



# Web Page Programming Reference

Version: 1.4  
Release date: 2011-12-22

© 2011 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Specifications are subject to change without notice.

## Document Revision History

---

Revision	Date	Author	Description
Draft	Initial Draft	2008/7/1	Lai Wei-Sheng
1.0	Reformat	2008/7/15	Zheng Rui-Xing
1.1	Added ACL and ipfilter function descriptions	2008/10/21	Zheng Rui-Xing
1.2	Added AppFilter and UrlFilter function descriptions, and included check box lessons.	2008/10/30	Liu Zheng-Hong
1.3	Through cfg_manger enhancement, made changes to the steps for user to invoke a new node.	2009/2/16	Liu Zheng-Hong
1.4	Multilanguage support	2011/10/25	Lisa.dai

## Table of Contents

<b>Document Revision History .....</b>	<b>2</b>
<b>Table of Contents .....</b>	<b>3</b>
<b>1 Introduction.....</b>	<b>6</b>
<b>2 System Architecture .....</b>	<b>7</b>
<b>3 Supported Configure node Parameters Table List .....</b>	<b>9</b>
3.1 Wan Node : Information for setting CPE Wan PVC0~PVC7.....	9
3.2 Lan Node: information for setting CPE LAN .....	11
3.3 Dhcpd Node: Setting the Dhcp Server Contents .....	12
3.4 DhcpLease: Setting the Dhcp Client List Contents Dhcp Client List.....	14
3.5 Dproxy Node: Information for setting the DNS Relay.....	14
3.6 DhcpRelay Node: Information for setting the DNS Relay.....	14
3.7 WLAN Node: Setting the CPE WLAN Information .....	14
3.8 Firewall Node: Function for Firewall Settings .....	22
3.9 Route Node: Static Route Entry Settings .....	22
3.10 Dmz Node: Setting DMZ Information .....	23
3.11 VirServer Node: Setting Virtual Server Information .....	23
3.12 Adsl Node: Setting ADSL Mode Information.....	23
3.13 AutoPVC Node : Information for setting AutoPVC, currently is 16 Entry lines maximum..	24
3.14 Snmpd Node: Setting Snmpd Information .....	25
3.15 Upnpd Node: Setting UPNP Information .....	26
3.16 Ddns Node: Setting Dynamic DNS Information .....	26
3.17 ACL: Setting System Service Access Authorization .....	26
3.18 Ipfilter: Filter Internet Connection Function.....	26
3.19 AppFilter node: Filter Specific Internet Applications.....	27
3.20 UrlFilter node: Filter Specific URL Webpages .....	27
3.21 Account Node: Setting User Password .....	28
3.22 Timezone Node: Setting Time zone Information.....	28
3.23 Mac Node: Setting MAC Address Information .....	29
3.24 Autoexec Node : Autoexec Node: Setting the commands that will be automatically executed upon startup of machine.....	29
3.25 System Node Setting System Restart Categories, skbmgr_hot_len values and also generates the current ROMFILE .....	29
3.26 Diagnostic Node: Diagnose PVC status.....	30
3.27 WebCurSet Node: Setting Webpage Pull-down Menu ID .....	31
3.28 Info Node: Retrieves Ethernet, ADSL, WLAN, ra0-ra3, nas0-nas7 Information .....	32
3.29 DeviceInfo Node: Retrieves Device information .....	40
3.30 wifiMacTab Node: Retrieves some information from wifi mac table .....	42

3.31	TR069Attr Node: Stores tr069 parameter attribute information .....	43
3.32	Cwmp Node: Stores Cwmp parameters .....	43
3.33	SslCA Node: Stores SSL Ca parameters .....	45
3.34	CDSLDiagnostic: Temporary stores DSL diagnostic information .....	46
3.35	CATMDiagnostic: Temporary stores ATM diagnostic information .....	46
3.36	SysInfo: Stores system parameters .....	46
3.37	LanAlias Node: Store Alias IP Parameters .....	47
3.38	DhcpClient Node: Stores TR111 parameters .....	47
3.39	DhcpClientLimit Node: Stores TR111 time parameters .....	47
3.40	VoIPCodecs Node: Stores cpe supported voip codec parameters (TR104) .....	48
3.41	IPInterface Node: Stores IPInterface Enabled status (null function) .....	48
3.42	Radvd Node: Stores Radvd status parameters .....	48
3.43	Dhcp6s Node: Stores Dhcp6s status parameters .....	48
3.44	PortBind Node: Stores portbind information .....	49
3.45	IpAddrMapNode: store IP address mapping information .....	49
3.46	wifiMacTab Node: store wifi mac information .....	49
3.47	wifiMacIp Node: store IP information of related to wifi mac .....	50
3.48	dynDisp Node: store dynamic display information .....	50
3.49	QoS Node: store QoS information .....	50
3.50	VoIPBasic Node : store basic VoIP information .....	51
3.51	VoIPCallCtrl Node : store VoIP Call control information .....	52
3.52	VoIPMedia Node : storeVoIP media information .....	52
3.53	VoIPCodecs Node : storeVoIP CODEC information .....	52
3.54	VoIPLog Node : store VoIP Log information .....	53
3.55	VoIPAdvanced Node: store VoIP advances configuration information .....	53
3.56	VoIPDigitMap Node: store VoIP DigitMap information .....	53
3.57	SysLog Node: store syslog information .....	54
3.58	ALGSwitch Node:store information for application layer .....	54
3.59	DyVLAN Node: store dynamic VLAN information .....	54
3.60	GUI Node: store GUI information .....	55
3.61	GUITemp Node: store dynamic GUI temp information .....	55
3.62	Schedule Node: store dynamic schedule information .....	55
3.63	portTriggering Node: store dynamic portTriggering information .....	55
3.64	timeOfDay Node: store dynamic timeofday information .....	56
3.65	keywordTime Node: store dynamic keywordTime information .....	57
3.66	natSessions Node: store dynamic nat sessions information .....	57
3.67	DMS Node: store dynamic DMS information .....	57
3.68	signature Node : store signature information .....	58
3.69	DefaultWan Node : store default setting of WAN PVC .....	58
3.70	LanHost: Setting the Host List Contents Dhcp Client List and Static Client List .....	59
3.71	VendorCfgFile: Setting the Vendor Config File Info .....	60
3.72	LanguageSwitch Node: Node for multi-language switch .....	60
<b>4</b>	<b>ASP Support Functions Table Listing .....</b>	<b>61</b>

<b>5</b>	<b>CFG Manager adds a new configure node</b>	<b>69</b>
5.1	ROMFILE(romfile.cfg) Format	69
5.2	Adding a new configure node into romfile.cfg	73
5.3	Understanding the data structure of cfg_node_t	73
5.4	To add a new configure node to the all_cfg_node Table	77
5.5	Declare and Invoke for an added configure node operation	79
<b>6</b>	<b>Webpage Development Lessons</b>	<b>83</b>
<b>7</b>	<b>Multi-language Webpage Development</b>	<b>93</b>
7.1	Webpage Modification	93
7.2	mlCheckHash introduction	97
<b>8</b>	<b>Conclusion</b>	<b>99</b>
8.1	AutoPVC Node : Setting the AutoPVC information	99

## 1 Introduction

---

This system provides the user with the function to control the low-level operations of the system through a webpage. The webpage interface may be self-designed by the user and through functions provide by us, communicate with the low-level programs in the system, thus achieving the purpose of controlling the system. The system provides a complete set of mechanisms so that, when developing or maintaining webpages, the user needs only to focus on the development of the webpage and does not need to understand the program codes associated with the operations of the system. Hence, it greatly shortens the programming time needed for the development of the webpage.

The aim of this document is to educate the user in understanding the system architecture so that when required to develop a new function, the user will have sufficient knowledge to proceed with the development. In addition, we have provided an example as a simple webpage lesson, and users developing webpages could reference the example and be able to complete their function.

The main aim of this document is to educate development personnel how to use the functions provided by our company to develop their own webpage, so that their webpage will communicate with the low-level system. However, general knowledge regarding the relationship between HTTP and HTML is not within the introduction boundary of this document. If readers have questions regarding information on this matter, please refer to related books.

## 2 System Architecture

We hope to provide a system architecture that is: one, intuitive and ease development and maintenance of the webpage interface; two, helps user to speed up on the development of the webpage. Diagram 1 shows the System Architecture that was developed by us.

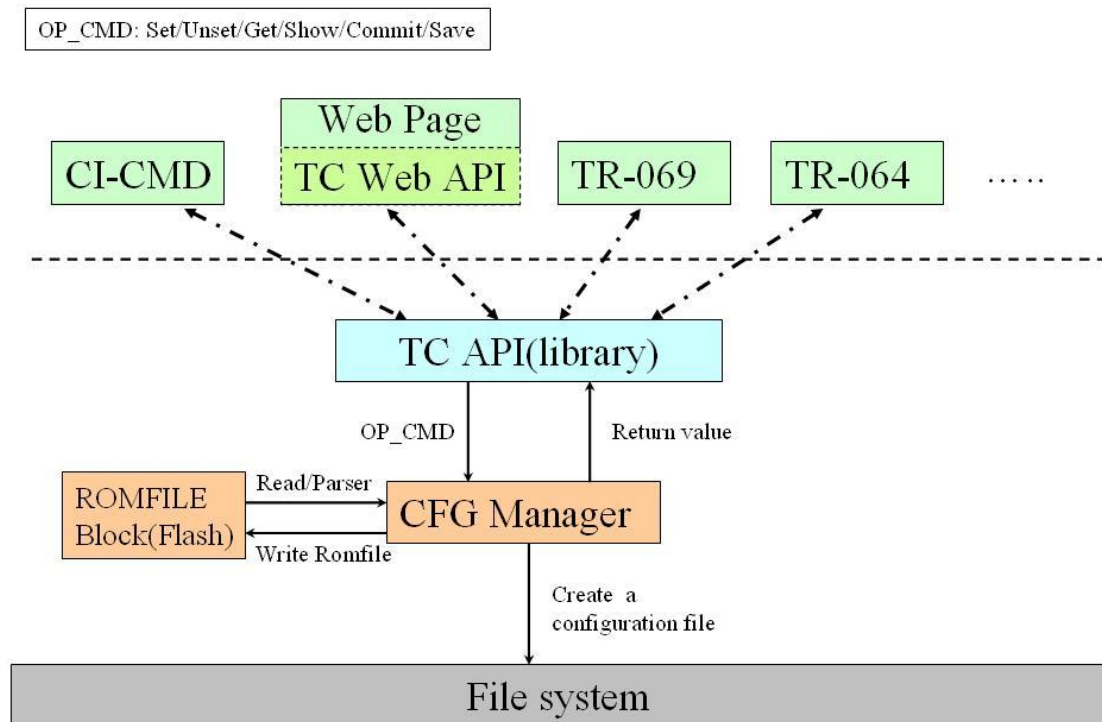


Diagram 1

- Diagram 1: Module Introduction
- CI-CMD: Customer Interface Command (# tcapi set/unset/get/show/commit/save xxx)
- Web Page: HTML Webpage (ex. home\_wan.asp, home\_wlan.asp...)
- TC Web Interface: ASP function provide by our company with direct link to TCAPI function database
- TR-069: Manages equipment agreement via the Wide Area Network (WAN) (function currently not available)
- TR-064: Manages equipment agreement via Local Area Network (LAN) (function currently not available)
- TCAPI: Function database provided by our company to allow external programs with common interface to access the values of the system parameters

- ROMFILE Block: One of the regions of the Flash system used to store the values of the system parameters configured by the user.
- CFG Manager: Coordinates the TSR for the system parameters control system
- File System: Linux File System

Refer to above Diagram 1, the system provides a common interface known as “TCAPI” for accessing system parameters. All external programs, with intention to access the system parameters, must operate via the interface provided by TCAPI, as the parameters cannot be retrieved through any other programs. Hence, on the one hand maintaining the integrity of the programs, while on the other hand, there is no necessity to self-develop other programs.

Below is the system startup operation process:

- Upon system startup, it will automatically evoke the executable for the CFG Manager
- CFG Manager retrieves the system parameters configured by the current system (system parameters which are recorded within the ROMFILE, in the format of XML) via Flash
- A configure node tree is constructed with the system parameters recorded in the ROMFILE
- A corresponding setup file is generated based on the information from the individual configure node

Configure node can be viewed as a collection of relevant parameters of a certain function in the system. TCAPI provides a common interface to allow external programs to access the system parameters but the physical access and operation of the system parameters is completed independently by the CFG Manager.

Figure 1 Linux Software Architecture



### 3 Supported Configure node Parameters Table List

CFG Manager controls all of the information of the configure node and it also has the function to reactivate or stop the configure node service. Each configure node exist basically to correspond to an executable program or an executable shell script, and the execution procedure and shell script will execute corresponding actions based on the contents of the node configuration file created by the CFG Manager. Below is the configuration Node parameters Tables List provided by the current CFG Manager. User may compare the system file /userfs/romfile.cfg as a reference.

At the end of the field, information has been provided as to whether we can read/write (R/W) the attribute. R(Read) denotes that the parameter can be read by using get; W(Write) denotes that the parameter can be set using set.

#### 3.1 Wan Node : Information for setting CPE Wan PVC0~PVC7

- Wan\_Common:common information of WAN node

Item	Node Name	Attribute	Value	RW
1	Wan_Common	UniqueMac	{0,1}	RW
2	Wan_Common	CDVT_Enable	{0,1}	RW

- PVC0 : Wan\_PVC0, PVC1 : Wan\_PVC1 ..., PVC7 : Wan\_PVC7

Item	Node Name	Attribute	Value	RW
1	Wan_PVC0	Active	{Yes, No}	RW
2	Wan_PVC0	VPI	0~255	RW
3	Wan_PVC0	VCI	1~65535	RW
4	Wan_PVC0	QOS	{ubr, cbr, rt-vbr , nrt-vbr}	RW
5	Wan_PVC0	PCR	0~5500	RW
6	Wan_PVC0	SCR	0~5500(SCR < PCR)	RW
7	Wan_PVC0	MBS	0~655325	RW
8	Wan_PVC0	ISP	0~3 {0:Dynamic IP Address, 1:Static IP Address, 2: PPPoA/PPPoE, 3: Bridge Mode}	RW
9	Wan_PVC0	USERNAME	64 characters	RW
10	Wan_PVC0	PASSWORD	64 characters	RW

11	Wan_PVC0	ENCAP	{1483 Bridged IP LLC, 1483 Bridged IP VC-Mux, 1483 Routed IP LLC(IPoA), 1483 Routed IP VC-Mux, PPPoE LLC, PPPoE VC-Mux, PPPoA LLC, PPPoA VC-Mux}	RW
12	Wan_PVC0	CONNECTION	{Connect_Keep_Alive, Connect_on_Demand}	RW
13	Wan_PVC0	CLOSEIFIDLE	0~999	RW
14	Wan_PVC0	MSS	100~1452(0: default value)	RW
15	Wan_PVC0	PPPGETIP	{Dynamic,Static}	RW
16	Wan_PVC0	IPADDR	X.X.X.X(IP Address format)	RW
17	Wan_PVC0	NETMASK	X.X.X.X(Netmask Address format)	RW
18	Wan_PVC0	GATEWAY	X.X.X.X(IP Address format)	RW
29	Wan_PVC0	NATENABLE	{Enable,Disabled}	RW
20	Wan_PVC0	DEFAULTROUTE	{Yes, No}	RW
21	Wan_PVC0	MTU	576~1500(0: default value)	RW
22	Wan_PVC0	RIPVERSION	{RIP1, RIP2-B, RIP2-M}	RW
23	Wan_PVC0	DIRECTION	{None, Both, IN Only, OUT Only}	RW
24	Wan_PVC0	IGMP	{IGMP v1, IGMP v2}	RW
25	Wan_PVC0	AUTHEN	{CHAP, PAP, BOTH}	RW
26	Wan_PVC0	ACNAME	0~256 characters	RW
27	Wan_PVC0	SRVNAME	0~256 characters	RW
28	Wan_PVC0	CONNAME	0~256 characters	RW
29	Wan_PVC0	BRIDGEDTYPE	{IP, PPP}	RW
30	Wan_PVC0	PPPo6Mode	{"0","1"}	RW
31	Wan_PVC0	PPPo6PD	{Yes, No}	RW
32	Wan_PVC0	MLDproxy	{Yes, No}	RW
33	Wan_PVC0	IPADDR6	{IPv6 address format, eg: 2001:db8:1::1}	RW
34	Wan_PVC0	PREFIX6	16~64 characters	RW

35	Wan_PVC0	DEFGATEWAY 6	{IPv6 address format, eg: 2001:db8:1::1}	RW
37	Wan_PVC0	DNSIPv61st	{IPv6 address format, eg: 2001:db8:1::1}	RW
38	Wan_PVC0	DNSIPv62nd	{IPv6 address format, eg: 2001:db8:1::1}	RW
39	Wan_PVC0	PVCScanRese rved	{Yes, No}	RW
40	Wan_PVC0	dot1q	{Yes, No}	RW
41	Wan_PVC0	VLANID	0~4095	RW
42	Wan_PVC0	BridgeInterface	{Yes, No}	R
43	Wan_PVC0	IGMPproxy	{Yes, No}	RW
44	Wan_PVC0	DSLITEEnable	{Yes, No}	RW
45	Wan_PVC0	DSLITEMode	{0:Auto, 1:Manual}	RW
46	Wan_PVC0	DSLITEAddr	{IPv6 address format, eg: 2001:db8:1::1}	RW
47	Wan_PVC0	EnableDynIPv 6	{0:SLAAC, 1:DHCP}	RW

### 3.2 Lan Node : information for setting CPE LAN

- Dhcp: Select activate/stop DHCP Server or select to use DHCP Relay
- Entry0~N Multiple LAN IP will be required in future development

Item	Node Name	Attribute	Value	RW
1	Lan_Dhcp	type	{0: Disable, 1: Enable, 2: Relay}	RW
2	Lan_Ripd	RIPVERSION	{RIP1, RIP2-B, RIP2-M}	RW
3	Lan_Ripd	DIRECTION	{None, Both, IN Only, OUT Only}	RW
4	Lan_Entry0	IP	X.X.X.X(IP Address format)	RW
5	Lan_Entry0	netmask	X.X.X.X(Netmask Address format)	RW
6	Lan_Entry0	IP6	{IPv6 address format, eg: 2001:db8:1::1}	RW
7	Lan_Entry0	PREFIX6	16~64 characters	RW
8	Lan_EtherMedia	enable	{“none”, “support”}	RW

9	Lan_EtherMedia	port0	{“Auto”, “100_Full”, “100_Ha lf”, “10_Full”, “10_Half”}	RW
10	Lan_EtherMedia	port1	{“Auto”, “100_Full”, “100_Ha lf”, “10_Full”, “10_Half”}	RW
11	Lan_EtherMedia	port2	{“Auto”, “100_Full”, “100_Ha lf”, “10_Full”, “10_Half”}	RW
12	Lan_EtherMedia	port3	{“Auto”, “100_Full”, “100_Ha lf”, “10_Full”, “10_Half”}	RW
13	Lan_IgmpSnoop	igmpsnoopEn able	{Yes, No}	RW

### 3.3 Dhcpcd Node : Setting the Dhcp Server Contents

- Dhcpd\_Entry0~7:Store static ip-mac items
- Dhcpd\_Common: substitute the original Dhcpd\_Entry and stores the Dhcp server configuration

