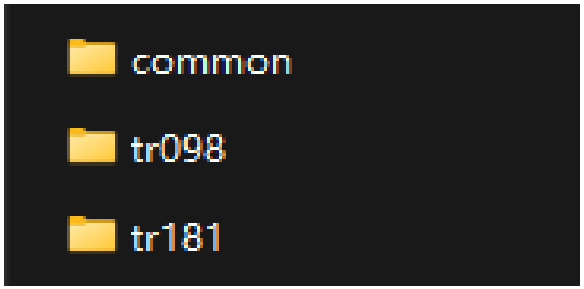
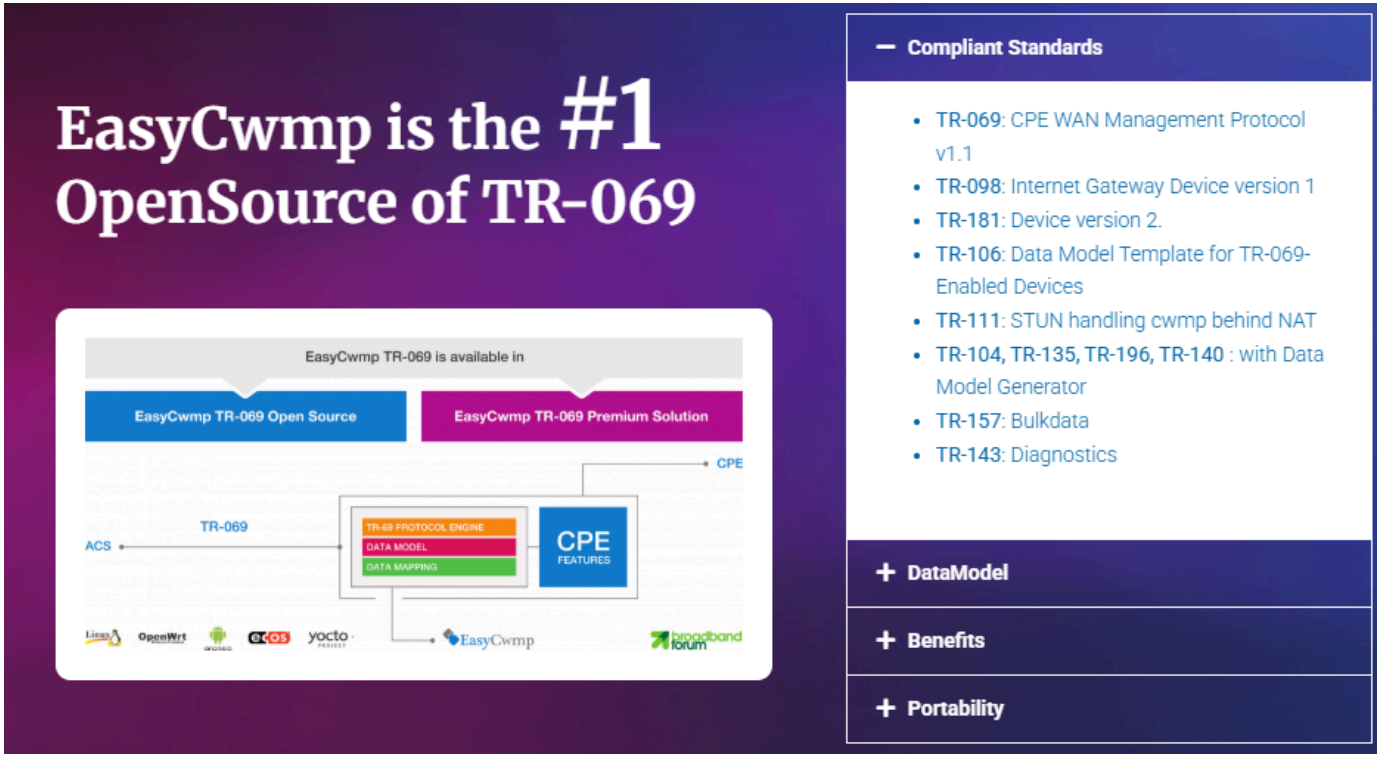
【How to】在 easycwmp 中自定义数据模型 TR069 (TR181)

