Mgmt Unified

业务开发指南

--- 1.0

1.0

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **修订人** | **修订内容** |
| 1.0 | 2022/9/14 | 曾毅 | 初稿 |
|  |  |  |  |

目录

[Mgmt Unified 1](#_Toc18968)

[业务开发指南 1](#_Toc24425)

[--- 1.0 1](#_Toc9231)

[1. 目的 4](#_Toc319)

[2. 代码目录结构 5](#_Toc32116)

[2.1 组件包 5](#_Toc19957)

[2.2 代码目录 5](#_Toc19395)

[2.3 业务需要关注的代码 6](#_Toc2136)

[3. 代码编写 7](#_Toc25802)

[3.1 接口层代码编写原则 7](#_Toc15438)

[3.1.1 设计原则 7](#_Toc17639)

[3.1.2 命名规则 7](#_Toc15521)

[3.2 JSON数据适配原则 7](#_Toc14438)

[3.2.1 设计原则 7](#_Toc14386)

[3.2.2 命名规则 8](#_Toc27720)

[3.3 Ubus代码编写原则 8](#_Toc14277)

[3.3.1 设计原则 8](#_Toc22400)

[3.3.2 命名规则 8](#_Toc1387)

[4. 开发流程 9](#_Toc22853)

[4.1 JSON交付格式制定 9](#_Toc30230)

[4.1.1 JSON格式约定 9](#_Toc755)

[4.2 JSON格式内部评审以及外部评审 10](#_Toc27936)

[4.3 开发和单元调试 10](#_Toc3133)

[4.4 联合调试 10](#_Toc9763)

[5. 调试方法 11](#_Toc554)

[5.1 前置条件检查 11](#_Toc667)

[5.1.1 ubusd服务 11](#_Toc20135)

[5.1.2 Uhttpd服务 11](#_Toc8602)

[5.1.3 MGMT-UNIFIED进程 11](#_Toc26096)

[5.2 通过ubus工具调试 12](#_Toc31250)

[5.3 通过远端调试 12](#_Toc4141)

[5.4 升级和调试 13](#_Toc13446)

# 目的

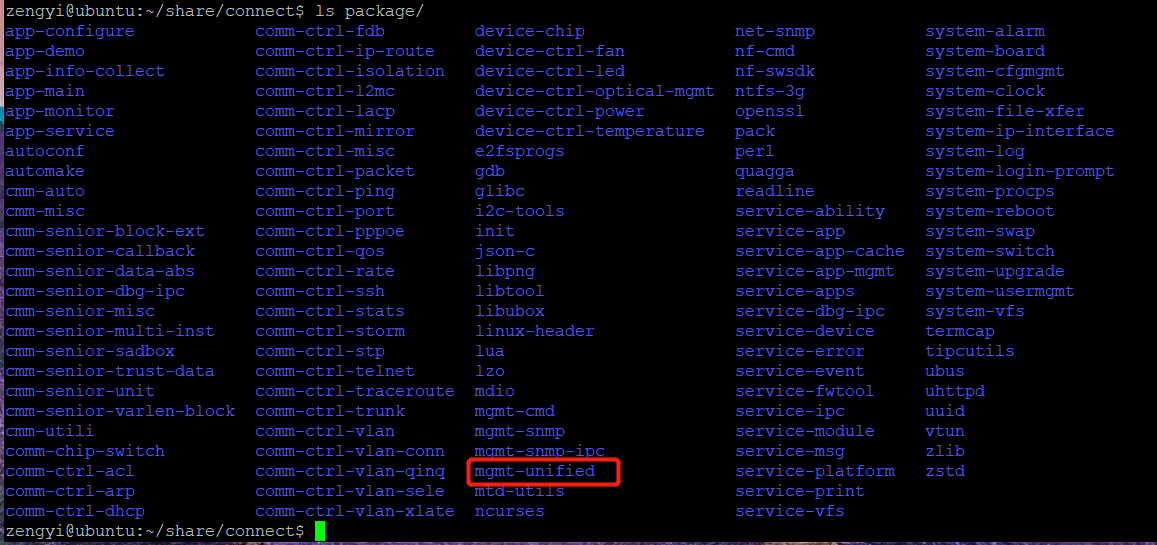
本文档基于《(03管理层统一数据管理-1.0.doc》文档的设计描述对管理层统一数据管理进行落地实现，并且指导业务模块编写对应代码。

