UGW-7.1 - TCS TRAINING

08th Oct ,15

Connected Home Division

Intel Confidential

Agenda

- **Introduction**
- **UGW-7.1 Key features**
- **UGW-7.1 USPs**
- **UGW SW architecture**
- **FAPI concept**
- **Packaging and Simplified Documentation**
- **Debug tools and logging mechanism**
- **UGW-7.1 Customer use cases**
- **Q&A**

Introduction

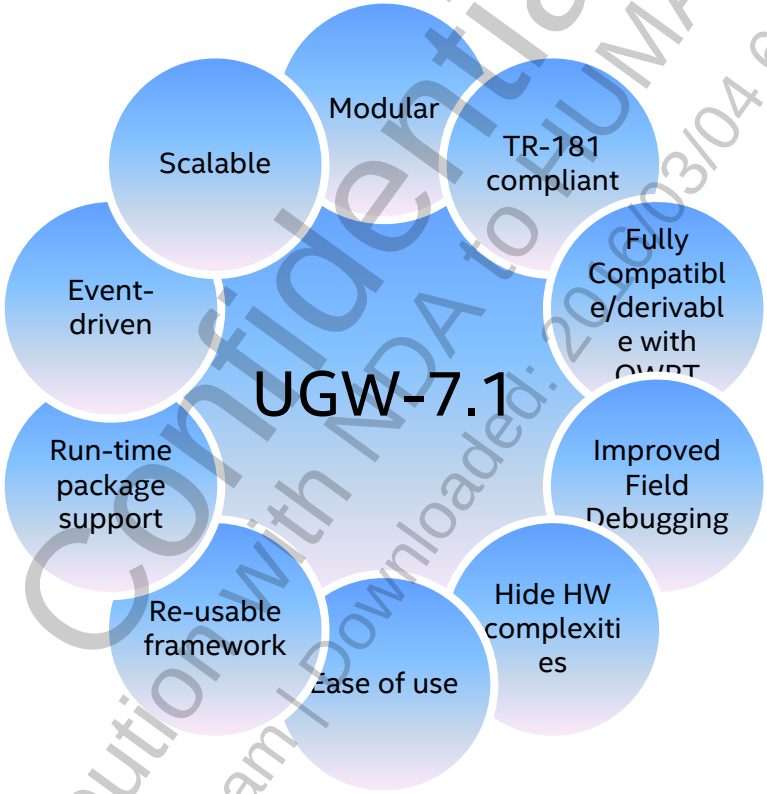
- Motivation for the training
  - Empower TCS with latest UGW SW offering – UGW-7.1

- Expectation
  - Resonate the USPs – help built Customer confidence
  - Feedback from the field
    - active TCS participation
    - helps improve UGW SW package

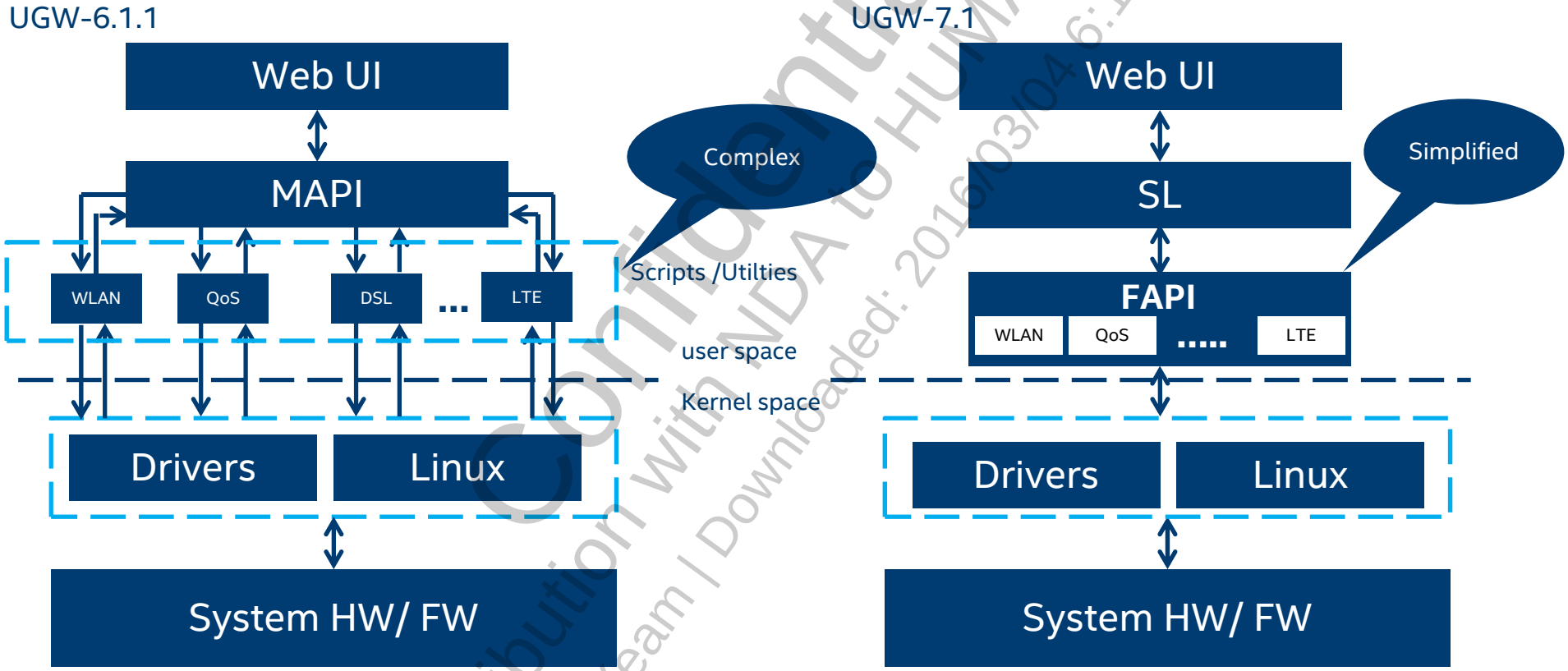
UGW-7.1 – Key features

- Functional API (FAPI) for faster development
- New Configuration Management Framework
- Web Framework supporting auto generation of web pages
  - UI customizations with less effort
- TR-181 Data Model support
  - Future proof
- Improved Debugging and Logging support
  - Faster debugging and development
- Pre-integrated QCA WLAN drivers, Intel LTE, Voice FW

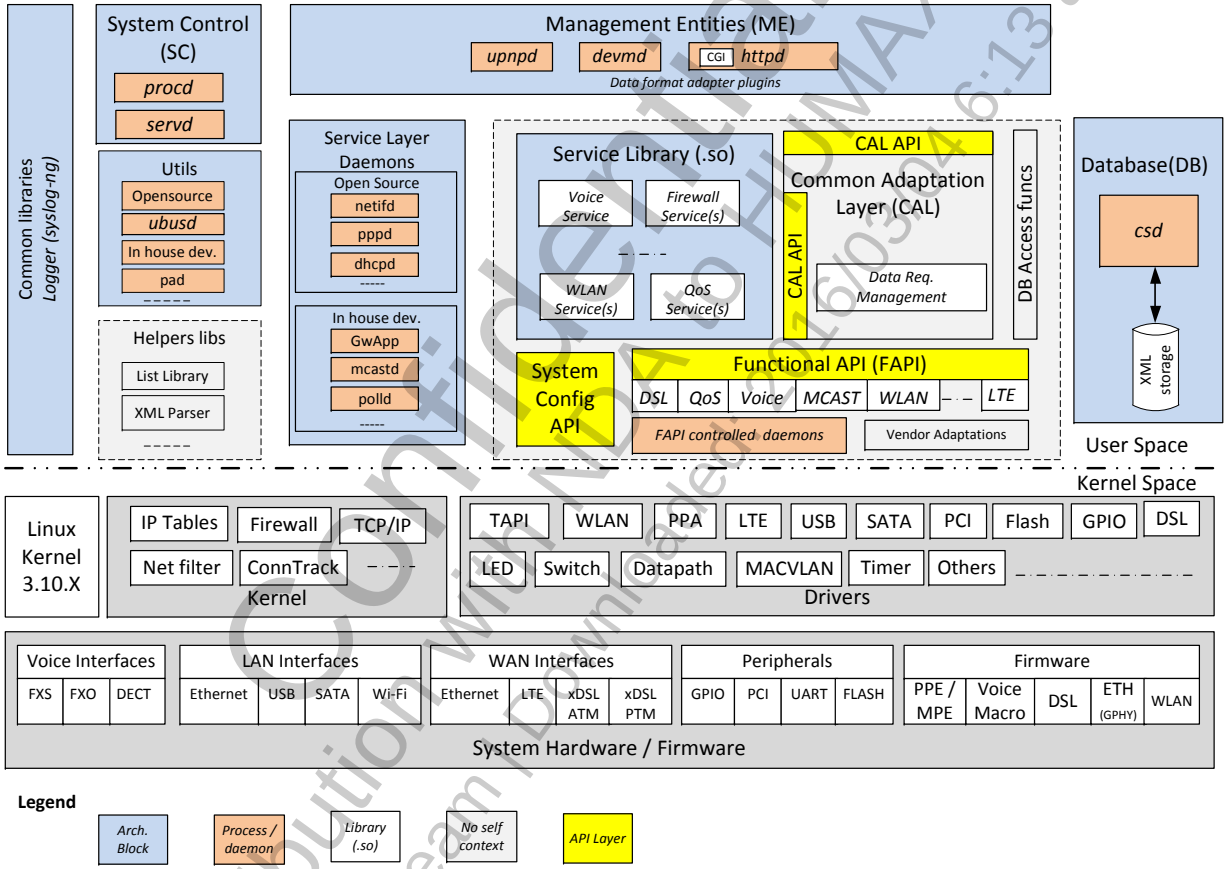
UGW-7.1 USPs



UGW SW architecture – comparison



UGW-7.1: High level architecture



Key changes - UGW-6.1.1 v/s UGW-7.1

Topic	UGW-6.1.1	UGW-7.1	Benefits
FAPI	✗	✓	• Hides complexity of the hardware
CLI utilities	~	✓	• Exclusive utilities to configure FAPI separately
TR-181	✗	✓	• TR-181 compatible (available in W7.2 release)
New web framework	Static web pages – ASP, HTML	Auto generated web pages – XML, AJAX, JSON, HTML5	• Easy to use and customize
Config Storage	Text based	XML based	• High readability • Easy extensible
Improved boot-up times (WiFi 2.4 Ping to CPE)	2.11 secs	1.23 secs	• 52.69% improvement
Scripts	>	<	• Easy to manage • Improved response time
OpenWRT Components	**	*****	• Reduces integration effort for openWRT model
Build system	Attitude Adjustment (12.09)	Chaos Calmer(15.05)	• Update to latest openWRT components, toolchain (available in W7.2 release)

FAPI

Introduction to FAPI
- Overview of FAPI
- Key benefits of FAPI

FAPI use cases
- Importance of FAPI with respect to some of the key features
- Comparison with Traditional approach vs. UGW-7.1

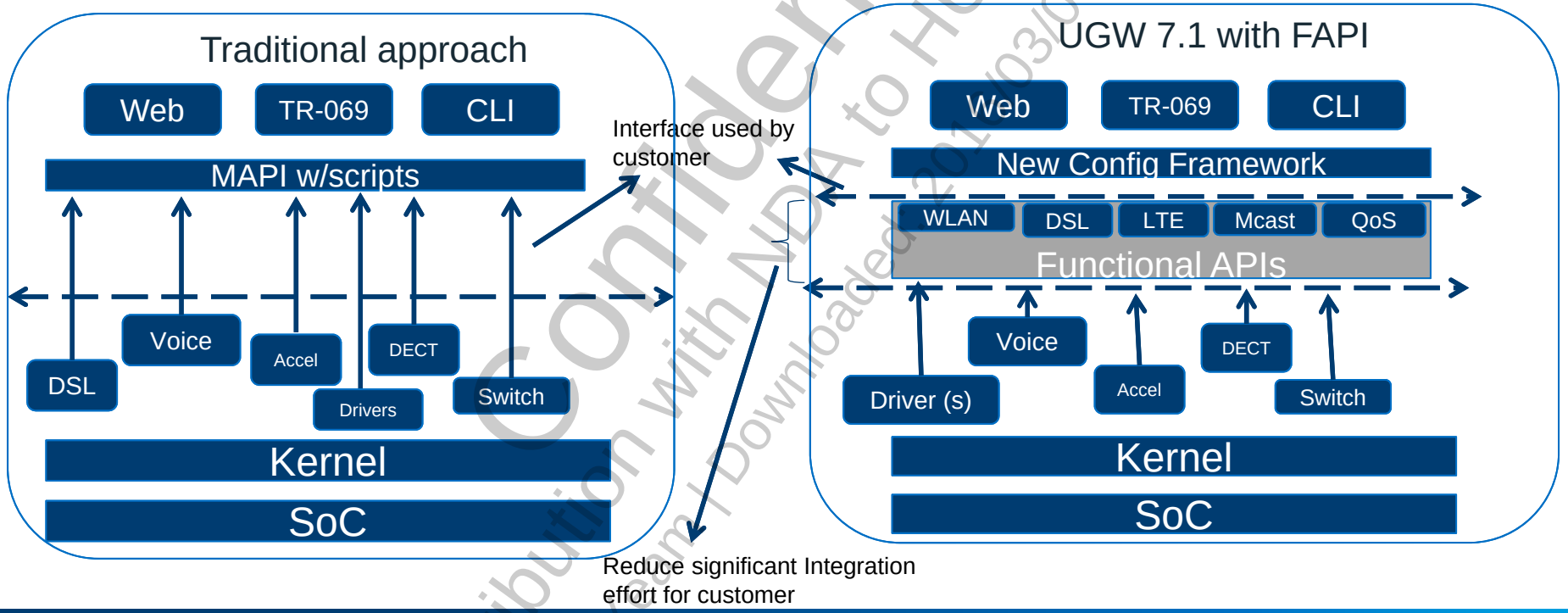
FAPI coverage
- Overview of key FAPIs and their corresponding utilities