由 earth 朱梦园 or 曾澳创建, 最后修改于五月 06, 2025

TR069 模型在 easycwmp 中默认已经实现，在源码：\easycwmp-1.8.6\ext\openwrt\scripts\functions\ 中集成了三类数据模型：



分别是： 通用数据模型、TR098数据模型、TR181数据模型



根据TR069数据模型标准，目前适合我 CPE 产品的模型数据为 TR181

而其他适用的标准，如 TR106 TR111 等，由于我们拿到的免费开源的 easycwmp，仅使用shell作为免费解决方案，并非商业版，更多模型和配置则需自行开发

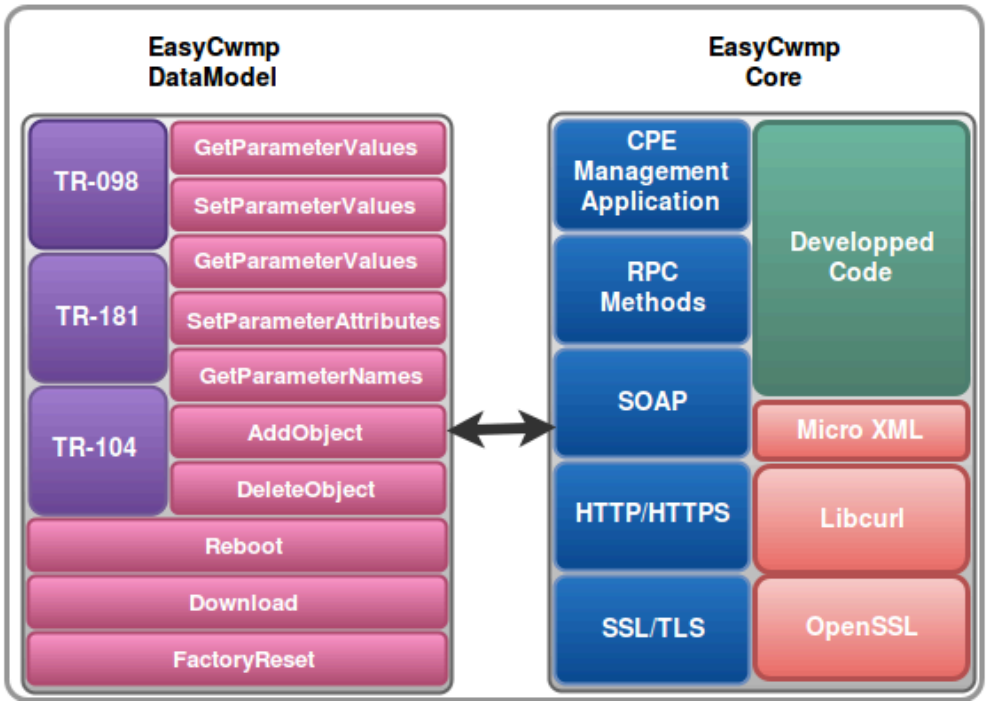
1. EasyCwmp design

The EasyCwmp design includes 2 parts:

- EasyCwmp core: it includes the TR069 CWMP engine and it is in charge of communication with ACS server. It is developed with C.
- EasyCwmp DataModel: it includes the DATAModel of TR-06 and it is compliant to some DataModel standards such as TR-098, TR-181, TR-104, ...

The key design goal is to separate the CWMP method execution from the CWMP engine. That makes **easy** to add and test new features.

DataModel is developped with shell as free solution and with C as commercial solution.



