MEDIATEK

MediaTek Mechanism for TR069 Parameter

Content

- Why Need New Mechanism
- New Mechanism Introduce
- Some Rule about New Mechanism

Why Need New Mechanism

- Redundant code exists in previous CWMP parameter tree ,and it’s not easy for customers to add nodes by themselves.
- It’s a disaster when reading CWMP source code since all the CWMP parameters locate in one single file(cwmpparameter.c), which can be more than 40,000 lines in Linos-TC2 Trunk.

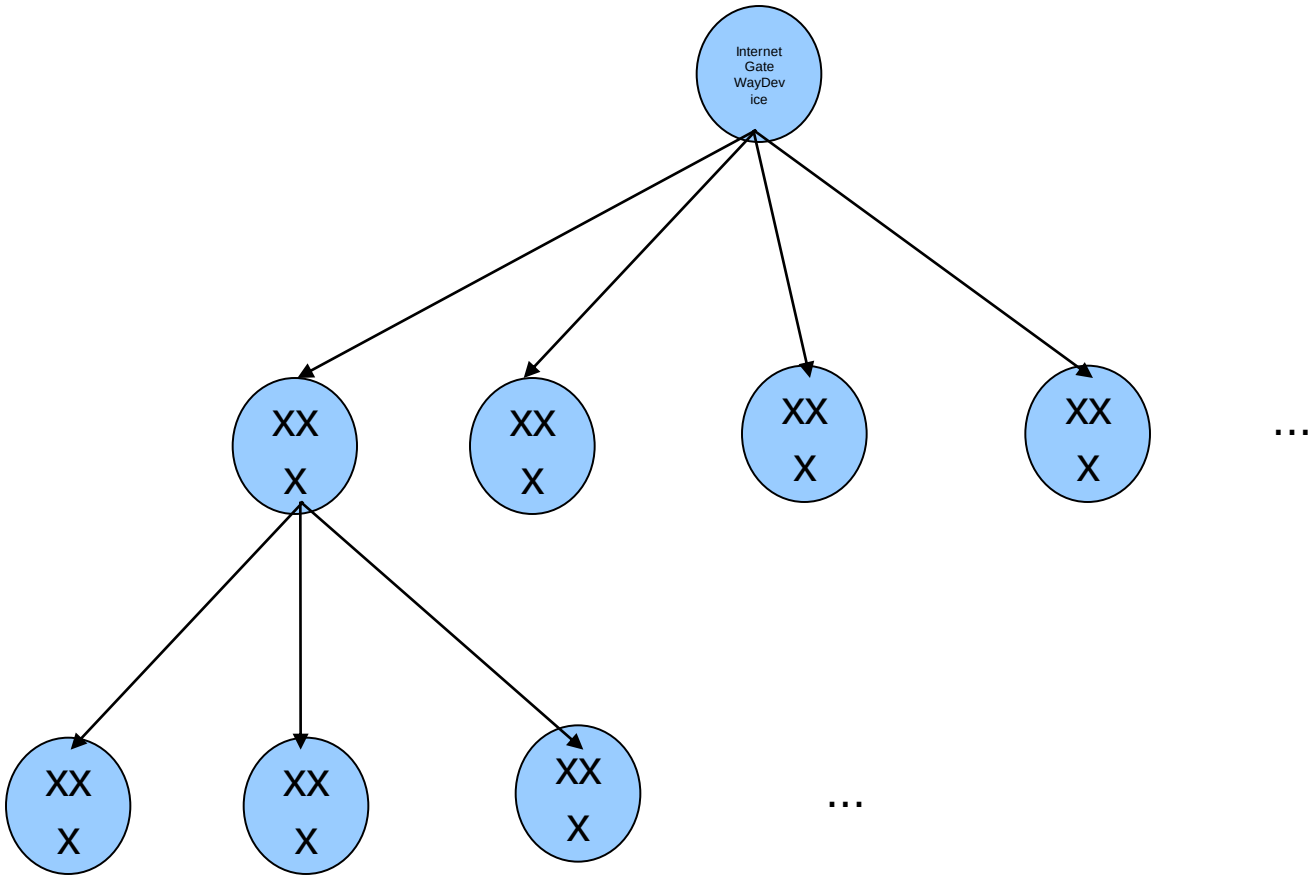
Why Need New Mechanism

```
TempLevel1 = DeviceNodeAllocStrW("URL",getURLValue, setURLValue, getURLAttribute, setURLAttribute);
if(TempLevel1 != NULL){
    cwmpNodeAppendChild(deviceNode, TempLevel1);
}else{
    return -1;
}

TempLevel1 = DeviceNodeAllocStrW("Username",getUsernameValue, setUsernameValue, getUsernameAttribute, setUsernameAttribute
if(TempLevel1 != NULL){
    cwmpNodeAppendChild(deviceNode, TempLevel1);
}else{
    goto Error_Handle;
}

TempLevel1 = DeviceNodeAllocStrW("Password",getPasswordValue, setPasswordValue, getPasswordAttribute, setPasswordAttribute);
if(TempLevel1 != NULL){
    TempLevel1->flag = 2;
    cwmpNodeAppendChild(deviceNode, TempLevel1);
}else{
    goto Error_Handle;
}

TempLevel1 = DeviceNodeAllocBooleanW( "PeriodicInformEnable", getPeriodicInformEnableValue,setPeriodicInformEnableValue, getPeric
if(TempLevel1 != NULL){
    cwmpNodeAppendChild(deviceNode, TempLevel1);
}else{
    goto Error_Handle;
}

TempLevel1 = DeviceNodeAllocUnsignIntW("PeriodicInformInterval",getPeriodicInformIntervalValue, setPeriodicInformIntervalValue, get
if(TempLevel1 != NULL){
    cwmpNodeAppendChild(deviceNode, TempLevel1);
}else{
    goto Error_Handle;
}
```