Item	Node Name	Attribute	Value	RW
1	Dhcpd_Common( replace the Dhcpd_Entry)	start	X.X.X.X(IP Address format)	RW
2	Dhcpd_Common( replace the Dhcpd_Entry)	pool_count	1~253	RW
3	Dhcpd_Common( replace the Dhcpd_Entry)	lease	seconds	RW
4	Dhcpd_Common	router	X.X.X.X(IP Address format)	RW
5	Dhcpd_Common	configurable	0/1	RW
6	Dhcpd_Common	DomainName	0~64 characters	RW
7	Dhcpd_Entry0	IP	X.X.X.X(IP Address format)	RW
8	Dhcpd_Entry0	MAC	XX:XX:XX:XX:XX:XX(MAC Address format)	RW
9	Dhcpd_Option60	Active	Yes/No	RW
10	Dhcpd_Option60	Vendor ID	0~63 characters	RW
11	Dhcpd_Option60	start	X.X.X.X(IP Address format)	RW
12	Dhcpd_Option60	pool_count	1~253	RW
13	Dhcpd_Option60	lease	seconds	RW

14	Dhcpd_Option60	router	X.X.X.X(IP Address format)	RW
15	Dhcpd_Option60	dns1	X.X.X.X(IP Address format)	RW
16	Dhcpd_Option60	dns2	X.X.X.X(IP Address format)	RW
17	Dhcpd_Option60	serverIP	X.X.X.X(IP Address format)	RW
18	Dhcpd_Option60	subnetMask	X.X.X.X(IP Address format)	RW
19	Dhcpd_Option60	domainName	0~38 characters	RW
20	Dhcpd_Option60	vendorIDEx	0/1	RW
21	Dhcpd_Option60	vendorIDMode	“Exact” “Prefix” “Suffix” “Substring”	RW
22	Dhcpd_Option60	clientID	0~64 characters	RW
23	Dhcpd_Option60	clientIDEx	0/1	RW
24	Dhcpd_Option60	userClassID	0~64 characters	RW
25	Dhcpd_Option60	userClassIDEx	0/1	RW
26	Dhcpd_Option60	chaddr	X:X:X:X:X:X(MAC Address format)	RW
27	Dhcpd_Option60	chaddrMask	X:X:X:X:X:X(MAC Address format)	RW
28	Dhcpd_Option60	chaddrEx	0/1	RW
29	Dhcpd_Option60	locallyServed	0/1	RW
30	Dhcpd_Option60	useAllocatedWAN	“Normal” “Passthrough”	RW
31	Dhcpd_Option240	Active	Yes/No	RW
32	Dhcpd_Option240	Value	0~87 characters	RW
33	Dhcpd	Empty_Entry	1~253	R
34	Dhcpd	Static_Num	1~253	R

### 3.4 DhcpLease : Setting the Dhcp Client List Contents Dhcp Client List

- DhcpLease\_Entry0~254:store dhcp active client list items

Item	Node Name	Attribute	Value	RW
1	DhcpLease_Entry 0	IP	X.X.X.X(IP Address format)	RW
2	DhcpLease_Entry 0	MAC	XX:XX:XX:XX:XX:XX(MAC Address format)	RW
3	DhcpLease_Entry 0	ExpireDay	0~n	RW
4	DhcpLease_Entry 0	ExpireTime	xx:xx:xx(time format)	RW
5	DhcpLease_Entry 0	Hostname	0~32 characters	RW
6	DhcpLease	LeaseNum	1~253	R

### 3.5 Dproxy Node : Information for setting the DNS Relay

Item	Node Name	Attribute	Value	RW
1	Dproxy_Entry	type	0:Manually ,1: Automatically	RW
2	Dproxy_Entry	Primary_DNS	X.X.X.X(IP Address format)	RW
3	Dproxy_Entry	Secondary_D NS	X.X.X.X(IP Address format)	RW

### 3.6 DhcpRelay Node : Information for seting the DNS Relay

Item	Node Name	Attribute	Value	RW
1	DhcpRelay_Entry	Server	X.X.X.X(IP Address format)	RW

### 3.7 Wlan Node : Setting the CPE WLAN Information

- Common: WLAN information shared by configurations
- Entry0~N : BSSID 0    Entry0, BSSID 3    Entry3

Item

Item	Node Name	Attribute	Value	RW
1	Wlan_Common	APOn	1 0	RW

			{ALBANIA, ALGERIA, ARGENTINA, ARMENIA, AUSTRALIA, AUSTRIA, AZERBAIJAN, BAHRAIN, BELARUS, BELGIUM, BELIZE, BOLVIA, BRAZIL, BRUNEI DARUSSALAM, BULGARIA, CANADA, CHILE, CHINA, COLOMBIA, COSTA RICA, CROATIA, CYPRUS, CZCH REPUBLIC, DENMARK, DOMINICANRE PUBLIC, ECUADOR, EGYPT, ELSALVADOR, ESTONIA, FINLAND, FRANCE, GEORGIA, GERMANY, GREECE, GUATEMALA, HONDURAS, HONGKONG, HUNGARY, ICELAND, INDIA, INDONESIA, IRAN, IRELAND, ISAREL, ITALY, JAPAN, JORDAN, KAZAKHSTAN, NORTH KOREA, KORE AREPUBLIC, KUWAIT,	RW	
--	--	--	---	----	--



3	Wlan_Common	CountryRegion	0~6	RW
4	Wlan_Common	Channel	01~13	RW
5	Wlan_Common	BeaconPeriod	20~1000	RW
6	Wlan_Common	RTSThreshold	1500~2347	RW
7	Wlan_Common	FragThreshold	256~2346	RW
8	Wlan_Common	DtimPeriod	1~255	RW
9	Wlan_Common	BssidNum	1~4	RW
10	Wlan_Common	WirelessMode	{11B/G Mixed: 0, 11B: 1, 11G-Only: 2}	RW
11	Wlan_Common	BasicRate	{11B/G Mixed: 15, 11B: 3, 11G-Only: 351}	RW
12	Wlan_Common	HT_BW	0:20MHz 1:20/40MHz	RW
13	Wlan_Common	HT_GI	0:800 nsec 1:400 nsec	RW
14	Wlan_Common	HT_EXTCHA	0:extension channel below the control channel 1:extension channel above the control channel ADD for WIFI TxRate API	RW R
15	Wlan_Common	rt_device	The value show the WIFI driver such as 3390 or 3092	
16	Wlan_Common	EfuseBufferMode	{0:effuse mode, 1:buffer mode}	RW
17	Wlan_Common	TxPower	0~100	RW
18	Wlan_Common	TxPreamble	{0:Long Preamble, 1:Short Preamble}	RW

19	Wlan_Common	session_timeout_interval	0:to disable reauthentication for every session; >60: to set reauthentication interval with unit of second	RW
20	Wlan_Common	IdleTimeout	number	RW
21	Wlan_Common	IgmpSnEnable	{0:Disable, 1: Enable}	RW
22	Wlan_Common	BGProtection	{1:Disable, 0: Enable}	RW
23	Wlan_Common	TxBurst	{0:Disable, 1: Enable}	RW
24	Wlan_Common	PktAggregate	{0:Disable, 1: Enable}	RW
25	Wlan_Common	ShortSlot	{0:Disable, 1: Enable}	RW
26	Wlan_Common	RekeyMethod	{DISABLE,TIME}	RW
27	Wlan_Common	APSDCapable	{0:Disable, 1: Enable}	RW
28	Wlan_Common	HT_AMSDU	{0:Disable, 1: Enable}	RW
29	Wlan_Common	HT_STBC	{0:Disable, 1: Enable}	RW
30	Wlan_Common	HT_AutoBA	{0:Disable, 1: Enable}	RW
31	Wlan_Common	HT_RDG	{0:Disable, 1: Enable}	RW
32	Wlan_Common	HT_BADecline	{0:Disable, 1: Enable}	RW
33	Wlan_Common	HT_DisallowTKIP	{0:Disable, 1: Enable}	RW
34	Wlan_Common	HT_TxStream	{1:Support 1-Tx Stream for MCS0~MCS7;2: Support 2-Tx Stream for MCS0~MCS15}	RW
35	Wlan_Common	HT_RxStream	{1:Support 1-Rx Stream for MCS0~MCS7;2: Support 2-Rx Stream for MCS0~MCS15}	RW
36	Wlan_Common	IEEE80211H	{0:Disable, 1: Enable}	RW

37	WLAN_Common	11nMode	{0:Disable, 1: Enable}	RW
38	WLAN_Common	HT_OpMode	{0:Mixed Mode, 1: Green Field}	RW
39	WLAN_Entry0	SSID	{0~z, less than 32 characters}	RW
40	WLAN_Entry0	HideSSID	{0: Disable, 1: Enable}	RW
41	WLAN_Entry0	AuthMode	{OPEN, WEPAUTO, SHARED, WPAPSK, WPA, WPA2PSK, WPA2, WPA1WPA2, WPAPSKWPA2PSK}	RW
42	WLAN_Entry0	EncrypType	{NONE, WEP, TKIP, AES, TKIPAES}	RW
43	WLAN_Entry0	DefaultKeyID	1~4(1 ~ 4 for WEP, 2 for TKIP, AES(WPA/WPA2, WPAPSK/WPA2PSK))	RW
44	WLAN_Entry0	Key1Str	{5 ascii characters or 10 hex number or 13 ascii characters or 26 hex numbers}	RW
45	WLAN_Entry0	Key2Str	{5 ascii characters or 10 hex number or 13 ascii characters or 26 hex numbers}	RW
46	WLAN_Entry0	Key3Str	{5 ascii characters or 10 hex number or 13 ascii characters or 26 hex numbers}	RW
47	WLAN_Entry0	Key4St	{5 ascii characters or 10 hex number or 13 ascii characters or 26 hex numbers}	RW
48	WLAN_Entry0	WPAPSK	{8~63 ASCII or 64 HEX characters}	RW

49	Wlan_Entry0	AccessPolicy	{0: Disable, 1: Allow all, 2: Reject all}	RW
50	Wlan_Entry0	Wlan_MAC0	Mac address format	RW
51	Wlan_Entry0	Wlan_MAC1	Mac address format	RW
52	Wlan_Entry0	Wlan_MAC2	Mac address format	RW
53	Wlan_Entry0	Wlan_MAC3	Mac address format	RW
54	Wlan_Entry0	Wlan_MAC4	Mac address format	RW
55	Wlan_Entry0	Wlan_MAC5	Mac address format	RW
56	Wlan_Entry0	Wlan_MAC6	Mac address format	RW
57	Wlan_Entry0	Wlan_MAC7	Mac address format	RW
58	Wlan_Entry0	WPSConfStatus	1:unconfigured 2:configured	RW
59	Wlan_Entry0	WPSMode	0: pincode 1:PBC	RW
60	Wlan_Entry0	WscDefaultSSID1	When AP is registrar and status is unconfigured, a new profile will be generated after execute wps. Such attribute is used to configure the new SSID	RW

61	Wlan_Entry0	WPSKeyASCII	When AP is registrar and status is unconfigured, a new profile will be generated after execute wps. Such attribute is used to configure the new WPA key.( 0: general a key with 64 HEX characters, 1: general a with 1 ASCII, eg.8: general a with 8 ASCII)	RW
62	Wlan_Entry0	WPSConfMode	0: disable 1: enrollee 2:proxy 4: registrar 7: all	RW
63	Wlan_Entry0	WMM	0:disable 1:enable	RW
64	Wlan_Entry0	HT_MCS	0~15,32:fix MCS rate for HT rate 33:auto rate adaption,recommended	RW
65	Wlan_Entry0	HT_RATE	ADD for WIFI TxRate API b/g mode rate:1,2,5.5,6,9,11,12,18,24,36,48,54 adaption,recommended	RW
66	Wlan_Entry0	RADIUS_Server	X.X.X.X(IP Address format)	RW
67	Wlan_Entry0	RADIUS_Port	0~65535	RW
68	Wlan_Entry0	RADIUS_Key	String:longer than 8 ASCII characters	RW

69	Wlan_Entry0	IgmpSnEn	{0:Disable, 1: Enable}	RW
70	Wlan_Entry0	KeyPassphrase	{0:Disable, 1: Enable}	RW
71	Wlan_Entry0	WEPAuthType	{OpenSystem,Shared Key,WEPAuto}	RW
72	Wlan_Entry0	enrollePinCode	More than 8 digits	RW
73	Wlan_WDS	WdsEnable	{0:Disable, 1: Enable}	RW
74	Wlan_WDS	Wds_MAC0	Mac address format	RW
75	Wlan_WDS	Wds_MAC1	Mac address format	RW
76	Wlan_WDS	Wds_MAC2	Mac address format	RW
77	Wlan_WDS	Wds_MAC3	Mac address format	RW
78	Wlan_WDS	WdsEncrypType	{NONE,WEP,TKIP,AES}	RW
79	Wlan_WDS	WdsKey	10 or 26 hexadecimal characters(eg.1234567890) for WEP, 5 or 13 ASCII characters for WEP; 8 or 63 ASCII characters for TKIP or AES; 64 hexadecimal characters for TKIP or AES	RW

### 3.8 Firewall Node : Function for Firewall Settings

Item	Node Name	Attribute	Value	RW
1	Firewall_Entry	firewall_status	{0:Disable, 1: Enabled}	RW
2	Firewall_Entry	spi_status	{0:Disable, 1: Enabled}	RW

### 3.9 Route Node : Static Route Entry Settings

- Route\_EntryID, ID=0 Node name= Route\_Entry0, ID =15 Node name=Route\_Entry15, static route entry setting, User\_def\_num and Route\_num are generated by the system based on the information in the current route table. User is only allowed to query but cannot configure the information for these two attributes.

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	Route	User_def_num	{0~15}, Only for query information, Not support set	R
2	Route	Route_num	{0~50}, Only for query information, Not support set	R
3	Route_Entry0	DST_IP	X.X.X.X(IP Address format)	RW
4	Route_Entry0	Sub_mask	X.X.X.X(Submask Address format)	RW
5	Route_Entry0	Device	nas0~nas7 or ppp0~ppp7	RW
6	Route_Entry0	Gateway	X.X.X.X(IP Address format)	RW
7	Route_Entry0	metric	0~15	RW
8	Route_Entry0	Use	0	RW
9	Route_Entry0	User_def	Get information from Route configure node, {0: system route entry, 1 : user define route entry}	R

### 3.10Dmz Node : Setting DMZ Information

PVC0 : Dmz\_PVC0, PVC1 : Dmz\_PVC1 ..., PVC7 : Dmz\_PVC7

Item	Node Name	Attribute	Value	RW
1	Dmz_PVC0	Active	{Yes, No}	RW
2	Dmz_PVC0	DMZ_IP	X.X.X.X(IP Address format)	RW

### 3.11VirServer Node : Setting Virtual Server Information

VirServer\_PVC\_Num\_Rule\_Index, PVC\_Num : 0~7, Rule\_Index : 0~9

Item	Node Name	Attribute	Value	RW
1	VirServer_PVC0_Entry0	STARTPORT	1~65536	RW
2	VirServer_PVC0_Entry0	ENDPORT	1~65536	RW
3	VirServer_PVC0_Entry0	LOCALIP	X.X.X.X(IP Address format)	RW

### 3.12Adsl Node : Setting ADSL Mode Information

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	Adsl_Entry	MODULATION TYPE	{Auto Sync-Up, ADSL2+, ADSL2, G.DMT, T1.413, G.lite}	RW
2	Adsl_Entry	ANNEXTYPE A	{ANNEX A, ANNEX I, ANNEX A/L, ANNEX M, ANNEX A/I/J/L/M}	RW

### 3.13 AutoPVC Node : Information for setting AutoPVC, currently is 16 Entry lines maximum.

Item	Node Name	Attribute	Value	RW
1	AutoPVC_Comm on	Active	{1:Active,0:Deactive}	RW
2	AutoPVC_Comm on	Probe_OAM	{1:search oam,0:not search oam}	RW
3	AutoPVC_Comm on	Probe_ARP	{1:search arp,0:not search arp }	RW
4	AutoPVC_Comm on	Probe_PPPO E	{1:search pppoe,0:not search pppoe }	RW
5	AutoPVC_Comm on	Probe_DHCP	{1:search dhcp,0:not search dhcp }	RW



6	AutoPVC_Comm on	Encap	{1:VC,0:LLC}	RW
7	AutoPVC_Entry	PVC	{0~8,the pvc we want to replace}	RW
8	AutoPVC_Entry	VPI	{0~255,the vpi we want to search}	RW
9	AutoPVC_Entry	VCI	{1~65535,the vci we want to search}	RW

### 3.14Snmpd Node : Setting Snmpd Information

Item	Node Name	Attribute	Value	RW
1	Snmpd_Entry	Active	Yes, No	RW
2	Snmpd_Entry	rocommunity	{0~z, less than 15 characters}	RW
3	Snmpd_Entry	rwcommunity	{0~z, less than 15 characters}	RW
4	Snmpd_Entry	sysName	string	RW
5	Snmpd_Entry	sysContact	string	RW
6	Snmpd_Entry	sysLocation	string	RW
7	Snmpd_Entry	v3Name	string	RW
8	Snmpd_Entry	access	string	RW
9	Snmpd_Entry	authProto	string	RW
10	Snmpd_Entry	authPasswd	string	RW
11	Snmpd_Entry	privPasswd	string	RW
12	Snmpd_Entry	privProto	string	RW
13	Snmpd_Entry	v3Enable	string	RW
14	Snmpd_Entry	trustStartIPv6	{IPv6 address format, eg: 2001:db8:1::1}	RW

15	Snmpd_Entry	trustEndIPv6	{IPv6 address format, eg: 2001:db8:1::100}	RW
16	Snmpd_Entry	trap2sink_ipv6	{IPv6 address format, eg: 2001:db8:1::1}	RW

### 3.15 Upnpd Node : Setting UPNP Information

Item	Node Name	Attribute	Value	RW
1	Upnpd_Entry	Active	{Yes, No}	RW
2	Upnpd_Entry	autoconf	{0:Deactivated , 1 :Activated}	RW

### 3.16 Ddns Node : Setting Dynamic DNS Information

Item	Node Name	Attribute	Value	RW
1	Ddns_Entry	Active	{Yes, No}	RW
2	Ddns_Entry	SERVERNAME	www.dyndns.org	R
3	Ddns_Entry	MYHOST	Domain name	RW
4	Ddns_Entry	USERNAME	{0~z, less than 63 characters}	RW
5	Ddns_Entry	PASSWORD	{0~z, less than 31 characters}	RW
6	Ddns_Entry	WILDCARD	{Yes, No}	RW

### 3.17 ACL : Setting System Service Access Authorization

Item	Node Name	Attribute	Value	RW
1	ACL_Common	Activate	{Yes, No}	RW
2	ACL_Entry	Activate	{Yes, No}	RW
3	ACL_Entry	SrcIPAddrBegin	X.X.X.X(IP Address format)	RW
4	ACL_Entry	SrcIPAddrEnd	X.X.X.X(IP Address format)	RW
5	ACL_Entry	Application	{ftp, telnet, http, snmp, ping, all}	RW
6	ACL_Entry	Interface	{Lan, Wan, Both}	RW

### 3.18 Ipfiler : Filter Internet Connection Function

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	IpMacFilter_Common	ListType	{White, Black}	RW
2	IpMacFilter_Entry	Interface	{Yes, No}	RW
3	IpMacFilter_Entry	Direction	{Incoming, Outgoing, Both}	RW
4	IpMacFilter_Entry	RuleType	{IP, Mac}	RW
5	IpMacFilter_Entry	Active	{Yes, No}	RW
6	IpMacFilter_Entry	SrcIPAddr	X.X.X.X(IP Address format)	RW
7	IpMacFilter_Entry	SrcIPMask	X.X.X.X(Netmask Address format)	RW
8	IpMacFilter_Entry	SrcPort	0~65535	RW
9	IpMacFilter_Entry	DesIPAddr	X.X.X.X(IP Address format)	RW
10	IpMacFilter_Entry	DesIPMask	X.X.X.X(Netmask Address format)	RW
11	IpMacFilter_Entry	DesPort	0~65535	RW
12	IpMacFilter_Entry	Protocol	{TCP, UDP, ICMP}	RW
13	IpMacFilter_Entry	MacAddr	MAC address format , example : 00:00:aa:bb:cc:ff	RW

