Utopia

This page captures the RDK-B Utopia module, its elements, design and high level description of utilities involved. This document includes details of the usage of third party open source utilities as part of Utopia. To understand the internal workings of each of these open source utilities please refer the project links shared alongside the utilities.

Introduction

RDK-B has a layered architecture with layers having logically independent functionalities. Broadly the functionality of the gateway device is implemented through 3 main layers: Utopia, HAL and CCSP.

The HAL layer abstracts the underlying hardware like MOCA, Wi-Fi, etc. through a standard set of APIs defined as part of RDK-B HAL for the respective components. This HAL layer is implemented per platform and the rest of the components can be compiled to run on the new platform without major modifications. CCSP components implement the core of the gateway device functionality like, WiFi, user settings, parental control, reporting and configuration.

Utopia is a sub component within RDK-B that deals with a set of utilities and their initialization sequence to configure the base functionality of the gateway device.

The base functionality includes

- Configuring the DHCP Server
- L2 on board switches
- Setting up the iptables
- Process Monitors
- Configuring MultiLan interfaces and creating bridges
- Creating multiple VLans for isolating/securing the traffic across interfaces

High Level Architecture

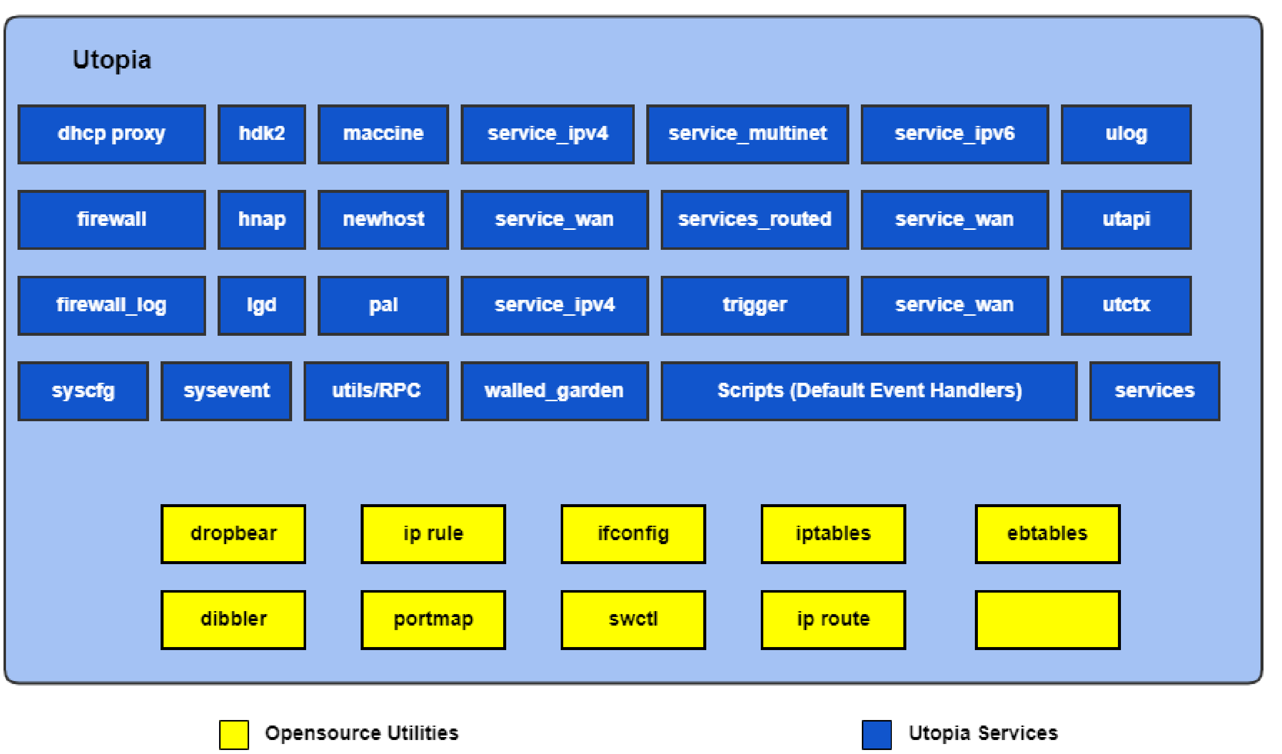


Figure 1 - High level architecture of Utopia module

Utopia is a package with multiple independent utilities. These utilities are launched through a startup sequence using shell scripts.
Few of the sub componentswithin Utopia are:

