LEPEIWARE CFG编程手册

# CFG介绍

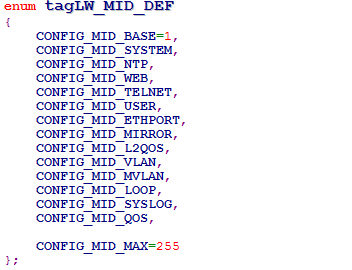
CFG 是配置管理层统称，用户通过此层面添加删除系统配置。

# Cfgoid

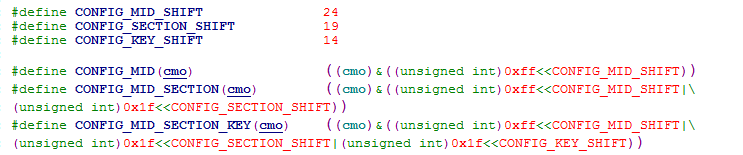
## Cfgoid 定义

一个OID分为模块module,段或表section,关键字key,索引index 四部分组成。

模块定义如下：



Cfgoid 定义如下：



举例：

定义一个syslog的各项配置

首先 添加

CONFIG\_MID\_SYSLOG 枚举

然后定义模块cfgoid



定义section OID



定义KEYid：



注意：

为了定义一个属性的ID，其所在module和section的OID也要定义

Module oid的secion,key,index字段全部是0

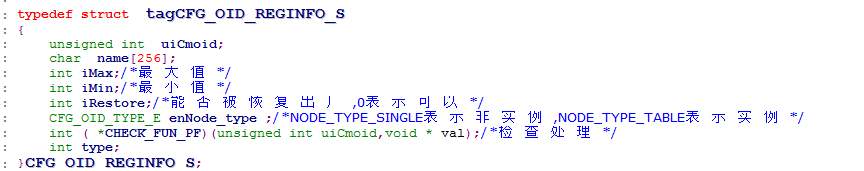
Section oid 的key和index 字段全部为0

Index 并不需要定义，使用时才用到

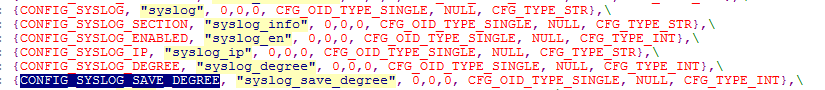
## Cfgoid 注册

Cfgoid 定义完成后，还不能使用，需要添加到注册表中

注册表定义如下：



举例：



注意：

Oid的名称不能带有数字，字母之间不能有空格，名称不能和其他配置的名称冲突

如果一个配置数据表象节点，则此配置对应的KEYOID，SECTIONOID对应的table 属性全部都要设置为CFG\_OID\_TYPE\_TABLE。

如果不是表节点则全部设置为CFG\_OID\_TYPE\_SINGLE

Moduleoid 全部为CFG\_OID\_TYPE\_SINGLE

## 添加出场配置文件

添加了一个新配置项，对应的出场配置文件需要添加，出场配置文件为

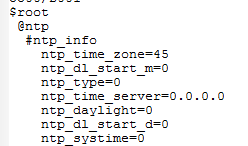
Product/xxx/config/yyy/startup 中

Xxx为：产品线名

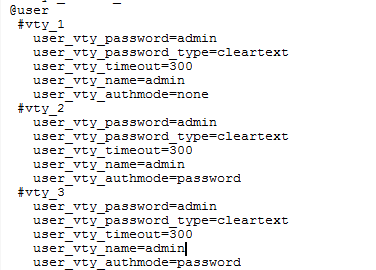
Yyy为：产品名

举例：

添加一个非表节点的配置



添加一个表象节点的配置



注意：

如果是全局配置则添加在root接口下面，如果需要存在特定接口下面，要找到对应的接口添加

模块名以@开头，段或者表项名以#开头

这些名称都是OID注册时的名称，不能随便写

如果是表象节点，则段名=注册的Section名称+编号，编号从1开始

# CFG API

## 初始化函数

int Cfginit(int flag);