### 3.19AppFilter node : Filter Specific Internet Applications

Item	Node Name	Attribute	Value	RW
1	AppFilter_Entry	Activate	{1, 0}	RW
2	AppFilter_Entry	Block_ICQ	{1, 0}	RW
3	AppFilter_Entry	Block_MSN	{1, 0}	RW
4	AppFilter_Entry	Block_YMSG	{1, 0}	RW
5	AppFilter_Entry	Block_RTSP	{1, 0}	RW

### 3.20UrlFilter node : Filter Specific URL Webpages

Item	Node Name	Attribute	Value	RW
1	UrlFilter_Common	Activate	{1, 0}	RW
2	UrlFilter_Entry	Activate	{1, 0}	RW
3	UrlFilter_Entry	URL	URL, e.g. http://tw.yahoo.com	RW

### 3.21 Account Node : Setting User Password

Item	Node Name	Attribute	Value	RW
1	Account_Entry	web_passwd	{0~z, less than 31 characters}	RW
2	Account_Entry	username	{0~z, less than 31 characters}	RW
3	Account_Entry	console_password	{0~z, less than 31 characters}	RW
4	Account_Entry	display_mask	xx xx xx xx xx xx xx xx (x:0-F)	RW

### 3.22 Timezone Node : Setting Time zone Information

Item	Node Name	Attribute	Value	RW
1	Timezone_Entry	TYPE	{0: NTP Server automatically, 1: PC's Clock, 2: Manually}	RW
2	Timezone_Entry	TZ	GMT, GMT+01:00, GMT+02:00, GMT+03:00, GMT+04:00, GMT+04:30, GMT+05:00, GMT+06:00, GMT+07:00, GMT+08:00, GMT+09:00, GMT+09:30, GMT+10:00, GMT+11:00, GMT+12:00, GMT-12:00, GMT-11:00, GMT-10:00, GMT-9:00, GMT-8:00, GMT-7:00, GMT-6:00, GMT-5:00, GMT-4:00, GMT-3:00, GMT-2:00, GMT-1:00	RW
3	Timezone_Entry	DAYLIGHT	{Enable, Disable}	RW
4	Timezone_Entry	SERVER	X.X.X.X (IP Address format)	RW

5	Timezone_Entry	PC_CLOCK	{Time format ,ext: 2008/6/1 19:17:01}	RW
6	Timezone_Entry	Year	XXXX, ext: 2008	RW
7	Timezone_Entry	Month	1~12	RW
8	Timezone_Entry	Date	1~31	RW
9	Timezone_Entry	Hour	0~23	RW
10	Timezone_Entry	Min	0~59	RW
11	Timezone_Entry	Sec	0~59	RW

### 3.23Mac Node : Setting MAC Address Information

Item	Node name	Attribute	Value	RW
1	Mac_Entry	WAN_MAC	MAC address format , example : 00:00:aa:bb:cc:ff	RW
2	Mac_Entry	LAN_MAC	MAC address format , example : 00:00:aa:bb:cc:ff	RW

### 3.24Autoexec Node : Autoexec Node: Setting the commands that will be automatically executed upon startup of machine

Attribute: command0 ~ commandN: System configure node provides system parameters, and it is not necessary to save into ROMFILE.

Item	Node Name	Attribute	Value	RW
1	Autoexec_Entry	command0	ci-command format	RW

### 3.25System Node Setting System Restart Categories, skbmgr\_hot\_len values and also generates the current ROMFILE

Item	Node Name	Attribute	Value	RW
1	System_Entry	reboot_type	{0:Do nothing, 1:reboot with current settings, 2: reset to default factory settings}	RW

2	System_Entry	skbmgr_hot_le n	{0: When the system have not enough free memory, you can write 0 to get more free memory, but the throughput will be decreasing, 512: Default value}	RW
3	System_Entry	bkrom	{0:Do nothing, 1:To create a current ROMFILE at /var/tmp/romfile.cfg}	RW
4	System_Entry	upgrade_fw	{0:Do nothing, 1:Upgrade ROMFILE, 2: Upgrade Kernel Image,3:Upgrade file system }	RW
5	System_Entry	update_wan_i p	{0:Do nothing, 1: Trigger CPE to update wan interface's IP address, 2: Trigger CPE to release wan interface's IP address}	RW
6	System_Entry	Start_up	{0:Do nothing, other value: Trigger CPE to run all boot function of the nodes in node tree}	RW

The below four configure node, WebCurSet, Diagnostic, DeviceInfo, Info configure node, are used primarily to provide the information required by the web interface. The information from these four configure node will not be saved in the ROMFILE. The main reason is that this system information cannot be configured via the webpage but is used only to retrieve relevant information.

### 3.26Diagnostic Node : Diagnose PVC status

Diagnostic\_PVC\_Num, PVC\_Num: 0~7

Item	Node Name	Attribute	Value	RW
1	Diagnostic_PVC0	EtherLanConn	{Pass, Skipped, Fail}	R
2	Diagnostic_PVC0	ADSLSyn	{Pass, Skipped, Fail}	R
3	Diagnostic_PVC0	ATMOAMSeg	{Pass, Skipped, Fail}	R

4	Diagnostic_PVC0	ATMOAMEnd 2End	{Pass, Skipped, Fail}	R
5	Diagnostic_PVC0	PingPriDNS	{Pass, Skipped, Fail}	R
6	Diagnostic_PVC0	PingYahoo	{Pass, Skipped, Fail}	R
7	Diagnostic_PVC0	PingOther	{Pass, Skipped, Fail}	R
8	Diagnostic_PVC0	ATM_LookBack	{Success, Fail}	R

### 3.27WebCurSet Node : Setting Webpage Pull-down Menu ID

Item	Node Name	Attribute	Value	RW
1	WebCurSet_Entry	<b>wan_pvc</b>	Get value from global variable record information,{0~7}	RW
2	WebCurSet_Entry	nat_pvc	Get value from global variable record information,{0~7}	RW
3	WebCurSet_Entry	diag_pvc	Get value from global variable record information,{0~7}	RW
4	WebCurSet_Entry	dev_pvc	Get value from global variable record information,{0~7}	RW
5	WebCurSet_Entry	virServ_id	Get value from global variable record information,{0~9}	RW
6	WebCurSet_Entry	ipAddr_id	Get value from global variable record information,{0~15}	RW
7	WebCurSet_Entry	route_id	Get value from global variable record information,{0~15}, system route id	RW
8	WebCurSet_Entry	lan_id	{0~N}	RW
9	WebCurSet_Entry	wlan_id	{0~3}	RW
10	WebCurSet_Entry	statis_type	{0: Ethernet, 1: ADSL, 2: WLAN}	RW
11	WebCurSet_Entry	qos_id	{0~N}	RW

12	WebCurSet_Entry	DyVlan_Index	Get value from global variable record information,{0~7}	RW
13	WebCurSet_Entry	IanAlias_id	0	RW
14	WebCurSet_Entry	dhcpd_id	Get value from global variable record information,{0~7}	RW
15	WebCurSet_Entry	portbind_id	Get value from global variable record information,{0~15}	RW
16	WebCurSet_Entry	acl_id	Get value from global variable record information,{0~15}	RW
17	WebCurSet_Entry	ipfilter_id	Get value from global variable record information,{0~15}	RW
18	WebCurSet_Entry	url_filter_id	Get value from global variable record information,{0~15}	RW
19	WebCurSet_Entry	autopvc_id	Get value from global variable record information,{0~15}	RW
20	WebCurSet_Entry	LineNum	{0,1}	RW
21	WebCurSet_Entry	CurrentAccess	{0~2}	RW

### 3.28Info Node : Retrieves Ethernet, ADSL, WLAN, ra0-ra3, nas0-nas7 Information

Ether : Retrieves Ethernet information

Item	Node Name	Attribute	Value	RW
1	Info_Ether	inOctets	Get value from /proc/tc3162/eth_stats	R
2	Info_Ether	inUnicastPkts	Get value from /proc/tc3162/eth_stats	R
3	Info_Ether	inMulticastPkts	Get value from /proc/tc3162/eth_stats	R



4	Info_Ether	outOctets	Get value from /proc/tc3162/eth_stats	R
5	Info_Ether	outUnicastPkts	Get value from /proc/tc3162/eth_stats	R
6	Info_Ether	outMulticastPkts	Get value from /proc/tc3162/eth_stats	R
7	Info_Ether	outErrors	Get value from /proc/tc3162/eth_stats	R
8	Info_Ether	txCollisionCnt	Get value from /proc/tc3162/eth_stats	R
9	Info_Ether	txUnderRunCnt	Get value from /proc/tc3162/eth_stats	R
10	Info_Ether	rxCrcErr	Get value from /proc/tc3162/eth_stats	R
11	Info_Ether	rxFrames	Get value from /proc/tc3162/eth_stats	R
12	Info_Ether	txFrames	Get value from /proc/tc3162/eth_stats	R
13	Info_Ether	ip	Get value from "ifconfig br0" ci-command	R
14	Info_Ether	mac	Get value from "ifconfig br0" ci-command	R
15	Info_Ether	status	{Up, NoLink}	R
16	Info_Ether	is4PortsEther	{Yes,No}	R
17	Info_Ether	isAliasIPSupport	{Yes,No}	R
18	Info_Ether	isDSLITESupported	{Yes,No}	R
19	Info_Ether	isIPv6Supported	{Yes,No}	R
20	Info_Ether	isUSBSupported	{Yes,No}	R

Adsl : Adsl: Retrieves ADSL information

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	Info_Adsl	outPkts	Get value from /proc/tc3162/tsarm_stats	R
2	Info_Adsl	inPkts	Get value from /proc/tc3162/tsarm_stats	R
3	Info_Adsl	outDiscards	Get value from /proc/tc3162/tsarm_stats	R
4	Info_Adsl	inDiscards	Get value from /proc/tc3162/tsarm_stats	R
5	Info_Adsl	fwVer	Get value from /proc/tc3162/adsl_fwver	R
6	Info_Adsl	lineState	Get value from /proc/tc3162/adsl_stats	R
7	Info_Adsl	Opmode	Get value from /proc/tc3162/adsl_stats	R
8	Info_Adsl	SNRMarginDown	Get value from /proc/tc3162/adsl_stats	R
9	Info_Adsl	SNRMarginUp	Get value from /proc/tc3162/adsl_stats	R
10	Info_Adsl	AttenDown	Get value from /proc/tc3162/adsl_stats	R
11	Info_Adsl	AttenUp	Get value from /proc/tc3162/adsl_stats	R
12	Info_Adsl	DataRateDown	Get value from /proc/tc3162/adsl_stats	R
13	Info_Adsl	DataRateUp	Get value from /proc/tc3162/adsl_stats	R
14	Info_Adsl	outBytes	Get value from /proc/tc3162/adsl_stats	R
15	Info_Adsl	inBytes	Get value from /proc/tc3162/adsl_stats	R
16	Info_Adsl	ATURID	Get value from /proc/tc3162/adsl_stats	R
17	Info_Adsl	ATUCID	Get value from /proc/tc3162/adsl_stats	R
18	Info_Adsl	WanListMode	Get value from /proc/tc3162/adsl_stats	R

19	Info_Adsl	FECDown	Get value from /proc/tc3162/adsl_stats	R
20	Info_Adsl	FECUP	Get value from /proc/tc3162/adsl_stats	R
21	Info_Adsl	HECDown	Get value from /proc/tc3162/adsl_stats	R
22	Info_Adsl	HECUP	Get value from /proc/tc3162/adsl_stats	R
23	Info_Adsl	CRCDown	Get value from /proc/tc3162/adsl_stats	R
24	Info_Adsl	CRCUp	Get value from /proc/tc3162/adsl_stats	R
25	Info_Adsl	ADSLUpTime	Get value from /proc/tc3162/adsl_stats	R
26	Info_Adsl	ADSLActiveTime	Get value from /proc/tc3162/adsl_stats	R
27	Info_Adsl	PowerDown	Get value from /proc/tc3162/adsl_stats	R
28	Info_Adsl	PowerUp	Get value from /proc/tc3162/adsl_stats	R
29	Info_Adsl	ATURID	Get value from /proc/tc3162/adsl_stats	R
30	Info_Adsl	ATUCID	Get value from /proc/tc3162/adsl_stats	R
31	Info_Adsl	AttainUp	Get value from /proc/tc3162/adsl_stats	R
32	Info_Adsl	AttainDown	Get value from /proc/tc3162/adsl_stats	R
33	Info_Adsl	ShowtimeStart	Get value from /proc/tc3162/adsl_stats	R
34	Info_Adsl	TotalStart	Get value from /proc/tc3162/adsl_stats	R
35	Info_Adsl	ATURANSIR v	Get value from /proc/tc3162/adsl_stats	R
36	Info_Adsl	ATUCANSIR v	Get value from /proc/tc3162/adsl_stats	R

37	Info_Adsl	ATURANSISStd	Get value from /proc/tc3162/adsl_stats	R
38	Info_Adsl	ATUCANSISStd	Get value from /proc/tc3162/adsl_stats	R
39	Info_Adsl	InterleaveDepth	Get value from /proc/tc3162/adsl_stats	R
40	Info_Adsl	ATURChanInterleaveDelay	Get value from /proc/tc3162/adsl_stats	R
41	Info_Adsl	ATURChanPrevTXRate	Get value from /proc/tc3162/adsl_stats	R
42	Info_Adsl	ATURChanCurrTXRate	Get value from /proc/tc3162/adsl_stats	R
43	Info_Adsl	ATURChanCrcBlkLen	Get value from /proc/tc3162/adsl_stats	R
44	Info_Adsl	mtenStandard	T1.413, G.DMT, G.lite, ADSL2, ADSL2+, G.dmt.bis, G.dmt.bisplus	R

WLAN : Retrieves WLAN information

Item	Node Name	Attribute	Value	RW
1	Info_WLan	isExist	{On: Existence, Off: no WLAN module, Error: WLAN EEPROM is empty, Diable:WLAN function disable }	R
2	Info_WLan	wlanTxFrames	Get value from "iwpriv ra0 stat" ci-command	R
3	Info_WLan	wlanTxErrFrames	Get value from "iwpriv ra0 stat" ci-command	R
4	Info_WLan	wlanTxDropFrames	Get value from "iwpriv ra0 stat" ci-command	R
5	Info_WLan	wlanRxFrames	Get value from "iwpriv ra0 stat" ci-command	R
6	Info_WLan	wlanRxErrFrames	Get value from "iwpriv ra0 stat" ci-command	R

7	Info_WLan	wlanRxDropFrames	Get value from "iwpriv ra0 stat" ci-command	R
8	Info_WLan	wlanTxRetryCount	Get value from "iwpriv ra0 stat" ci-command	R
9	Info_WLan	wlanRTSSuccessRcvCTS	Get value from "iwpriv ra0 stat" ci-command	R
10	Info_WLan	wlanRTSFailRcvCTS	Get value from "iwpriv ra0 stat" ci-command	R
11	Info_WLan	wlanSNR0	Get value from "iwpriv ra0 stat" ci-command	R
12	Info_WLan	wlanSNR1	Get value from "iwpriv ra0 stat" ci-command	R
13	Info_WLan	wlanSNR2	Get value from "iwpriv ra0 stat" ci-command	R
14	Info_WLan	wlanSelfPinCode	Get value from "iwpriv ra0 stat" ci-command	R
15	Info_WLan	wlanWPSStatus	Get value from "iwpriv ra0 stat" ci-command	R
16	Info_WLan	wlanWPStimerRunning	Get value from "iwpriv ra0 stat" ci-command	R
17	Info_WLan	DLS	{0,1}	R
18	Info_WLan	M2U	{0,1}	R
19	Info_WLan	DFS	{0,1}	R
20	Info_WLan	Carrier	{0,1}	R
21	Info_WLan	TXPath	{1,2}	R
22	Info_WLan	RXPath	{1,2}	R
23	Info_WLan	CurrentChannel	1~13	R

ra0-ra7: Retrieves some statistical information on wlan , retrieves from ifconfig

Item	Node Name	Attribute	Value	RW
1	Info_ra0	rxpackets	Get value from "ifconfig ra0"	R
2	Info_ra0	txpackets	Get value from "ifconfig ra0"	R
3	Info_ra0	rxbytes	Get value from "ifconfig ra0"	R
4	Info_ra0	txbytes	Get value from "ifconfig ra0"	R
5	Info_ra0	hwaddr	Get value from "ifconfig ra0"	R

eth0- eth3: Retrieves some statistical information on eth0 , retrieves from ifconfig

Item	Node Name	Attribute	Value	RW
1	Info_eth0	rxpackets	Get value from "ifconfig eth0"	R
2	Info_eth0	txpackets	Get value from "ifconfig eth0"	R
3	Info_eth0	rxbytes	Get value from "ifconfig eth0"	R
4	Info_eth0	txbytes	Get value from "ifconfig eth0"	R
5	Info_eth0	hwaddr	Get value from "ifconfig eth0"	R

nas0-nas7: Retrieves some statistical information on pvc, retrieves from ifconfig

Item	Node Name	Attribute	Value	RW
1	Info_nas0	rxpackets	Get value from "ifconfig ra0"	R
2	Info_nas0	txpackets	Get value from "ifconfig ra0"	R
3	Info_nas0	rxbytes	Get value from "ifconfig ra0"	R
4	Info_nas0	txbytes	Get value from "ifconfig ra0"	R
5	Info_nas0	hwaddr	Get value from "ifconfig ra0"	R

AutoPVC : Retrieves AutoPVC information

Item	Node Name	Attribute	Value	RW
1	Info_AutoPVC	State	Get value from global"autopvc_state"	R

Showtime: Retrieves statistical information on ADSL showtime, retrieves from adsl\_stats\_show

LastShowt: Retrieves statistical information on ADSL LastShowtime, retrieves from adsl\_stats\_last

Totaltime: Retrieves statistical information on ADSL total showtime, retrieves from adsl\_stats\_total

CurdayStShowt: Retrieves statistical information on ADSL Showtime 24h prior, retrieves from adsl\_stats\_curdaySt

QutHourStShowt: Retrieves the statistical information on ADSL Showtime 15m prior, retrieves from adsl\_stats\_quthourSt

Item	Node Name	Attribute	Value	RW
1	Info_Showtime	lineState	Get value from /proc/tc3162/ adsl_stats_show	R
2	Info_Showtime	ReceiveBlock	Get value from /proc/tc3162/ adsl_stats_show	R
3	Info_Showtime	TransmitBlock	Get value from /proc/tc3162/ adsl_stats_show	R
4	Info_Showtime	CellDelin	Get value from /proc/tc3162/ adsl_stats_show	R
5	Info_Showtime	LinkRetrain	Get value from /proc/tc3162/ adsl_stats_show	R
6	Info_Showtime	InitErrors	Get value from /proc/tc3162/ adsl_stats_show	R
7	Info_Showtime	InitTimeouts	Get value from /proc/tc3162/ adsl_stats_show	R
8	Info_Showtime	LossOfFramin g	Get value from /proc/tc3162/ adsl_stats_show	R
9	Info_Showtime	ErroredSecs	Get value from /proc/tc3162/ adsl_stats_show	R
10	Info_Showtime	SeveErrSecs	Get value from /proc/tc3162/ adsl_stats_show	R
11	Info_Showtime	FECDown	Get value from /proc/tc3162/ adsl_stats_show	R
12	Info_Showtime	FECUp	Get value from /proc/tc3162/ adsl_stats_show	R
13	Info_Showtime	CRCDown	Get value from /proc/tc3162/ adsl_stats_show	R
14	Info_Showtime	CRCUp	Get value from /proc/tc3162/ adsl_stats_show	R
15	Info_Showtime	HECDown	Get value from /proc/tc3162/ adsl_stats_show	R
16	Info_Showtime	ATURChanRx Blks	Get value from /proc/tc3162/ adsl_stats_show	R
17	Info_Showtime	ATURChanTx Blks	Get value from /proc/tc3162/ adsl_stats_show	R