- dhcp_proxy - Utility to modify the Network Processor (NP) bridge and setup a dhcp proxy between dhcp server (WAN) and dhcp client (LAN CPEs)
- Firewall - Utility which is used to set all the IPv4 and IPv6 rules on device
- service_routed - Utility to set routes using ip rule for IPv4 and IPv6.
- service_wan - Event triggered utility used to bring up the wan services (static and dhcp)
- utctx - Standalone batch get/set application. This provides functionality such as Utopia_Free, Utopia_Init, Utopia_RawGet, Utopia_RawSet. This also has the list of Utopia events.

Utopia also contains Open Source Utilities like brctl, vconfig, dropbear, dibbler, ifconfig, iptables, ip rule, ip route, ebtables, portmap and swctl.

Utopia also contains swctl:

- Switch control utility is designed to address dual switches: internal and external.
- The internal switch connects two processors, network and application processors, in additional to MoCA port and external switch.
- The external switch consists of 4 external Ethernet ports and the port connecting to internal switch.

Code Flow

Utopia Initialisation Sequence

As described in the previous section Utopia is launched and initialised through a set of shell scripts. This section details the scripts and the initialisation sequence.

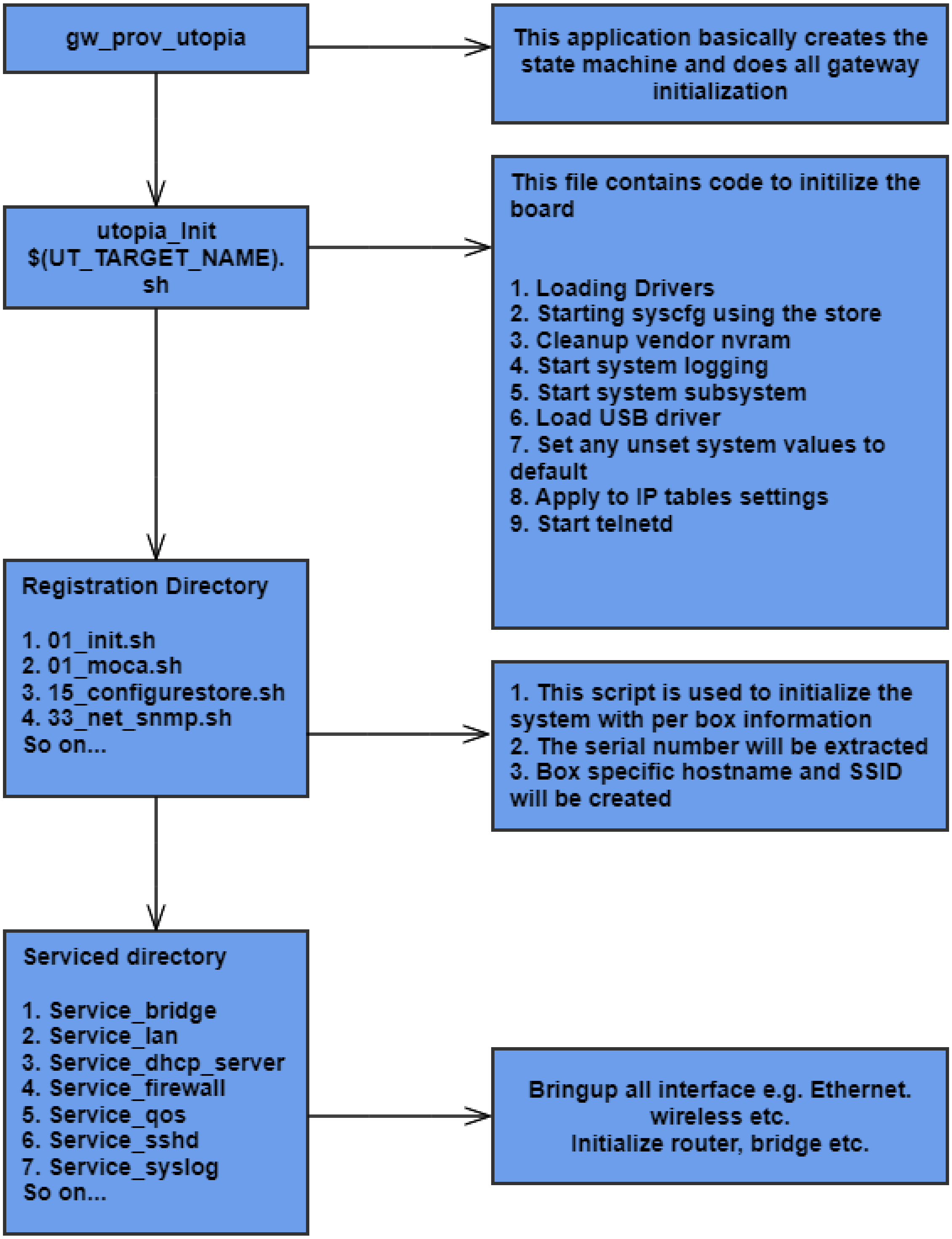


Figure 2 - Utopia Initialization Sequence Diagram

When device boots up following process will take place to initialise Utopia:

1. Application processor CPU kernel comes up
2. Initialize GWSDK using a PCD script present in /etc/scripts/gwsdk.pcd
3. L2 Switch driver initialization
4. RPC management server initialized
5. Start gw_prov_utopia which will initialise CCSP system configuration through utopia_init.sh.

Utopia Initialization Steps from *utopia_init.sh*