Introduction to FAPI

- Overview of FAPI
  - Generic APIs, hides HW complexities
  - FAPI provides uniform interface across different SoC
- Key components
  - FAPI for initialization of System & Acceleration
  - Switch configuration FAPI
  - Multicast FAPI
  - QoS FAPI

Functional API Layer

- Higher Layer Functional API
  - Reduces the complexity for customer to plug-in their services



Acceleration sub-system

Action	Traditional approach (using scripts)	UGW-7.1 (using FAPI)	Overall benefits
Load modules	load_ppa_modules.sh	int32_t fapi_eth_init (void)	HW complexities hidden. Say goodbye to scripts!
Initialize	ppa.sh		Initialization of acceleration simplified
Add LAN/WLAN/WAN to PPA	$ ppacmd addlan –i <lan intf> $ ppacmd addwan –i <wan intf>	int32_t fapi_sys_if_attach (IN ifcfg_t *ifCfg)	Clean APIs to register/de-register with Acceleration system
Remove LAN/WLAN/WAN from PPA	$ ppacmd dellan –i <lan intf> $ ppacmd delwan –i <wan intf>	int32_t fapi_sys_if_detach (IN ifcfg_t *ifCfg)	

Acceleration sub-system – code comparison

Traditional approach

```
datapath_driver=ppa_datapath_${platform}_${module}.ko
hal_driver=ppa_hal_${platform}_${module}.ko
if [ -r /lib/modules/*/$datapath_driver ]; then
        if [ "$ppe_module" = "D5" -a "$target" = "GRX2" -a "$CONFIG_FEATURE_WWAN_LTE_SUPPORT" = "1" ]; then
                insmod /lib/modules/*/$datapath_driver ethwan=$phy_mode wanqos_en=0 wanitf=$wan_itf
        elif [ "$ppe_module" = "D5" -a "$target" = "GRX2" ]; then
                insmod /lib/modules/*/$datapath_driver ethwan=$phy_mode wanqos_en=0
        elif [ "$ppe_module" = "E5" -a "$CONFIG_FEATURE_DSL_BONDING_SUPPORT" = "1" ]; then
                insmod /lib/modules/*/$datapath_driver ethwan=$phy_mode wanqos_en=$wan_qos dsl_bonding=$bonding_mode
        else
                insmod /lib/modules/*/$datapath_driver ethwan=$phy_mode wanqos_en=$wan_qos wanitf=$wan_itf
        fi
        insmod /lib/modules/*/$hal_driver
    fi
fi

insmod /lib/modules/*/ltqmips_dtlk.ko
insmod /lib/modules/*/dlrx_fw.ko
insmod /lib/modules/*/ppa_api.ko
insmod /lib/modules/*/ppa_api_proc.ko
insmod /lib/modules/*/ppa_api_sw_accel_mod.ko
```

Based on static info

Hardcoded module names, not generic

UGW-7.1

System init - one API does the trick!

```
int32_t fapi_eth_init(void)
{
    sys_cfg_t sys_cfg;
    PPAInit_cfg_t PPA_Cfg;

    int32_t platform;
    int32_t ret = UGW_SUCCESS;

    memset(&PPA_Cfg, 0, sizeof(PPA_Cfg));

    switch (platform = get_hw_platform()) {
    case HW_PLATFORM_GRX350:
            ret = load_wan_modules(&sys_cfg, platform);
```

Dynamically determine HW

Initialize PPA

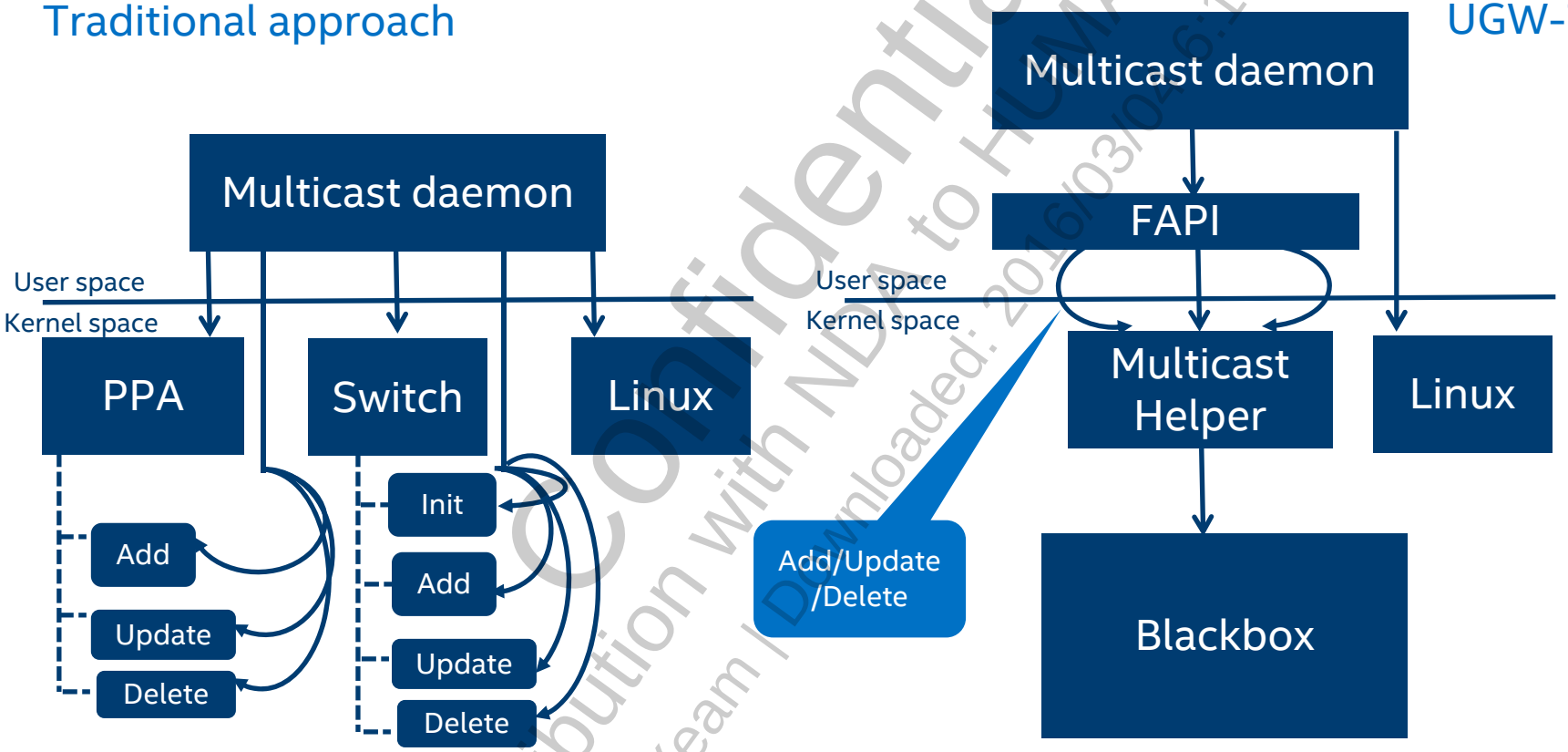
```
ret = fapi_sys_ppa_init(&PPA_Cfg);
```

Clean APIs to register/ de-register interfaces with PPA

```
int32_t fapi_sys_if_attach(IN ifcfg_t * ifCfg)
int32_t fapi_sys_if_detach(IN ifcfg_t * ifCfg)
```

Multicast

Traditional approach

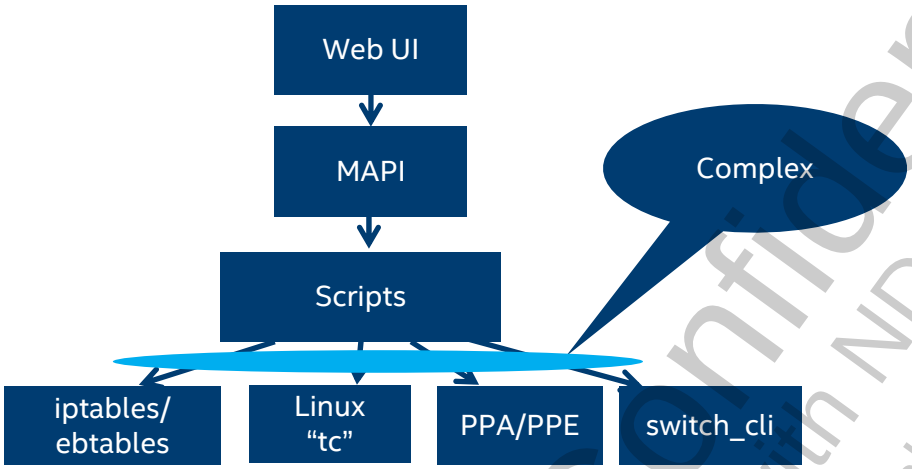


Multicast FAPI

Actions	
Add Group (Join)	int fapi_mch_add_entry (MCAST_MEMBER_t *mch_mem) This FAPI perform ADD i.e it informs MC-Helper module about new Join request received by mcastd . More...
Update Group	int fapi_mch_update_entry (MCAST_MEMBER_t *mch_mem) This FAPI perform Update i.e it informs MC-Helper module about Join request for already playing group is received by mcastd . More...
Remove Group (Leave)	int fapi_mch_del_entry (MCAST_MEMBER_t *mch_mem) This FAPI perform DELETE i.e it informs MC-Helper module about Leave request received by mcastd . More...
Initialize	int fapi_mch_init (void) This FAPI perform initilization i.e it insmod mcast_helper and create /dev/mcast device . More...
Un-initialize	int fapi_mch_uninit (void) This FAPI perform uninitilization i.e it rmmod mcast_helper . More...