18	Info_Showtime	ATURChanCorrectBlks	Get value from /proc/tc3162/adsl_stats_show	R
19	Info_Showtime	ATURChanUnCorrectBlks	Get value from /proc/tc3162/adsl_stats_show	R
20	Info_Showtime	CurdayStTimeElapsed	Get value from /proc/tc3162/adsl_stats_show	R
21	Info_Showtime	QutHrStATURChanTimeElapsed	Get value from /proc/tc3162/adsl_stats_show	R
22	Info_Showtime	PrevdayStATURChanMoniSecs	Get value from /proc/tc3162/adsl_stats_show	R
23	Info_Showtime	ATURChanPerfValidIntervals	Get value from /proc/tc3162/adsl_stats_show	R
24	Info_Showtime	ATURChanPerfInValidIntervals	Get value from /proc/tc3162/adsl_stats_show	R

### 3.29 DeviceInfo Node : Retrieves Device information

DeviceInfo\_PVCNum, PVCNum: PVC0~PVC7

Item	Node Name	Attribute	Value	RW
1	DeviceInfo	FwVer	Get value from fwver.conf file	R
2	DeviceInfo	cur_time	Get the value from "date" ci-command, ext: Thu May 22 15:06:11 UTC 2008	R
3	DeviceInfo_PVC0	Status	{0:Not Connected, 1:Connected}	R
4	DeviceInfo_PVC0	pppConnTime	Get value from /tmp/pppuptime-ppp0 file,{00:00:00:00: Date format}	R
5	DeviceInfo_PVC0	DispPPPconn	Check DMT is up and the PVC is set on PPP mode, {0:Not display, 1: Display ppp connection time}	R



6	DeviceInfo_PVC0	DispBtnType	Check DMT is up and the PVC is set on dynamic MER mode, {0:Not display renew and release button, 1: display renew and release button}	R
7	DeviceInfo_PVC0	DispConnBtnType	Check DMT is up and the PVC is set on dynamic MER mode, {0:Not display renew and release button, 1: display renew and release button}	R
8	DeviceInfo_PVC0	WanIP	{IP address format: 192.168.3.10}	R
9	DeviceInfo_PVC0	WanSubMask	{IP subnet mask address: 255.255.255.0}	R
10	DeviceInfo_PVC0	WanDefGW	{IP address format: 192.168.3.254}	R
11	DeviceInfo_PVC0	DNSIP	{IP address format: 192.168.3.10}	R
12	DeviceInfo_PVC0	connType	Connection type,{ Dynamic IP, Static IP, PPPoE, PPPoA, Bridge}	R
13	DeviceInf_PVC0	SECDNSIP	The second dns. gateway of current PVC	R
14	DeviceInfo_PVC0	gateway	Value: N/A or gateway ip {IP address format: 192.168.3.254}	R
15	DeviceInfo_PVC0	IP6Status	{0:Not Connected, 1:Connected}	R
16	DeviceInfo_PVC0	IP6pppConnTime	Get value from /tmp/pppuptime-ppp0 file,{00:00:00:00: Date format}	R

17	DeviceInfo_PVC0	IP6DispPPPconn	Check DMT is up and the PVC is set on PPP mode, {0:Not display, 1: Display ppp connection time}	R
18	DeviceInfo_PVC0	IP6DispConnBtnType	Check DMT is up and the PVC is set on dynamic MER mode, {0:Not display renew and release button, 1: display renew and release button}	R
19	DeviceInfo_PVC0	IP6DispBtnType	Check DMT is up and the PVC is set on dynamic MER mode, {0:Not display renew and release button, 1: display renew and release button}	R
20	DeviceInfo_PVC0	IP6WanIP	{IPv6 address format, eg: 2001:db8:1::1}	R
21	DeviceInfo_PVC0	IP6WanDefGW	{IPv6 address format, eg: 2001:db8:1::1}	R
22	DeviceInfo_PVC0	IP6DNSIP	{IPv6 address format, eg: 2001:db8:1::1}	R
23	DeviceInf_PVC0	IP6SECDNSIP	The second dns. {IPv6 address format, eg: 2001:db8:1::1}	R
24	DeviceInfo_PVC0	IP6gateway	{IPv6 address format, eg: 2001:db8:1::1}	R
25	DeviceInfo_PVC0	IP6PreDelegation	{IPv6 address format, eg: 2001:db8:1::1}	R
26	DeviceInfo_PVC0	IP6DSLiteAddr	{IPv6 address format, eg: 2001:db8:1::1}	R

### 3.30wifiMacTab Node: Retrieves some information from wifi mac table

wifiMacTab\_Common

wifiMacTab\_Entry{i} i = 0 ~ 64

Item	Node Name	Attribute	Value	RW
1	wifiMacTab_Common	valueChanged	{0,1} Get value from iwpriv ra{i} get_mac_table	R
2	wifiMacTab_Common	index	ssid index	RW
3	wifiMacTab_Common	NUM	The num under specific ssid.	R
4	wifiMacTab_Common	flag	{0, 1} 0: if want to get valueChanged 1: if want to get wifi mac table info.	RW
5	wifiMacTab_Entry 0	MAC	The mac addr of station connect to specific ssid.	R
6	wifiMacTab_Entry 0	IP	The IP addr of station connect to specific ssid.	R

### 3.31 TR069Attr Node: Stores tr069 parameter attribute information

Item	Node Name	Attribute	Value	RW
1	TR069Attr_SpeAttr	undetermine	{0,1,2}	RW
2	TR069Attr_Entry	undetermine	0~512 characters	RW

### 3.32 Cwmp Node: Stores Cwmp parameters

Item	Node Name	Attribute	Value	RW
1	Cwmp_Entry	Active	{Yes, No}	RW
2	Cwmp_Entry	periodInterval	0~n (s)	RW
3	Cwmp_Entry	periodActive	{Yes, No}	RW
4	Cwmp_Entry	acsUrl	The format is like <a href="http://122.193.99.166:80/">http://122.193.99.166:80/</a>	RW
5	Cwmp_Entry	acsUserName	0~255 characters	RW
6	Cwmp_Entry	acsPassword	0~255 characters	RW
7	Cwmp_Entry	conReqPath	The format is like /tr69	RW

8	Cwmp_Entry	conReqUserName	0~255 characters	RW
9	Cwmp_Entry	conReqPassword	0~255 characters	RW
10	Cwmp_Entry	CPHostName	The format is like 122.193.99.166	RW
11	Cwmp_Entry	CpePort	80	RW
12	Cwmp_Entry	CpePath	The format is like /comserver/node/tr069	RW
13	Cwmp_Entry	ManufacturerName	0~63 characters	RW
14	Cwmp_Entry	ManufacturerOUI	0~63 characters	RW
15	Cwmp_Entry	ProductClass	0~63 characters	RW
16	Cwmp_Entry	PrvCode	0~63 characters	RW
17	Cwmp_Entry	conReqPort	1~65535	RW
18	Cwmp_Entry	dbgflag	0~4	RW
19	Cwmp_Entry	Persistent_Data	0~255 characters	RW
20	Cwmp_Entry	WarrantyDate	xxxx-xx-xxTxx:xx:xxZ	RW
21	Cwmp_Entry	SerialNum	{0~z, less than 63 characters}	RW
22	Cwmp_Entry	FwUpgradeDelay	{0,1}	RW
23	Cwmp_Entry	FwUpgradeCfg	{0,1}	RW
24	Cwmp_Entry	RandomInform	{0,1}	RW
25	Cwmp_Entry	SendHostPortNumFlag	0	R
26	Cwmp_Entry	cwmpflags	String	R
27	Cwmp_Entry	tr069Commit	{0,1}	R
28	Cwmp_Entry	prdInformTime	Time formate eg.0001-01-01T00:00:00Z	RW
29	Cwmp_Entry	CusHWVersion	String	R

30	Cwmp_Entry	noValueTypeFlag	{0,1}	RW
31	Cwmp_Entry	arrayType	{0,1}	RW
32	Cwmp_Entry	Inform_Status	{0:initial status;1:acs url is invalid;2:ACS not response;3:connection interrupt;4:authentication failed;5:Success}	R
33	Cwmp_Entry	AcsConnStatus	{0:initial status;1:connection interrupt;2:authentication failed;3:Success}	R
34	Cwmp_tr143UDPEcho	Enable	{Yes, No}	RW
35	Cwmp_tr143UDPEcho	Interface	IGD.LANDevice.1.LANHost ConfigManagement.IPInterface.1 / IGD.WANDevice.1.WANConnectionDevice.{i}	RW
36	Cwmp_tr143UDPEcho	SourceIPAddress	Eg:192.168.1.10	RW
37	Cwmp_tr143UDPEcho	UDPPort	Eg:7	RW
38	Cwmp_tr143UDPEcho	EchoPlusEnabled	{Yes, No}	RW
39	Cwmp_tr143UDPEcho	EchoPlusSupported	{Yes}	R

### 3.33 SslICA Node: Stores SSL Ca parameters

SslICA\_Entry0~SslICA\_Entry4:Store CA File

Item	Node Name	Attribute	Value	RW
1	SslICA_Common	ValidIndex	1~4(MAX_CA_NUM)	RW
2	SslICA_Common	MaxCANum	4	RW
3	SslICA_Common	CurlIndex	0~4	R
4	SslICA_Entry0	FragNum	<a href="#">0~8</a>	RW
5	SslICA_Entry0	Frag0~8	0~255 characters	RW

### 3.34 CDSLDiagnostic: Temporary stores DSL diagnostic information

Item	Node Name	Attribute	Value	RW
1	CDSLDiagnostic_Common	DiagnosticState	Complete	R
2	CDSLDiagnostic_Common	ACTPSDds		R
3	CDSLDiagnostic_Common	ACTPSDus		R
4	CDSLDiagnostic_Common	ACTATPds		R
5	CDSLDiagnostic_Common	ACTATPus		R
6	CDSLDiagnostic_Common	HLINSCds		R
7	CDSLDiagnostic_Common	HLINpsds		R
8	CDSLDiagnostic_Common	QLNpsds		R
9	CDSLDiagnostic_Common	SNRpsds		R
10	CDSLDiagnostic_Common	BITSpds		R
11	CDSLDiagnostic_Common	GAINSpds		R

### 3.35 CATMDiagnostic: Temporary stores ATM diagnostic information

Item	Node Name	Attribute	Value	RW
1	CATMDiagnostic_Common	DiagnosticState	{“Complete”, “Error_internal”, “InProgress”, “Error_other”}	R
2	CATMDiagnostic_Common	DiagnosticTime	Unit:ms	R

### 3.36 SysInfo: Stores system parameters

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	SysInfo_Entry	SerialNum	{0~z, less than 63 characters}	RW
2	SysInfo_Entry	Description	{0~z, less than 255 characters}	RW
3	SysInfo_Entry	HWVer	{0~z, less than 63 characters}	RW
4	SysInfo_Entry	Vendor	0~31 characters	RW
5	SysInfo_Entry	ProductName	0~31characters	RW
6	SysInfo_Entry	SWVer	0~31characters	RW
7	SysInfo_Entry	FWVer	0~255characters	RW
8	SysInfo_Entry	PubDate	0~31characters	RW
9	SysInfo_Entry	BatchNum	String	RW
10	SysInfo_Entry	sysStartTime	Time formate	R

### 3.37LanAlias Node: Store Alias IP Parameters

Entry0 : LanAlias\_Entry0

Item	Node Name	Attribute	Value	RW
1	LanAlias_Entry0	Active	1~65535	RW
2	LanAlias_Entry0	IP	{IP address format: 192.168.3.10}	RW
3	LanAlias_Entry0	netmask	{IP subnet mask address: 255.255.255.0}	RW

### 3.38DhcpClient Node: Stores TR111 parameters

Item	Node Name	Attribute	Value	RW
1	DhcpClient_Info	Count	{Yes, No}	R
2	DhcpClient_Info	oui(0-N)	0~6 characters	R
3	DhcpClient_Info	sn(0-N)	0~64 characters	R
4	DhcpClient_Info	pclass(0-N)	0~64 characters	R

### 3.39DhcpClientLimit Node: Stores TR111 time parameters

Item	Node Name	Attribute	Value	RW
1	DhcpClientLimit_Entry	Time	1~65535	RW

### 3.40 VoIPCodecs Node: Stores cpe supported voip codec parameters (TR104)

Entry0 : VoIPCodecs\_Entry0; ..; Entry7 : VoIPCodecs\_Entry7.

Item	Node Name	Attribute	Value	RW
1	VoIPCodecs_Entry	codec	0~64 characters	R
2	VoIPCodecs_Entry	SIPPacketizationTime	0~64 characters	RW
3	VoIPCodecs_Entry	priority	1~65535	RW

### 3.41 IPInterface Node: Stores IPInterface Enabled status (null function)

Item	Node Name	Attribute	Value	RW
1	IPInterface_Entry	enable	{“0”, “1”}	R

### 3.42 Radvd Node: Stores Radvd status parameters

Item	Node Name	Attribute	Value	RW
1	Radvd_Entry	Enable	{“0”, “1”}	RW
2	Radvd_Entry	Mode	{“0”, “1”}	RW
3	Radvd_Entry	PrefixIPv6	{IPv6 address format, eg: 2001:db8:1::1}	RW
4	Radvd_Entry	Prefixv6Len	16~64	RW
5	Radvd_Entry	ValidLifetime	60~4294967295	RW
6	Radvd_Entry	PreferredLifetime	60~4294967295	RW

### 3.43 Dhcp6s Node: Stores Dhcp6s status parameters

Item	Node Name	Attribute	Value	RW
1	Dhcp6s_Entry	Enable	{“0”, “1”}	RW
2	Dhcp6s_Entry	Mode	{“0”, “1”}	RW
3	Dhcp6s_Entry	PDvar	{“0”, “1”}	RW



4	Dhcp6s_Entry	PrefixIPv6	{IPv6 address format, eg: 2001:db8:1::1}	RW
5	Dhcp6s_Entry	Prefixv6Len	16~64	RW
6	Dhcp6s_Entry	DNSserver	{IPv6 address format, eg: 2001:db8:1::1}	RW
7	Dhcp6s_Entry	SecDNSserver	{IPv6 address format, eg: 2001:db8:1::1}	RW
8	Dhcp6s_Entry	ValidLifetime	60~4294967295	RW
9	Dhcp6s_Entry	PreferredLifetime	60~4294967295	RW

### 3.44PortBind Node: Stores portbind information

Item	Node Name	Attribute	Value	RW
1	PortBind_Common	Active	{"Yes", "No"}	RW
2	PortBind_Entry	p0-p7	{"Yes", "No"}	RW
3	PortBind_Entry	e1-e4	{"Yes", "No"}	RW
4	PortBind_Entry	ra0-ra3	{"Yes", "No"}	RW

### 3.45IpAddrMapNode: store IP address mapping information

Item	Node Name	Attribute	Value	RW
1	IpAddrMap_PVC_Entry	RULETYPE	"One-to-One", "Many-to-One", "Many-to-Many-Overload", "Many-to-Many-No-Overload", "Server"	RW
2	IpAddrMap_PVC_Entry	LOCALSTART IP	x.x.x.x(x:0 - 255)	RW
3	IpAddrMap_PVC_Entry	LOCALENDIP	x.x.x.x(x:0 - 255)	RW
4	IpAddrMap_PVC_Entry	PUBLICSTAR TIP	x.x.x.x(x:0 - 255)	RW
5	IpAddrMap_PVC_Entry	PUBLICENDIP	x.x.x.x(x:0 - 255)	RW

### 3.46wifiMacTab Node: store wifi mac information

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	wifiMacTab_Common	NUM	number	R
2	wifiMacTab_Common	index	number	R
3	wifiMacTab_Common	valueChanged	{“0”, “1”}	R
4	wifiMacTab_Common	flag	{“0”, “1”}	R
5	wifiMacTab_Entry0	MAC	xx:xx:xx:xx:xx:xx(x:0-9)	R
6	wifiMacTab_Entry0	IP	x.x.x.x(x:0 - 255)	R

### 3.47wifiMacIp Node: store IP information of related to wifi mac

Item	Node Name	Attribute	Value	RW
1	wifiMacIp_Common	NUM	number	R
2	wifiMacTab_Entry0	MAC	xx:xx:xx:xx:xx:xx(x:0-9)	R
3	wifiMacTab_Entry0	IP	x.x.x.x(x:0 - 255)	R

### 3.48dynDisp Node: store dynamic display information

Item	Node Name	Attribute	Value	RW
1	dynDisp	CurPage	0 – 50 chars	R
2	dynDisp	MainMaskBit0	Xxxx xxxx{x:“0”, “1”}	R
3	dynDisp	CurMaskBit0	Xxxx xxxx{x:“0”, “1”}	R

### 3.49QoS Node: store QoS informtion

Item	Node Name	Attribute	Value	RW
1	QoS_Common	Active	{“Yes”, “No”}	RW
2	QoS_Common	Discipline	{“SP”...}	RW
3	QoS_Common	DisciplineFlag	{“Yes”, “No”}	RW
4	QoS_Common	WeightHighest	number	RW
5	QoS_Common	WeightHigh	number	RW
6	QoS_Common	WeightMedium	number	RW
7	QoS_Common	WeightLow	number	RW

8	QoS_Common	Drop	{“TAIL”...}	RW
9	QoS_Common	Prio	{“DSCP”...}	RW
10	QoS_Common	Queue1	{“Yes”, “No”}	RW
11	QoS_Common	Queue2	{“Yes”, “No”}	RW
12	QoS_Common	Queue3	{“Yes”, “No”}	RW
13	QoS_Common	Queue4	{“Yes”, “No”}	RW
14	QoS_Common	QueueSW1	{“Yes”, “No”}	RW
15	QoS_Common	QueueSW2	{“Yes”, “No”}	RW
16	QoS_Common	QueueSW3	{“Yes”, “No”}	RW
17	QoS_Common	QueueSW4	{“Yes”, “No”}	RW
18	QoS_Entry0	Active	{“Yes”, “No”}	RW

### 3.50VoIPBasic Node : store basic VoIP information

VoIPBasic\_Common, store publicVoIP information

VoIPBasic\_Entry0~4, store information of entry 0~4

Item	Node Name	Attribute	Value	RW
1	VoIPBasic_Common	SIPProxyEnable	{“Yes”, “No”}	RW
2	VoIPBasic_Common	SIPProxyAddr	x.x.x.x{x:0-255}	RW
	VoIPBasic_Common	SIPProxyPort	0 - 65535	RW
2	VoIPBasic_Common	SIPOutboundProxyEnable	{“Yes”, “No”}	RW
3	VoIPBasic_Common	SIPOutboundProxyAddr	x.x.x.x{x:0-255}	RW
4	VoIPBasic_Common	SIPOutboundProxyPort	0 - 65535	RW
5	VoIPBasic_Common	SIPTransportProtocol	{“TCP”, “UDP”}	RW
6	VoIPBasic_Common	LocalSIPPort	0 - 65535	RW
7	VoIPBasic_Common	LocalRTTPort	0 - 65535	RW
8	VoIPBasic_Entry0	SIPEnable	{“Yes”, “No”}	RW