1. Set IPv4 and IPv6 network parameters such as tcp timeout, udp timeout, and generic timeout and threshold values
2. Starting log module from log_start.sh
3. Starting syscfg using filestore and creating syscfg.db database using syscfg_create
4. Read reset duration to check if the device was rebooted by pressing the HW reset button using /proc/P-UNIT/status
5. Set the factory reset key if it was pressed for longer than the threshold value. Remove syscfg, PSM storage files and the DHCP lease file. Restart syscfg and execute create_wifi_default
6. Start system logging using service 'service_syslog.sh' with event 'syslog-start' and Start sysevent subsystem using syseventd.
7. Setting the unset system values to defaults values using apply_system_defaults and apply iptables settings.
8. Registration: Run all executables in the sysevent registration directory /etc/utopia/registration.d.
9. Setting up private IPC VLAN on interface l2sd0 with vlan ID 500 using switch handler /etc/utopia/service.d/service_multinet/handle_sw.sh
10. Setting up RADIUS VLAN on interface l2sd0 with vlan ID 4090 using switch handler /etc/utopia/service.d/service_multinet/handle_sw.sh
11. Create IOT VLAN on ARM. Adding VLAN with ID 106 to internal switch using swctl and creating a virtual interface on l2sd0 with VLAN ID 106.
12. Start dropbear process from service 'service_sshd.sh' with event 'sshd-start'.
13. Setting Multicast MAC before any switch configuration using service 'service_multinet_exec' with event 'set_multicast_mac'
14. Utopia initialization is completed by creating utopia_inited flag

Utopia Scripts

Scripts are the sysevent handlers which are tied up with different events

Few scripts that bring up and initialize interfaces

service_bridge
./service.d/service_bridge_arm.sh
./service.d/service_bridge/dhcp_link.sh
./service.d/service_bridge.sh
./service.d/service_bridge_puma7.sh