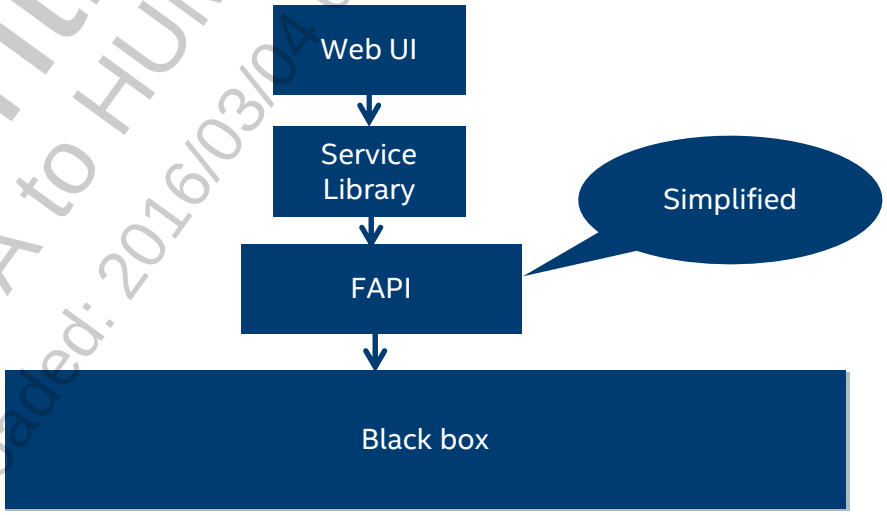
QoS FAPI – comparison

Traditional approach



- Manage multiple configuration objects - complex
- More time to adapt to New platforms
- Difficult for customers to integrate / customize
- Low readability
- Difficult to debug

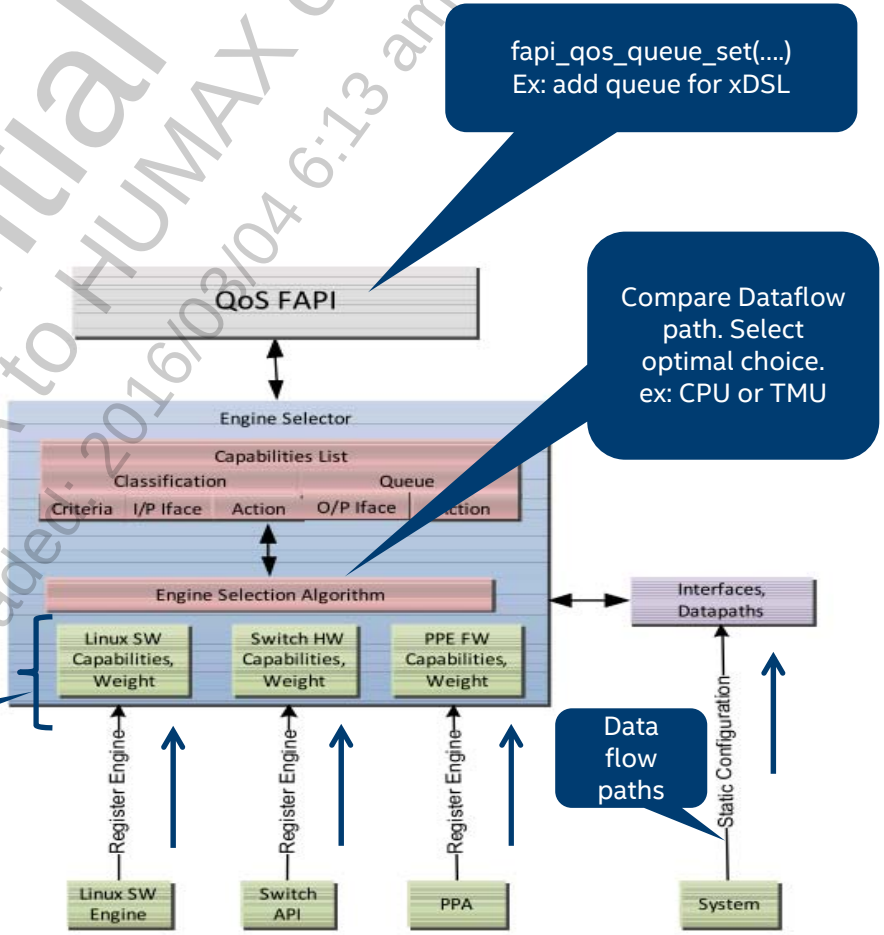
UGW-7.1



- Common APIs(FAPI) which abstracts HW/FW/SW
- Same FAPI across platforms
- Easy for customers to integrate / customize
- Better readability
- Easy to debug

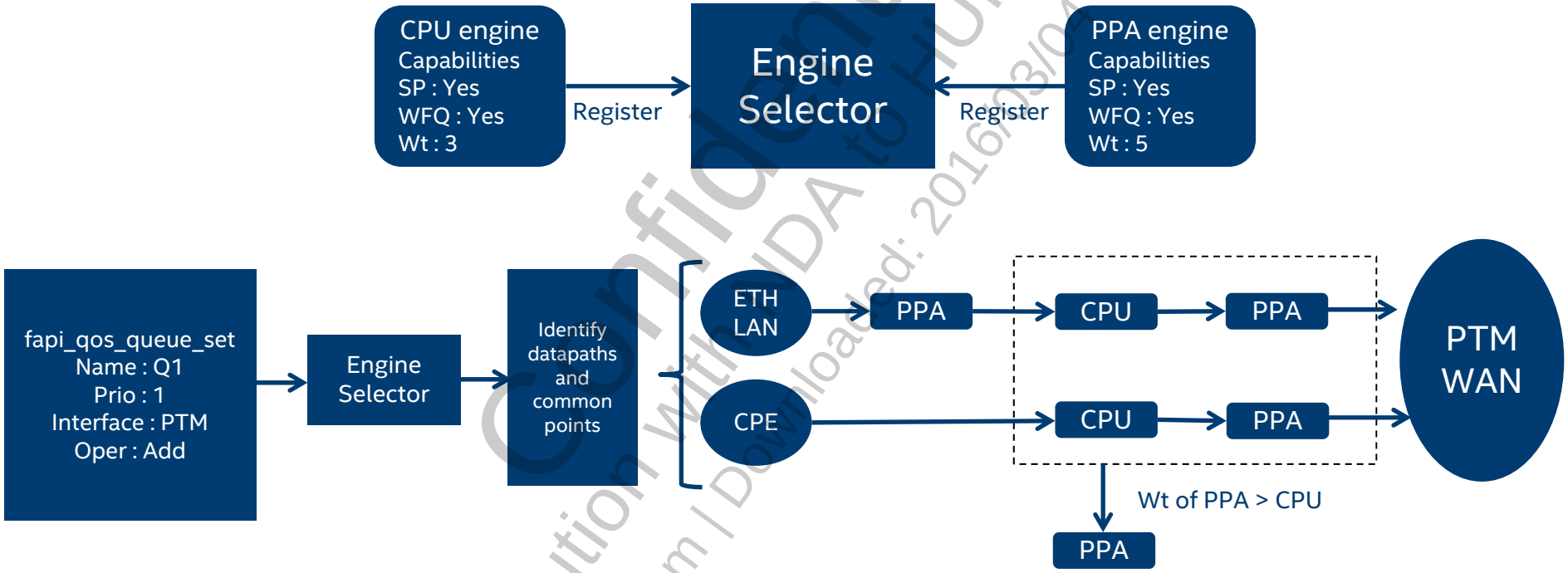
UGW-7.1 – QoS framework

- Modular design
- Centralized QoS config mechanism
- Abstracts HW



- Registration of QoS capabilities from each engine
- Weightage of each engines (ex: Switch < PPE FW < Linux)

Illustration of Queue add operation



QoS – code comparison

Traditional approach

```
#initialize eth wan queues
#Port Cfg to classify based on PCP or DSCP
switch_cli IFX_ETHSW_QOS_PORT_CFG_SET nPortId=6 eClassMode=3 nTrafficClass=0
switch_cli IFX_ETHSW_QOS_PORT_CFG_SET nPortId=5 eClassMode=3 nTrafficCla
#Downstream QoS Config Start
switch_cli dev=$CONFIG_SWITCH_DEVICE_ID IFX_ETHSW_QOS_PORT_CFG_SET nPortId=0 eClassMode=3 nTrafficClass=0
switch_cli dev=$CONFIG_SWITCH_DEVICE_ID IFX_ETHSW_QOS_PORT_CFG_SET nPortId=1 eClassMode=3 nTrafficClass=0
switch_cli dev=$CONFIG_SWITCH_DEVICE_ID IFX_ETHSW_QOS_PORT_CFG_SET nPortId=2 eClassMode=3 nTrafficClass=0
switch_cli dev=$CONFIG_SWITCH_DEVICE_ID IFX_ETHSW_QOS_PORT_CFG_SET nPortId=3 eClassMode=3 nTrafficClass=0
switch_cli dev=$CONFIG_SWITCH_DEVICE_ID IFX_ETHSW_QOS_PORT_CFG_SET nPortId=4 eClassMode=3 nTrafficClass=0
```

Switch QoS

```
if [ $QUEUE_TYPE -eq 0 ]; then
        ppacmd setwfq -p $PORT -q 0 -w 100
        ppacmd setwfq -p $PORT -q 1 -w 100
        ppacmd setwfq -p $PORT -q 2 -w 100
        ppacmd setwfq -p $PORT -q 3 -w 100
        ppacmd setwfq -p $PORT -q 4 -w 100
        ppacmd setwfq -p $PORT -q 5 -w 100
        ppacmd setwfq -p $PORT -q 6 -w 100
        ppacmd setwfq -p $PORT -q 7 -w 100
else
        ppacmd setwfq -p $PORT -q 0 -w 0
        ppacmd setwfq -p $PORT -q 1 -w 0
        ppacmd setwfq -p $PORT -q 2 -w 0
        ppacmd setwfq -p $PORT -q 3 -w 0
        ppacmd setwfq -p $PORT -q 4 -w 0
        ppacmd setwfq -p $PORT -q 5 -w 0
        ppacmd setwfq -p $PORT -q 6 -w 0
        ppacmd setwfq -p $PORT -q 7 -w 0
fi
```

PPE FW QoS

UGW-7.1

Clean APIs – complexities hidden inside the APIs

int32_t **fapi_qos_queue_set** (char *ifname, **qos_queue_cfg_t** *q, uint32_t flags)

Set QoS Queue Configuration - involves adding a new queue or modifying an existing queue, or deleting an existing queue. More...

int32_t **fapi_qos_queue_get** (char *ifname, char *queue_name, int32_t *num_queues, **qos_queue_cfg_t** **q, uint32_t flags)

Get one of all QoS Queue Configuration on given interface. More...

int32_t **fapi_qos_queue_stats_get** (char *ifname, char *queue_name, int32_t *num_queues, **qos_queue_stats_t** **qstats, uint32_t flags)

Get statistics of one or all Queues on given interface. More...

int32_t **fapi_qos_port_config_set** (char *ifname, **qos_shaper_t** *shaper, int32_t weight, int32_t priority, uint32_t flags)

Set Port QoS characteristics like rate shaper, and/or weight/priority if this port is cascaded into another scheduler. More...

int32_t **fapi_qos_port_config_get** (char *ifname, **qos_shaper_t** *shaper, int32_t *weight, int32_t *priority, uint32_t flags)

Get Port QoS characteristics like rate shaper, and/or weight/priority if this port is cascaded into another scheduler. More...

FAPI coverage

Module	Description	CLI Utilities(s)	FAPI
System	System initialization, LAN port connectivity, Acceleration sub-system initialization, Switch statistics, etc...	sys_cli	fapi_eth_init fapi_sys_set, fapi_sys_get fapi_sys_if_attach, fapi_sys_if_detach fapi_get_portid fapi_port_set_status, fapi_port_get_status fapi_port_getbitrate, fapi_rmon_get fapi_port_setDuplexMode
QoS	QoS configuration – add/delete queue, add/delete classifier, etc...	qoscfg, ifcfg, qcfg, classcfg	fapi_qos_init fapi_qos_if_abs_set fapi_qos_if_abs_get fapi_qos_if_base_set fapi_qos_queue_set fapi_qos_classifier_set fapi_qos_port_config_get fapi_qos_port_config_set
Multicast	Multicast group membership management – add groups to Acceleration, Switch, etc...	mcast_cli	fapi_mch_add_entry fapi_mch_update_entry fapi_mch_del_entry

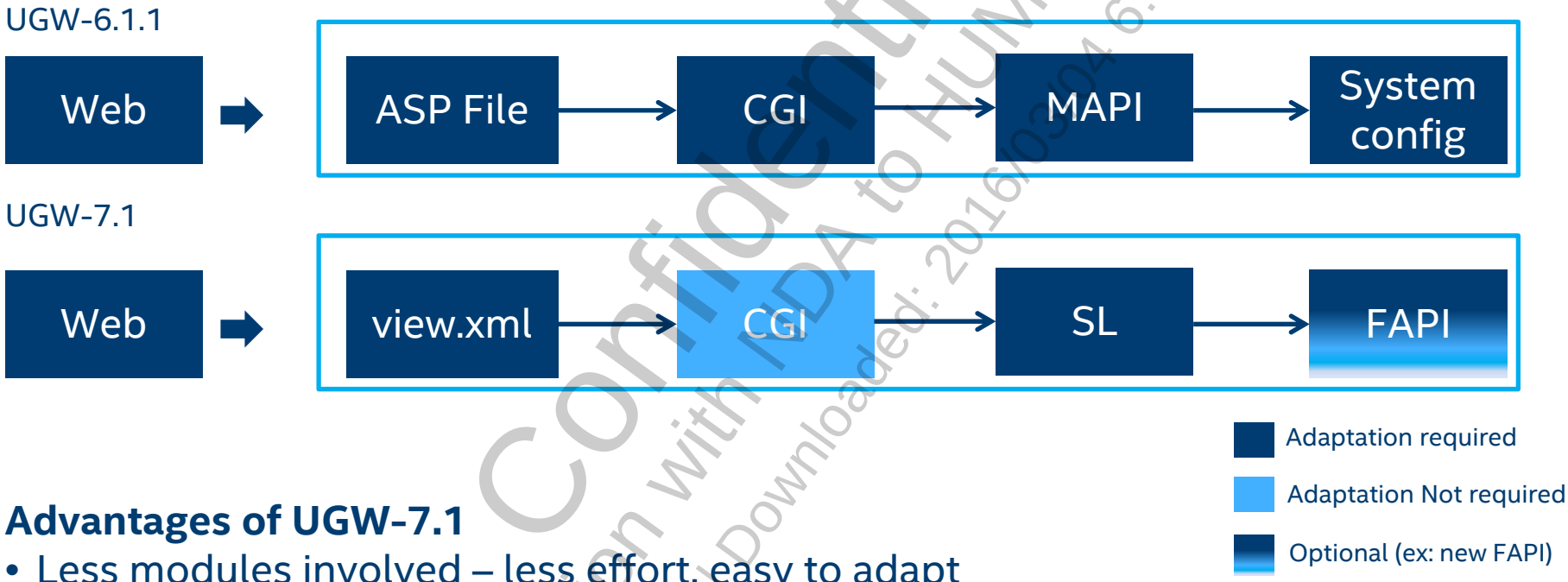
Contd.,

Module	Description	CLI Utilities(s)	FAPI
LTE	LTE module configuration and status check, etc...	wwan_cli	wwanfapi_do_init wwanfapi_do_connect wwanfapi_do_disconnect wwanfapi_get_interfacestatus wwanfapi_get_signalstrength wwanfapi_get_usimstatus wwanfapi_do_setpin wwanfapi_do_unlock wwanfapi_at_set
DSL	DSL initialization, status check, statistics, etc...	dsl_fapid	fapi_dsl_open fapi_dsl_init fapi_dsl_close fapi_dsl_uninit fapi_dsl_line_get fapi_dsl_line_set fapi_dsl_channel_stats_showtime_get fapi_dsl_channel_stats_showtime_set fapi_dsl_channel_stats_last_showtime_get fapi_dsl_channel_stats_last_showtime_set fapi_dsl_channel_stats_current_day_get fapi_dsl_channel_stats_current_day_set and more