所有模块如果需要和CFG打交道，初始时都要调用此函数。

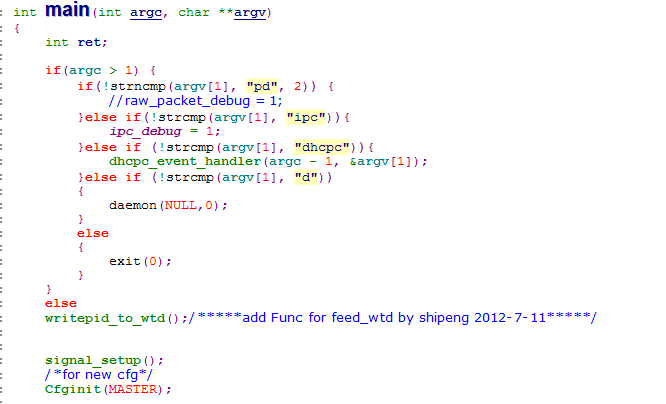
Flag 为

#define MASTER 1

#define SLAVER 0

只有master 初始化时设置为 MASTER ,其他所有模块都为SLAVER

举例：



## 配置读取函数

int cfg\_getval(int ifindex,unsigned int oid,void \*val,void\* default\_val,unsigned retlen);

ifindex 接口索引，根据配置存在的位置去指定

oid 为对应的oid

val 为buffer

default 为 默认配置，如果获取不到对应的值，则val返回值为default指定的值

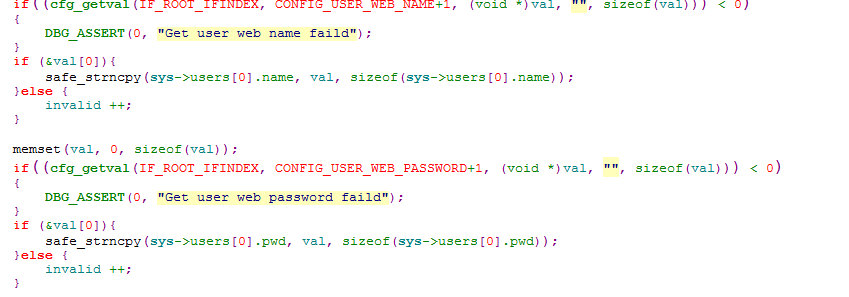
retlen 为buffer大小

注意：

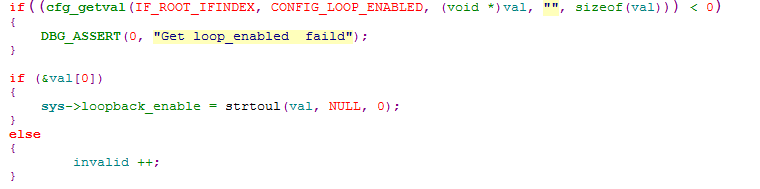
val ，default 仅支持字符串

举例：

获取表象节点配置



获取非表象节点配置：



## 配置设置函数

注意：此接口只能在master 中使用

int master\_cfg\_setval(int ifindex,unsigned int oid,void \*val)

参数说明

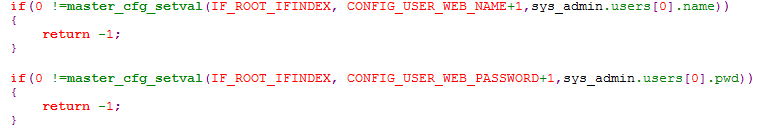
ifindex 接口索引，根据配置存在的位置去指定

oid 为对应的oid

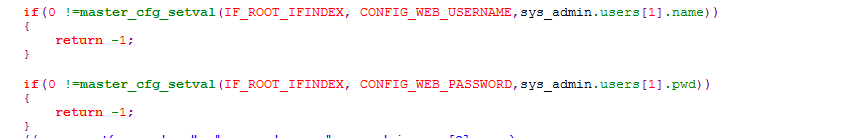
val 为设置的buffer

举例：

设置表象节点



设置非表象节点



## 配置删除函数

注意：此接口只能在master 中使用

int master\_cfg\_del(int ifindex,unsigned int oid)