LAN Service
./service.d/service_lan.sh
./service.d/service_lan/lan_hooks.sh
./service.d/service_lan/dhcp_lan.sh
./service.d/service_lan/wlan.sh
./service.d/lan_handler.sh
./service.d/bring_lan.sh

DHCP Server
./service.d/service_dhcp_server/dhcp_server_functions.sh
./service.d/service_dhcp_server.sh

SSH Service
./service.d/service_sshd.sh

Firewall
These scripts are replaced by C utility defined in ./source/firewall/firewall.c and nfq_handler.c.
Another utility ./source/firewall_log/GenFWLog.c is also defined to generate firewall log and write firewall rules in /tmp/.ipt_rule file.
./service.d/firewall_log_handle.sh
./service.d/service_firewall/firewall_log_handle.sh
./service.d/service_firewall/firewall_nfq_handler.sh
./service.d/service_firewall/log_reader.awk
./service.d/service_firewall/newhost_monitor.sh
./service.d/service_firewall/trigger_monitor.sh

service_syslog
./service.d/service_syslog/syslog_rotate_monitor.sh
./service.d/service_syslog.sh

Default Event Handlers present in Utopia

Each service has three default events that it should handle:
${SERVICE_NAME}-start
${SERVICE_NAME}-stop
${SERVICE_NAME}-restart

For each case following functionality is implemented:

1. Clear the service's errinfo
2. Set the service's status
3. Do the work (Actual Functionality)
4. Check the error code (check_err will set service's status and service's errinfo upon error)
5. If no error then set the service's status

Registration of Event Handlers with SYSEVENT

Sysevent is the utility that will activate respective handlers upon events. It provides quite a bit of flexibility to how events are triggered, and how handlers are run. This flexibility is controlled by activation flags (describing how to run the handler), and tuple flags (describing how to interpret events). The default is to trigger an event only when the tuple value changes and to serialise the activation of each unique handler.

When an event is triggered, the handler will be called with a parameter specifying the name of the event. It is also possible to specify additional parameters to be passed to a handler. The parameters may be constants, and/or run-time values of syscfg, and/or run-time values of sysevent.

The following example demonstrate the range of behaviours:

Name of a handler to be activated upon some event:

```
HANDLER="/etc/utopia/service.d/new_service_handler.sh”
```

Register for $HANDLER to be activated whenever <event_name> changes value. Ensure that if multiple value changes occur, then only one instance of $HANDLER will be run at a time.

```
sysevent async event_name $HANDLER
```

Register for $HANDLER to be activated whenever any value is SET for <event_name>

```
sysevent async event_name $HANDLER
sysevent setoptions event_name $TUPLE_FLAG_EVENT
```

Register for $HANDLER to be activated whenever <event_name> changes value. If multiple value changes occur, do NOT enforce that only one instance of $HANDLER will be run at a time.

```
sysevent async_with_flags $ACTION_FLAG_NOT_THREADSAFE event_name $HANDLER
```

Register for $HANDLER to be activated whenever <event_name> changes and pass the parameter "new_param" as the second parameter in the activation of the handler

```
sysevent async event_name $HANDLER new_param
```

Unregistering

The calls to sysevent async or sysevent async_with_flags will return an async id. The async id can be used to cancel notifications.
Example:

```
asyncid=`sysevent async event_name $HANDLER`;
sysevent set event_name_asyncid_1 $asyncid
```