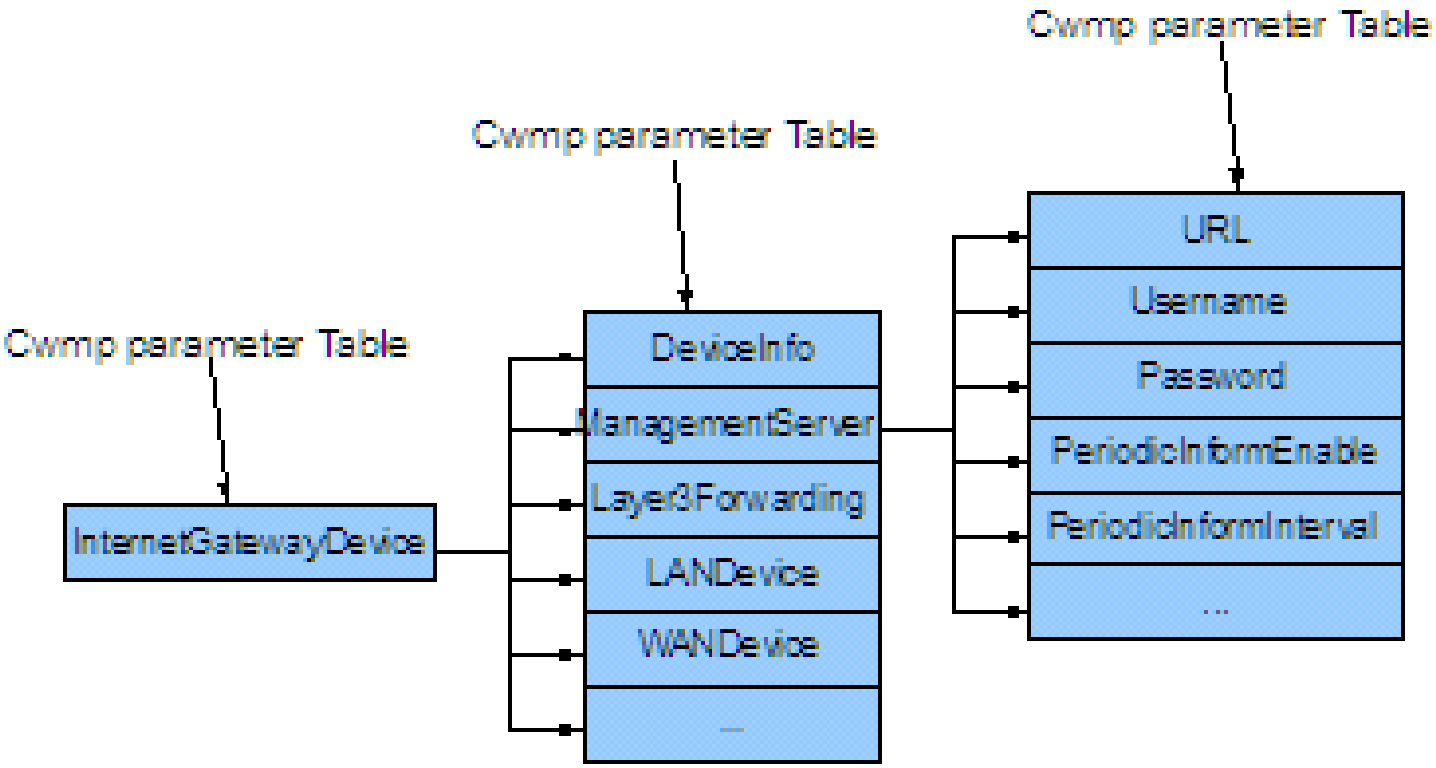
New Mechanism Introduce

- **Define the requisite node in different node table, and generate the parameter tree with the recursive function RegisterNodeFromTable();**
- **Assign the nodes to respect files following the TR098 Spec, which makes the node maintenance work more efficient.**
- **Provide a dynamic node registration mechanism, so the customers can add new parameters by themselves.**

Former implementation: parameter tree with List structure



New implementation : Table structure



Registration mechanism for table structure

- **Static register:** define the associate nodes in static arraies, which will be handled by function RegisterNodeFromTable()
- **Dynamic register:** fill the node information in DynamicNode_Table of cp_dynamicnodetable(), including parent node name(Node in Object type, full name),node table to be registered and new added node type(NodeAddType)

--- Dynamic node: register under Node.1 ,.2 ,.3…..