Scalable and Modular

Ease of use	• Simplified approach to add/modify web UI • Auto-generated webpages without the need to hand-code for development purposes
Easy to customize	• Modular design • Ability to add/remove a package at run-time except for the base packages
DSL switch over optimizations	• UGW-7.1 includes improvements for DSL switchover time • Benefits seen during WAN switch over – ATM <-> PTM

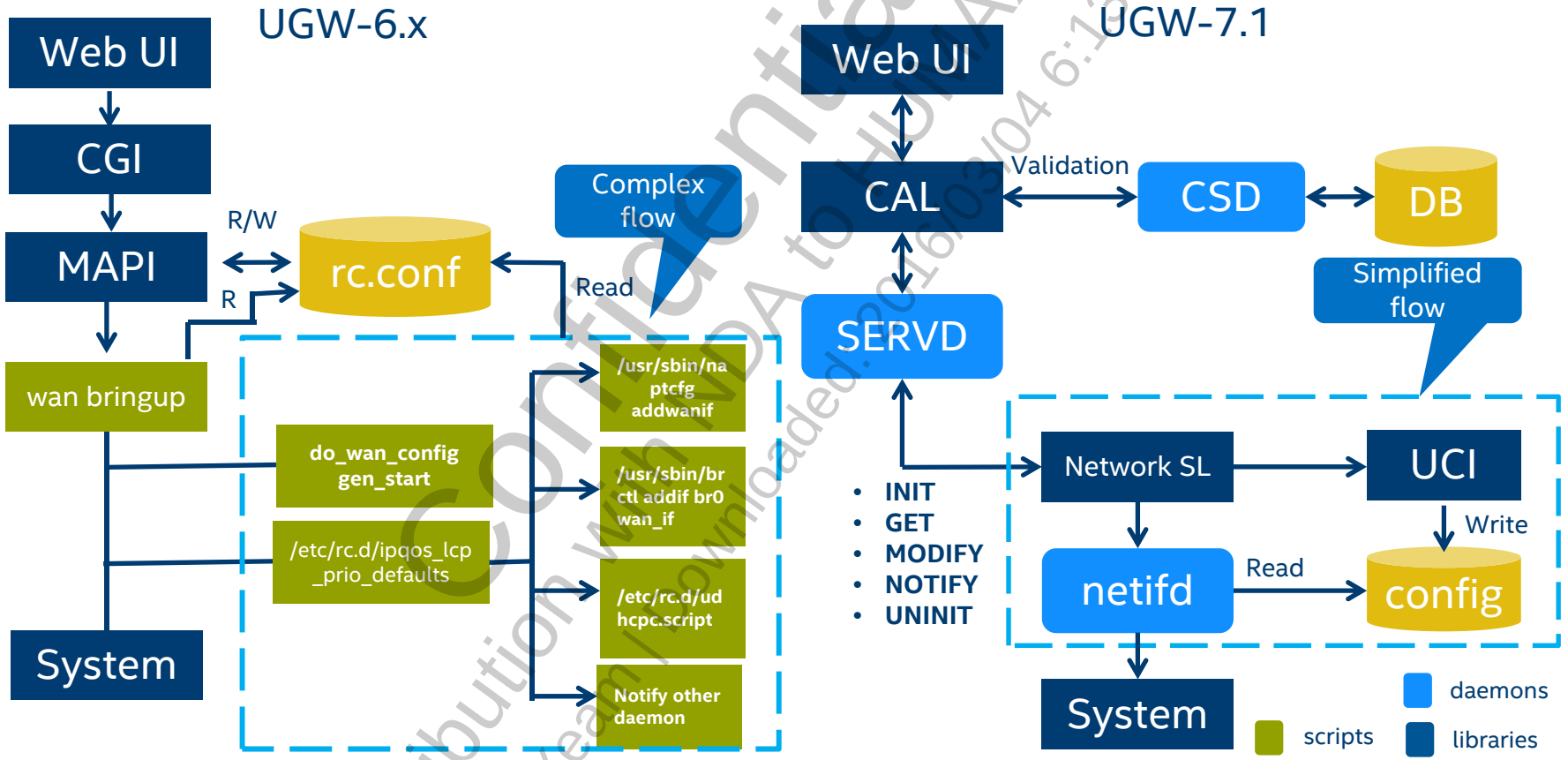
Ease of use - Illustration of adding a web page



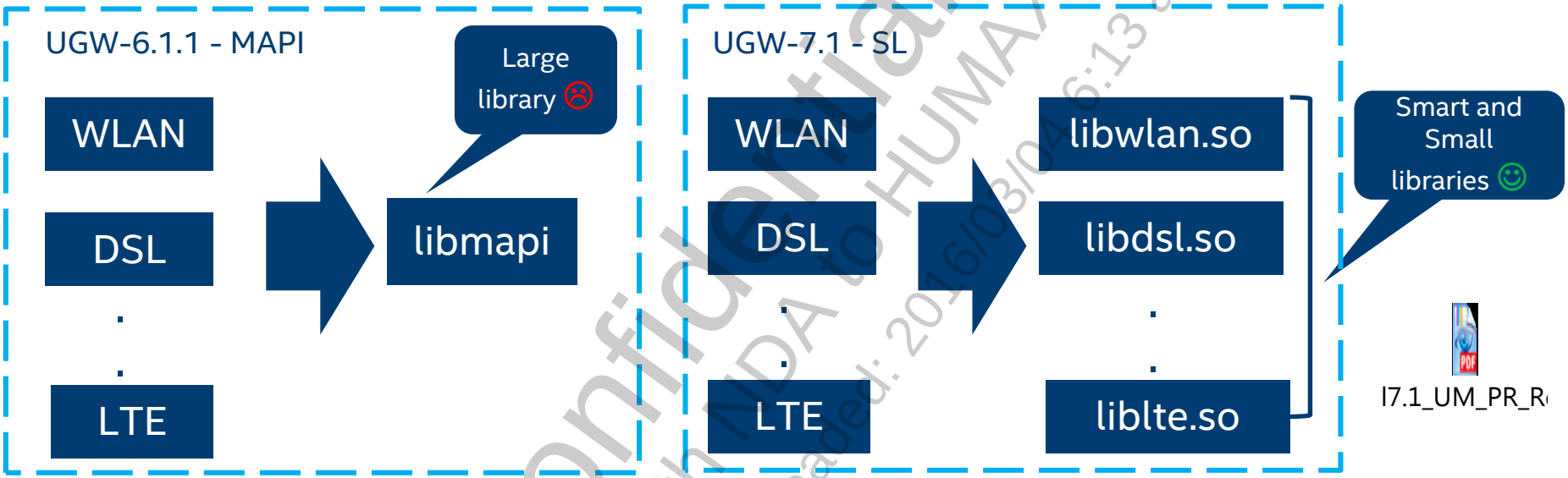
Advantages of UGW-7.1

- Less modules involved – less effort, easy to adapt
- Automated Web framework

WAN flow – comparison



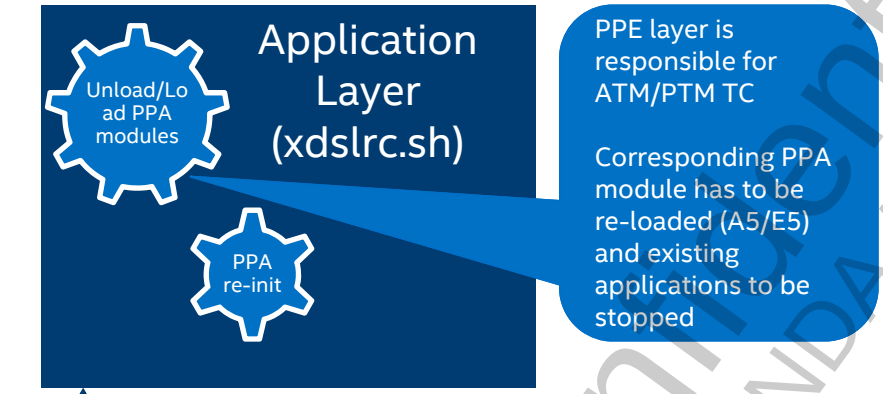
Easy to customize – Illustration of Service libraries



UGW-6.1.1	UGW-7.1
Monolithic library (libmapi.so) – not modular	Smart and Small libraries – modular
Customization – not so easy to add/remove features at compile time. Run time customization is ruled out.	Customization – easy to add/remove features at compile time/run time

xDSL switchover optimization

Traditional approach



UGW-7.1

Database access: UGW-6.1.1 GRX350 vs. UGW-7.1

- Fetching all Parameter Values from the database - most intensive operation with maximal access to system configuration & drivers

	UGW-6.1.1 on GRX350	UGW-7.1 on GRX350
Trail-1	61 Secs for 1638 Params	10.4 Secs for 6644 Params
Trail-2	44 secs for 1613 Params	17.7 Secs for 6394 Params
Trail-3	40 secs for 1641 Params	17.3 secs for 6290 Params

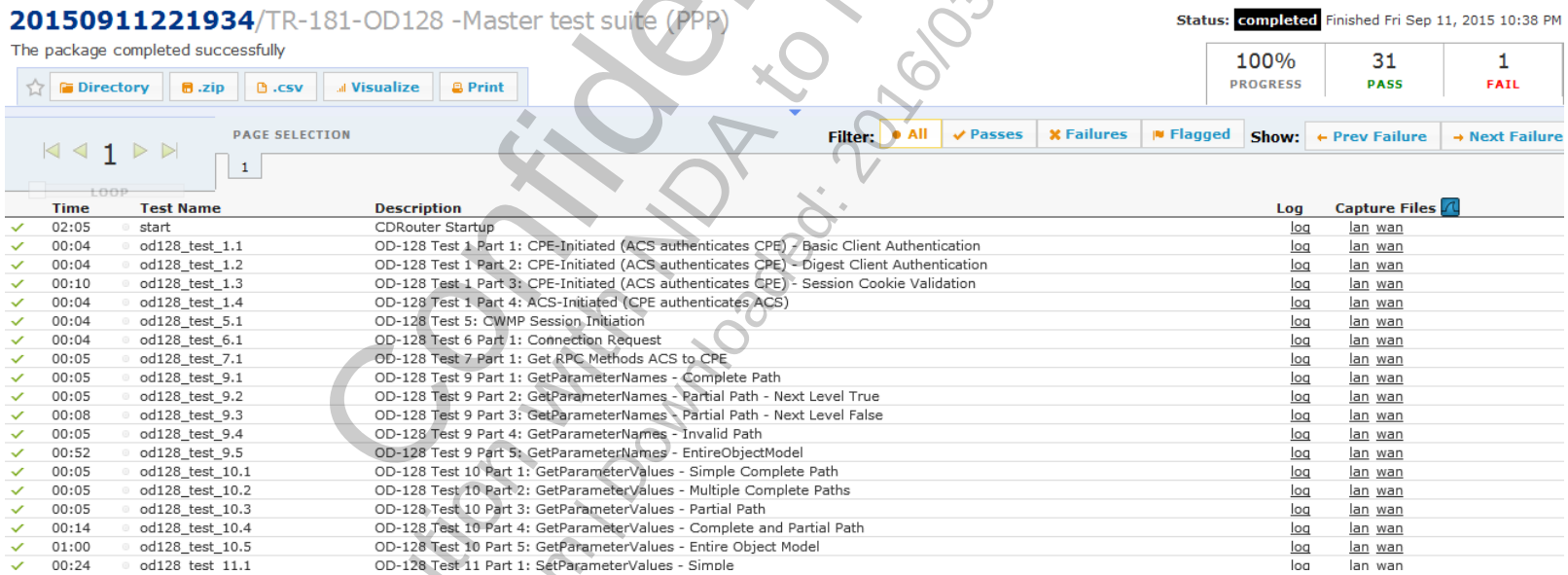
Huge improvement! 3x to 5x faster access