# 代码目录结构

本章节介绍代码结构和代码目录。

## 组件包

管理层统一数据管理的所有代码都放在mgmt-unified组件包内，类似于mgmt-cmd和mgmt-snmp，这样设计主要还是考虑代码结构稳定性和工程工作量可控。



## 代码目录

基于设计代码目录如下，

|  |
| --- |
| zengyi@ubuntu:~/share/connect$ tree -L 2 ./package/mgmt-unified/code/  ./package/mgmt-unified/code/  ├── abs //系统控制接口屏蔽层  │   ├── Makefile  │   └── unified\_abs.c  ├── include //头文件目录  │   ├── abs  │   ├── json  │   └── ubus  ├── json //json数据转换层  │   ├── jubus //基于ubus格式的json数据转换  │   ├── Makefile  │   ├── unified\_json.c  │   └── wrap //通用json数据操作封装  ├── Makefile  ├── startup //main函数和初始化  │   ├── ct\_unified.c  │   └── Makefile  ├── ubus //ubus适配层  │   ├── Makefile  │   ├── unified\_ubus.c  │   ├── unified\_ubus\_demo.c  │   └── unified\_ubus\_demo.h  └── xml //xml数据转换层，暂时没有使用 |

## 业务需要关注的代码

根据2.2的代码目录，一个业务模块要做业务适配需要在abs，json和ubus目录下分别增加业务自己相关的业务代码。

# 代码编写

为了保证代码质量和一致性，在统一数据管理组件中的代码需要遵循一定的编写规则。

## 接口层代码编写原则

### 设计原则

接口层（abs）的主要目标是屏蔽系统实现细节，这一层对JSON数据适配层提供实现无关的接口和参数，对于Connect系统来说目前的IPC和API接口已经比较抽象，大部分业务在这一层可能只需要做简单的参数检查或者参数转换即可，但是由于IPC调用相对API调用来说会消耗更多的时间，如果需要获取大量数据可能会导致ubus等服务访问超时，这时业务需要在这里设计分段获取方法或者异步获取方法。

增加接口层的另外一个目的是如果以后系统增加数据库操作，只需修订接口层代码即可从而不会影响JSON和UBUS的业务逻辑。

### 命名规则

在这一层的所有“.c/.h”都应该以“unified\_abs\_”作为前缀。

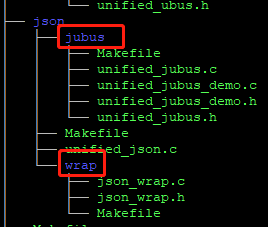
业务模块是否在这里增加子目录由自身的业务评估。

## JSON数据适配原则

### 设计原则

JSON本身是通用的数据格式定义，但是针对不同的管理服务会使用JSON定义出自身的格式要求，这导致了在JSON数据适配层有多套不同的数据格式解析。

目前JSON目录下有两个子目录，



Jubus目录下的代码就是针对我们Web服务通过ubus调用的格式处理，这些格式会被WEB前端使用，定时JSON的数据格式也是我们当前的工作之一。

wrap目录是针对JSON数据格式解析的通用接口函数，考虑采用通过封装的原因是，如果以后更换JSON解析库不用修改JSON业务代码，但是目前jubus和ubus耦合度较高，而且使用blobmsg库进行json格式解析，ubus，blob和libubox是OpenWrt中联合使用的三个套件，由于一般情况不考虑拆分单独使用，所以在Jubus中的代码直接使用blobmsg库进行JSON格式解析即可。

### 命名规则

在jubus目录下的“.c/.h”都应该以“unified\_jubus\_”作为前缀。

业务模块是否在这里增加子目录由自身的业务评估。

## Ubus代码编写原则

### 设计原则

ubus是基于unix socket的IPC服务模型，业务向ubus注册“对象”和“函数”，ubus在IPC通信时通过JSON格式携带“对象”和“函数”以便找到对应的函数调用。