--- Static node: register under parent Node straightforward

Difference against Linos

- Linos: The DeviceNodeAlloc() function and CwmpNodeAppendChild() function will be called repeatedly in InitParameterTree() to generate the parameter tree.
- Linux: the RegisterNodeFromTable() will be called once by InitParameterTree() after filling out the respective node table and the dynFunc()
- The InitDynamicNode() will be called to dynamically register the node，so the customers can develop nodes by themselves.

Memory status

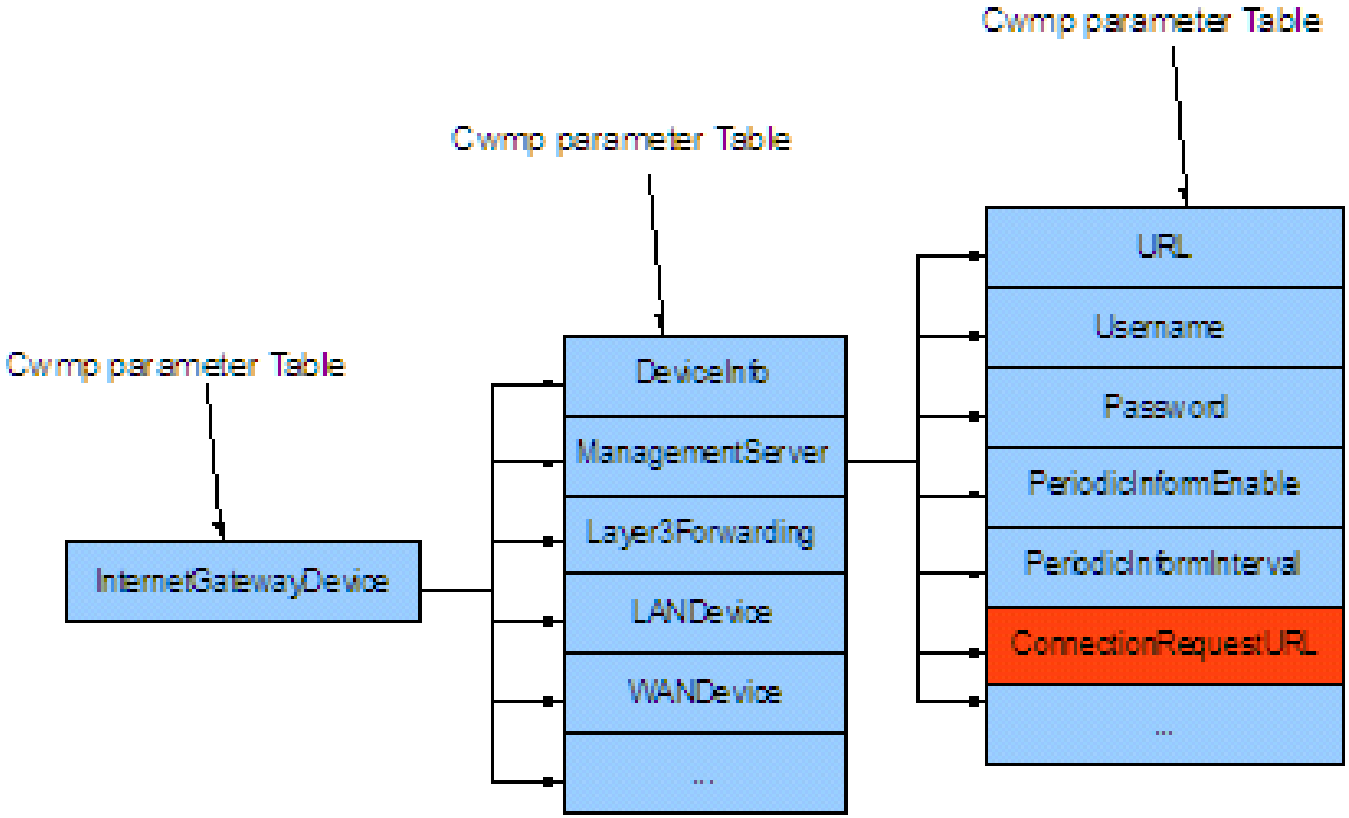
- **Node No: 186**
- **APP size difference:**

--- Code size is smaller compare to former implementation in Linos.