故接下来将简述如何使用 shell 增加设备的模型

数据模型位置以及如何获取

当安装 easycwmp 在 openwrt 中之后，模型就默认集成在了 /usr/share/easycwmp/functions 目录下：

```
root@OpenWrt:/usr/share/easycwmp/functions# ls
common                ip                    management_server
device_info           ipping_diagnostic     root
dhcpv4                ipping_launch         wifi
```

这里存放的便是上述源码中 common 和 TR181（编译时选择TR181） 中的模型

首先测试获取本地的模型数据，这些 shell 脚本会在设备中，根据模型的设置，在对应路径去查找数据，相关命令为：

```
easycwmp get
```

执行效果如下：

```
root@OpenWrt:/usr/share/easycwmp/functions# easycwmp get
{ "parameter": "Device.DeviceInfo.SpecVersion", "value": "1.0" }
{ "parameter": "Device.DeviceInfo.ProvisioningCode", "value": "" }
{ "parameter": "Device.DeviceInfo.Manufacturer", "value": "OpenWrt\nhttps___openwrt.org_" }
{ "parameter": "Device.DeviceInfo.ManufacturerOUI", "value": "FFFFFF" }
{ "parameter": "Device.DeviceInfo.ProductClass", "value": "Generic" }
{ "parameter": "Device.DeviceInfo.SerialNumber", "value": "FFFFFF123456" }
{ "parameter": "Device.DeviceInfo.HardwareVersion", "value": "v0" }
{ "parameter": "Device.DeviceInfo.SoftwareVersion", "value": "r16847-f8282da11e" }
{ "parameter": "Device.DeviceInfo.test_date", "value": "123" }
{ "parameter": "Device.DeviceInfo.wifi_ssid", "value": "test_tr069" }
{ "parameter": "Device.DeviceInfo.UpTime", "value": "2618", "type": "xsd:unsignedInt" }
{ "parameter": "Device.DeviceInfo.DeviceLog", "value": "" }
{ "parameter": "Device.DeviceInfo.MemoryStatus.Total", "value": "637576" }
{ "parameter": "Device.DeviceInfo.MemoryStatus.Free", "value": "136020" }
{ "parameter": "Device.DHCPv4.Server.Enable", "value": "1", "type": "xsd:boolean" }
{ "parameter": "Device.IP.Interface.1.Enable", "value": "1", "type": "xsd:boolean" }
{ "parameter": "Device.IP.Interface.1.Name", "value": "loopback" }
{ "parameter": "Device.IP.Interface.1.Type", "value": "Loopback" }
{ "parameter": "Device.IP.Interface.1.IPv4AddressNumberOfEntries", "value": "1", "type": "xsd:unsignedInt" }
{ "parameter": "Device.IP.Interface.1.IPv4Address.1.IPAddress", "value": "127.0.0.1" }
{ "parameter": "Device.IP.Interface.1.IPv4Address.1.AddressingType", "value": "Static" }
```

当然也可以单独去获取某个参数信息：

```
root@OpenWrt:/usr/share/easycwmp/functions# easycwmp get Device.WiFi.wifi24g.mai
n.SSID
{ "parameter": "Device.WiFi.wifi24g.main.SSID", "value": "DefaultSSID-E5EE" }
```