9	VoIPBasic_Entry0	SIPDiaplayName	0 – 50 chars	RW
10	VoIPBasic_Entry0	SIPAuthenticationName	0 – 50 chars	RW

### 3.51 VoIPCallCtrl Node : store VoIP Call control information

Item	Node Name	Attribute	Value	RW
1	VoIPCallCtrl_Entry	SIPCallWaitingEnable	{“Yes”, “No”}	RW
2	VoIPCallCtrl_Entry	SIPCallForwardingEnable	{“Yes”, “No”}	RW

### 3.52 VoIPMedia Node : store VoIP media information

VoIPMedia\_Common, store public VoIP media information

VoIPMedia\_Entry0~1 store information of entry 0~1

Item	Node Name	Attribute	Value	RW
1	VoIPMedia_Common	T38Enable	{“Yes”, “No”}	RW
2	VoIPMedia_Common	EchoCancellationEnable	{“Yes”, “No”}	RW
3	VoIPMedia_Common	VAD	{“Yes”, “No”}	RW
4	VoIPMedia_Entry0	SIPSupportedCodec	{“G.711 a-law”, “G.711 u-law”, “G.729”, “G.723”}	RW
5	VoIPMedia_Entry0	SIPPacketizationTime	number	RW

### 3.53 VoIPCodecs Node : store VoIP CODEC information

VoIPCodecs\_Entry0~1, store information of entry 0~1

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	VoIPCodecs_Entry 0	codec	{“G.711 a-law”, “G.711 u-law”, “G.729”, “G.726 - 16”, “G.726 - 24”, “G.726 - 32”, “G.726 - 40”}	RW
2	VoIPCodecs_Entry 0	SIPPacketization Time	number	RW
3	VoIPCodecs_Entry 0	proirity	number	RW

### 3.54VoIPLog Node : store VoIP Log information

Item	Node Name	Attribute	Value	RW
1	VoIPLog_Entry0	LogEnable	{“Yes”, “No”}	RW
2	VoIPLog_Entry0	LogServerIP	x.x.x.x{x:0-255}	RW
3	VoIPLog_Entry0	LogServerPort	number	RW
4	VoIPLog_Entry0	LogDisplay	0 - 50 chars	RW

### 3.55VoIPAdvanced Node : store VoIP advances configuration information

VoIPAdvanced\_Common, store public VoIP advancedconfiguration information

VoIPAdvanced\_Entry0~1 store information of entry 0~1

Item	Node Name	Attribute	Value	RW
1	VoIPAdvanced_Co mmon	VoIPRegion	{“CHN-China” ...}	RW
2	VoIPAdvanced_Co mmon	RegitrationExpire	number	RW
3	VoIPAdvanced_Co mmon	RetransInterval	number	RW
4	VoIPAdvanced _Entry0	VoiceVolumnListen	number	RW
5	VoIPAdvanced _Entry0	VoiceVolumnSpeak	number	RW

### 3.56VoIPDigitMap Node : store VoIP DigitMap information

Item	Node Name	Attribute	Value	RW
------	-----------	-----------	-------	----

1	VolPDigitMap_Entry	DigitMapEnable	{“1”,“0”}	RW
2	VolPDigitMap_Entry	DigitMap	“x.”	RW
3	VolPDigitMap_Entry	DailTimeout	number	RW
4	VolPDigitMap_Entry	DigitTimeout	number	RW

### 3.57 SysLog Node : store syslog information

Item	Node Name	Attribute	Value	RW
1	Syslog_Entry	remoteSyslogEnable	{“0”,“1”}	RW
2	Syslog_Entry	remoteHost	Ip address	RW
3	Syslog_Entry	Log_Path	Log path (default : /tmp/var/log/messages )	RW

### 3.58 ALG Switch Node: store information for application layer

item	Node Name	Attribute	Value	RW
1	ALGSwitch_Entry	FTPSW	{“on”,enable; “off”,disable}	RW
2	ALGSwitch_Entry	SIPSW	{“on”,enable; “off”,disable}	RW
3	ALGSwitch_Entry	H323SW	{“on”,enable; “off”,disable}	RW
4	ALGSwitch_Entry	RSTPSW	{“on”,enable; “off”,disable}	RW
5	ALGSwitch_Entry	L2TPSW	{“on”,enable; “off”,disable}	RW
6	ALGSwitch_Entry	IPSECSW	{“on”,enable; “off”,disable}	RW

### 3.59 DyVLAN Node: store dynamic VLAN information

DyVLAN\_Common, store public dynamic vlan information

DyVLAN\_Entry0~4: store information of entry 0~4

item	Node Name	Attribute	Value	RW
1	DyVLAN_Common	Active	{“Yes”,“No”}	RW
2	DyVLAN_Common	Type	{“Yes”,“No”}	RW
3	DyVLAN_Entry0	STB_OUI	string	RW
4	DyVLAN_Entry0	STB_PVC	{“0” – “7”}	RW
5	DyVLAN_Entry0	Voip_PVC	{“0” – “7”}	RW

### 3.60GUI Node: store GUI information

GUI\_Entry0~2:store information of entry 0~2

item	Node Name	Attribute	Value	RW
1	GUI_Entry0	portTriggering	{"support"}	R
2	GUI_Entry0	natSessionsPerIP	{"support"}	R
3	GUI_Entry0	timeOfDay	{"support"}	R
4	GUI_Entry0	Custom	{"C9"}	R
5	GUI_Entry0	WMM	{"1"}	R
6	GUI_Entry0	WPS	{"1"}	R
7	GUI_Entry0	WDS	{"1"}	R
8	GUI_Entry0	Schedule	{"1"}	R
9	GUI_Entry0	802.1x	{"1"}	R
10	GUI_Entry0	tmp_UserFlag	{"0", "1", "2"}	R

### 3.61GUITemp Node: store dynamic GUI temp information

DyVLAN\_Entry0~31:information of entry 0~31

item	Node Name	Attribute	Value	RW
1	GUITemp_Entry0	urlfilterTRLLine	string	R

### 3.62Schedule Node: store dynamic schedule information

Schedule\_Common,store public schedule information

Schedule\_Entry0~34 : store information of entry 0~34

item	Node Name	Attribute	Value	RW
1	Schedule_Common	Active	{"Yes", "No"}	R
2	Schedule_Entry0	Day	number	R
3	Schedule_Entry0	OnOff	{"0", "1"}	R
4	Schedule_Entry0	StartHour	"0" – "23"	R
5	Schedule_Entry0	StartMin	"00" – "59"	R
6	Schedule_Entry0	EndHour	"0" – "23"	R
7	Schedule_Entry0	EndMin	"00" – "59"	R

### 3.63portTriggering Node: store dynamic portTriggering information

portTriggering\_Common, store public sportTriggering information

portTriggering\_Entry0~7 store information of entry 0~7

item	Node Name	Attribute	Value	RW
1	portTriggering_setting	canUseNum	number	RW
2	portTriggering_Entry0	Name	0 – 50 chars	RW
3	portTriggering_Entry0	NameTmp	number	RW
4	portTriggering_Entry0	TSPort	number	RW
5	portTriggering_Entry0	TSPortTmp	number	RW
6	portTriggering_Entry0	TEPort	number	RW
7	portTriggering_Entry0	TEPortTmp	number	RW
8	portTriggering_Entry0	TProtocol	{“TCP”, “UDP”..}	RW
9	portTriggering_Entry0	TProtocolTmp	{“TCP”, “UDP”..}	RW
10	portTriggering_Entry0	OSPort	number	RW
11	portTriggering_Entry0	OSPortTmp	number	RW
12	portTriggering_Entry0	OEPort	number	RW
13	portTriggering_Entry0	OEPortTmp	{“TCP”, “UDP”..}	RW
14	portTriggering_Entry0	OProtocol	{“TCP”, “UDP”..}	RW
15	portTriggering_Entry0	OProtocolTmp	{“TCP”, “UDP”..}	RW

### 3.64timeOfDay Node: store dynamic timeofday information

item	Node Name	Attribute	Value	RW
1	timeOfDay_Common	Active	{“1”, “0”}	RW



2	timeOfDay_Entry	all	{“on”, “off”}	RW
3	timeOfDay_Entry	msn	{“on”, “off”}	RW
4	timeOfDay_Entry	http	{“on”, “off”}	RW
5	timeOfDay_Entry	ftp	{“on”, “off”}	RW
6	timeOfDay_Entry	url	{“on”, “off”}	RW
7	timeOfDay_Entry	clock	number	RW

### 3.65 keywordTime Node: store dynamic keywordTime information

keywordTime\_Entry0~7 : store information of entry 0~15

item	Node Name	Attribute	Value	RW
1	keywordTime_Entry 0	keyword	0 – 64 chars	RW

### 3.66 natSessions Node: store dynamic nat sessions information

natSessions\_Common, store public natSessions information

natSessions\_Entry0~9: store information of entry 0~9

item	Node Name	Attribute	Value	RW
1	natSessions_setting	MaxNatSessions	0 – 64 chars	RW
2	natSessions_setting	SessionMode	{“0”, “1”}	RW
3	natSessions_setting	SessionPerUser	number	RW
4	natSessions_setting	add_num	{“0”, “1” ... “full”}	RW
5	natSessions_Entry0	IPAddress	x.x.x.x{x:0 – 255}	RW
6	natSessions_Entry0	sessionNum	number	RW

### 3.67 DMS Node: store dynamic DMS information

item	Node Name	Attribute	Value	RW
1	DMS_Entry	Enable	{“Yes”, “No”}	RW
2	DMS_Basic	IPAddress	x.x.x.x{x:0 – 255}	RW
3	DMS_Basic	PortNumber	number	RW
4	DMS_Basic	NumPath	number	RW
5	DMS_Basic	Path	string	RW
6	DMS_Basic	MaxContent	number	RW

7	DMS_Device	Manufacturer	0~63 characters	RW
8	DMS_Device	ManufacturerUr l	string	RW
9	DMS_Device	ModelName	0~63 characters	RW
10	DMS_Device	ModelNumber	number	RW
11	DMS_Device	ModelUrl	string	RW
12	DMS_Device	SerialNumber	0~63 characters	RW

### 3.68signature Node : store signature information

item	Node Name	Attribute	Value	RW
1	signature_Entry	Status	{“Enable”, “Disable”}	RW

### 3.69DefaultWan Node : store default setting of WAN PVC

Item	Node Name	Attribute	Value	RW
1	DefaultWan_Entr y	Active	{Yes, No}	RW
2	DefaultWan_Entr y	VPI	0~255	RW
3	DefaultWan_Entr y	VCI	1~65535	RW
4	DefaultWan_Entr y	QOS	{ubr, cbr, rt-vbr , nrt-vbr}	RW
5	DefaultWan_Entr y	PCR	0~5500	RW
6	DefaultWan_Entr y	SCR	0~5500(SCR < PCR)	RW
7	DefaultWan_Entr y	MBS	0~655325	RW
8	DefaultWan_Entr y	ISP	0~3 {0:Dynamic IP Address, 1:Static IP Address, 2: PPPoA/PPPoE, 3: Bridge Mode}	RW

9	DefaultWan_Entry	ENCAP	{1483 Bridged IP LLC, 1483 Bridged IP VC-Mux, 1483 Routed IP LLC(IPoA), 1483 Routed IP VC-Mux, PPPoE LLC, PPPoE VC-Mux, PPPoA LLC, PPPoA VC-Mux}	RW
10	DefaultWan_Entry	RIPVERSION	{RIP1, RIP2-B, RIP2-M}	RW
11	DefaultWan_Entry	PPPo6Mode	{“0”, “1”}	RW
12	DefaultWan_Entry	PPPo6PD	{Yes, No}	RW
13	DefaultWan_Entry	MLDproxy	{Yes, No}	RW
14	DefaultWan_Entry	PVCScanReserved	{Yes, No}	RW

### 3.70LanHost : Setting the Host List Contents Dhcp Client List and Static Client List

HostNum: store active host client number value

Item	Node Name	Attribute	Value	RW
1	LanHost	HostNum	0~254	RW

[1] LanHost\_Entry0~253:store active host client list items

Item	Node Name	Attribute	Value	RW
1	LanHost_Entry0	IP	X.X.X.X(IP Address format)	RW
2	LanHost_Entry0	MAC	XX:XX:XX:XX:XX:XX(MAC Address format)	RW
3	LanHost_Entry0	ExpireDay	0~n	RW
4	LanHost_Entry0	ExpireTime	xx:xx:xx(time format)	RW
5	LanHost_Entry0	Hostname	0~64 characters	RW
6	LanHost_Entry0	AddressSrc	{DHCP, Static, AutoIP}	RW
7	LanHost_Entry0	InterfaceType	{Ethernet, USB, 802.11, HomePNA, HomePlug, MoCA, Other}	RW

8	LanHost_Entry0	Active	{“0”, “1”}	RW
---	----------------	--------	------------	----

### 3.71 VendorCfgFile : Setting the Vendor Config File Info

- VendorCfgFile\_Common: store the vendor config file number

Item	Node Name	Attribute	Value	RW
1	VendorCfgFile_Common	VendorCfgFile Num	0-2	RW

- VendorCfgFile\_Entry0~1:store the vendor config file list items

Item	Node Name	Attribute	Value	RW
1	VendorCfgFile_Entry0	Name	0~64 characters	RW
2	VendorCfgFile_Entry0	Version	0~16 characters	RW
3	VendorCfgFile_Entry0	Date	xxxx-xx-xxTxx:xx:xx(dateTime format) 0~256 characters	RW
3	VendorCfgFile_Entry0	Description	“The customer setting of CPE”	RW

### 3.72 LanguageSwitch Node: Node for multi-language switch

item	Node Name	Attribute	Value	RW
1	LanguageSwitch_Entry	Type	{“0”, “1”}	RW

## 4 ASP Support Functions Table Listing

---

In order to allow user to set system parameters via the Web interface, this system uses the Boa Web Server, version V0.94. In Boa, to provide development personnel with the ease of accessing the system parameters, we have designed several ASP functions. With the use of these functions, users can easily achieve their motive of accessing the system parameters. In addition, we have also provided functions to ease the writing of dynamic webpages. An introduction on the ASP functions provided by us is written below.

Item	Function	Explanation
1	<code>void asp_write(char *string)</code>	<p><b>PURPOSE:</b> Specify the contents of the output string to the browser</p> <p><b>PARAMETER:</b> string: for output string</p> <p><b>EXAMPLE:</b> If you wish to add the line "Hello World." to the HTML document, call function <code>asp_write("&lt;br&gt;Hello World.");</code></p>

2	void request_form(char *string)	<p>PURPOSE: Retrieves the value of the HTML component specified in the HTTP Request</p> <p>PARAMETER: string: HTMLcomponent name</p> <p>EXAMPLE: Assuming that there is a field named "Name" on the HTML screen of the submitted Request, on the page receiving the Request, in order to retrieve the value of this field, call function request_form("Name");</p>
---	---------------------------------	--

3	<pre>void tcWebApi_set(char *nodeName, char *attr, char* value)</pre>	<p>PURPOSE: Sets certain system Attributes</p> <p>PARAMETER: nodeName: is the parent node path of the set attribute but does not include the root node (ROMFILE) attr: for setting Attribute name value: specifys the updated system parameter value; the HTML component name must be specified; unable to enter string values directly; will set a certain HTML component value to a certain attribute of the configure node.</p> <p>EXAMPLE: Assuming there is a Username field on the HTML screen, and you wish to write the value of this field to the USERNAME attribute of the child node PVCO under the configure node WAN, call function tcWebApi_set("WAN_PVC 0", "USERNAME", "Username")</p>
---	---	--

4	void tcWebApi_unset(char *nodeName)	<p>PURPOSE: Deletes certain system parameters from configure node</p> <p>PARAMETER: nodeName: name of parameter for deletion</p> <p>EXAMPLE: Assuming that today we are going to delete the PVC0 under configure node WAN, call function tcWebApi_unset("WAN_PVC0");</p> <p>NOTE: tcWebApi_unset can only delete the child node for certain configure node, but it is not allowed to delete the entire configure node (for example: the PVC0 under WAN may be deleted, but WAN is a configure node and hence cannot be deleted). Furthermore, it is also not allowed to delete individual attributes (for example: not allowed to delete USERNAME under PVC0, under WAN).</p>
---	-------------------------------------	---



5	<pre>void tcWebApi_get(char *nodeName, char     *attr, char* type)</pre>	<p>PURPOSE: Retrieves certain system parameter values from configure node</p> <p>PARAMETER: nodeName: is the parent node path for the defined attribute but does not include the root node (ROMFILE) attr: for setting Attribute name type: separated into 2 types: "s": abbreviation for show, displays corresponding system parameter at client's browser; "h": abbreviation for hidden, retrieves corresponding system parameter but does not display at client's browser.</p> <p>EXAMPLE: If you wish to display the attribute USERNAME under child node PVC0 under configure node WAN, onto the HTML screen, you will have to call function tcWebApi_get("WAN_PVC0", "USERNAME", "s"); if you do not wish to display onto the screen, then call function tcWebApi_get("WAN_PVC0", "USERNAME", "h");</p>
---	--	--

6	void tcWebApi_commit(char *nodeName)	<p><b>PURPOSE:</b> Allows the initial configuration of the information to take effect and reactivates the node service. This program will call the function execute of the configure node. This will be elaborated later.</p> <p><b>PARAMETER:</b> nodeName: configure node name</p> <p><b>EXAMPLE:</b> Assuming that we have already modified certain attributes in WAN, at this point we have to call function tcWebApi_commit("WAN") to allow the system to apply the new attribute values immediately.</p>
---	--------------------------------------	--

7	void tcWebApi_save(void)	<p><b>PURPOSE:</b> After calling tcWebApi_set or tcWebApi_unset, the modified values are not saved in flash; the purpose of this function is to write the current system parameters to the ROMFILE region that is in flash.</p> <p><b>PARAMETER:</b> none</p> <p><b>EXAMPLE:</b> None</p>
8	int tcWebApi_CurrentDefaultRoute(void)	<p><b>PURPOSE:</b> Returns the default route number in the current system</p> <p><b>PARAMETER:</b> None</p> <p><b>EXAMPLE:</b> None</p>

Other than the above-mentioned functions, we have also provided basic conditional expressions, at present we have provided if, else elseif conditional expressions, please refer to the below program paragraph:

```
<%
if request_Form("QOS_Flag") = "0" then
...
elseif Request_Form("QOS_Flag")="1" then
...
elseif Request_Form("QOS_Flag")="2" then
...
end if
```

»

Presently, we have not developed loop function as loop functions in webpage programming are mostly used for drawing tables and currently we are using CGI to write these related pages. Loop functions will be provided in the future, to ease the development of webpage programming. When developing programs, all users, please take special note that we have yet to provide a function to declare variables.

## 5 CFG Manager adds a new configure node

This chapter introduces the architecture of the CFG Manager and details of related contents. In order for the programs on the webpage to execute correctly after adding a configure note for a new function, how should user modify and set CFG Manager.