Web Improvements - Ease of Use, Faster Configuration times

Configuration	Steps		Pages		Time taken (Sec)	
	UGW6.1.1-350 – S7.9	UGW-7.1-W7	UGW6.1.1-350 – S7.9	UGW-7.1-W7	UGW6.1.1-350 – S7.9	UGW-7.1-W7
Add VLAN based DHCP ETHWAN connection	9	4	5 (WAN mode, VLAN(Summary, Add), WAN Setting(Summary, Add))	2(Summary, Add)	28.29 (5.59+2.32+20.38)	10
Delete a WAN connection	3	3	1	1	10	4.62
WLAN VAP Add	8	2	4 (Radio settings, SSID Config(Summary, Add), Security)	1	56.52 (21.89+20+14.63)	9.22
WAN Status Load (5 WAN Connections)	2	2	1	1	2.3	1.55

UGW-7.1: TR-181

TR-181 successfully tested on the CPE in W7 release



20150911221934/TR-181-OD128 -Master test suite (PPP)

The package completed successfully

Status: completed Finished Fri Sep 11, 2015 10:38 PM

Directory | .zip | .csv | Visualize | Print

100% PROGRESS	31 PASS	1 FAIL

PAGE SELECTION 1 1 LOOP

Filter: All | Passes | Failures | Flagged Show: Prev Failure | Next Failure

	Time	Test Name	Description	Log	Capture Files
✓	02:05	start	CDRouter Startup	log	lan wan
✓	00:04	od128_test_1.1	OD-128 Test 1 Part 1: CPE-Initiated (ACS authenticates CPE) - Basic Client Authentication	log	lan wan
✓	00:04	od128_test_1.2	OD-128 Test 1 Part 2: CPE-Initiated (ACS authenticates CPE) - Digest Client Authentication	log	lan wan
✓	00:10	od128_test_1.3	OD-128 Test 1 Part 3: CPE-Initiated (ACS authenticates CPE) - Session Cookie Validation	log	lan wan
✓	00:04	od128_test_1.4	OD-128 Test 1 Part 4: ACS-Initiated (CPE authenticates ACS)	log	lan wan
✓	00:04	od128_test_5.1	OD-128 Test 5: CWMP Session Initiation	log	lan wan
✓	00:04	od128_test_6.1	OD-128 Test 6 Part 1: Connection Request	log	lan wan
✓	00:05	od128_test_7.1	OD-128 Test 7 Part 1: Get RPC Methods ACS to CPE	log	lan wan
✓	00:05	od128_test_9.1	OD-128 Test 9 Part 1: GetParameterNames - Complete Path	log	lan wan
✓	00:05	od128_test_9.2	OD-128 Test 9 Part 2: GetParameterNames - Partial Path - Next Level True	log	lan wan
✓	00:08	od128_test_9.3	OD-128 Test 9 Part 3: GetParameterNames - Partial Path - Next Level False	log	lan wan
✓	00:05	od128_test_9.4	OD-128 Test 9 Part 4: GetParameterNames - Invalid Path	log	lan wan
✓	00:52	od128_test_9.5	OD-128 Test 9 Part 5: GetParameterNames - EntireObjectModel	log	lan wan
✓	00:05	od128_test_10.1	OD-128 Test 10 Part 1: GetParameterValues - Simple Complete Path	log	lan wan
✓	00:05	od128_test_10.2	OD-128 Test 10 Part 2: GetParameterValues - Multiple Complete Paths	log	lan wan
✓	00:05	od128_test_10.3	OD-128 Test 10 Part 3: GetParameterValues - Partial Path	log	lan wan
✓	00:14	od128_test_10.4	OD-128 Test 10 Part 4: GetParameterValues - Complete and Partial Path	log	lan wan
✓	01:00	od128_test_10.5	OD-128 Test 10 Part 5: GetParameterValues - Entire Object Model	log	lan wan
✓	00:24	od128_test_11.1	OD-128 Test 11 Part 1: SetParameterValues - Simple	log	lan wan

User-friendly

Structured Packaging

- Easy to extract/modify and generate a CD with Lantiq, 3rd party and open source components

Simplified documentation

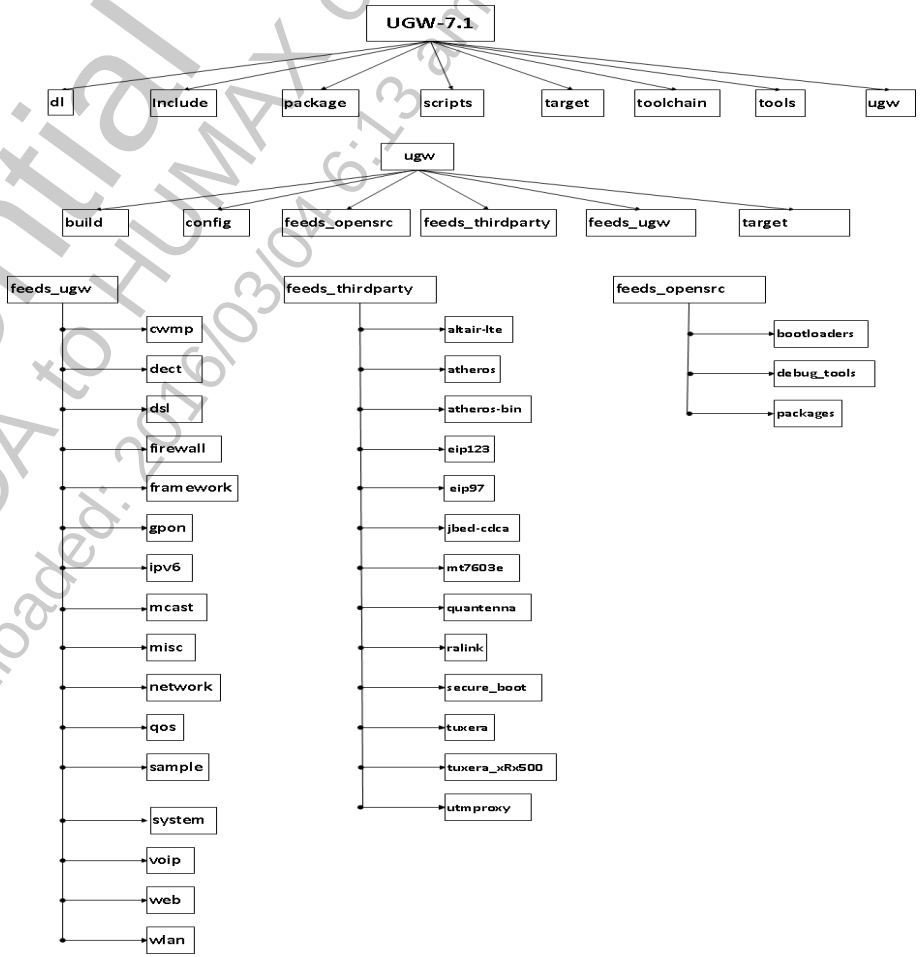
- API documentation is doxygen complaint and hence always in sync with the code
- Clear Usage description for commonly used scenarios
- Detailed usage examples for the APIs provided in Programmers reference
- How to add a new service - Right from Config till it is applied to system

Config Generator

- DBTool to automatically generate the system configuration file based on TR-181 spec as an input

UGW-7.1 New CD Layout

- Clear segregation of open source, Lantiq and Third-party components
- Changes are maintained in patches, makes it easy for customers to know what has changed
  - Simple script provided to view the fully patched kernel sources (diff between versions)
- We ensure that no GPLv3 components are packaged for any components going on target