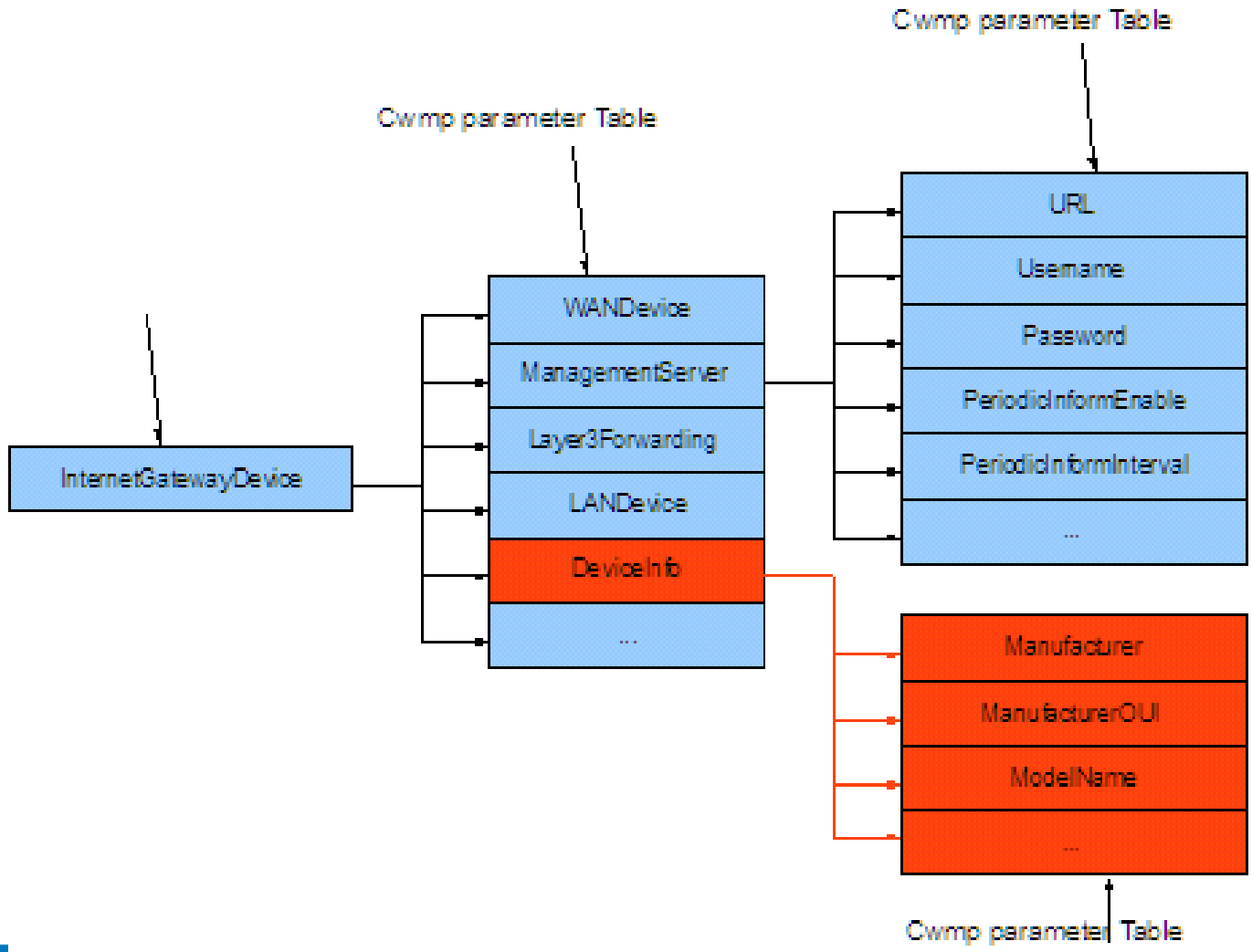
--- Overhead for table data is increased compared to former implementation in

parameter tree implementation	App Size（Bytes）
Former Linos type	1060296
Table structure type	1054216