分析模型源码

上述模型的输出，是基于现有设备中的 config 文件获取的，easycwmp get 从某种意义上来说，和 uci show 类似，可以这样理解：

easycwmp的数据模型，其实就是对 uci config 文件的封装，接下来以 device_info 新增模型数据来举例：

```
root@OpenWrt:/usr/share/easycwmp/functions# cat device_info
#!/bin/sh
# Copyright (C) 2015 PIVA Software <www.pivasoftware.com>
#       Author: MOHAMED Kallel <mohamed.kallel@pivasoftware.com>

#common_execute_method_param "$parameter" "$permission" "$get_cmd" "$set_cmd" "xsd:$type" "$forcedinform"
#  $forcedinform should be set to 1 if the parameter is included in the inform message otherwise empty
#  Default of $type = string

##############################
#   Entry point functuons   #
##############################

prefix_list="$prefix_list $DMROOT.DeviceInfo."
entry_execute_method_list="$entry_execute_method_list entry_execute_method_root_DeviceInfo"
entry_execute_method_list_forcedinform="$entry_execute_method_list_forcedinform  entry_execute_method_root_DeviceInfo"

entry_execute_method_root_DeviceInfo() {
        case "$1" in ""|"$DMROOT."|"$DMROOT.DeviceInfo."*)
                common_execute_method_obj "$DMROOT.DeviceInfo." "0"
                common_execute_method_param "$DMROOT.DeviceInfo.SpecVersion" "0" "echo 1.0" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.ProvisioningCode" "1" "$UCI_GET easycwmp.@local[0].provisioning_code" "device_i
                common_execute_method_param "$DMROOT.DeviceInfo.Manufacturer" "0" "$UCI_GET easycwmp.@device[0].manufacturer" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.ManufacturerOUI" "0" "$UCI_GET easycwmp.@device[0].oui" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.ProductClass" "0" "$UCI_GET easycwmp.@device[0].product_class" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.SerialNumber" "0" "$UCI_GET easycwmp.@device[0].serial_number" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.HardwareVersion" "0" "$UCI_GET easycwmp.@device[0].hardware_version" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.SoftwareVersion" "0" "$UCI_GET easycwmp.@device[0].software_version" "" "" "1"
                common_execute_method_param "$DMROOT.DeviceInfo.UpTime" "0" "device_info_get_uptime" "" "xsd:unsignedInt"
                common_execute_method_param "$DMROOT.DeviceInfo.DeviceLog" "0" "dmesg | tail -n1"
                common_execute_method_obj "$DMROOT.DeviceInfo.MemoryStatus." "0"
                common_execute_method_param "$DMROOT.DeviceInfo.MemoryStatus.Total" "0" "device_info_get_total_memory"
                common_execute_method_param "$DMROOT.DeviceInfo.MemoryStatus.Free" "0" "device_info_get_free_memory"

                return 0;
                ;;
```

```
        esac
        return $E_INVALID_PARAMETER_NAME;
}

########################################
#   Data model parameters functions   #
########################################

device_info_get_total_memory() {
                cut -d: -f 2 /proc/meminfo | head -n 1| tail -n 1 |  tr -d " kKbB"
}

device_info_get_free_memory() {
                cut -d: -f 2 /proc/meminfo | head -n 2| tail -n 1 | tr -d " kKbB"
}

device_info_set_provisioningcode() {
        local val="$1"
        $UCI_SET easycwmp.@local[0].provisioning_code="$val"
        return 0
}

device_info_get_uptime() {
        awk -F '.' '{ print $1 }' /proc/uptime
}
```

common_execute_method_param 函数

```
# 函数格式
common_execute_method_param "$parameter" "$permission" "$get_cmd" "$set_cmd" "xsd:$type" "$forcedinform"


# 示例
common_execute_method_param "$DMROOT.DeviceInfo.SerialNumber" "0" "$UCI_GET easycwmp.@device[0].serial_number" "" "" "1"
```

参数	含义	示例
parameter	TR069模型参数的完整路径	$DMROOT.DeviceInfo.SerialNumber 表示参数在TR069模型中的路径为 Device.DeviceInfo.SerialNumber
permission	只读 "0" 可写 "1"	0 表示该参数仅为只读参数
get_cmd	如何获取参数数值（命令）	$UCI_GET easycwmp.@device[0].serial_number 表示该参数获取的命令是 uci get easycwmp.@device[0].serial_number
set_cmd	如何设置参数（命令）	留空表示该参数不需要设置，对应 permission
type	数据类型： "xsd:string" 字符串 "xsd:boolean" 布尔 "xsd:unsignedInt" 无符号整型 "" 默认设置为字符串	留空表示默认为 字符串
forcedinform	"1" 表示这个参数为必须上报类型（关键字段） "0" 表示这个参数非必须，不强制出现在报文中	1 表示该参数为关键字段，每次都需要上报

确认模型输出

确认模型设置正确有两种方法，一种十分简单，调用 easycwmp 命令即可，我们以 device_info 模型的 SerialNumber 参数为例

```
easycwmp get


# 可以看到输出中存在
{ "parameter": "Device.DeviceInfo.SerialNumber", "value": "FFFFFF123456" }

# 由于在 parameter 中指定过路径，也可以直接：
easycwmp get Device.DeviceInfo.SerialNumber

# 输出
{ "parameter": "Device.DeviceInfo.SerialNumber", "value": "FFFFFF123456" }
```