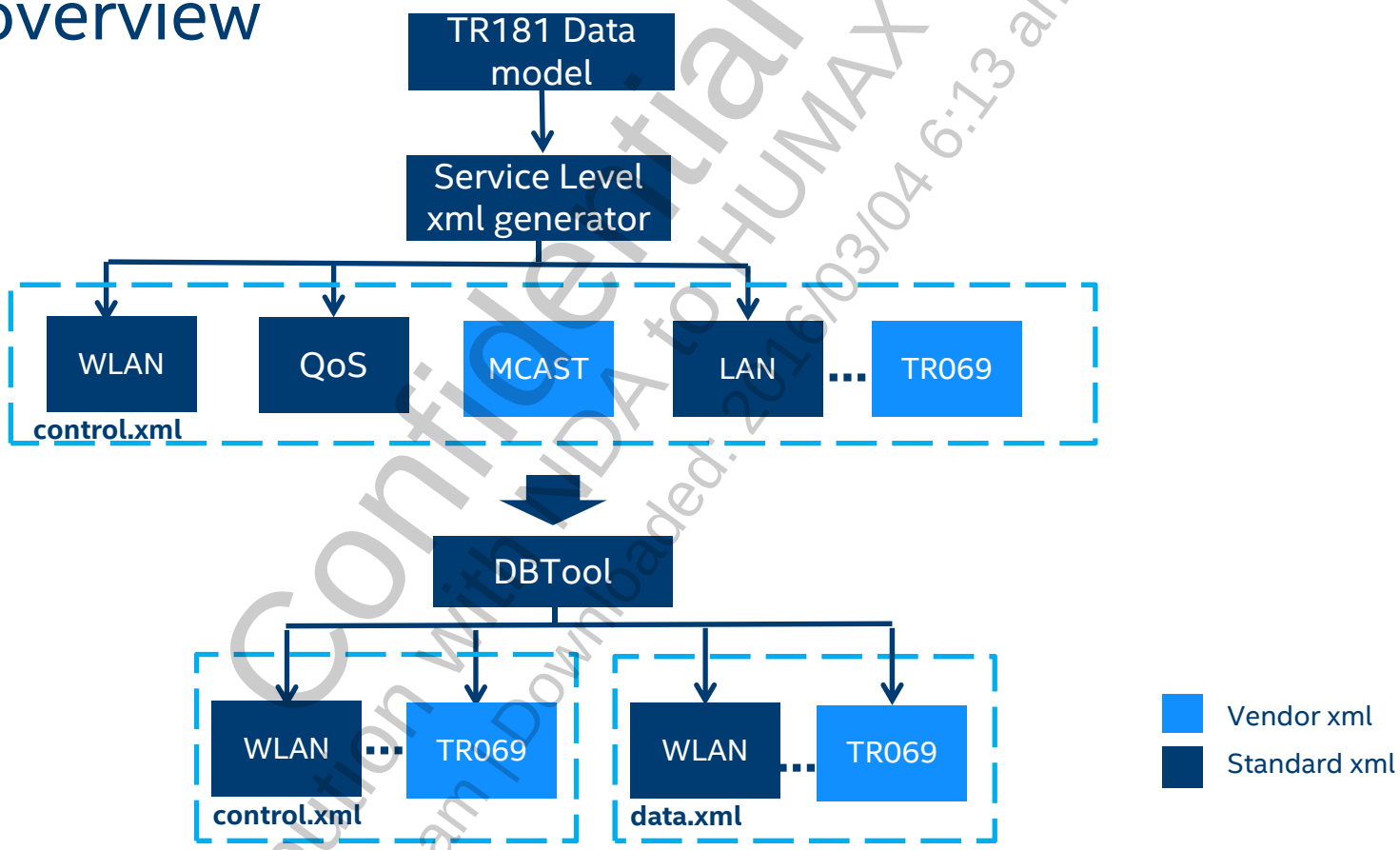


UGW-7.1 Documentation

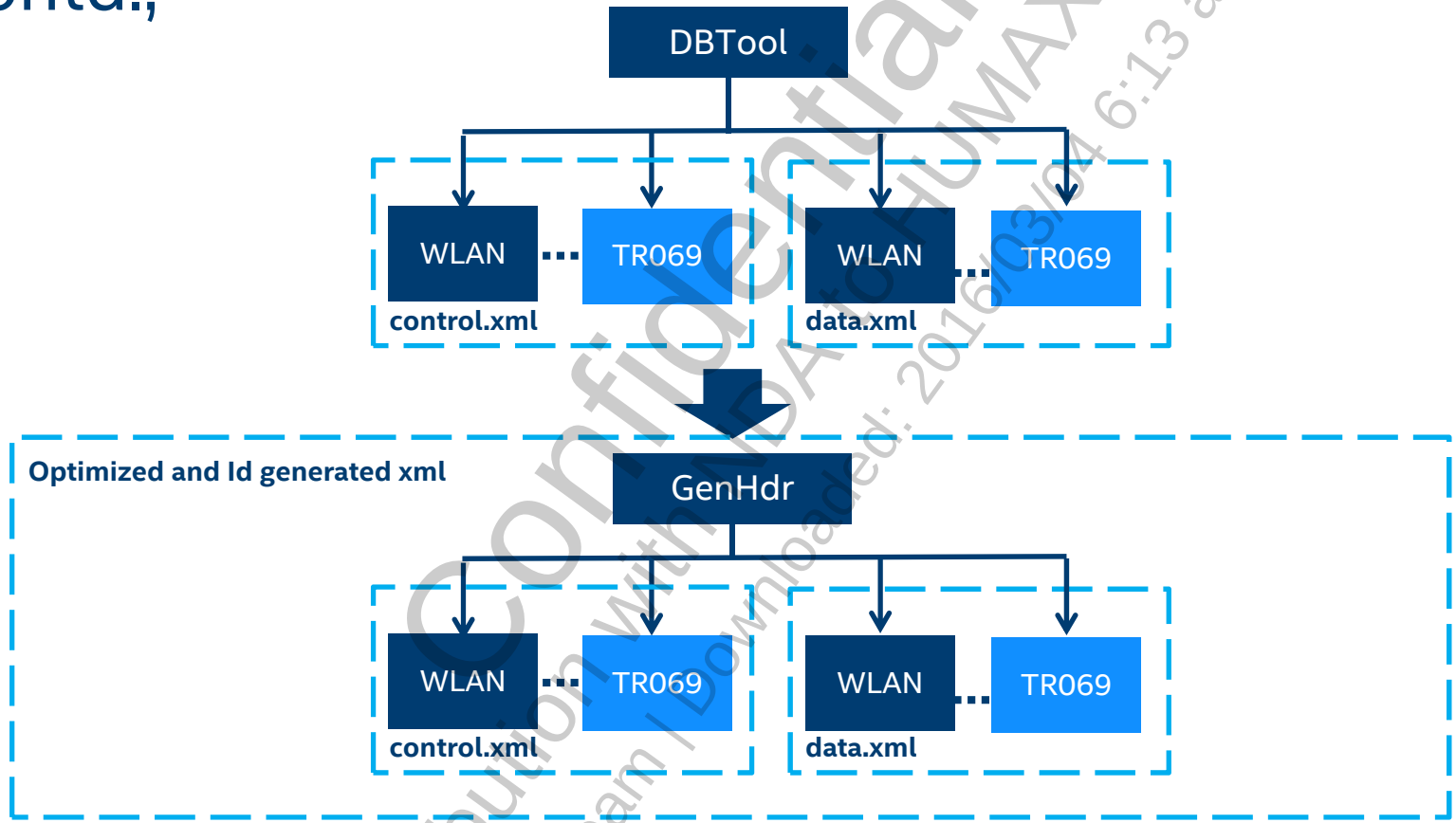
Improvements:

- Release Notes optimized from 83 to ~20 pages
- Software Overview (UMSO) optimized from 60 to 38 pages
- Programmer's Reference auto-generated (866 Vs 44 pages)
  - API definitions are Auto-generated (doxygen)
  - Use-case references provided in document

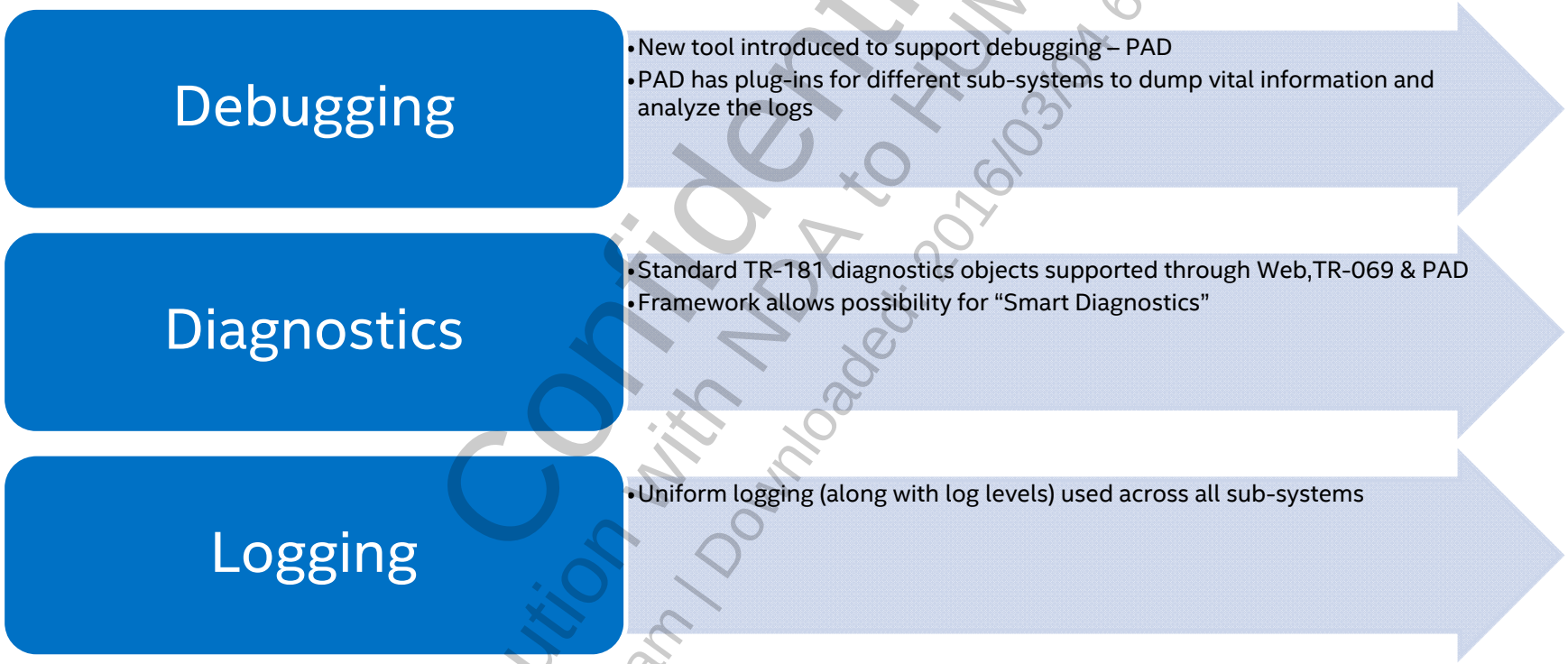
DBTool overview



Contd.,

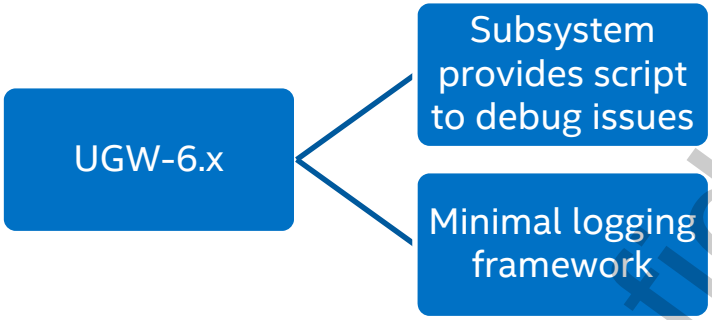


Better Field Debugging



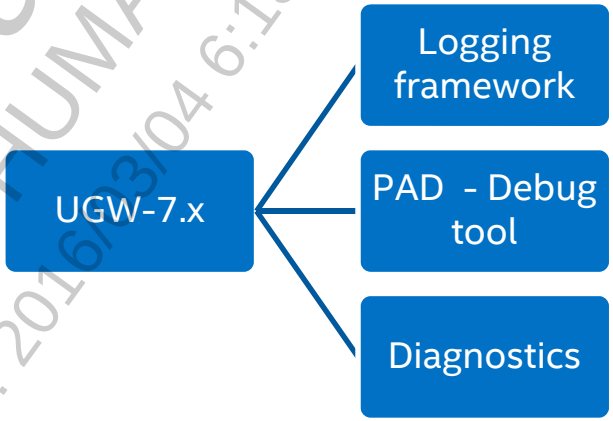
Interactive analysis and Debugging support

Traditional approach



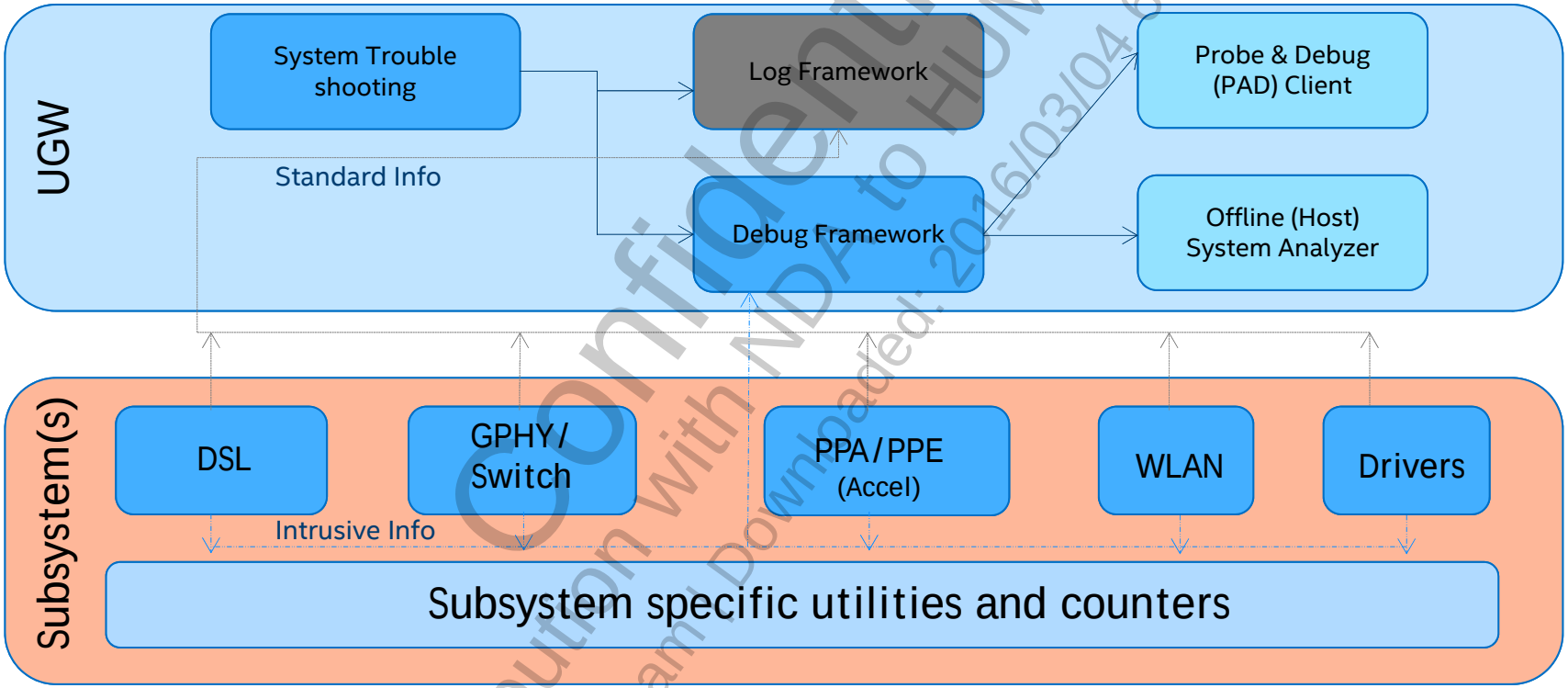
- Standalone commands/scripts to collect debug info
- Several iterations involved to collect required info
- No user friendly tools to help analyze the logs (sometimes huge log files)

UGW-7.1



- Improved debug support with new tools – ex: PAD
- PAD collects entire system debug info in one go
- Reduced email traffic between Customer and TCS/Experts
- In house developed System analyzer tool to analyze the logs