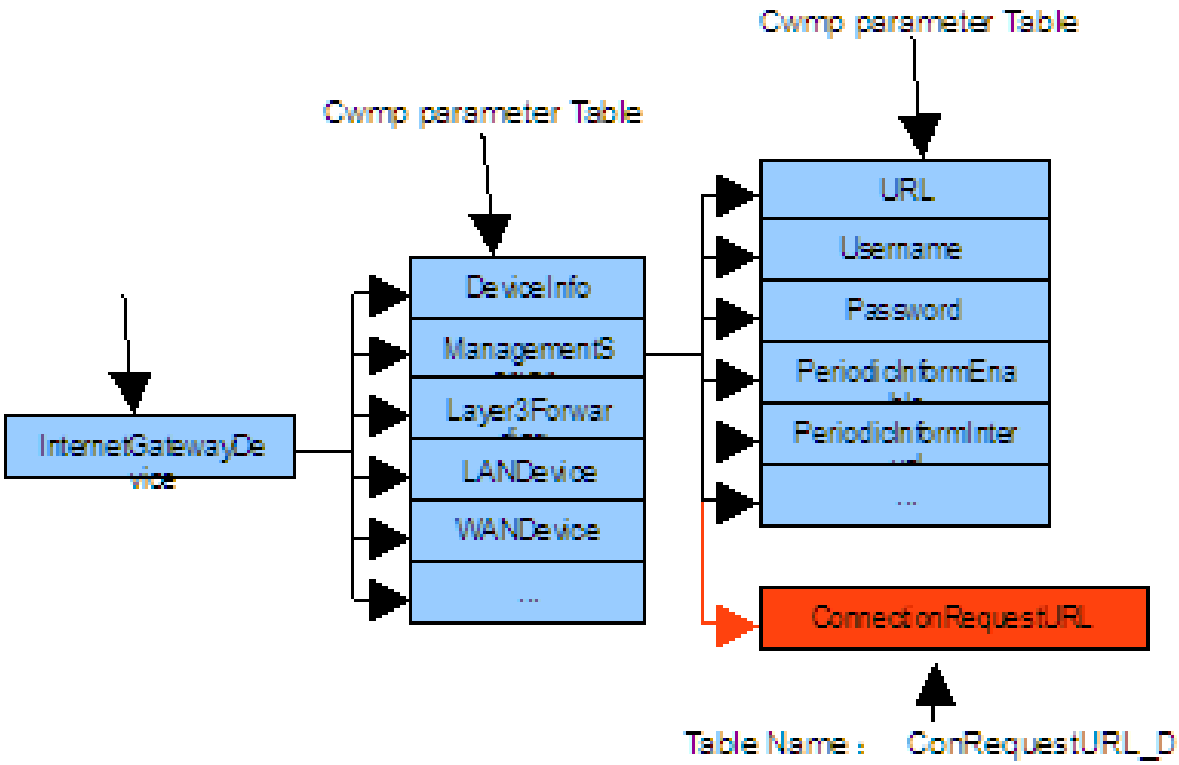
Static registration- leaf node



Static registration- sub tree node

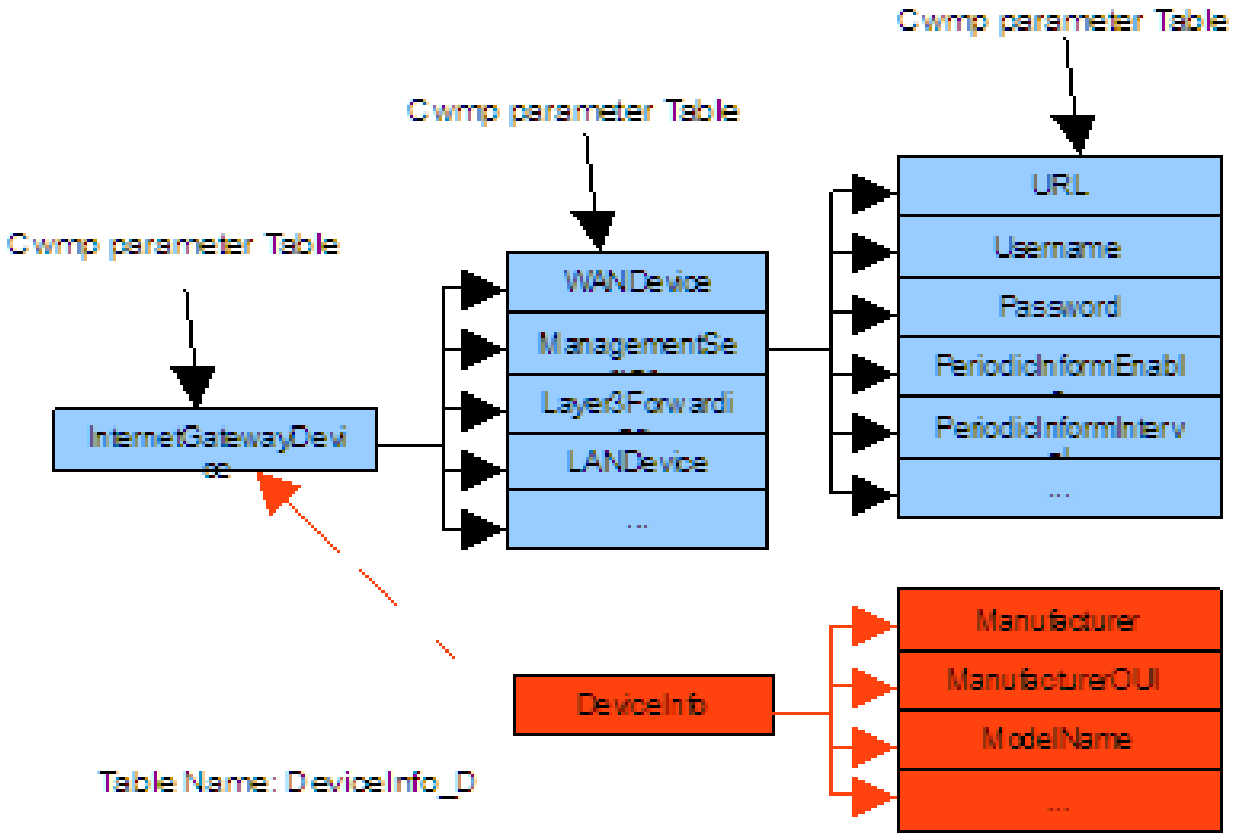


Dynamic registration- static leaf node



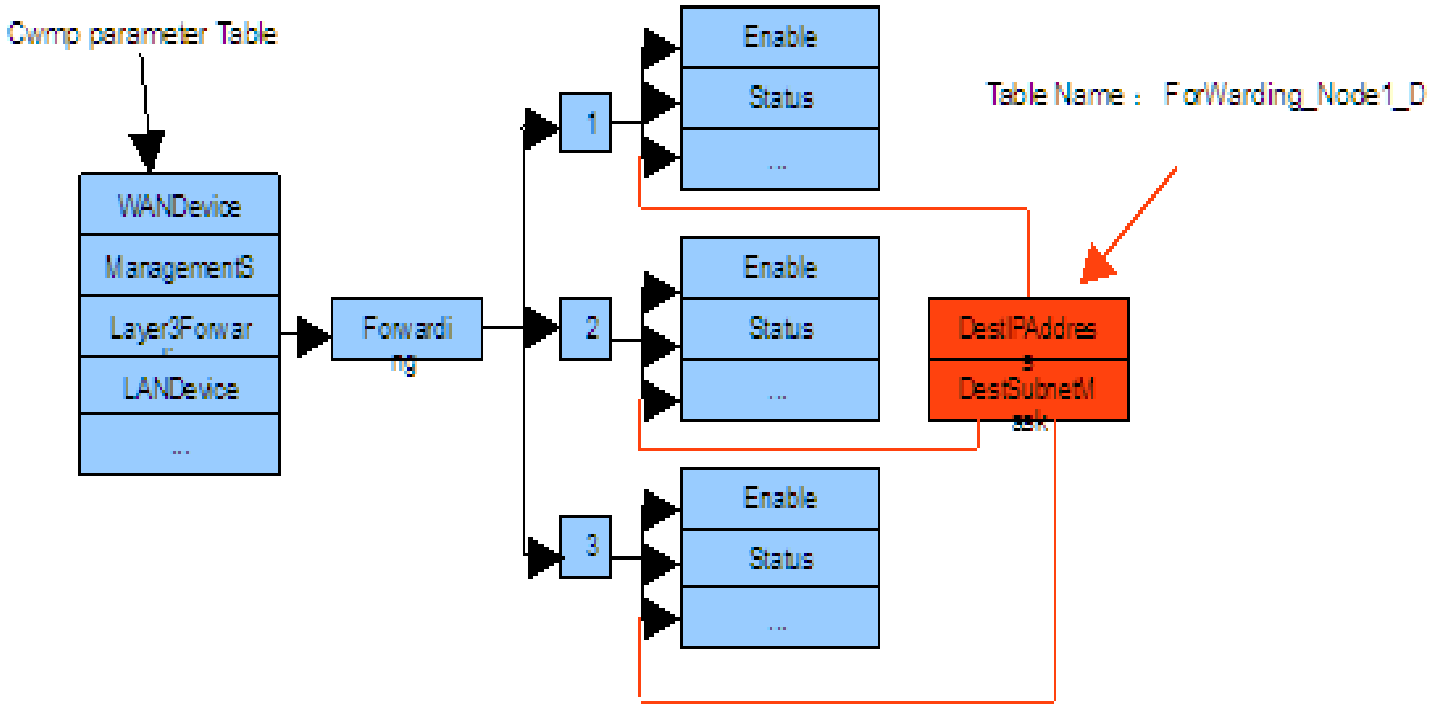

```
CwmpDynamicNodeStruct DynamicNode_table[] =
{
    {"InternetGatewayDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.", DeviceInfo_D, Dynamic},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Dynamic},
    {"InternetGatewayDevice.ManagementServer.", ConRequestURL_D, Static},
    {NULL, NULL, -1},
};
```

Dynamic registration: static sub tree node




```
CwmpDynamicNodeStruct DynamicNode_table[] =
{
    {"InternetGatewayDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.", DeviceInfo_D, Dynamic},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Dynamic},
    {NULL, NULL, -1},
};
```

Dynamic registration: dynamic node




```
CwmpDynamicNodeStruct DynamicNode_table[] =
{
    {"InternetGatewayDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.", DeviceInfo_D, Dynamic},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Dynamic},
    {"InternetGatewayDevice.Layer3Forward.Forwarding.", ForWarding_Node1_D,
    Dynamic},
    {NULL, NULL, -1},
};
```

Dynamic,

Dynamic registration mechanism

- **function prototype**

int InitDynamicNode（DeviceNodePtr root, char *ReInitNodeName）

Parameter name	Meaning
Root	Root node pointer of the registering node