上述只是简单的在命令行调用 easycwmp 工具，若想看到完整的报文上传，需要使用的是 easycwmpd 命令：

```
/usr/sbin/easycwmpd -f -b
# 该命令会重新尝试去连接 ACS 服务器，并主动上报一次数据，数据在报文中会组成 xml 格式的信息（以下截取部分）：


<?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <soap_env:Envelope
xmlns:soap_env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soap_enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:cwmp="urn:dslforum-org:cwmp-1-2">
  <soap_env:Header>
   <cwmp:ID soap_env:mustUnderstand="1">1</cwmp:ID>
  </soap_env:Header>
  <soap_env:Body>
   <cwmp:Inform>
    <DeviceId>
     <Manufacturer>OpenWrt
https___openwrt.org_</Manufacturer>
     <OUI>FFFFFF</OUI>
     <ProductClass>Generic</ProductClass>
     <SerialNumber>FFFFFF123456</SerialNumber>
    </DeviceId>
    <Event soap_enc:arrayType="cwmp:EventStruct[1]">
     <EventStruct>
      <EventCode>1 BOOT</EventCode>
      <CommandKey />
     </EventStruct>
    </Event>
    <MaxEnvelopes>1</MaxEnvelopes>
    <CurrentTime>2025-05-06T06:27:12+00:00</CurrentTime>
    <RetryCount>0</RetryCount>
    <ParameterList soap_enc:arrayType="cwmp:ParameterValueStruct[57]">
     <ParameterValueStruct>
      <Name>Device.DeviceInfo.SpecVersion</Name>
      <Value xsi:type="xsd:string">1.0</Value>
```

```
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.ProvisioningCode</Name>
       <Value xsi:type="xsd:string"></Value>
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.Manufacturer</Name>
       <Value xsi:type="xsd:string">OpenWrt
https___openwrt.org_</Value>
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.ManufacturerOUI</Name>
       <Value xsi:type="xsd:string">FFFFFF</Value>
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.ProductClass</Name>
       <Value xsi:type="xsd:string">Generic</Value>
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.SerialNumber</Name>
       <Value xsi:type="xsd:string">FFFFFF123456</Value>
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.HardwareVersion</Name>
       <Value xsi:type="xsd:string">v0</Value>
      </ParameterValueStruct>
      <ParameterValueStruct>
       <Name>Device.DeviceInfo.SoftwareVersion</Name>
       <Value xsi:type="xsd:string">r16847-f8282da11e</Value>
      </ParameterValueStruct>
```

可以清楚的看到 Device.DeviceInfo.SerialNumber 这个参数的数据以及数据类型

这便说明 common_execute_method_param 函数中的配置正确，若两种方法均无参数输出，则说明模型设置错误

所谓TR069模型

从源码中可以看出，所谓模型数据，实际上就是 uci 的 config 文件数据，而恰好 OpenWrt 的所有配置参数都放在 /etc/config 路径下，所以增加数据模型就十分简单了，只要 /etc/config 中存在对应的数据，就可以将其添加在 TR069 模型中，而发送给ACS服务器的报文数据，其实就是将所有的配置文件数据，重新命名并组成 XML 数据，然后通过 http POST 发送致 ACS 服务器