CFG Manager Files Tree Diagram

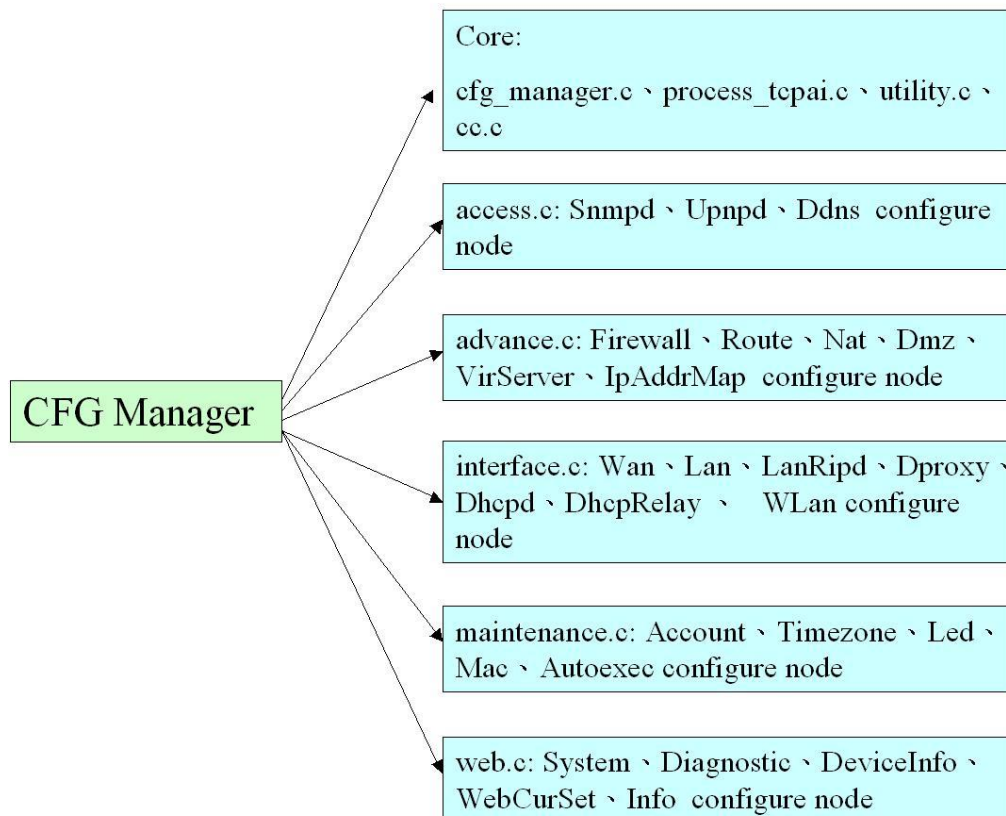


Diagram 2

CFG Manager is the main program in the system; containing the information for all of the configure nodes and having the ability to activate and stop the services corresponding to each configure node. When we have a new request and needs to add a new webpage but is unable to expand the existing configure node, at this point we will need to add a new configure node. This chapter introduces how to add a new configure node in CFG Manager.

### 5.1 ROMFILE(romfile.cfg) Format

All system parameters (configure node) are recorded inside the ROMFILE. ROMFILE uses XML format to record the data of all of the current system parameters (configure node). As shown in Diagram 3 below, "ROMFILE" is tag as the root node and below there are 25 child nodes at present, and these child nodes are what we call

configure node. Based on the names of each individual configure node, the information of its corresponding system parameters will be recorded. For example, configure node LAN will record CPE LAN IP address information.

```

- <ROMFILE>
+ <Wan>
+ <Lan>
+ <Dhcpd>
+ <Dproxy>
+ <DhcpRelay>
+ <WLan>
+ <Firewall>
+ <Route>
+ <Dmz>
+ <VirServer>
+ <Adsl>
+ <Snmpd>
+ <Upnpd>
+ <Ddns>
+ <Account>
+ <Timezone>
+ <Mac>
+ <Led>
+ <Autoexec>
+ <System>
+ <Diagnostic>
+ <WebCurSet>
+ <Info>
+ <ACL>
+ <IpMacFilter>
</ROMFILE>

```

Diagram 3

The data structure for Configure node is classified into four main categories. Each configure node must fulfil either one of these four structures. Likewise, if user wishes to add a new configure node, it must also fulfil one of these data structure.

- Category 1:

configure node contains only one XML Element named Entry. All attributes belonging to this configure node will be recorded as a XML Attribute under Entry.

Format:

```

<ConfigureNodeName>
  <Entry Attribute1="value1" Attribute2="value2" .../>
</ConfigureNodeName>

```

Example: Dhcpd

```

<Dhcpd>
  <Entry start="10.0.0.6" pool_count="0" lease="0" />
</Dhcpd>

```

- Category 2:

configure node contains several XML Element defined as Entry with a serial number suffix (Entry0, Entry1...). Each individual Entry has the same XML Attribute fields to allow for individual settings for each different Entry.

Format:

```
<ConfigureNodeName>
  <Entry0 Attribute1="value1" Attribute2="value2" ... />
  <Entry1 Attribute1="value1" Attribute2="value2" ... />
  ...
  <EntryN Attribute1="value1" Attribute2="value2" ... />
</ConfigureNodeName>
```

Example: Lan

```
<Lan>
  <Dhcp type="0" />
  <Entry0 IP="192.168.10.1" netmask="255.255.255.0" />
  <Entry1 IP="192.168.10.2" netmask="255.255.255.0" />
</Lan>
```

- Category 3:

configure node contains several Element defined as PVC with a serial number suffix (PVC0 、PVC1...). Individual PVC has the same XML Attribute fields to allow for individual settings to be performed on different PVC.

Format:

2 level configure node

```
<ConfigureNodeName>
  <PVC0 Attribute1="value1" Attribute2="value2" ... />
  <PVC1 Attribute1="value1" Attribute2="value2" ... />
  ...
  <PVC7 Attribute1="value1" Attribute2="value2" .../>
</ConfigureNodeName>
```

3 level configure node

```
<ConfigureNodeName>
  <PVC0>
    <Entry0 Attribute1="value1" Attribute2="value2" .../>
    <Entry1 Attribute1="value1" Attribute2="value2" .../>
    ...
    <EntryN Attribute1="value1" Attribute2="value2" .../>
  </PVC0>
  ...
  <PVC7>
    <Entry0 Attribute1="value1" Attribute2="value2" .../>
    <Entry1 Attribute1="value1" Attribute2="value2" .../>
    ...
    <EntryN Attribute1="value1" Attribute2="value2" .../>
  </PVC7>
</ConfigureNodeName>
```

```
</PVC7>
</ConfigureNodeName>
```

Example:

2 level configure node : Dmz

```
<Dmz>
  <PVC0 Active="Yes" DMZ_IP="10.10.10.1" />
  <PVC1 Active="No" DMZ_IP="10.10.10.2" />
</Dmz>
```

3 level configure node : VirServer

```
<VirServer>
  <PVC0>
    <Entry0 STARTPORT="" ENDPORT="" LOCALIP="" />
  </PVC0>
  <PVC1>
    <Entry0 STARTPORT="" ENDPORT="" LOCALIP="" />
    ...
    <Entry9 STARTPORT="" ENDPORT="" LOCALIP="" />
  </PVC1>
  <PVC2>
    <Entry0 STARTPORT="" ENDPORT="" LOCALIP="" />
    <Entry1 STARTPORT="" ENDPORT="" LOCALIP="" />
  </PVC2>
</VirServer>
```

#### ● Category 4:

configure node contains common attributes, which are listed as self-defined XML Element in the configure node. These attributes belong specifically to configure node and they do not belong to any other Entry. Readers may refer to introductions in the adv\_table at the back regarding method of use for category 4.

Format:

```
< ConfigureNodeName >
  <CommonAttribute Attribute1="value1" Attribute2="value2".../>
  <Entry0 Attribute1="value1" Attribute2="value2" />
  ...
  <EntryN Attribute1="value1" Attribute2="value2" />
</ ConfigureNodeName >
```

Example 1: WLAN

```
<Wlan>
  <Common APOn="1" Country="TAIWAN" .../>
  <Entry0 SSID="Trendchip_AP" HideSSID="0" ... />
  <Entry1 SSID="Trendchip_AP" HideSSID="0" ... />
</Wlan>
```



#### Example 2: Lan

```
<Lan>
  <Dhcp type="0" />
  <Ripd RIPVERSION="RIP1" DIRECTION="None" />
  <Entry0 IP="10.10.10.5" netmask="255.255.255.0" />
</Lan>
```

The ROMFILE introduced above is saved within a certain region inside flash. After the ROMFILE has been modified, the system will save the new ROMFILE in a certain region inside flash. Later, the CFG Manager will be able to access and use the parameters from this region. In our system, there is also a separate file, /userfs/romfile.cfg containing configure node default values. When an error occurs while reading the ROMFILE under flash, the default values from this file will be retrieved instead.

## 5.2 Adding a new configure node into romfile.cfg

After understanding the classification of the current configure nodes, based on a new configure node that you wish to add, select one of the four categories above, and add this system data into the file romfile.cfg. The romfile.cfg file is located at /userfs/romfile.cfg and contains the default system parameters.

## 5.3 Understanding the data structure of cfg\_node\_t

After completing the settings of the configure node inside ROMFILE, next is to add relevant programming codes to the program, so that the CFG Manager will operate correctly. Because the information of the newly added configure node does not exist in the system originally, hence we need to make minute modifications to the programs to allow the program to complete our requests. Below shows our additions to the program:

- Relevant information added to configure node, are configure node type, configure node name, certain Entry or PVC.
- For each configure node, there are 4 corresponding actions (read, write, verify, execute) that must be declared and invoked in the programs.
- If configure node belongs to category 4 (common attributes) of the above-mentioned, the names of the common attributes must be stated in the programs.

File Location: cfg\_manager.h

cfg\_node\_t data structure shown below:

```
typedef struct
cfg_node_s{
    char name[MAX_NODE_NAME];
    /*
```

Type Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADv	Type							MAX							

```

*/
#define SINGLE_TYPE    (1 << 8)
#define PVC_TYPE       (1 << 9)
#define ENTRY_TYPE     (1 << 10)
#define WEB_TYPE       (1 << 11)
#define UPDATE_TYPE    (1 << 12)
#define ADV_FLAG       (1 << 15)
    unsigned short int    type;
    char **adv_table;
    char **merge_table;
    int (*cfg_read)(mxml_node_t *top, char name[][MAX_NODE_NAME]);
    int (*cfg_write)(mxml_node_t *top, mxml_node_t *node);
    int (*cfg_verify)(mxml_node_t *node);
    int (*cfg_execute)(mxml_node_t *top, char name[][MAX_NODE_NAME]);
    int (*cfg_boot)(mxml_node_t *top);
}cfg_node_t;

```

1. name: configure node name, maximum length of name must not exceed 15 characters.
2. type:
  - 0th~7th bits: defines the total number of identical Entry. Ex. 00000101 denotes 5 identical Entry.
  - 8th~15th bits: defines the configure node category
    - 8th bit(SINGLE\_TYPE): When this bit is set to 1, denotes only one Entry, which is configure node category 1 as mentioned previously.
    - 9th bit(PVC\_TYPE): When this bit is set to 1, denotes that this configure node has a PVC structure, which is configure node category 3 as mentioned previously.
    - 10th bit(ENTRY\_TYPE): When this bit is set to 1, denotes that this configure node has an Entry structure, which is configure node category 2 as mentioned previously.
    - 11th bit(WEB\_TYPE): To allow the system to be able to support Web applications, we need to add some configure node to record the relevant information from the Web. When this bit is set to 1, it denotes that the configure node is added for Web Interface. Presently, within our system, we have System, WebCurSet, Diagnostic and Info, four configure nodes that belong to WEB\_TYPE. For example, the configure node WebCurSet is to allow us to set the rule or PVC for the pull-down menu on the webpage (WebCurSet is further explained in the introduction of the merge\_table below).
    - 12<sup>th</sup> bit (UPDATE\_TYPE): Because certain system parameter values recorded by configure node are stored in other files within the system, these values might have been updated frequently, hence when

we are retrieving these values from the configure node tree, we need to first access the current set values in order to update the values in the configure node tree. This way, we can avoid using old data. When this bit is set to 1, it denotes that when accessing configure node system data via TCAPI, we must first update the configure node tree data immediately. Presently, within our system, there are three configure nodes, Diagnostic, Info and Route that belong to UPDATE\_TYPE. For example, UnicastPkts under the node Info\_Ether denotes the Ethernet Unicast Packet Size received by CPE, and the data is recorded in the file /proc/tc3162/eth\_stats. Since the received packet size might increase at any time, hence when we are retrieving this parameter value, we must re-access file /proc/tc3162/eth\_stats in order to update the configure node tree data, so as to avoid retrieving an old value.

- 15th bit(ADV\_FLAG): This is a flag defined for configure node category 4. When this bit is set to 1, it denotes that the child node of this configure node is saved under the name of the child configure node that was defined by the user. For example: Wlan configure node besides having child nodes named Entry0~Entry3, it also has a child configure node defined as “Common”.
- adv\_table character array: this is prepared for the configure node of the above-mentioned category 4 (common attributes) and it stores the array of the user-defined child configure node names. For example: the common attribute for Lan configure node is known as Dhcp, hence it shows “Common” in the adv\_table. Likewise, Wan configure node shows “Common”, while IpAddrMap shows “Server”.

Using LAN as an example, the below program codes must be added to cfg\_manager.c, so that the system will know the names of the common attributes for Lan.

```
char *
lan_adv_table[MAX_NODE_NAME]=
{
    "Dhcp", "",
};
```

- merge\_table character array: When we wish to retrieve or set the Attribute value for certain configure node via the CFG\_Manager, the parent node path and the Attribute name must be provided. For example: If you wish to set the attribute USERNAME of the child node PVC0 under the configure node WAN, we must provide the parent node path (WAN\_PVC0) and the Attribute name (USERNAME). At present, the ASP of the system does not provide functions for string format, hence we are not able to combine strings into a complete path (which means WAN\_PVC and 0 cannot combine into WAN\_PVC0 with a ASP function) and then transferred to CFG\_Manager, hence CFG Manager has provided a special configure node known as WebCurSet, and uses this attribute to handle the combination for the parent node path. This method is as follows: Using the above USERNAME as an example, the parent node path for

USERNAME is WAN\_PVC0, therefore we will locate the attribute wan\_pvc from the configure node WebCurSet and set its value to 0, then we will know that the path for the parent node is WAN\_PVC0. We will need to record the string “wan\_pvc” in the merge\_table of the configure node WAN so as to inform us that the attribute wan\_pvc in the configure node WebCurSet has the information for the complete path.

```
/*
inside cfg_manager, must first declare the merge table for WAN, method
is as follows:
*/
char * wan_merge_table[MAX_NODE_NAME] = {"wan_pvc", "", };

/*
next, inside web.c must add the corresponding merge table names
*/
int
webCurSet_read(mxml_node_t *top, char name[][MAX_NODE_NAME]) {
    char webCurSet_attr[][16]=
    {
        {"wan_pvc"}, {"nat_pvc"},
        {"diag_pvc"}, {"dev_pvc"},
        {"virServ_id"}, {"ipAddr_id"},
        {"route_id"}, {"statis_type"},
        {"lan_id"}, {"wlan_id"},
        {"qos_id"},
        {"acl_id"},
        {"ipfilter_id"},
        {"url_filter_id"},
        {""}
    };
};
```

For each configure node, there are five corresponding operations (read, write, verify, execute, boot), and the notation format is configureNodeName\_operationName (for example, the write operation for WAN is noted as wan\_write). After adding the data for the configure node, we need to define the operation for the newly added configure node. Below is a detailed introduction on the five types of operations:

- **cfg\_node\_read:** Because configure node might already have another file in the system for storing the parameter values, hence the need to provide a function to read this corresponding system file and then base on this file to update the data in the configure node tree. This function is provided in the operation “read”.

Example: The function info\_read in Info configure node reads the system file to obtain the current system parameter. If this configure node does not exist in the recorded configure node tree in the current CFG Manager, then Info configure node is added, whereas if it exist then the attributes values of Info configure node are updated directly.

- `cfg_node_write`: Writes the system parameters recorded in the configure node back to the corresponding system file.

- Example: The system parameters recorded in Dhcpd configure node, generates a setting file (`udhcpd.conf`) via the `dhcpd_write` function and the information in this setting file will be provided for use by the dhcpd server.

- 
- `cfg_node_verify`: Verifys whether the system parameters recorded in configure node are valid values.

Example: Verifys whether the system parameters recorded in Dhcpd configure node are valid contents.

- 
- `cfg_node_execute`: activates/stops node service.

Example: Dhcpd configure node activates/stops the dhcpd server service via the function `dhcpd_execute`.

- 
- `cfg_node_boot` : is used to initialize this node during machine startup
- 
- Example: During the initial startup of the system, firewall configure node will determine via the function `firewall_boot`, whether user has activated the firewall prior to machine startup, if yes, it will take corresponding actions based on user's settings.

The above-mentioned five functions will have to be self-developed based on the requirements of the newly added configure node. And the operation information for the configure node must be added to `cfg_manager.c`, and the system will then operation normally.

## 5.4 To add a new configure node to the `all_cfg_node` Table

After understanding the data structure for `cfg_node_t`, we may start to add a new configure node. First we must add the init function to the `init_table` Table.

Init\_table Table

File location: `cfg_manager.c`

```
node_init
all_cfg_node[]=
{
    wan_init,
```

```

lan_init,
ripd_init,
dproxy_init,
dhcpd_init,
dhcpRelay_init,
wlan_init,
firewall_init,
route_init,
nat_init,//10
dmz_init,//11
virServ_init,
adsl_init,
snmpd_init,
upnpd_init,
ddns_init,
account_init,
timezone_init,
mac_init,
led_init,//20
autoexec_init,
system_init,
webCurSet_init,
diagnostic_init,
deviceInfo_init,
info_init,
acl_init,
ipfilter_init,
app_filter_init,
url_filter_init,//30
/*Node should be added before this
   line!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
NULL,
};

```

One point to note, any newly added init function must be added **in front of NULL** so that the function will then be called.

What this init function does is to declare the structure for a `cfg_node_t`, and all its contents must be filled up. Then call the function `register_cfg_node` under `cfg_manager` export, and the init task will be completed. Next `cfg manager` will automatically input into one of our table, the values of the dynamic structure which were just declared and entered and finally all operations performed by `cfg manger` to the node will be based on this table.

Below is an example for an init function (within `access.c`):

```

int
url_filter_init(void){
    int ret=0;
    cfg_node_t node={
        .name="UrlFilter",

```

```

.type=ADV_FLAG|ENTRY_TYPE|MAX_URL_FILTER_RULE,
.adv_table=url_filter_adv_table,
.merge_table=url_filter_merge_table,
.cfg_read=NULL,
.cfg_write=url_filter_write,
.cfg_verify=url_filter_verify,
.cfg_execute=url_filter_execute,
.cfg_boot=url_filter_boot,

};
ret=register_cfg_node(&node);
return ret;
}

```

## 5.5 Declare and Invoke for an added configure node operation

Since the application for the each configure node is different, user does not necessary need to invokes all five functions, but will only need to invoke the required function based on the characteristics of the configure node. Below is an individual introduction on the timing for the use of the five functions.

- **cfg\_read**

When the configure node type is WEBTYPE or UPDATETYPE, invoke cfg\_read.

- Timing for function call:

1. WEBTYPE: Because the data characteristics of web type is not recorded inside the flash ROMFILE block, CFG\_Manager will only call function cfg\_read during machine startup to read the data for the default parameters from the system and write to configure node tree, the information is provided for the webpage to function normally, hence we must invoke the function cfg\_read for the configure node for web type so as to allow CFG\_Manager to read the parameter value during machine startup.
2. UPDATETYPE: Data that are tag as update type denotes system parameter values that we have obtained might be update frequently. When retrieving values via TCAPI, to ensure that we have retrieved the most updated value, we must re-read the most updated parameter values, hence we must invoke the function cfg\_read for the update type configure node so that CFG\_Manager is able to read the most updated parameter value.

- **cfg\_write**

- Timing for function call:

cfg\_write is used when tcWebApi\_commit is being called. When tcWebApi\_commit is being called, it will first execute cfg\_write based on the current value settings and writes a file (setting file or shell) to allow cfg\_execute to execute the service using the new parameter. Take Dhcp for instance, dhcpd\_write will generate a

setting file (udhcpd.conf) and the information will be provided for use by the dhcpd server. Citing Adsl as another example, adsl\_write will read the data in configure node tree and then write a shell file for adsl\_execute to execute. There will be times when the system does not need to invoke cfg\_write, for example, when configure node System reactivates its service, it does not need to read new system parameters, hence it does not need to invoke System\_write.