ReIntNodeName	Initialize the node tree with full node object name again(in case the DeleteObject() and AddObject() need to partially initial the node tree again)

- **Execution Flow**

– Pick up node in DynamicNode_Table one by one till NULL.

– Determine if the node need registration according to its parent node and ReInitNodeName, which assure the node is not multiple registered

InitDynamicNode Examples

- initialization of the whole TR069 parameter tree

```
int cwmpInitParameterTree()
{
    DeviceNodePtr   TempLevel1 = NULL;
    DeviceNodePtr   TempLevel2 = NULL;
    char deviceName[PARAMETER_NAME_LEN];
    int rtnCode = 0;

    DeviceNodePtr top = NULL;
    top = RegisterNodeFromTable(NULL, TopIGDNode);
    if(top == NULL)
    {
        tc_printf("cwmpInitParameterTree: RegisterNodeFromTable: ERROR\n");
        return -1;
    }

    if(InitDynamicNode(top, "InternetGatewayDevice.") != 0)
    {
        tc_printf("cwmpInitParameterTree: RegisterNode Dynamic ERROR\n");
    }

    rpc.IGDDeviceRootPtr = top;
    return 0;
}
```

InitDynamicNode Example

- initialization of the WANDevice parameter tree

```
    cwmpFreeDeviceTable(WANDeviceNode);

    //rtnCode = cwmpInitWANDeviceSubTree(WANDeviceNode);
    if ( RegisterNodeFromTable(WANDeviceNode, WANDeviceIndex) == NULL ) {
        rtnCode = -1;
    }
    if(InitDynamicNode(rpc.IGDDeviceRootPtr,
"InternetGatewayDevice.WANDevice.") != 0)
    {
        tc_printf("DeviceNodeDelPVCObject: RegisterNode Dynamic ERROR\n");
        rtnCode = -1;
    }
    return rtnCode;
}
```