ACS 服务器下发命令

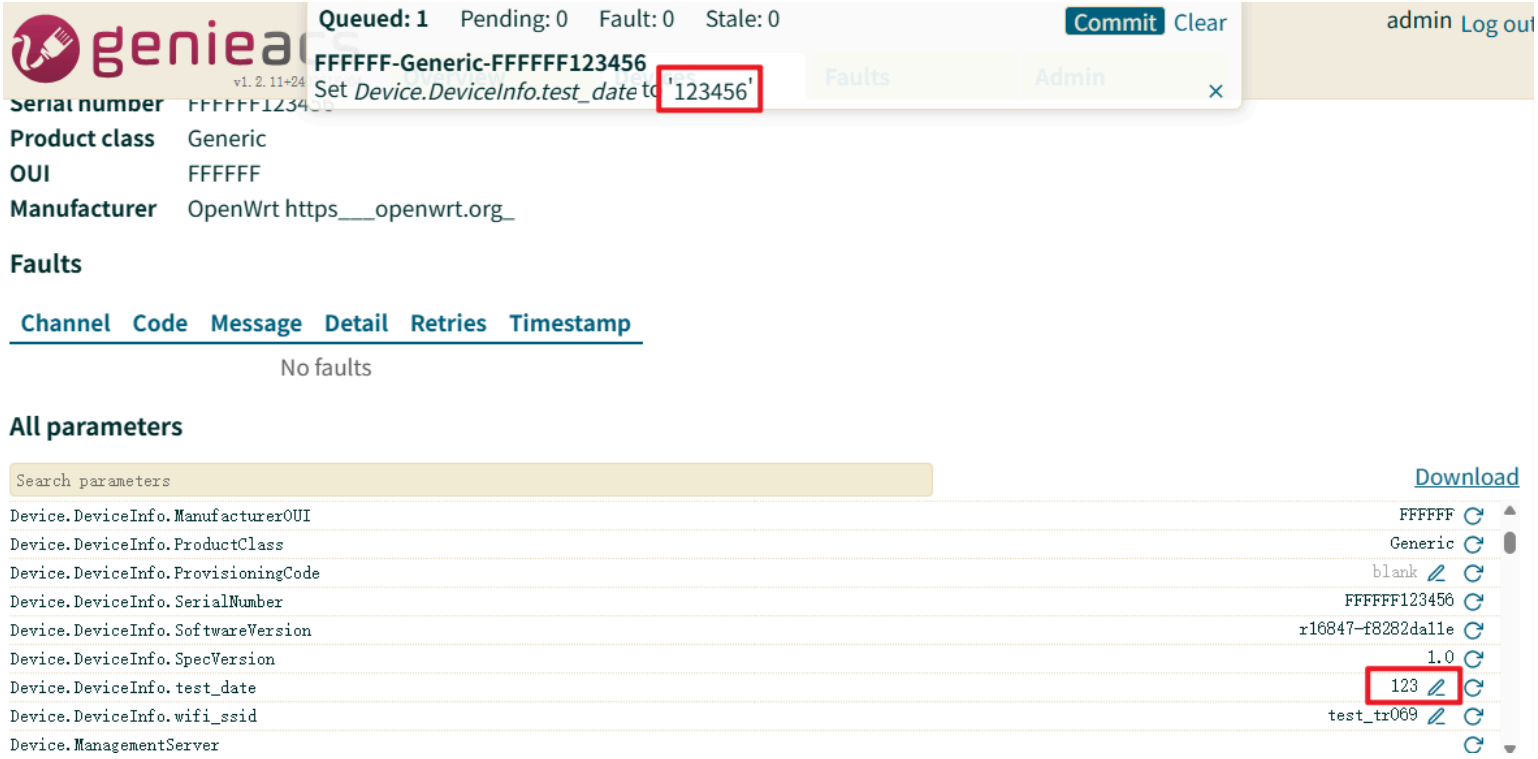
理解了模型的本质，那么ACS服务器的下发命令也就更好理解了，ACS 服务器的 UI 界面记录了 操作者 更改的参数信息，然后通过 http 发送 XML 给 CPE，CPE 中的 easycwmp 则根据这个 XML 包中的信息，对应到实际的 UCI 配置参数，然后执行 uci set 指令，更新参数

这里以新增一个 test_date 来详细说明：

```
# 首先在/usr/share/easycwmp/functions/device_info 中增加一个新的数据模型
common_execute_method_param "$DMROOT.DeviceInfo.test_date" "1" "$UCI_GET easycwmp.@device[0].test_date" "device_info_set_test_date" "" "1"

# 然后需要增加对应的处理函数
########################################
#   Data model parameters functions   #
########################################


device_info_set_test_date() {
        local val="$1"
        $UCI_SET easycwmp.@device[0].test_date="$val"
        uci commit easycwmp
        return 0
}




# 需要确保 test_date 存在且有输出
uci get easycwmp.@device[0].test_date
123


# 上述新增加了一个可以更改的模型数据，模型名为 Device.DeviceInfo.test_date
# 执行上报
```

```
/usr/sbin/easycwmpd -f -b


# 可以看到 XML 中存在

<ParameterValueStruct>
<Name>Device.DeviceInfo.test_date</Name>
<Value xsi:type="xsd:string">123</Value>
</ParameterValueStruct>
```