- **cfg\_verify**

Verifies whether the system parameters recorded in configure node are valid values. Only when there is a need to validate the parameter for the self developed functions then it is necessary to invoke this function. Therefore, it is up to the user to evaluate the use of this function.

- Timing for function call:
- During machine startup, the system will call the function verify of each of the configure node, to verify the attribute within the ROMFILE, to confirm that the attribute value is valid. If the validation returns an error, CFG\_Manager will read the default values inside userfs/romfile.cfg to generate the configure node tree.
- When uploading ROMFILE via Firmware under webpage Maintenance, the system will call the function verify of all the configure node to verify all the attributes that requires verification. If an error is returned, the system will display an error message on the Firmware page.

- **cfg\_execute**

- Timing for function call:  
When tcWebApi\_commit is being called, system will apply the new parameter values and then reactive the corresponding services (executable shell or executable file), cfg\_execute is the function used for calling the corresponding services. For example, the function adsl\_execute is used to execute /etc/adsl.sh to activate Adsl. If we need to reactive the service for TCAPI, then we must invoke this function.

- **cfg\_boot**

- Timing for function call:

During the initial startup of the system, cfg manager will locate the node to invoke the cfg\_boot function, and call their boot functions, so these boot functions basically can be regarded as the execute function for this node during machine startup. The exception is that they will only be executed during machine startup and hence the contents will differ from the contents of cfg\_execute. For example, based on the contents of the node tree, the function firewall\_boot is used to determine whether firewall has been set to open prior to machine startup, and if it is yes, it will execute /usr/script/fw\_start.sh to activate firewall. This way, the status of the system will stay the same as before restarting machine.



In order to allow CFG Manager to operate the newly added configure node, we must declare and invoke configure node operation. Using QoS as an example here, on our webpage, the function QoSis is placed under Advanced Setup, within CFG Manager there are two corresponding programs, advanced.h and advanced.c. First, we must declare the four operations (qos\_read, qos\_write, qos\_verify, qos\_execute) in advanced.h.

Note: Based on the special characteristics of the configure node QoS, it does not need to declare and invoke the function qos\_read, it was declared and invoked here in order to help the user understand the program structure, so please take special note. Furthermore, qos currently does not have a boot function and therefore app\_filter\_boot instead is used for the boot example here.

```
int qos_read(mxml_node_t *top);
int qos_write(mxml_node_t *top, mxml_node_t *parant);
int qos_verify(mxml_node_t *node);
int qos_execute(mxml_node_t *top, char name[][MAX_NODE_NAME]);
int app_filter_boot(mxml_node_t *top);
```

After declaring, we will proceed to advanced.c to invoke these four functions individually. Below is a list of the structures for these five functions, which consist of input, output, program execution and machine startup procedure. Users may refer to these data to invoke their own functions. Within the programs, third party programs are used to help parse xml. Presently, we are using mxml for analysis and if user wishes to understand the use of mxml, please refer to [relevant documents](#).

```

    .

/*
 *description: Read system parameter from configuration file and      *
 record those information to mxml_node_t.
 *param *top: root of mxml_node_t structure
 *return int:
 *          success 0
 *          otherwise -1
 */
int qos_read(mxml_node_t *top) {
/*program codes to invoke*/
}

/*
 *description: Find element from mxml tree and then write system
 *              parameters to specific file(configure file orshell...).
 *param *top: root of mxml_node_t structure
 *param *parant: parant node of qos cfg node
 *return int:
 *          success 0
 *          otherwise -1
 */
int qos_write(mxml_node_t *top, mxml_node_t *parant) {
/*program codes to invoke*/
}

/*
```

```
*description: Check if the system parameters of cfg node element is *
valid or not.
*param *node: node that going to be validated
*return int:
*           success 0
*           otherwise -1
*/
int qos_verify(mxml_node_t *node) {
/*program codes to invoke*/
}

/*
*description: Execute a service belongs to specific configure node.
*param *top: root of mxml_node_t structure
*param name[][]: record path of every configure node entries
*              (ex. {{WAN_PVC0},{WAN_PVC1}}...)
*return int:
*           success 0
*           otherwise -1
*/
int qos_execute(mxml_node_t *top, char name[][MAX_NODE_NAME]) {
/*program codes to invoke*/
}

/*
** app_filter_boot
**
** descriptions:
**     the boot function of app filter.
** parameters:
**     top:          Specify the root of mxml_node_t structure.
** return:
**     Success:      0
**
**
*/
int
app_filter_boot(mxml_node_t *top) {
/*program codes to invoke*/
}
```

Earlier, when introducing the ASP functions supported by us, we have mentioned the function `tcWebApi_commit`, its main usage is when the system parameter has been changed for a particular system configure node, it allows the new parameter to take effect. `tcWebApi_commit` achieves this step by executing the `execute` function of the corresponding configure node.

After completing the above function, in future when we call on CFG Manager via TCAPI to perform operations on configure node, CFG Manager will then be able to execute the actions as expected by us.

## 6 Webpage Development Lessons

---

Important Items on Webpage Writing:

- Webpage will attempt to retrieve the system parameters via the tcWebapi\_get ASP function, if these parameters currently do not contain any set values, system will return the string “N/A”, and user will be able to identify that the string “N/A” means that currently there is no such system parameter and will set the webpage components to the default options. For example: The Radio Button for WAN Default Gateway will be set to “No” if the obtained return string is “N/A”.
- Please take special note that when a certain field of the webpage is to be set into the system, the tcWebApi\_set ASP function must be used to input simultaneously the system parameter field name and the HTML screen field name, instead of directly entering the changing values.
- When delete is provided on the page, configure node information must be deleted via the tcWebApi\_unset ASP function, CFG Manager can only delete the child node for a certain configure node, but is not allowed to delete the entire configure node (for example, the PVC0 under WAN can be deleted, but WAN cannot be deleted as it is a configure node), and furthermore, it is also not allowed to delete individual attributes (for example, it is not allowed to delete USERNAME under PVC0 under WAN).
- Current ASP functions do not provide loop functions, such as loop, while loop, do while loop. When writing webpage, loop function is mostly used for drawing tables, if looping is required during your webpage development, please change to CGI to develop your webpage. Detailed information could be referenced from pages with relevant examples (home\_pvclist.cgi、virsvr\_table.cgi、ipaddr\_table.cgi).
- Drop-down menus is a combination of the option and select components. When there are too many option selections, it is recommended to use the Option function provided by javascript to overcome the lengthy computations of the ASP function. Detailed information can be reference from the country options at home\_wlan.asp channel.
- When a Check Box is not checked, HTML will not return this object, hence take special note when invoking Check Box, when it is possible that a Check Box on the page might be changed from tick to untick, it is recommended to set a Hidden component to record the current status of Check Box so that it will still return the object. This way, regardless whether a Check Box has been ticked or not, it is ensured that its status would be returned to the Server.

We have provided some webpage examples below. For a simple introduction, in the examples below, we will focus on some of the commonly seen components of a

HTML webpage, such as Text Field, Radio Button, Check Box and Select and by coordinating with the characteristics of our system, show how to set the webpage values into our system and how to retrieve values from the system and then display them onto the webpage.

- Example: SNMP Page

HTML Element : Radio Button

Diagram 4

Radio Button is to allow us to select one option out of a selection, like the region circled in red, in the above diagram, displays the options where we select whether to activate SNMP.

When opening this webpage, at first, the system will check the value of the attribute Activate under configure node. If the value is “Yes”, the circle option on the screen for Activated would have been selected, whereas if the value is “No”, the circle option on the screen for Deactivated would have been selected.

Below is the information for the SNMP configure node:

```
<Snmpd>
  <Entry Active="No" rocommunity="" rwcommunity="" />
</Snmpd>
```

Below is the information for the HTML page:

```
<td class="tabdata">
<INPUT TYPE="RADIO" NAME="SNMP_active" VALUE="Yes"
onClick="snmpOff(0)"
<% If TCWebApi_get("Snmpd_Entry","Active","h") = "Yes" then
asp_Write("checked") end if%> >
<font color="#000000">Activated</font>
<INPUT TYPE="RADIO" NAME="SNMP_active" VALUE="No"
onClick="snmpOff(1)" <%
If TCWebApi_get("Snmpd_Entry","Active","h") = "No" then
asp_Write("checked") end if
If TCWebApi_get("Snmpd_Entry","Active","h") = "N/A" then
asp_Write("checked") end if
%> >
```

```
<font color="#000000">Deactivated</font>
</td>
```

The action that we have to perform is to check inside ROMFILE, the attribute value for whether to activate SNMP, and then select the corresponding Radio Button, and hence we need to add ASP function codes (those shown in red above) to the webpage.

In order to retrieve the setting value inside ROMFILE, we must use the function TCWebApi\_get. TCWebApi\_get has been introduced in an earlier chapter. Here we will cite a simple example. The attribute that we are retrieving is Active; from configure node SNMP, the path for Active starts as SNMP\_Entry (underscore as separator), hence the first two parameters of TCWebApi\_get are SNMP\_Entry and Active, and since we do not need to display these values on screen, the last parameter will be set to "h"(hidden), the complete program code will be: TCWebApi\_get("Snmpd\_Entry","Active","h"), hence, the value for Active will be retrieved.

At the Radio Button for Activate, after we have retrieved the value for Active from the ROMFILE, we will then determine whether this value is "Yes", if it is so, then use asp\_Write("checked") to click this Radio Button, and if it is not so, then no action will be taken. Likewise, at the Radio Button for Deactivate, the actions will be the same, except that the determining value is "No". Otherwise, if the retrieved value is "N/A", it denotes that currently there is no set value and the system will select Deactivate by default.

- Example: SNMP Page

HTML Element : Text Field

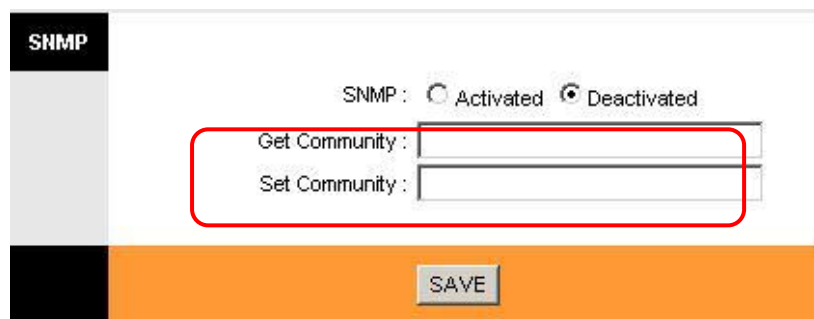


Diagram 5

Text Field allows us to enter string values. On our webpage, if there is a value for this attribute, it will be displayed at the corresponding field upon opening the webpage.

When setting this attribute, the value in this field on the webpage must be retrieved and then set into the system ROMFILE.

Below is the information for the SNMP configure node:

```
<Snmpd>
  <Entry Active="No" rocommunity="" rwcommunity="" />
</Snmpd>
```

Below is the information for the HTML page:

```
<td class="tabdata">
<INPUT TYPE="TEXT" NAME="SNMP_get" SIZE="30" MAXLENGTH="15" VALUE="
<% If TCWebApi_get("Snmpd_Entry","rocommunity","h") <> "N/A" then
  TCWebApi_get("Snmpd_Entry","rocommunity","s") end if%>" >
</td>
```

In this example, the action that we are performing is to check the attribute value for rocommunity under SNMP, if there is a value it should be displayed on the page, if there is no value, no action is taken and the field will be displayed as blank.

Likewise, we are using the function TCWebApi\_get. The attribute that we are retrieving is rocommunity; from configure node SNMP, the path for rocommunity starts as SNMP\_Entry (underscore as separator), hence the first two parameters of TCWebApi\_get are SNMP\_Entry and rocommunity. Since we have to retrieve the value to determine if it is a null value, it is not necessary to display this action on the screen, so the last parameter will be set to “h”(hidden), the complete program code will be: TCWebApi\_get("Snmpd\_Entry","rocommunity","h")

Following, if the retrieved value is not “N/A”, we will retrieve the value for rocommunity again and then displayed it on the webpage. Likewise, the first two parameters are SNMP\_Entry and rocommunity, and since we intend to display the retrieved values on the webpage, the third parameter should be set to “s” (show), the complete program code will be: TCWebApi\_get("Snmpd\_Entry","rocommunity","s")

- Example: SNMP Page

HTML Element : Button



Diagram 6

When we are developing a webpage, very often we have to use a button to submit or cancel our setting values, so here we will introduce some techniques for designing buttons.

When we send out a Request to the Web Server, at the same time the actions that we want to perform must be clearly indicated with the Request, or else the system will have no way of knowing the behaviour of the execution. Using the below program code as an example

```
<td width="440" class="orange">
<INPUT TYPE="HIDDEN" NAME="Snpflag" VALUE="">
<INPUT TYPE="BUTTON" NAME="SaveBtn" VALUE="SAVE"
onClick="SNMPsave();">
</td>
```

The line written in red is a HTML hidden component, and the value of this component will not be displayed on the webpage. We may inform the system of the nature of our executing actions by setting the value of this component. For example, value 0 denotes delete; value 1 denotes save.

The green line in the above program code indicates: when we click on the button, the webpage will execute the JavaScript program section in SNMPsave(). Below is a section of the SNMPsave() program code:

```
function SNMPsave()
{
...
document.forms[0].Snpflag.value=1;
document.SNMP_form.submit();
}
```

We can see that SNMPsave() initially sets the value for the hidden field for Snpflag to 1(save), then submits the webpage.

At the webpage (location specified by Action in Form) receiving this Request, we will process the Request that was sent. Below is the program that we have processed after receiving the Request:

```
<%  
/*received snmpflag value is 1, denotes to add a new data*/  
If Request_Form("Snmpflag")="1" then  
    TCWebApi_set("Snmpd_Entry","Active","SNMP_active")  
    TCWebApi_set("Snmpd_Entry","rocommunity","SNMP_get")  
    TCWebApi_set("Snmpd_Entry","rwcommunity","SNMP_set")  
    TCWebApi_commit("Snmpd_Entry")  
End If  
>
```

First, line 1 of the program determines the value of the field Snmpflag, since the value sent is 1, it means to save the value that is on the webpage, hence program lines 2~4 will start to set the values to the corresponding fields in the ROMFILE.

Here we shall use the function TCWebApi\_set, and similar to the function TCWebApi\_get, we must specify the parent node path and the attribute name. Taking line 2 as an example, the attribute name is Active, parent node path is SNMP\_Entry, as for the third parameter, it will be the name in the corresponding HTML field, that was entered on the webpage, which in this case is SNMP\_active. Please take note that TCWebApi\_set cannot be set directly, hence TCWebApi\_set("Snmpd\_Entry","Active","Yes") would be a wrong application. Please take care when developing programs.

Last, line 5 of the program calls TCWebApi\_commit("Snmpd\_Entry") to allow the set value to take effect so that the system will be able to apply the new attribute values.

- Example: Lan Page

HTML Element : Drop-down menus



IP Address : 10.10.10.1

IP Subnet Mask : 255.255.255.0

Dynamic Route : RIP1 Direction None

DHCP : ☒ Disabled ☐ Enabled ☐ Relay

SAVE CANCEL

Diagram 7

Drop-down menus allows us to choose one option out of multiple options. On our webpage, if the system has recorded the value for a certain option, then upon the opening of the webpage, the option would have been selected. If there is no value set in the system, it may temporarily select a default option. When we are setting the value back to the system, we must also write back the value of the selected option.

Below is the information for the LanRipd configure node:

```
<LanRipd>
  <Entry RIPVERSION="RIP1" DIRECTION="None" />
</LanRipd>
```

Below is the information for the HTML page:

```
<td class="tabdata">
<SELECT NAME="lan_RIP" SIZE="1">
<OPTION <% if tcWebApi_get("LanRipd_Entry","RIPVERSION","h") = "RIP1"
then asp_Write("selected") end if %>>RIP1
<OPTION <% if tcWebApi_get("LanRipd_Entry","RIPVERSION","h") = "RIP2-
B" then asp_Write("selected") end if %>>RIP2-B
<OPTION <% if tcWebApi_get("LanRipd_Entry","RIPVERSION","h") = "RIP2-
M" then asp_Write("selected") end if %>>RIP2-M
</SELECT>
<font color="#000000">Direction</font>
<SELECT NAME="lan_RIP_Dir" SIZE="1">
<OPTION <% if tcWebApi_get("LanRipd_Entry","DIRECTION","h") = "None"
then asp_Write("selected") end if %>>None
<OPTION <% if tcWebApi_get("LanRipd_Entry","DIRECTION","h") = "Both"
then asp_Write("selected") end if %>>Both
<OPTION <% if tcWebApi_get("LanRipd_Entry","DIRECTION","h") = "IN
Only" then asp_Write("selected") end if %>>IN Only
<OPTION <% if tcWebApi_get("LanRipd_Entry","DIRECTION","h") = "OUT
Only" then asp_Write("selected") end if %>>OUT Only
</SELECT>
</td>
```

In this example, we will check the attribute recorded in the system, and then display the corresponding option on the screen drop-down menu.

For each option, we must determine whether the value denoting the option is the same as the value recorded in the system, and if they are the same, it denotes that the value of this option should be displayed in the drop-down menu. Taking the option RIP1 (lines in red) as an example, we will use the function TCWebApi\_get, and since the attribute that we are retrieving is RIPVERSION; from configure node LanRipd, the path for RIPVERSION starts as LanRipd\_Entry (underscore as separator), hence the first two parameters of TCWebApi\_get are LanRipd\_Entry and RIPVERSION and since we do not need to display these value on screen, the third parameter will be set to "h"(hidden). After inputting these three parameters, the complete program code will be TCWebApi\_get("LanRipd\_Entry","RIPVERSION","h")

After retrieving the value for RIPVERSION, we will determine whether this value is the same as the value of our option, if it is so, use asp\_Write("selected") to set this option, if it is not the case, no action is taken (reference the red section in the top program). This way, the drop-down menu in the page will display the set values of the current options of the system.

#### 1. Example: VLAN Page

HTML Element : Check box

Active : ☒ Yes ☐ No

VLAN ID :  (Decimal)

Tagged	0	1	2	3	4	5	6	7
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

ATM VCs :

Port #	0	1	2	3	4	5	6	7
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Ethernet :

Tagged	1	2	3	4
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Diagram 8

When multiple options are to be repeatedly selected, use the Check Box interface. Need to note that when Check Box is not tick, the object will not be returned to the server, and hence the Server will not know that the tick for the Check Box has been cancelled. Therefore, we recommend to use Hidden component to record the current Check Box status, when user needs to set or retrieve the Check Box value, the value for Hidden component should also be

set together to ensure that regardless whether Check Box is ticked or not, Server is able to manage the status of these Check Box.

Below is the information for the Vlan configure node:

```
<Vlan>
  <Function ACTIVE="No"/>
    <Entry0 ACTIVE="No" VLAN_ID="0" VC_PORT0="0" VC_PORT1="0"
VC_PORT2="0" VC_PORT3="0" VC_PORT4="0" VC_PORT5="0" VC_PORT6="0"
VC_PORT7="0" VC_TAG0="0" VC_TAG1="0" VC_TAG2="0" VC_TAG3="0"
VC_TAG4="0" VC_TAG5="0" VC_TAG6="0" VC_TAG7="0" E_PORT0="0"
E_PORT1="0" E_PORT2="0" E_PORT3="0" E_TAG0="0" E_TAG1="0" E_TAG2="0"
E_TAG3="0" />
</Vlan>
```

Here we have created many Attributes to accept the Check Box status that are returned to the Server from the Client.