Dynamic register node

- **Aadvantages of dynamic registration**
  - Keep our existing nodes invisible
  - Custom need only focus on the implementation of target node table
  - Register the dynamic node once after the node information added in DynamicNode_Table
- **Attentions for dynamic registration**
  - Dynamic registration is not advisable for sub-nodes under multiple level dynamic node (for instance, node under WanConnectionDevice)
  - Actions like addObject and DeleteObject will operate parameter tree locally need to call InitDynamicNode() to register again.
  - Pay attention to the node type (Dynamic or Static) to add correct DynamicNode_Table

Example for attention 1

- **Customer want to add new node under parent node:**
- **Create node table for child node, suppose the table names WANIPConnection_SUBNode1_D**

  Add in the DynamicNode_Table as following figure.

```
CwmpDynamicNodeStruct DynamicNode_table[] =
{
    {"InternetGatewayDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.", DeviceInfo_D, Dynamic},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Static},
    //{"InternetGatewayDevice.LANDevice.", DeviceInfo_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.1.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.2.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.3.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.4.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.5.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.6.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.7.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    {"InternetGatewayDevice.WANDevice.1.WanConnectionDevice.8.IPConnection.",
    IPConnection_SUBNode1_D, Dynamic},
    /*You can Add Dynamic node before*/
    {NULL, NULL, -1},
```

Struct definition

```
typedef struct _CwmpParameterStruct
{
    char *name;
    NodeType type;
    NodeAttrType attrFlag;
    NodeFunc *func;
    struct _CwmpParameterStruct *next;
}CwmpParameterStruct, *pCwmpParameterStruct;
```

Parameter Name	Meaning
name	Node name
type	Node type
attrFlag	Flag indicates attribute if it can be “active” or “Passive”
func	Operate functions for node, (getvalue, setvalue, getattribute,setattribute,addobject,deleteobject,dynfunc)
next	Child node table in case the node type is Object

Node structure definition

Dynamic register node table definition

```
typedef struct _CwmpParameterStruct
{
    char *Parentname;
    struct CwmpParameterStruct *nodetable;
    NodeAddType type;
}CwmpParameterStruct, *pCwmpParameterStruct;
```

Parameter name	Meaning
Parentname	Parent node to be registered to
Nodetable	Nodetable to be registered
Type	Node register type(staic, dynamic)

NodeType definition

Node type	Meaning
ObjectR	Object type, Readonly
ObjectW	Object type, Writable
StrW	String type, writable
StrR	String type, Readonly
BooleanW	Boolean type, Writable
BooleanR	Boolean Type,ReadOnly
UnsignIntW	Unsigned Int,Writable
UnsignIntR	Unsigned Int,ReadOnly
IntW	Int type,Writable
IntR	Int type,Readonly
DateTimeTypeW	Date timetype,Writable
DateTimeTypeR	DateTime Type,ReadOnly

NodeAttrType deifinition

Parameter type	Meaning
NoActiveSet	This node can not be set to “Active”
NoLimite	This node has none attribute limite
NoPassiveActiveSet	This node can not be set to “Active ” or “Passive”

NodeFunc definition

Function Name	Meaning
addObject	For Node type ObjectW, function to be called when excute AddObject RPC
deleteObject	For Node type ObjectW, function to be called when excute AddObject RPC
getValueInfo	For none-object type node, function to be called when excute GetParameterValue RPC
setValueInfo	For none-object type node, function to be called when excute SetParameterValue RPC
getAttribute	For none-object type node, function to be called when excute GetParameterAttributeValue RPC
setAttribute	For none-object type node, function to be called when excute SetParameterAttributeValue RPC
dynFunc	For Object type node, function to be called when has dynamic child node

Definition of the node register function

- **Function prototype**

DeviceNodePtr DeviceNodeAlloc(char *IndexStr, NodeType type, NodeAttrType attrFlag, NodeFunc *nodeFunc)

- **Return value**
  - Success: Pointer of the node
  - Failure：NULL
- **Parameter specification**
  - IndexStr-nodeName： Node name
  - type-: Node type
  - **attrFlag-**Flag bit for attribute of nodes
  - **nodeFunc-**Node assoicated function

Static register functions

- **function prototype**

DeviceNodePtr RegisterNodeFromTable(DeviceNodePtr parent, CwmpParameterStruct *pNodeStruct)

- **Return value**
  - Success: Pointer of the parent node
  - Failure: NULL
- **parameter specification**
  - Parent-pointer of the parent node for registering node, NULL stands for the parent node registration.
  - pNodeStruct- Node table structure address

Dynamic register function

- **function prototype**

int RegisterNodeDynamic(DeviceNodePtr root, char* parentNodeName, CwmpParameterStruct *pNodeStruct, NodeAddType Flag)

- **Return value**
  - Success ：0
  - Failure：-1
- **parameter specification**
  - Root –pointer for the root node
  - parentNodeName-Parent node name of the registering node
  - pNodeStruct- node table address
  - Flag-Flag indicates register type（Dynamic or Static）