此时ACS服务器中可以对其参数进行设置：



将其数值更新为 123456

通过对ACS服务器抓包，可以发现ACS服务器发送了这样的数据：

```
Content-Length: 751
Server: GenieACS/1.2.11+240321fd04
SOAPServer: GenieACS/1.2.11+240321fd04
Content-Type: text/xml; charset="utf-8"
Date: Tue, 06 May 2025 06:54:23 GMT
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:
06:54:23.491270 IP 14.31.18.138.33685 > wutong.7547: Flags [.], ack 1747, win 135, options [nop,nop,TS val 1331393181 ecr 79787961], length 0
E..4n.@.8.u.............{rf...?1..... ......
O[r...w.
06:54:23.585355 IP 14.31.18.138.33685 > wutong.7547: Flags [P.], seq 10426:11197, ack 1747, win 135, options [nop,nop,TS val 1331393249 ecr 797
E..7n.@.8.r.............{rf...?1......!.....
O[r...w.POST / HTTP/1.1
Host: 14.21.46.164:7547
Cookie: session=a5d589b4996f68db
User-Agent: easycwmp
Content-Type: text/xml; charset="utf-8"
SOAPAction:
Content-Length: 596
```

可以明显看到 将 Device.DeviceInfo.test_date 更新为 123456

而在 CPE 这边，收到这个数据之后，easycwmp 会立刻去调用 common_execute_method_param 配置的 $set_cmd 也就是 一开始新增的

device_info_set_test_date 函数，最后调用 uci 命令更新了 easycwmp.@device[0].test_date ，这就是 ACS 下发参数的全过程。

ACS 服务器下发的命令谁来执行

我们知道 ACS 服务器可以远程重启设备，对设备进行 reboot 或者 firstboot，实际上本质和下发参数完全一致，也是发送了一个http包给 CPE，CPE 的 easycwmpd 服务，收到包后解析出命令，然后调用 easycwmp 开始执行对应的命令，其实我们可以直接调用 easycwmp 命令来测试

```
# 查看easycwmp 支持的命令
easycwmp
```

```
USAGE: /usr/sbin/easycwmp command [parameter] [values]
command:
  get [value|notification|name]
  set [value|notification]
  apply [value|notification|object|service]
  add [object]
  delete [object]
  download
  upload
  factory_reset
  reboot
  inform [parameter|device_id]
  --json-input
invalid action ''
```

参数	作用
get	获取模型参数的值
set	设置模型参数的值
apply	应用更改
add	添加动态对象
delete	删除动态对象
download	触发下载任务
upload	触发上传任务
factory_reset	恢复出厂设置
reboot	重启设备
inform	立即汇报模型参数一次
--json-input	json格式批量更改

上述命令，在 easycwmpd 服务启动的过程中，会自行调用以用于和ACS服务器通信或处理命令，例如，ACS 服务器需要更新某个模型数据时，下发命令到 easycwmpd 服务，就会自动调用 easycwmp get 去获取值，然后使用easycwmp inform 上报数据，当 ACS 服务器下发了 重启的命令时，easycwmpd 服务就会自动调用 easycwmp reboot 去执行重启；factory_reset也同理，我们可以看一段代码解析：

```
# 截取自 /usr/sbin/easycwmp


if [ "$action" = "factory_reset" ]; then
    if [ "`which jffs2_mark_erase`" != "" ]; then
        jffs2_mark_erase "rootfs_data"
    else
        /sbin/jffs2mark -y
    fi
    sync
    reboot
fi

if [ "$action" = "reboot" ]; then
    sync
    reboot
fi




# 可以看到，调用 easycwmp reboot 实际上就是执行了
# sync
# reboot
```

这里就是 easycwmp reboot 命令的具体实现，其中也说明了其他命令（get 等）的实现方法，此处不多列举

无标签