Below is the information for the HTML page:

```
<INPUT TYPE="HIDDEN" NAME="A_Port0" VALUE="<% if
tcWebApi_get("Vlan_Entry","VC_PORT0","h") = "1" then asp_Write("1")
else asp_Write("0") end if %>" >
```

We have created a Hidden component called A\_Port0 to record the client's VC\_PORT0 Check Box status and the default value is based on the current CV\_PORT0 setting at the Server.

Next, we have written a JavaScript function, prior to sending the page back to the Server, activating this function will sweep through all the Check Box, and if a Check Box is ticked, it will set the corresponding hidden component value to 1, and if it is not ticked the value will be set to 0.

```
function checkcheckbox()
{
  var form = document.VlanGroup_form;
  for(i = 7; i <= 30; i++){
    if(form.elements[i].checked)
      form.elements[i + 28].value=1;
    else
      form.elements[i + 28].value=0;
  }
}
```

}

Finally, the ASP function at the Server will be changed into using this Hidden component to set the Attribute value:

```
tcWebApi_set("Vlan_Entry","VC_PORT0","A_Port0")
```

After the carrying out the above actions, Check Box will be able to accurately return the status regardless whether it is ticked or not ticked.

## 7 Multi-language Webpage Development

### Notes at beginning:

1. There're two buttons used for language switching on right corner of homepage if we turned on multi-language option. We can use these two buttons to switch shown language.
2. The node for multi-language is "LanguageSwitch", English is chosen with configuration "string1.conf" when the Type=1 and the other language correspond with "string2.conf" when Type=2.
  - 1) Two string file storing shown string(with two languages) should be prepared previously. The format of string file should be "key=[value]", the Key represents attribute name and value will be the string content, which will be used to modify shown content of the webpage later.
3. Note points for configure string file:
  - 1) The maximum line number in string file is 3529 currently, which is set as array size "hashsize" and this array should be enlarged if the effective line number is closed to the maximum number
  - 2) The maximum length of "Key"(string name) will be 32 bytes and the "value"(string content) should not be larger than 580 bytes. There're restrictions for single and double quote: if the webpage call with `alert("<%tcWebApi_get('String','Key','s')%>");` the "value" definition for this key should avoid double quote, similarly, single quote should be avoided if the call point has already included " '" (unless there's escape character).
4. Command of "tcapi get String\_Entry XXX" will be ineffective since the multi-language mechanism is implemented in "boa"
5. For multi-language related webpage resource, we have created subdirectory of 'ml' under Router( same level with "tc").
6. As the hash table can not avoid key conflict, we have involved a new tool "mlCheckHash" to check if the "key" in string1.conf has duplication when buildimage.

### 7.1 Webpage Modification

We will introduce the procedure of webpage modification when add new webpage or new element, includes the following parts:

- Add or modify asp webpage
- Add or modify "js" under /html
- Add or modify "cgi" embedded in asp file
- Modify the asp in GUITemp node.

- 1) Add or modify asp webpage

The procedure should be :

- a) Add elements for multi-language on asp.

- Call function of tcWebApi\_get("String\_Entry","newOption","s") to show dedicated string on webpage.
  - Modify the /apps/private/etc\_script/string%d.conf respectively.
- To explicity illustrate how to add in asp, we will show with the following example : if we wanna add a "MldSnoop" on Interface->lan:

Alias IP Subnet Mask : 0.0.0.0

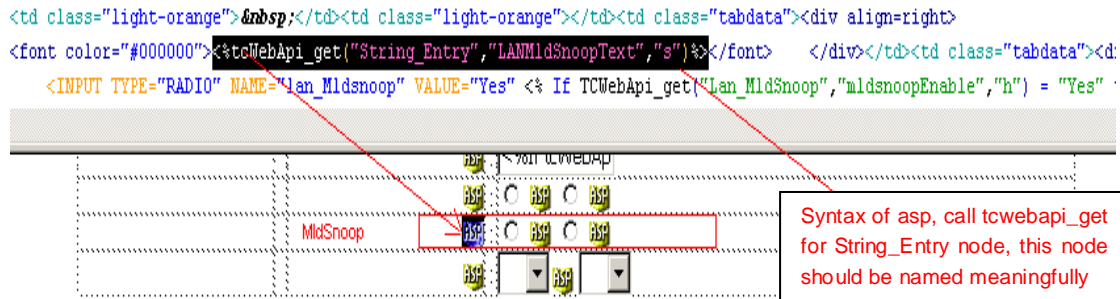
IcmpSnoop : ☐ Activated ☒ Deactivated

MldSnoop : ☐ Activated ☒ Deactivated

Dynamic Route : RIP1 Direction None

用户看到的最终效果

- Add new element on asp like this:



- Call with tcWebApi\_get("String\_Entry"," LanMldSnoopText","s") and the LanMldSnoopText should be defined in string%d.txt and the radio buttons of "Activated" and "Deactivated" represent as means.
- Add definition of Key and value in /apps/private/etc\_script/string1.conf and /apps/private/etc\_script/string2.conf, which should have the same "Key". Since this was invented for Lan page, we will add the associate elements under home\_lan frame(the location selection won't effect the application but for easy maintain).

```
872 ;
873 ; adv home_lan
874 ;
875 LANLocalIPText=Router Local IP
```

```
1001 LANAliasIPInfoText= (0.0.0.0 means to close the alias ip)
1002 LANMldSnoopText=MldSnoop
1003 ;
```

Attribute name defined by user

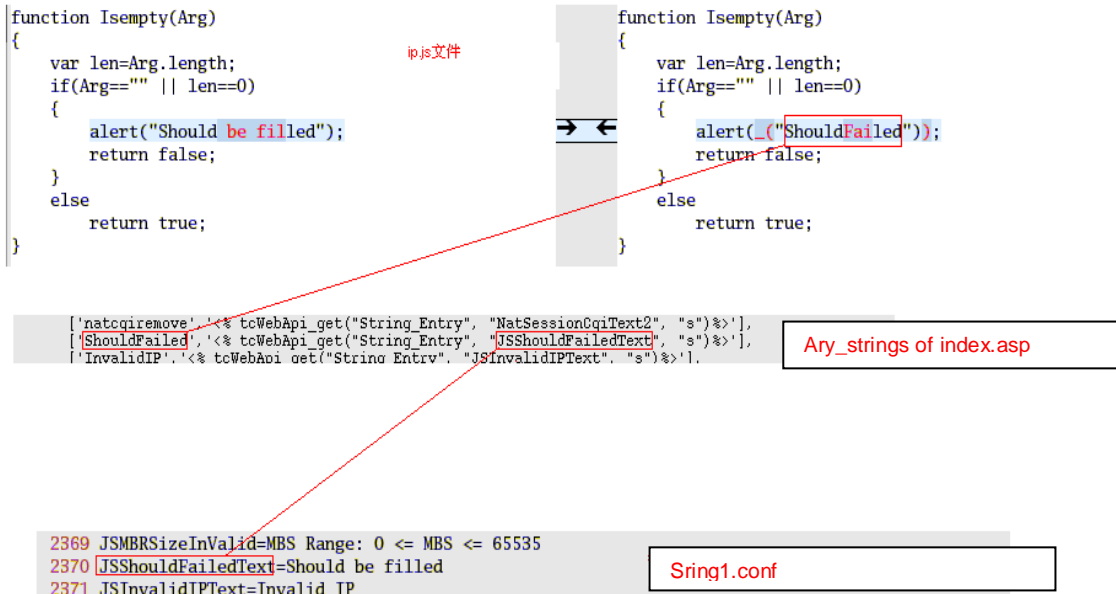
Words shown on the webpage

- Add or modify "js" under /html

If the independent js files contain items need multi-language support, for example, alert, confirmation, etc, we should modify the js as following:

- Modify the index.asp
- Modify /apps/private/etc\_script/string%d.conf
- Modify the asp file call this js.

For instance, if we wanna add a check() function in ip.js, which will induce some instruction with multi-language.



- Modify the "instruction" of alert(\_("key")) in index.asp.
- Add the mirror string for ary\_strings variable in index.asp, which represents the actual information. The true attribute name will be defined in string.conf.
- Modify the string1.conf and string2.conf for value assignment of key.
- Modify the home\_lan.asp which will call ip.js. plus, general.js will be called as the figure.

```

<script type="text/javascript" src="/style.css" tppabs="/style.css" type="text/javascript"></script>
<script type="text/javascript" src="/general.js"></script>
<script type="text/javascript" src="/js1.js"></script>
<script type="text/javascript" src="/val.js"></script>
<script type="text/javascript" src="/pvc.js"></script>
<script type="text/javascript" src="/ip.js"></script>
<script type="text/javascript" src="/ip_new.js"></script>
    
```

Tips: all asp pages calling ip.js will call general.js as well, other wise, js error will occur. Similarly , the general.js should be call by the asp if some new js files added to the asp.

Currently, the maximum length of "Key"(string name) will be 32 bytes and the "value"(string content) should not be larger than 580 bytes.

- Add or modify "cgi" "embedded in asp file  
cgi file embedded in asp (except the cgi that independent with the browser)  
should be modified to support multi-language as follow:
  - Add or modify the cgi file
  - Modify the index.asp and mirror in the ary\_sting array.
  - Modify th string.conf

Example:

If we add virtual server on adv\_nat\_virsvr.asp:

```
<TD class=tabdata>
  <iframe src="/cgi-bin/virsvr_table.cgi" id="cgi_frame" frameborder="0" width="550" height="250"></iframe>
</TD></TR></TBODY></TABLE>
```

#### a) Add js codes in virsvr\_table.cgi

```
var virstrtmp = top.ary_strings;
var vir_obj = {};
for(var i=0; virstrtmp[i][0] != ""; i++) vir_obj[virstrtmp[i][0]]=virstrtmp[i][1];
```

Ary\_strings of index.asp

As

there's nothing but table column need to support multi-language, we will modify the table column with `<script>document.writeln(vir_obj["VirsvrCgiRule"])</script>`, the VirsvrCgiRule is the Key of ary\_stings in index.asp.

```
<table id="CgiInnerTable" align=center width="440" border="1"
cellpadding="0" cellspacing="0" bordercolor="#CCCCC" bgcolor="#FFFFFF">
<tr height="30"><td width="70" align="center"
class="tabdata"><strong><script>document.writeln(vir_obj["VirsvrCgiRule"])</script></strong></td><td width="70" align="center"
class="tabdata"><strong><script>document.writeln(vir_obj["VirsvrCgiStartPort"])</script></strong></td>
<td width="70" align="center" class="tabdata"><strong><script>document.writeln(vir_obj["VirsvrCgiEndPort"])</script></strong></td>
width="120" align="center" class="tabdata"><strong><script>document.writeln(vir_obj["VirsvrCgiLocalIP"])</script></strong></td>
width="70" align="center" class="tabdata"><strong><script>document.writeln(vir_obj["VirsvrCgiEdit"])</script></strong></td>
align="center" class="tabdata"><strong><script>document.writeln(vir_obj["VirsvrCgiDrop"])</script></strong></td></tr>
```

Cgi source code

```
VirsvrCgiRule', '<% tcWebApi.get("String_Entry", "VirsvrCgiRule", "s")%>'],
VirsvrCgiStartPort', '<% tcWebApi.get("String_Entry", "NatVirsvrStartPortNumText", "s")%>'],
VirsvrCgiEndPort', '<% tcWebApi.get("String_Entry", "NatVirsvrEndPortNumText", "s")%>'],
VirsvrCgiLocalIP', '<% tcWebApi.get("String_Entry", "VirsvrCgiLocalIP", "s")%>'],
VirsvrCgiEdit', '<% tcWebApi.get("String_Entry", "VirsvrCgiEdit", "s")%>'],
VirsvrCgiDrop', '<% tcWebApi.get("String_Entry", "VirsvrCgiDrop", "s")%>'],
AndCgiIndexText', '<% tcWebApi.get("String_Entry", "AndCgiIndexText", "s")%>'].
```

Defined in ary\_strings in index.asp

- Modify the index.asp. Add string in ary\_string for cgi. The mapping relationship can refer to the instruction of modify independent js file.
- Modify the string.conf as below

```
579 VirsvrCgiRule=Rule
580 VirsvrCgiStartPort=External start Port
581 VirsvrCgiEndPort=External end port
582 VirsvrCgiIntPort=Internal port
583 VirsvrCgiLocalIP=Local IP Address
584 VirsvrCgiStatus=Active
585 VirsvrCgiEdit=Edit
586 VirsvrCgiDrop=Drop
```

Tips: the modifications apply to js embedded in asp only, the cgi independent with browser don't work here. If the cgi independent with browser need to support multi-language, the only way would be transfer to asp.

#### 4) Modify the asp in GUITemp node.

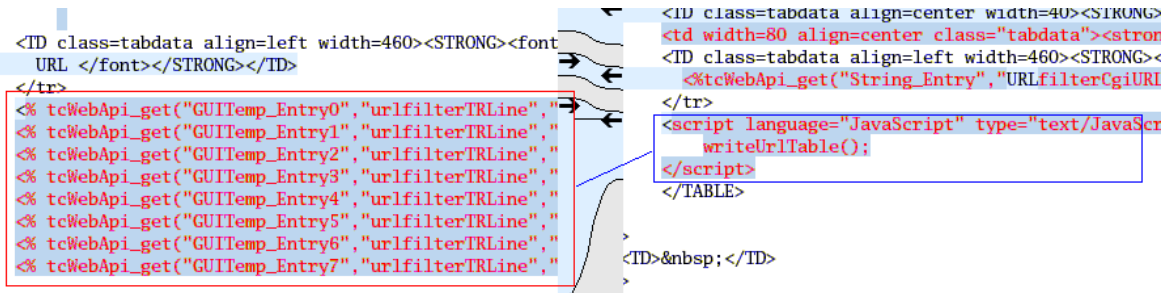
There's two methods of Multi-Language adaption for GUITemp:

a) Ignore the GUITemp node in asp( suitable for node with few attributes).Handle with js and store strings which need multi-language support into array.Read all information need to be shown with Multilanguage in the table as well. For example, if we want to modify the access\_URLfilter.asp:

Get all the attributes (need to be shown on table) of 8 entry and store them into the array. Substitute the information with multi-language for the content.



Drop the GUITemp and use the js as following figure:



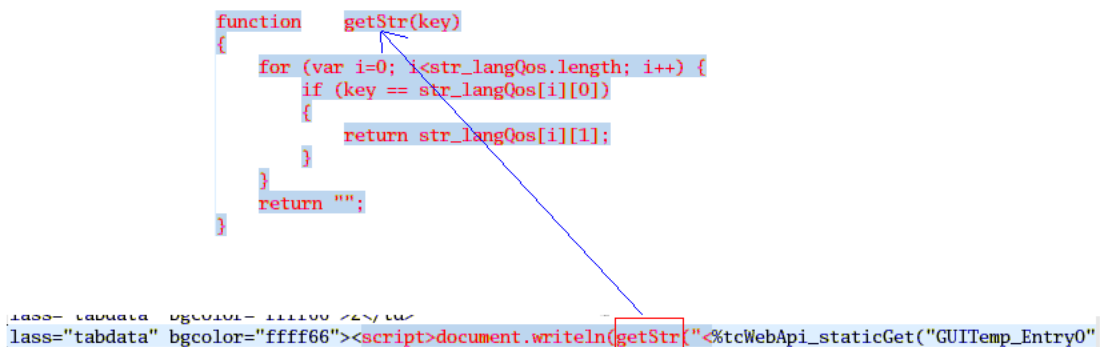
```
<TD class=tabdata align=left width=460><STRONG><font
URL </font></STRONG></TD>
</tr>
<% tcWebApi_get("GUITemp_Entry0", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry1", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry2", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry3", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry4", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry5", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry6", "urlfilterIRLine",
<% tcWebApi_get("GUITemp_Entry7", "urlfilterIRLine",
</tr>
<td width=80 align=center class="tabdata"><strong>
<TD class=tabdata align=left width=460><STRONG><
<%tcWebApi_get("String_Entry", "URLfilterCgiURL
</tr>
<script language="JavaScript" type="text/JavaScript"
writeUrITable();
</script>
</TABLE>
<TD>&nbsp;</TD>
```

b) Still using the GUITemp, but substitute the content on GUI(suitable for webpage with lots nodes and the attribute are scattered). For example, the adv\_qoslist.asp:

Store the information need to be shown in QOS table into array like following:

```
<script language="JavaScript" type="text/JavaScript">
var str_langQos = [
    ["Yes", "<%tcWebApi_get("String_Entry", "ACLYesText", "s")%>"],
    ["No", "<%tcWebApi_get("String_Entry", "ACLNoText", "s")%>"]
];
```

The "yes" and "no" are the value under Qos node, and the tcWebApi\_get("String\_Entry", "ACLYesText", "s") are the actual character need to be substituted. Other characters will be handled similarly. Call getStr function when drawing table in html code(the definition of getStr will be defined in this page, in which key means "Yes"/"No")



```
function getStr(key)
{
    for (var i=0; i<str_langQos.length; i++) {
        if (key == str_langQos[i][0])
        {
            return str_langQos[i][1];
        }
    }
    return "";
}

<table border="1" class="tabdata" bgcolor="ffffff"><tr>
<td colspan="2" class="tabdata" bgcolor="ffffff"><script>document.writeln(getStr("<%tcWebApi_staticGet("GUITemp_Entry0"
```

Tips: avoid using GUITemp node or cgi if too many information stored in the table, and do the showing with js in asp will be better.

## 7.2 mlCheckHash introduction

Location: /tools/mlCheckHash

Compile: sudo make PROFILE=tc3162u mlCheckHash

Tips: mlCheckHash does not need to be recompiled unless toolchain changed or code under /tools/mlCheckHash has been modified.



## 8 Conclusion

---

## 8 Conclusion

---

This system provides a common interface for users to construct their own webpage without the need to understand the internal structure of the system. Via the API provided by us, users can control the internal parameters of the system.

At the beginning of the document, we have introduced the format for ROMFILE. Next, we also introduced the seven ASP functions currently provided by the Web Server, and by using these seven functions, users will be able to develop their own functions for their own webpage.

When users are developing their own functions, these functions must be defined and invoked under CFG Manager. Inside this document, we have also introduced the program structure of CFG Manager and what program information to modify when required to add new functions.

At the end of the document, we have provided some webpage examples. These examples cover most of the applications currently in our system, and after looking at these examples, users should be able to write their own webpage.

Since the current ASP functions do not support looping, resulting in some functions requiring the use of CGI to process the development, but in future the system will support loop functions hence increasing the flexibility for users to develop their webpage. Furthermore, we have yet to provide a function to declare variables, all users kindly take note of this during development of programs.

### 8.1 AutoPVC Node : Setting the AutoPVC information

Currently the maximum Entry is 16 lines

Item	Node Name	Attribute	Value	RW
1	AutoPVC_Comm on	Active	{1:Active,0:Deactive}	RW

2	AutoPVC_Common	Probe_OAM	{1:search oam,0:not search oam}	RW
3	AutoPVC_Common	Probe_ARP	{1:search arp,0:not search arp }	RW
4	AutoPVC_Common	Probe_PPPOE	{1:search pppoe,0:not search pppoe }	RW
5	AutoPVC_Common	Probe_DHCP	{1:search dhcp,0:not search dhcp }	RW
6	AutoPVC_Common	Encap	{1:VC,0:LLC}	RW
7	AutoPVC_Entry	PVC	{0~8,the pvc we want to replace}	RW
8	AutoPVC_Entry	VPI	{0~255,the vpi we want to search}	RW
9	AutoPVC_Entry	VCI	{1~65535,the vci we want to search}	RW