在ubus这一层的业务代码需要向ubus注册业务的“对象”和“函数”（有点类似Connect系统中IPC注册需要提供的模块名和函数名），同时还需要为每个“函数”提供各自的参数格式定义（JSON）；业务模块向ubus提供的“函数”都在这一层实现，“函数”主要负责响应ubus的远程IPC访问以及将封装好的数据发送给ubus。

### 命名规则

在ubus目录下的“.c/.h”都应该以“unified\_ubus\_”作为前缀。

业务模块是否在这里增加子目录由自身的业务评估。

# 开发流程

开发流程大体要遵循“JSON交付格式制定”，“JSON格式内部评审以及外部评审”，“开发和单元调试”以及“联合调试”几个过程。

## JSON交付格式制定

JSON的格式最终会被Web前端以及其他客户端使用，作为对外开放的格式信息和命令行，MIB节点一样需要经过严格的审查和版本控制。

JSON由模块负责人制定，制定的格式要满足

1. 易于理解；
2. 易于使用；
3. 受制于当前ubus框架一次传输的数据不能太大，如果是传输数据表等信息要考虑通过某些关键字提供”get next”的格式方法，比如获取路由表，路由表默认16K太大，那么JSON格式可以增加”get next”同时传入上一次获取的最后一个子网IP，然后通过”get next”获得接下来的256个路由表项。

另外绝不能为了图方便提供模块内部信息。

### JSON格式约定

为了保证所有业务的格式风格大体一致，这里对JSON数据格式进行一些通用的基本约束，

1. Path约定

前端会通过路径来获取JSON对象和函数，比如，

|  |
| --- |
| https://192.168.111.87:443/ubus/1/unified\_ubus/async  ubus: uhttps web server识别JSON的关键字  1: 当前uhttps web server需要支持session id，目前被我写死为1  unified\_ubus: 对象  async: 函数 |

所以我们约定PATH的格式一定为：/ubus/1/对象/函数

对象和函数为各个业务自行定义，但是所有的对象都必须以unified\_开头。

另外对象和函数应该简明，既然清晰又要尽可能短，我们约定对象不要超过16个字符（不计算“unified\_”），函数也不要超过16个字符。

1. HTTP协议约定

对于HTTP协议，我们约定，

获取动作使用Method: GET

创建/删除/配置/清除动作使用Method: POST

针对GET和POST在5.3章节有不同的工具可以调试。

1. 数据主体

不管是获取还是配置，创建还是删除，每个JSON格式都是以对象和函数作为一个基本单元存在，即“对象”+“函数”有唯一一组“JSON格式”，JSON格式又分为前端传递到设备的格式，我们称为“查询”格式，以及设备返回的格式，我们称为“返回”格式。

对于“查询”格式，所有的JSON格式必须携带版本号，

|  |
| --- |
| “Ver”:“1” |

名字为Ver，而后面为int32的数字，这个数字代表了当前“对象”+“函数”对应的JSON格式的版本号，这样做的目的是考虑到后续格式调整带来的兼容性问题。

对于“返回”格式，通用的前缀部分为，

|  |
| --- |
| “Ver”:“1”  “Ret”：“0”  “RetMsg”：“错误提示信息字符串” |

“Ver”和查询相同，但是这是设备返回给前端的版本号，前端同样要考虑兼容性问题，比如前端携带版本号2获取参数，但是返回版本1，这是前端要知道使用版本1的格式进行解析。