and later

```
sysevent rm_async `sysevent get event_name_asyncid_1`
```

Default Event Flags defined

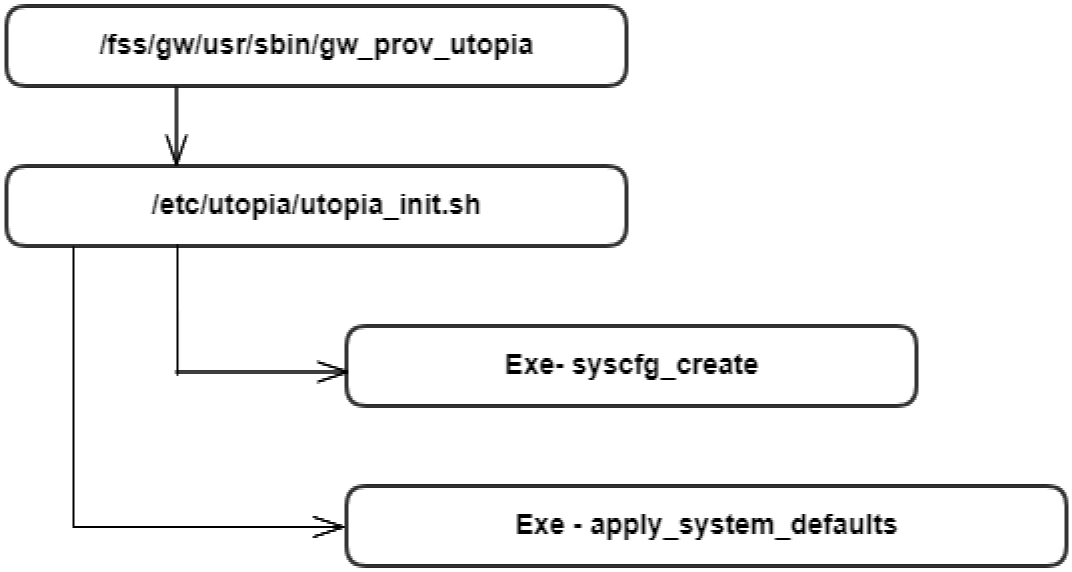
```
TUPLE_FLAG_NORMAL=0x00000000

TUPLE_FLAG_SERIAL=0x00000001

TUPLE_FLAG_EVENT=0x00000002

ACTION_FLAG_NORMAL=0x00000000

ACTION_FLAG_NOT_THREADSAFE=0x00000001

ACTION_FLAG_COLLAPSE_PENDING_QUEUE=0x00000002
```

Utopia Open source utilities

Utility	Description	Reference
brctl	It is a tool used to configure Ethernet bridge (Network Bridging)	https://linux.die.net/man/8/brctl
vconfig	It allows user to create and remove vlan-devices on a vlan enabled kernel. Vlan-devices are virtual Ethernet devices which represents the virtual lans on the physical lan.	https://linux.die.net/man/8/vconfig
dropbear	It is a lightweight SSH2 server designed to be small enough to be used in small memory environments, while still being functional and secure enough.	https://linux.die.net/man/8/dropbear https://matt.ucc.asn.au/dropbear/dropbear.html
dibbler	It is an implementation of DHCPv6 Server/Client	http://klub.com.pl/dhcpv6/
ifconfig	Utility used to configure a network interface	https://linux.die.net/man/8/ifconfig
iptables	Administration tool for IPv4 packet filtering and NAT	https://linux.die.net/man/8/iptables
ip rule	Utility used to manipulate rules in the routing policy database control the route selection algorithm	http://man7.org/linux/man-pages/man8/ip-rule.8.html
ip route	Utility used to manipulate routing tables	http://linux-ip.net/html/tools-ip-route.html
ebtables	It is an application program used to set up and maintain the tables of rules (inside the Linux kernel) that inspect Ethernet frames. It is analogous to the iptables application, but less complicated, due to the fact that the Ethernet protocol is much simpler than the IP protocol.	https://linux.die.net/man/8/ebtables
portmap	It is a server that converts RPC program numbers into DARPA protocol port numbers. It must be running in order to make RPC calls.	https://linux.die.net/man/8/portmap
walled garden	Walled garden is used to restrict internet for devices prior to activation. Once the activation is completed, the device downloads a walled garden config file and the internet is provisioned.	https://www.computerhope.com/jargon/w/walled-garden.htm
igd	Internet Gateway Device (IGD) Standardized Device Control Protocol[1] is a protocol for mapping ports in network address translation (NAT) setups, supported by some NAT-enabled routers. It is a common communications protocol for automatically configuring port forwarding, and is part of an ISO/IEC Standard rather than an Internet Engineering Task Force standard.	https://en.wikipedia.org/wiki/Internet_Gateway_Device_Protocol
HNAP	The Home Network Administration Protocol (HNAP) is an HTTP-Simple Object Access Protocol (SOAP)-based protocol that can be implemented inside of network devices to allow advanced programmatic configuration and management by remote entities.	HNAP