Some Rule about New Mechanism

- **Node assignment and file naming rule**
  - Node assignment follow the TR098 Spec
  - New file need to be created in case Large node object under IGD node
  - New file need to be created in case the new Object has lots objects or sub nodes
  - **Naming rule(can use shortcut)**
    - cp_**nodeName**.h, cp_**nodeName**_**d**.h(Dynamic register)
    - cp_**nodeName**_nodetable.c,cp_**nodeName**_nodetable_**d**.c （Dynamic register）
    - cp_**nodeName**_func.c,cp_**nodeName**_func_**d**.c（Dynamic register）

File rules

File Name	Definition
Cwmpparameter_table.c	CwmpInitParameterTree()definition and functions and variables to be shared for all nodes
Cp_topnodetable.c	IGD table definition and child node table definition
Cp_topnodetable.h	IGD table declaration and child node declaration
Cp_dynamicnodetable.c	Dynamic registered node table definition and child node table definition
Cp_dynamicnodetable.h	Dynamic registered node table declaration and child node table declaration
Cp_core.h	Definition of table structure and declaration of node register function
Cp_Core.c	Definition of node register function and node apply fuction
Cp_xxx_noetable.c Cp_xxx_nodetable_d.c	Definition of Node table(_d stands for dynamic register type)
Cp_xxx.h Cp_xxx_d.h	Declaration of function (_d stands for dynamic register type)

Some Rule about New Mechanism

- **File Locations**
- **Create new directory cwmpParameter under CWMP**
  - Includes
  - Sources
    - Source code for the new added nodes be put in the dir respectively

Some Rule about New Mechanism

Example 1:New added node(InternetGatewayDevice.DeviceInfo.）and its child node

3 files added:

cp_deviceinfo.h

cp_deviceinfo_nodetable.c

cp_deviceinfo_func.c

Example 2:New added node (InternetGatewayDevice.WANDevice.） and its child node

3 files added:

cp_wandevice.h

cp_wandevice_nodetable.c

cp_wandevice_func.c

Example 3:New added node (InternetGatewayDevice.WANDevice.）and its child node WANConnectionDevice

3 files added:

cp_wandevice_wanconnectiondevice.h

cp_wandevice_wanconnectiondevice_nodetable.c

cp_wandevice_wanconnectiondevice_func.c

Some Rule about New Mechanism

- **Definition rule for node table**
  - Use the structure of CwmpParameterStruct, and end with {NULL, -1, -1, NULL, NULL}
  - Add comments of the parent node name and associate location.

```
/*
      This Node: InternetGatewayDevice.LANDevice.{i}.WLanConfiguration.{i}
    Parent Node: InternetGatewayDevice.LANDevice.{i}.WLanConfiguration.
    Parent Node Table Location: cp_landevice_nodetable.c
*/

CwmpParameterStruct WLANConfigurationIndex[]=
{
    {NULL,ObjectR, NoLimite,NULL,WLANConfiguration},
    {NULL, -1, -1, NULL, NULL}
};
```

Additional remarks

- The source code also includes the cwmpParameter_table.h, which gives the definition of the Attribute structure and flag.
- All the attributes will be defined in the cwmpParameter_table.c

Naming rule for the node table

- (1)RPC functions for the leaf node be named following the TR098.
- (2)Node tables be named following the TR098.
- (3)Dynamic nodes use the “node name in TR098”+”Index”

```
/*
        This Node: InternetGatewayDevice.LANDevice.{i}.WLanConfiguration.{i}
    Parent Node: InternetGatewayDevice.LANDevice.{i}.WLanConfiguration.
    Parent Node Table Location: cp_landevice_nodetable.c
*/

CwmpParameterStruct WLANConfigurationIndex[]=
{
    {NULL,ObjectR, NoLimite,NULL,WLANConfiguration},
    {NULL, -1, -1, NULL, NULL}
};
```

(4)RPC function in Nodes not leaf name with “TR098nodeName” + “Funcs”

```
static NodeFunc WLANConfigurationFuncs=
{
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    cwmpInitWLANConfiguration1SubTree
};

/*
    Parent Node: InternetGatewayDevice.LANDevice.{i}.
    Parent Node Table Location: cp_landev_nodetable.c
*/
static CwmpParameterStruct LanDevice[] =
{
    {"WLANConfiguration", ObjectR, NoLimite, &WLANConfigurationFuncs, NULL},
    {NULL, -1, -1, NULL, NULL}
};
```

- **RPC naming rule**
  - (1)Use the existing name for the node created in Linos
  - (2) New added nods
    - get/set+NodeName+value/Attribute
    - DeviceNodeAdd+NodeName+Object
    - DeviceNodeDel+NodeName+Object
    - Name confilct must be avoided if use shortcut for the NodeName

Additional rules:

- cp_xxx_nodetable.c/cp_xxx_func.c just need to include cp_xxx.h and Global_res.h
- Declare the table you needed and omit the unnecessary;
- Define the RPC function with “static” attribute to avoid names conflict.
- To omit the declaration, the structure should be defined before table and the child table should be defined before parent table.
- All the tables should add comments to implicit the parent table and its definition path.