“Ret”是本次操作的结果，值为int32数字，0代表成功，非0代表各种错误。

“RetMsg”是对错误的字符串描述，当Ret为0时RetMsg为“OK”，当Ret为非0时RetMsg对应特定的错误信息。

## JSON格式内部评审以及外部评审

格式制定以后进行内部评审，由部门负责人，模块负责人和开发人员一起评审。

内部评审通过以后，就当前的情况发送给前端人员评审，如果前端人员没有意见即可以开发。

## 开发和单元调试

开发人员根据第3章节约束进行接口适配，并完成单元测试（必须在web端验证）。

## 联合调试

开发完成以后提交代码，由部门负责人定时统一打版本提供给前端人员进行联调，当联调出现问题可以自行通过升级联调。

联调通过以后对应接口标记并固定版本。

# 调试方法

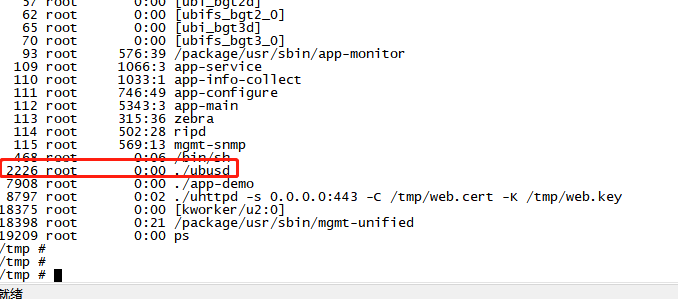
本章节介绍Mgmt unified组件的调试方法。

## 前置条件检查

在进行业务调试前先确定相关服务是否正常，

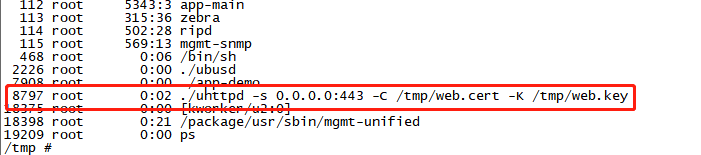
### ubusd服务

ubus服务由ubusd进程提供，首先要检查ubusd是否正常运行，使用ps查看，



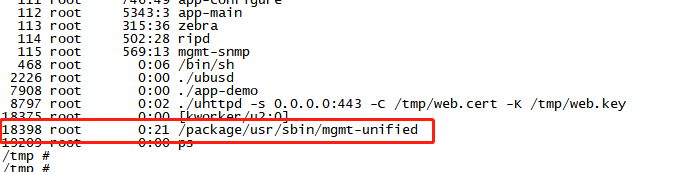
### Uhttpd服务

理论上只需要ubusd即可使用ubus工具配合调试，但是这样调试不一定能发现时序或者超时等问题，完整的测试还需要通过Web Server服务，检查web server是否正常运行，使用ps查看，



### MGMT-UNIFIED进程

我们的业务都在mgmt-unifed进程中实现，所以在进行调试前也需要关注mgmt-unified进程是否存在。



## 通过ubus工具调试

在进行Web server拉通调试前，可以通过ubus进行简单检查和验证，ubus是ubus组件提供的一个工具，可以通过命令查看当前注册的对象和函数，并且可以通过命令发起访问调用，

|  |
| --- |
| //查看当前对象和函数  / # /package/usr/sbin/ubus list  session  unified\_ubus  //触发对象和函数访问  / # ./ubus call unified\_bus async '{"id":1,"msg":"zengyi"}' |