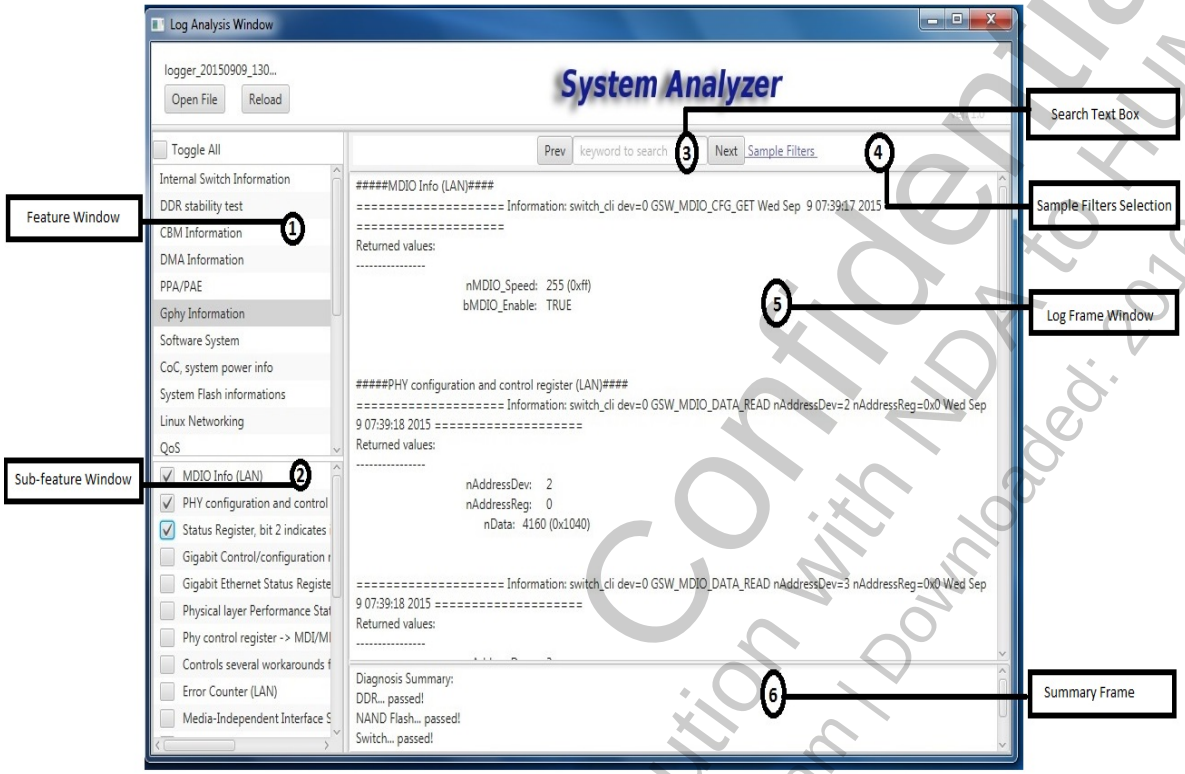
Introduction : Troubleshooting



PAD Improvements (on top of UGW-6.1.1-GRX350)

- Integrated missing sub systems plugin files (System, Multicast, DSL)
- Platform abstraction within Plugin files
  - plugin file and command level support
- Data flows incorporated for different interfaces (WANoE, xDSL, WLAN)
- Basic diagnostics implemented
  - ex: DSL, LAN and WLAN connectivity/status
- Support for Logging during System endurance
- Analysis of past HDT issues
  - 29 tickets analyzed (65% can be solved by logs provided by PAD)

System analyzer : Tool overview



- Feature and Sub-Feature windows.
- Search and Filter options
- Periodic check for updated logger_XXXXXX.zip files
- Multiple debug window support
- Support to invoke other applications like wireshark, teraterm etc,.,

Port mirroring

- "sys_cli" – configure Port mirroring
- Command: $ sys_cli --help
  - -M configure Port Mirror
  - provide US interface, DS interface, Mirror interface (one of LAN ports)
  - ex: enable port mirror : ./sys_cli -M eth1 eth0_1 eth0_2 1
  - ex: disable port mirror : ./sys_cli -M eth1 eth0_1 eth0_2 0

Logging Framework

- **Changing Service Libraries Log level and Type at run time**
  - "slloglevel" command to change the log level and log type of SL or servd at run time.
  - Usage
    - **Set log level and type :**
      slloglevel <sl name or servd or all> <level> <type>
    - **Print log level and type:**
      slloglevel print <sl name or servd or all>
- **Changing CSD Log level and Type at run time**
  - ubus call csd setlog '{ "loglevel" : 1- CRITICAL | 2 - ERROR | 3 - INFO | 4 - DEBUG , "logtype": 1 - File | 2 - Console | 3 - Both }'

Logging Framework: UGW-6.1.1 vs. UGW-7.1

```
Sep 29 19:29:04 syslog:notice - syslog: do_session_mgmt returned fail for LAN
Sep 29 19:29:04 syslog:notice - syslog: do_session_mgmt returned fail for WAN
Sep 29 19:29:11 syslog:warning - syslog: Timeout waiting for PADO packets
Sep 29 19:29:34 syslog:notice - syslog: do_session_mgmt returned fail for LAN
Sep 29 19:29:34 syslog:notice - syslog: do_session_mgmt returned fail for WAN
Sep 29 19:29:36 syslog:warning - syslog: Timeout waiting for PADO packets
Sep 29 19:29:50 syslog:debug - syslog: PADS: Service-Name: \'\'
Sep 29 19:29:50 syslog:info - syslog: PPP session is 9
Sep 29 19:29:55 syslog:debug - syslog: [mapi_delete_old_virtualserver:2254:0]
Sep 29 19:29:55 syslog:debug - syslog: [mapi_set_vlan_ch_entry:3484] VLAN channel modified [1]
Sep 29 19:29:57 syslog:debug - syslog: Event Notification Invoked.
Sep 29 19:29:57 syslog:debug - syslog: Event Notify Successful
Sep 29 19:30:00 syslog:debug - syslog: Event Notification Invoked.
Sep 29 19:30:00 syslog:debug - syslog: Event Notify Successful
Sep 29 19:30:04 syslog:notice - syslog: do_session_mgmt returned fail for LAN
Sep 29 19:30:04 syslog:notice - syslog: do_session_mgmt returned fail for WAN
Oct  7 11:41:44 syslog:debug - syslog: Event Notification Invoked.
Oct  7 11:41:44 syslog:debug - syslog: Event Notify Successful
Sep 29 19:28:50 syslog:debug - syslog: [IFX_GetTr69IdFromCpeId:814] [-27] Unable to get TR69 Id for [Xram_1] from tr69_map
Sep 29 19:28:50 syslog:debug - syslog: [IFX_SendNotify:1336] [-27] Error in getting TR69 ID
Sep 29 19:28:50 syslog:debug - syslog: Notification protocol api failed
Sep 29 19:28:53 syslog:debug - syslog: Event Notification Invoked.
Sep 29 19:28:53 syslog:debug - syslog: Event Notify Successful
Sep 29 19:28:32 syslog:debug - syslog: Notification protocol api failed
Sep 29 19:28:32 syslog:debug - syslog: [IFX_GetTr69IdFromCpeId:814] [-27] Unable to get TR69 Id for [Rfpi_1] from tr69_map
Sep 29 19:28:32 syslog:debug - syslog: [IFX_SendNotify:1336] [-27] Error in getting TR69 ID
Sep 29 19:28:32 syslog:debug - syslog: Notification protocol api failed
Sep 29 19:28:34 syslog:notice - syslog: do_session_mgmt returned fail for LAN
Sep 29 19:28:34 syslog:notice - syslog: do_session_mgmt returned fail for WAN
```

- Quite difficult to identify call flow of a given process/daemon/library
- Log level control only at Process/Daemon level

```
Oct  7 06:42:20 servd:debug - servd: libcal{cal_setValue, 207}:SET Request from Owner - 4
Oct  7 06:42:20 servd:debug - servd: libcal{cal_setValue, 276}:CAPI_SET called to SET to DB...
Oct  7 06:42:20 servd:debug - servd: libcal{cal_setValue, 283}:CAL_SET Return : 0
Oct  7 12:12:21 syslog:debug - syslog: libcal{cal_getValue, 101}:Dynamic Flag is set on Object Device.SelfTestDiagnostics
Oct  7 12:12:21 syslog:info - syslog: libcal{cal_getValue, 110}:Dynamic List is Created. Sending to ServD..
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_diagnostics_handler, 344}:Enter Handler INIT function: Opcode = 1
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_diagnostics_get, 191}:Entry...!!
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_diagnostics_runSelfTestDiagnostics, 1004}:Entry...!!
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_diagnostics_checkLanHostReachability, 440}:Entry...!!
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_diagnostics_discoverDefaultGatewayAndDns, 984}:Entry...!!
Oct  7 06:42:21 servd:debug - servd: libcal{cal_getValue, 126}:CAL GET Return : 0
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_getDefaultWanObjName, 62}:Default GW Wan interface is Device.IP.Interface.5.
Oct  7 06:42:21 servd:debug - servd: libcal{cal_getValue, 126}:CAL GET Return : 0
Oct  7 06:42:21 servd:debug - servd: libdiagnostics{sl_diagnostics_checkDnsReachability, 389}:Entry...!!
Oct  7 06:42:24 servd:debug - servd: libdiagnostics{sl_diagnostics_checkGatewayReachability, 413}:Entry...!!
Oct  7 06:42:24 servd:debug - servd: libdiagnostics{sl_diagnostics_checkGoogleReachability, 492}:Entry...!!
Oct  7 06:42:24 servd:debug - servd: libdiagnostics{sl_diagnostics_DnsLookUp, 861}:Entry...!!
Oct  7 06:42:25 csd:err - csd: csd{cfgapi_12GetLeafNode, 1019}: Invalid Objname [Device.DNS.Diagnostics.NSLookupDiagnostics.Result.]
Oct  7 06:42:25 servd:err - servd: libcal{cal_getValue, 65}:CAPI GET Failed. Return Value : -200
Oct  7 06:42:25 servd:debug - servd: libcal{cal_getValue, 126}:CAL GET Return : -200
Oct  7 06:42:25 servd:err - servd: SERVD{servd_calGetValueDB, 267}:CAL Get Failed
Oct  7 06:42:25 servd:err - servd: libdiagnostics{sl_diagnostics_DnsLookUp, 899}:Get
Device.DNS.Diagnostics.NSLookupDiagnostics.Result. Failed with Return code -200
Oct  7 06:42:25 servd:debug - servd: libcal{cal_getValue, 126}:CAL GET Return : 0
Oct  7 06:42:25 servd:debug - servd: libdiagnostics{sl_diagnostics_updateReachabilityResults, 529}:Entry...!!
Oct  7 06:42:25 servd:debug - servd: libcal{cal_getValue, 126}:CAL GET Return : 0
Oct  7 06:42:25 servd:debug - servd: libdiagnostics{sl_diagnostics_handler, 361}:Exit Handler INIT function: Opcode = 1
Oct  7 12:12:25 syslog:debug - syslog: libcal{cal_getValueCb, 303}:GET Response Received from Servd
Oct  7 12:12:25 syslog:debug - syslog: libcal{cal_getValue, 126}:CAL GET Return : 0
Oct  7 06:42:26 servd:debug - servd: liblanservices{sl_lanservices_handler, 41}:LAN SERV >> HANDLER function: ops = 16
Oct  7 06:42:26 servd:debug - servd: liblanservices{sl_lanservices_notify, 105}:LAN >> NOTIFY function : Id = 31
Oct  7 06:42:26 servd:info - servd: liblanservices{dhcp_notify, 293}:Received Notify Id - 31
Oct  7 06:42:26 servd:debug - servd: libcal{cal_getValue, 126}:CAL GET Return : 0
```

- Easy to identify the call flows of daemon/SL/process
- Possible to control log level for Service Libraries and Process/Daemon levels

Fully Compatible with OpenWRT

Customer use cases

- Illustration of different Customer use cases to help migrate to UGW-7.1
- Availability of sample openWRT model with FAPI

