

**03\_interface\_properties**

Display the *properties* of each interface discovered in step 2.

**03\_interface\_configuration**

Display the *configuration* of each interface discovered in step 2.

**03\_interface\_configuration\_update**

One of the interfaces from step 2 is chosen such that it is not a management interface. The initial configuration is displayed, then updated and re-displayed, then restored and re-displayed. The fields that are updated include the IP address and mask and the description. The interface name is not modified. This script is capable of shutdown and startup.

This script is particularly well suited for workshop participants to ‘hack’.

**03\_interface\_shutdown**

One of the interfaces from step 2 is chosen such that it is not a management interface and is currently *up*. The initial state is displayed, then updated to be *down*, then re-displayed. The state is not restored to *up*. Thus this script can be run repeatedly to shutdown multiple interfaces. There is an informative message when there are no more *up* interfaces.

**03\_interface\_startup**

Reciprocal of the *shutdown* script to *startup* one interface. Can be run repeatedly until all interfaces are *up*.

PS: I experienced an inconsistency with ODL which prevented the scripts from working as intended. I accidentally discovered a work-around for the issue, which I have applied to the scripts. The problem is that ODL reports stale information when the configuration is shutdown or startup. That is, it appears that the update has failed because the state does not change. Waiting for time to pass does not improve this scenario. However, it is possible to prompt ODL to refresh itself by executing the scripts from step 2. Thus there are lines of code in the scripts that appear to be unnecessary yet they do have a purpose.

**Step 04**

* 04\_routes
* 04\_topology

Each script requests information from ODL and successfully receives a response. However, the response will not contain rich content unless the appropriate configuration exists.

At the time of writing I have not been successful at configuring my local network simulation for BGP. The network devices were configured into a *group* but I did not *reflect* the group to ODL. A goal that our team could pursue is to configure ODL as a BGP client *programatically* (via the Python scripts). The information in the settings would be used to achieve this goal. Information from the ODL team on how to do this has been forthcoming this week.

The stories that can be told with the route and topology information require further development. In the Korean workshop the participants can hack at the responses if sufficient configuration is achieved to provide rich content. In the future, our team could pursue goals such as displaying graphs of this information.

**05\_acl\_list**

List the name of every ACL on each network device.

**05\_acl\_detail**

Show the detail of every ACL on each network device. Same ACLs as per the list of names but with a *drill down*.

**05\_acl\_delete**

Attempt to delete every ACL from each network device. An ACL will not be deleted if it is *in use*, meaning that it is currently applied to an interface.

**05\_port\_deny**

Create an ACL on each network device and apply it to each interface that is not the management interface. Each ACL is applied as a *packet filter*. The following attributes of the ACL can be *hacked* by editing the settings near the top of the script: grant (deny or permit); port (www, echo, etc); protocol (tcp, udp, etc); bound (inbound or outbound).

**05\_port\_permit**

The same as the *port\_deny* script with *grant* set to *permit* instead of *deny*.

PS: The effectiveness of these scripts is inconsistent. ODL requires a deep clean quite often. ODL reports of success or failure are unreliable. For example, ODL will report that a new ACL has been created on a device and applied to an interface when the ACL is not evident on the device or in the list of ACLs.