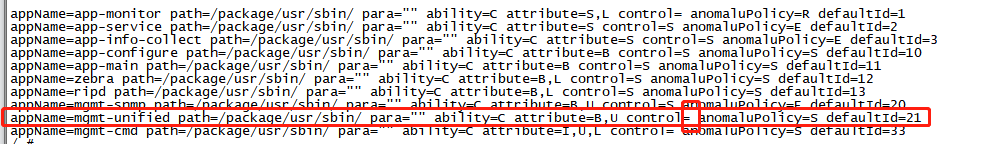
## 通过远端调试

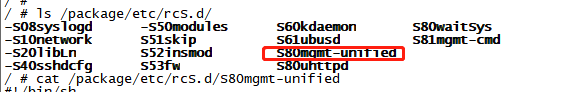
在浏览器或者主机linux系统上，

|  |
| --- |
| zengyi~/share/ubus$ wget --no-check-certificate --post-data='{"id":1,"msg":"zengyi"}' https://192.168.111.87:443/ubus/1/test/hello  --2022-09-07 00:10:45-- https://192.168.111.87/ubus/1/test/hello  Connecting to 192.168.111.87:443... connected.  WARNING: cannot verify 192.168.111.87's certificate, issued by ‘CN=DEBI,O=DEBI,ST=SICUAN,C=CN’:  Unable to locally verify the issuer's authority.  WARNING: certificate common name ‘DEBI’ doesn't match requested host name ‘192.168.111.87’.  HTTP request sent, awaiting response... 200 OK  Length: 49 [application/json]  Saving to: ‘hello’  hello 100%[==================================================>] 49 --.-KB/s in 0s  2022-09-07 00:10:46 (7.11 MB/s) - ‘hello’ saved [49/49]  zengyi~/share/ubus$ cat ./hello  {  "message": "test received a message: zengyi"  }zengyi~/share/ubus$ |

## 升级和调试

为了加快调试，目前将mgmt-unified进程调整为独立启动，即不受monitor启动引导，由其自身的脚本启动，这样做的好处是调试业务时可以将其杀掉再重新拉起，不用重启整个设备以及其他进程，





假设设备已经稳定，重新刷新mgmt-unified进程的步骤为，

1. 编译代码

|  |
| --- |
| zengyi~/share/connect$ ./cppf-s.sh dc mgmt-unified |

1. 打包代码

|  |
| --- |
| zengyi~/share/connect$./cppf-s.sh pack mgmt-unified  zengyi~/share/connect$ ls  build configure CPPF-S cppf-s.sh image.tar.gz package script target tmp toolchain tools  zengyi~/share/connect$ |

1. 将组件包image.tar.gz刷进系统

|  |
| --- |
| cd /tmp  rm -f image.tar.gz  rm -rf image  ifconfig eth0 192.168.111.87 //注意：不要用我的IP  tftp -g -l image.tar.gz 192.168.111.88 //注意：不要用我的IP  tar xzf image.tar.gz  cp -rf ./image/\* /package/  sync |

1. 杀掉mgmt-unified并重启

|  |
| --- |
| cd /tmp  rm -f image.tar.gz  rm -rf image  ifconfig eth0 192.168.111.87 //注意：不要用我的IP  tftp -g -l image.tar.gz 192.168.111.88 //注意：不要用我的IP  tar xzf image.tar.gz  cp -rf ./image/\* /package/  sync |

目前还存在的一些问题，

1. 设备断电以后，不知道什么原因（可能是运算量的问题）TLS初始化特别慢，导致Web Server要启动很长时间，大概至少5分钟，

|  |
| --- |
| port = 50 , hss\_num = 12  port = 49 , hss\_num = 12  system no ready, command wait...  port = 48 , hss\_num = 12  set vid 1 tpid 0 fail !  system no ready, command wait...  system no ready, command wait...  system no ready, command wait...  mgmt unified init complete.  system no ready, command wait...  Web server: starting...  Web server: tls init...  ws4484\_fan\_lvl\_set,175, modify fan level to:4, duty\_cycle:24000.  User Access Verification  Username:adminad  Password:\*\*\*\*\*[CT-V]: User [admin] login!  ...  Web server: tls init complete  Web server: certificate load...  Web server: certificate load success  Web server: complete. //一定要出现complete，Web Server才会正常工作 |

1. Mgmt-unified杀掉拉起稍微等一下再进行ubus访问，大概30秒左右，因为Mgmt-unified进程虽然不被Monitor进程管理但是它还需要加入域，

|  |
| --- |
| 93 root 1:26 /package/usr/sbin/app-monitor  96 root 0:00 /package/usr/sbin/ubusd  99 root 0:40 /package/usr/sbin/mgmt-unified  120 root 1:42 app-service  121 root 2:06 app-info-collect  122 root 1:34 app-configure  124 root 10:12 app-main  125 root 0:40 zebra  126 root 1:01 ripd  127 root 1:07 mgmt-snmp  487 root 0:00 /bin/sh  489 root 0:01 /package/usr/sbin/uhttpd -s 0.0.0.0:443 -C /package/etc/u  517 root 0:00 ps  / #  / #  / #  / # kill -9 99  / # /package/usr/sbin/mgmt-unified &  / # mgmt unified init complete. //因为打印重定向一些其他原因，这个打印不一定有，如果有这个打印就说明可以使用了。 |