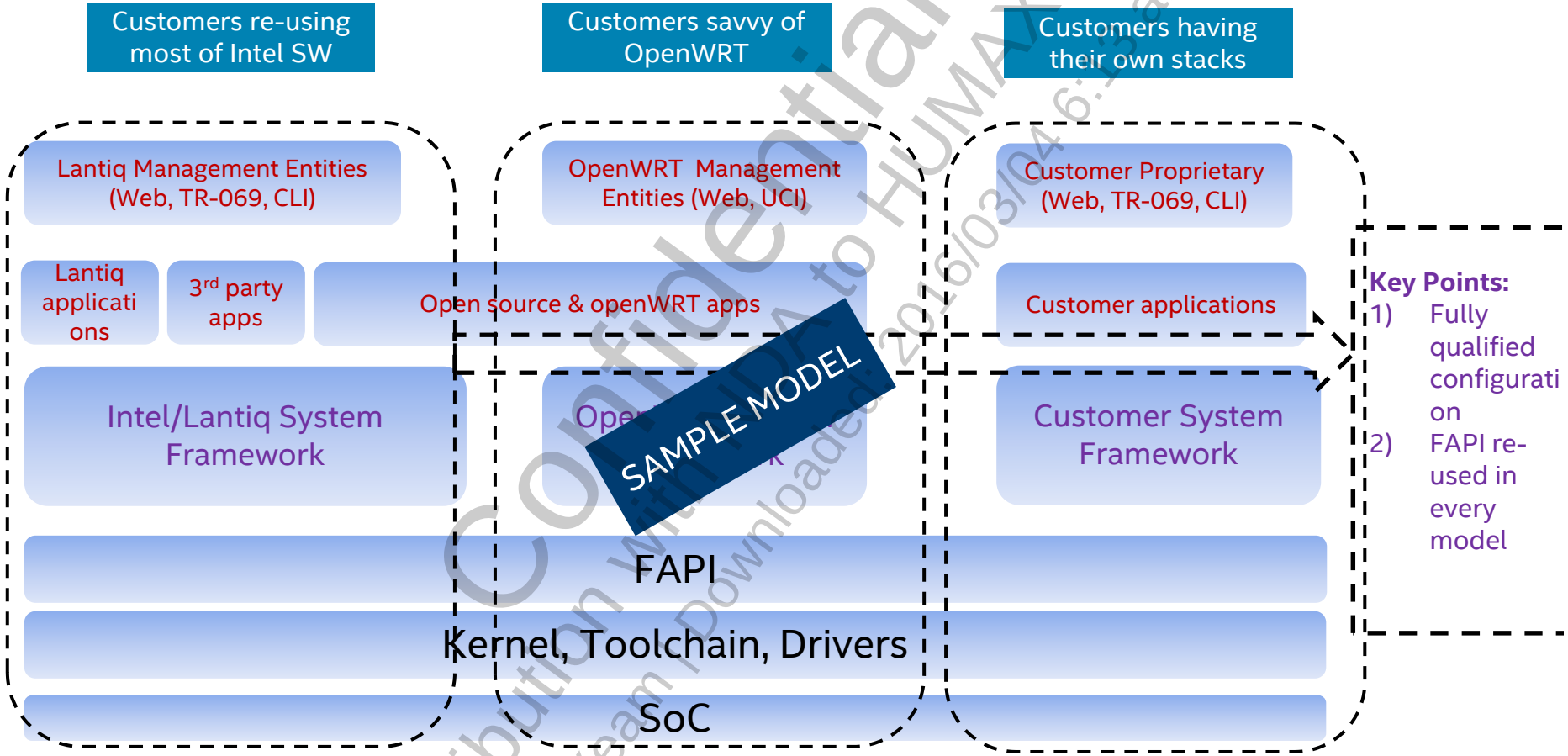
Release mapping

- Mapping information of UGW-6.1.1 GRX350 → UGW-7.1

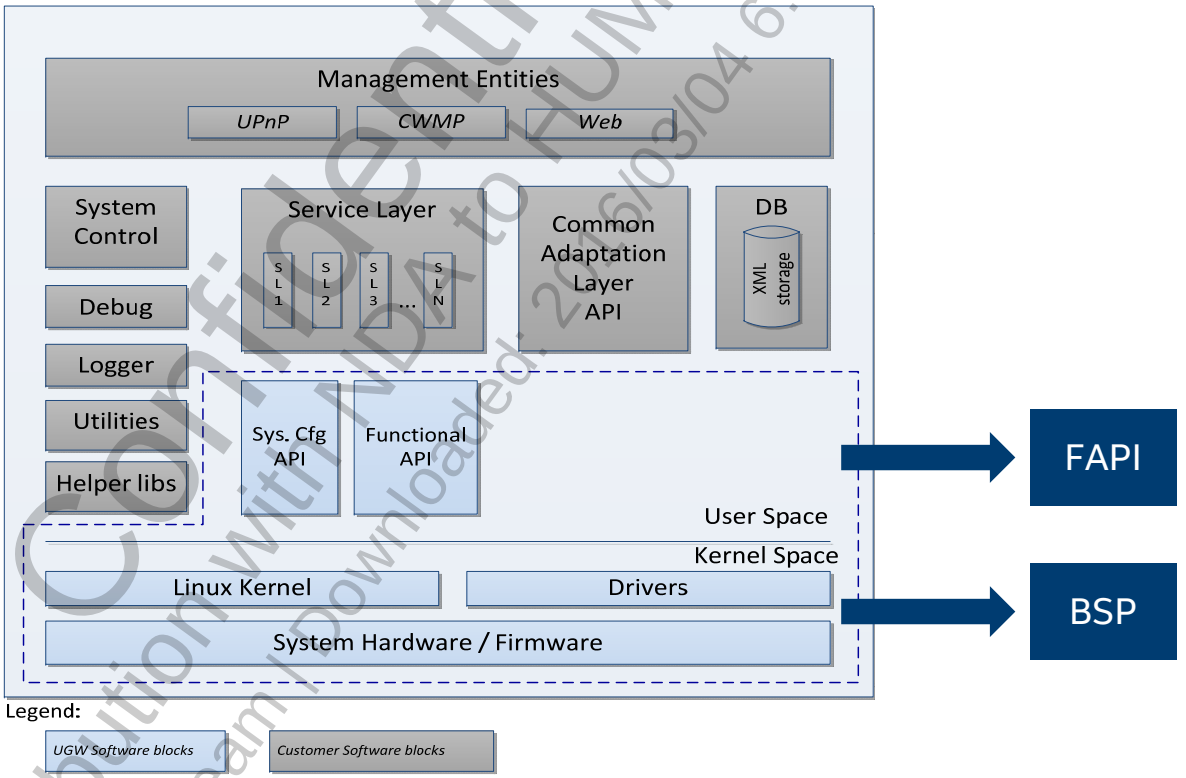
Release Calendar

- Overall release schedule of UGW-7.1

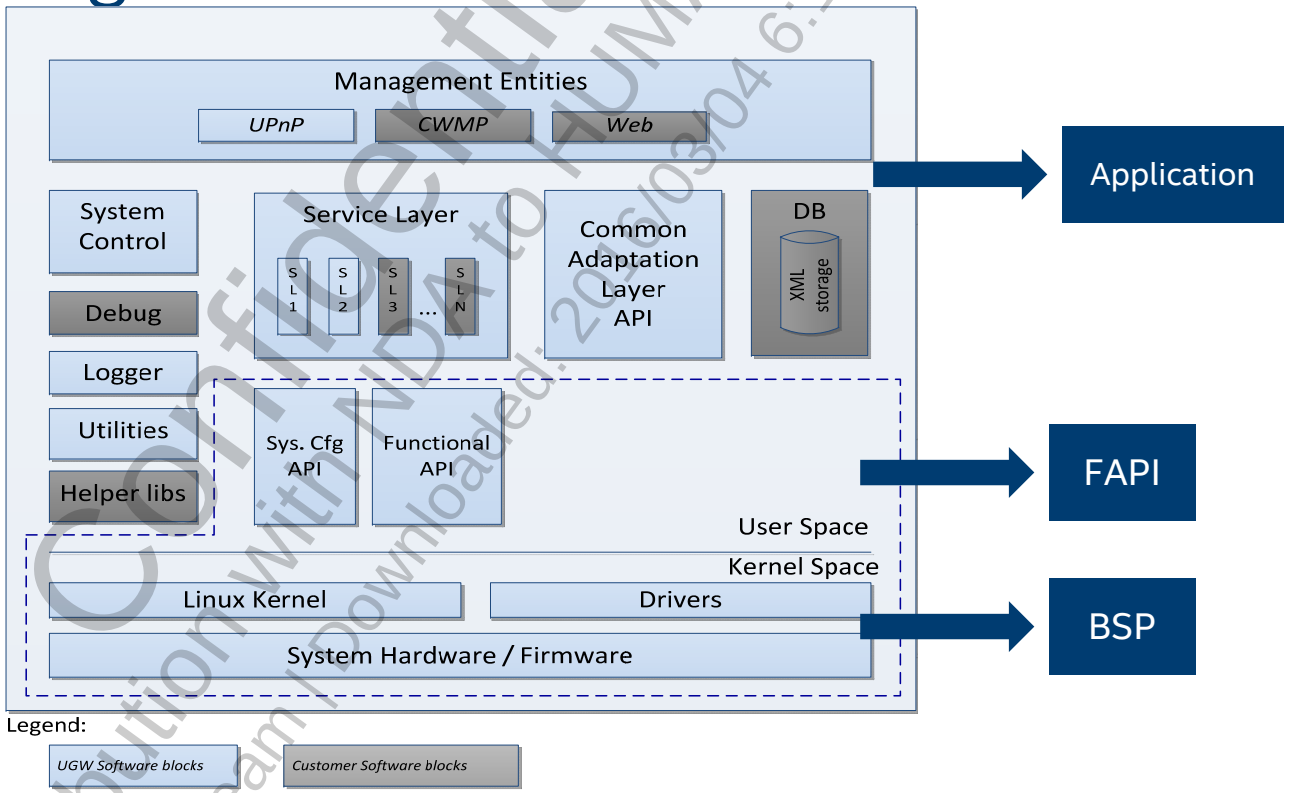
UGW-7.1: Customer Use-cases



Customer Use-cases: Customer has own System SW



Customer use-cases: Customer re-uses/extends Intel UGW package



UGW-7.1 GRX350 (UGW-6.1.1) System Release Mapping

UGW-7.1 Milestone	Target Schedule	SoC	GRX350 Release Mapping	Remarks
(Pre) Engineering (W7)	Available (Sep '15)	GRX350 V1.1	S7 (V1.1) + (PPP-3.3 W7 Rel)	SL+FAPI Release to address BSP customer Engagements (ALPHA)
Engineering (W7.2)	Dec '15	GRX350 V1.2	S7.9 (V1.1)	GRX350 System package Engineering Release (BETA)
Productive Software (W8)	Jan '16	GRX350 V1.2	S8 (V1.2)	GRX350 System package Productive Release (GA)

UGW-7.1: Release Calendar

	Sep15	Oct 15	Nov 15	Dec 15	Jan 16	Feb 16	Mar 16	Apr 16	May 16
GRX350 V1.1 Dual QCA	W7 Dual QCA 11/09								
GRX350 V1.2 Dual QCA				W7.2 Dual QCA 30/11	W8 Dual QCA 29/01				
GRX550 V1.2 Dual QCA				S/W7 Dual QCA 18/12	S/W8 Dual QCA 29/01				
GRX550 V1.2 Dual Wave									W8 Dual WAVE 30/05
GRX300/330 xRX220				W7.3 30/12					W8.2 31/05

Engineering Release

Productive Release

Exercises for TCS

Topic	Description	Module	Pointers
System FAPI utility usage	Use “sys_cli” to load PPA modules, initialize Acceleration, Add system interfaces to PPA	FAPI utility	Check loaded modules, PPA status, Configure Switch interfaces (link rate, duplex modes, find switch port no. corresponding to interface) UGW_Rel7.1_UM_PR_Rev1.0.pdf -> Sec-2.5
Debug	Using PAD, collect datapath log. Analyze following HDTs and feedback if the log helped to resolve following issues HDT-43588, HDT-43467, HDT-43330, HDT-42327, HDT-43650, HDT-41564	PAD	UGW_Rel7.1_Debug_AN_Rev1.0.pdf Feel free to check other HDTs apart from those listed here and share feedback on PAD tool usage
Logging	Ex: enter wrong PPPoE password. Check system log to find out the error message. Repeat test by entering correct password	Logging framework	https://wiki.lantiq.com/display/SW/Logging+Framework
Web UI	Add a new web page using view.xml Change the fonts, color of the webpage	Web UI	https://wiki.lantiq.com/display/SW/Web+UI
Service Library	Add a new Service Library, call existing FAPI and verify on the target	SL	UGW_Rel7.1_UM_PR_Rev1.0.pdf

References

- UGW Programmers Reference (covers FAPI)
  - UGW_Rel7.1_UM_PR_Rev1.0.pdf (UGW-7.1 → Documentation)
- Debug
  - Introduction to PAD https://wiki.lantiq.com/pages/viewpage.action?pageId=50135164
  - Video tour of PAD (prepared by Platform team) https://www.dropbox.com/sh/vubpxuoj8lcdy94/AABMA3_jzixSRHSVM0YllanKa?dl=0
  - Application Note – UGW_Rel7.1_Debug_AN_Rev1.0.pdf
- Logging
  - Introduction to System logging: https://wiki.lantiq.com/display/SW/Logging+Framework
- DBTool
  - https://wiki.lantiq.com/display/SW/DBTool
  - https://www.broadband-forum.org/cwmp/tr-181-2-9-0.html

Q & A

Connected Home Division

Intel Confidential