User and default configuration initialization

Syscfg_create executable creates shared memory with user configuration data (/nvram/syscfg.db). This is present in the code base at the location /ccsp/utopia/source/syscfg



Apply_system_defaults executable reads the data from system_defaults file (path: /etc/utopia/system_defaults) and compares with syscfg.db, in case of any data is missing in syscfg, those defaults are written in to shared memory. On start of any module, data is read from the shared memory during initialization.

If syscfg.db does not exists (e.g in case of factory reset) apply_system_default executable writes all default data on to shared memory and syscfg_commit() gets called which in turn creates syscfg.db.

syscfg variable definations are defined under utopia(syscfg_lib.c file).

/nvram/syscfg.db, is a database of all the syscfg variables info. when ever we set a value using "syscfg set" the value will be updated in syscfg.db file.

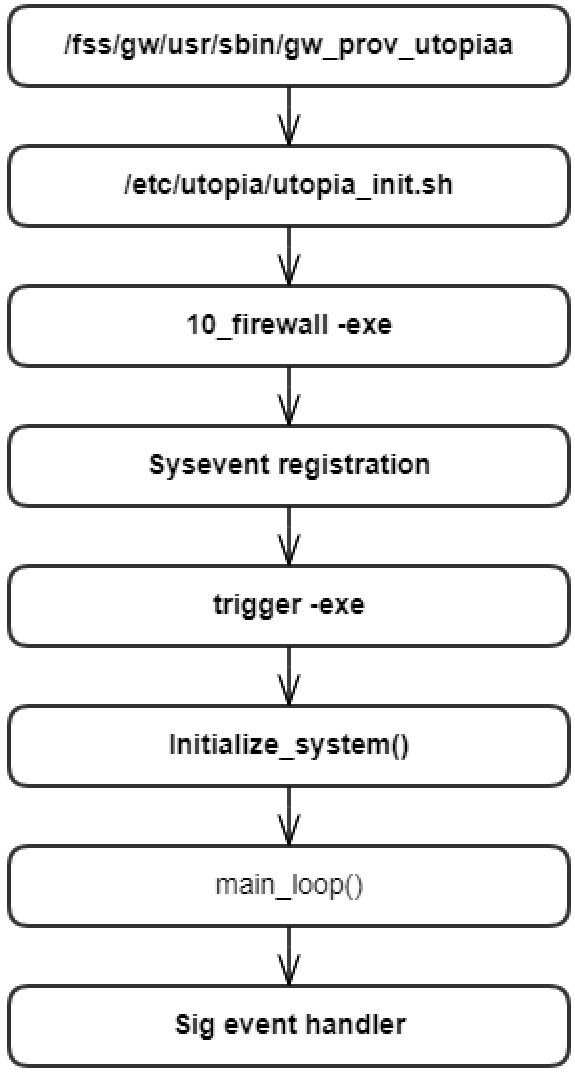
Examples:

```
syscfg get wan_physical_ifname
syscfg get lan_ifname
syscfg get ecm_wan_ifname
syscfg get lan_ipaddr
syscfg get X_RDKCENTRAL-COM_LastRebootReason
```

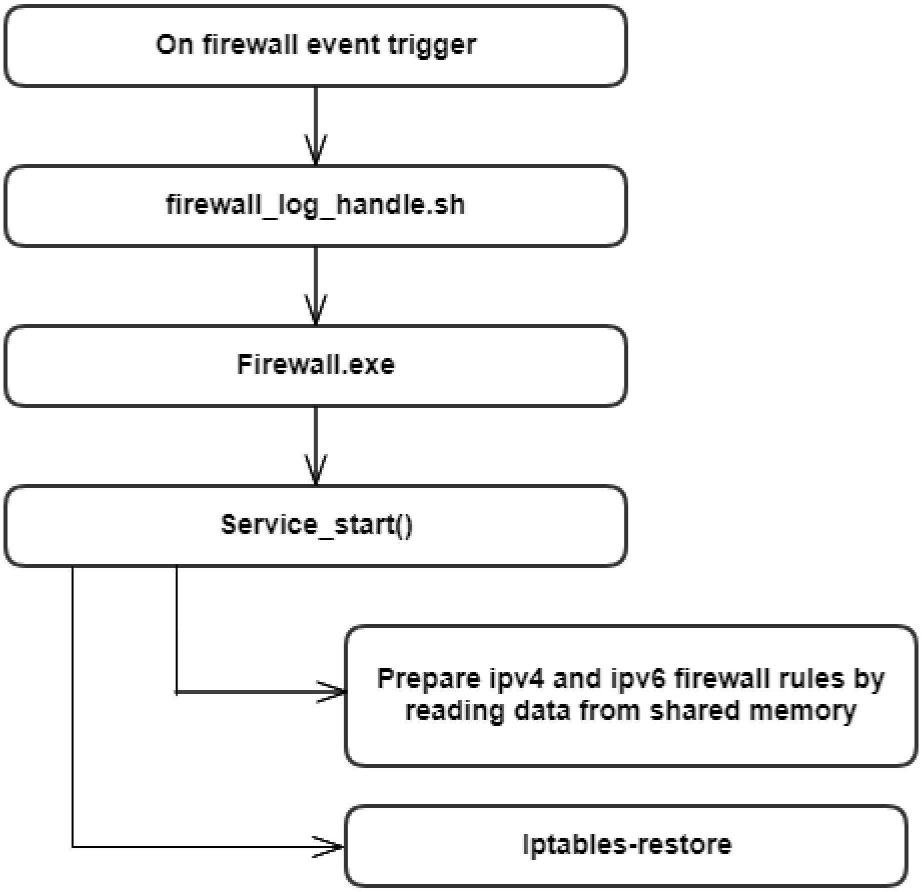
Note: The RDK data model naming convention prefix was changed in March 2020 to “X_RDK_”. We request you use the new prefix going forward.

Example Firewall Initialization

Gw_prov_utopia exe calls Init script. Init script executes all executables present in /etc/utopia/registration.d/ directory. 10_firewall exe is responsible for firewall events and it registers for sysevent callback with service name as firewall. Handler script is firewall_log_handle.sh. If any firewall event occurs sysevent is triggered with firewall-restart event name.



Firewall Initialization Process



On firewall-restart event service_start() method gets called. Ip4table and Ip6table rules are prepared by reading data from shared memory, written into /tmp/.ipt and /tmp/.ipt_v6 files respectively. Iptable rules are restored using these files.

Example Set flow

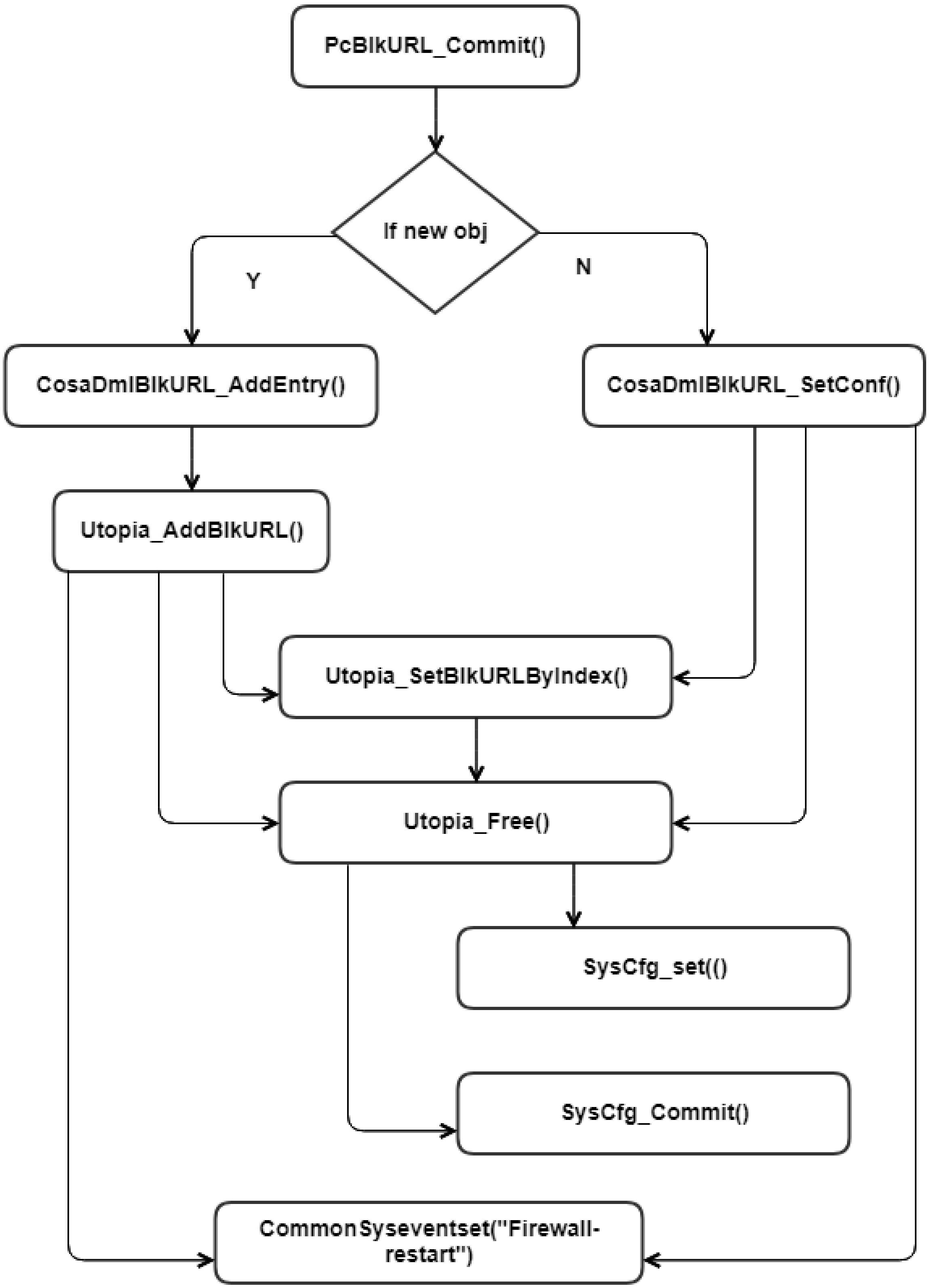
Following sequence explains flow when a SET from SNMP, TR69 or CLI is done:

- When commit needs to be done xml mapped API PCBlkURL_Commit() api gets called.
- If object is new CosaDmlBlkURL_AddEntry() is called, else CosaDmlBlkURL_SetConf() api is called.
- Utopia_SetBlkURLByIndex() prepares the syscfg data by appending proper index to be added to the cfg file.

Eg. Second row entry details are saved as shown below in xml.

pcms_2::method=URL
pcms_2::always=1
pcms_2::end_time=
ManagedSiteBlock_2=pcms_2
pcms_2::alias=cpe-BlockedURL-2
pcms_2::days=
pcms_2::site=https://www.wellsfargo.com
pcms_2::ins_num=2
pcms_2::start_time=

- syscfg_set() checks if the syscfg value exists, if not allocates memory and add it to the end of the linked list and sets the value.
- SysCfg_Commit() – calls syscfg_commit().
- syscfg_commit API updates the persistent memory with the shared memory details.



Example set flow for PC URL