ifindex 接口索引，根据配置存在的位置去指定

oid 为对应的oid

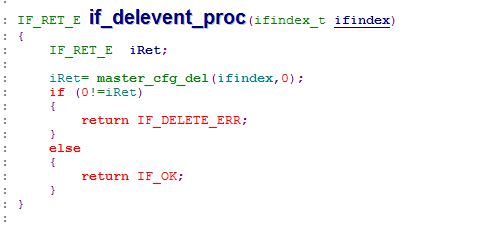
说明：

如果要删除一个模块的全部配置，oid为模块OID

如果要删除接口下面的所有配置 ，oid为0

如果要删除一个section下面的全部配置，oid为section oid

举例：



LEPEIWARE 接口编程手册

# ifm介绍

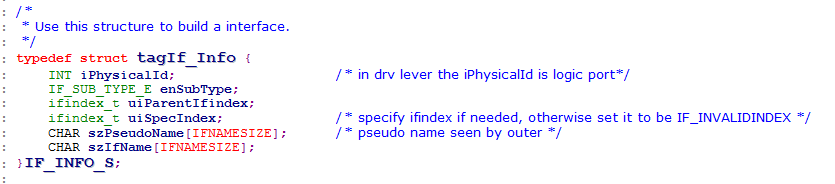
ifm 是用户态接口抽象层，此抽象层为编程用户提供统一的接口读取，设置，遍历，删除，创建等接口，屏蔽LINUX 设备层，方便用户开发。

# 接口API

## 接口创建函数

IF\_RET\_E IF\_CreatInterface( IN IF\_INFO\_S \* pstInfo, OUT ifindex\_t \* puiIfIndex );

入参IF\_INFO\_S 定义如下



enSubType 指定创建接口类型，具体定义在lw\_if\_pub.h中

uiParentIfindex 为父接口索引，存在父接口的，要填写此参数，例如L3VLAN和ETH接口往往存在父子关系。不存在则设置为IF\_INVALID\_IFINDEX

uiSpecIndex 为指定接口索引，如果此参数不为IF\_INVALID\_IFINDEX，系统则为用户创建的接口分配指定的索引，如果设置IF\_INVALID\_IFINDEX，则系统自动分配接口索引

szPseudoName 为伪接口名，用户可以随意设置，接口创建成功后，用户也可以更改

szIfName 为接口名，此接口名一旦设置后，后面将不能变更，其和ifindex都是此接口的

身份标志

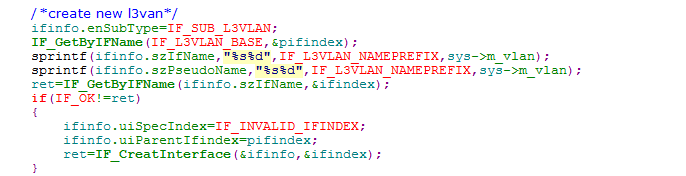
szIfName 的命名有一套规范，其中ETH网口，其名称是netdevice设备名，其他接口命名如下：

L3VLAN IF\_L3VLAN\_NAMEPREFIX+vid  
ETHPORT IF\_ETHPORT\_NAMEPREFIX+logicport

puiIfIndex 为出参，接口创建成功后，系统分配的接口索引

注意：此接口只能在master中使用，一般此接口只用在三层逻辑接口方面，其他物理接口系统会自动创建

举例：



## 接口删除函数

IF\_RET\_E IF\_DeleteInterface( IN ifindex\_t uiIfIndex );

uiIfIndex 为要删除接口的索引

注意： 如果此接口存在子接口，则要先删除字接口，否则无法删除

## 接口属性获取函数

IF\_RET\_E IF\_GetAttr( IN ifindex\_t uiIfindex, IN IF\_ATTR\_E enAttrId, INOUT VOID \* pValue ,IN UINT uiLen);

参数说明：

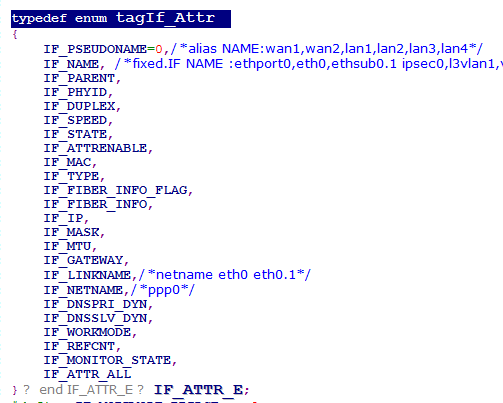
uiIfindex 接口索引

enAttrId 要获取的属性ID

pValue 返回值的BUFFER

uiLen BUFFER 的大小

可以获取的属性ID如下：

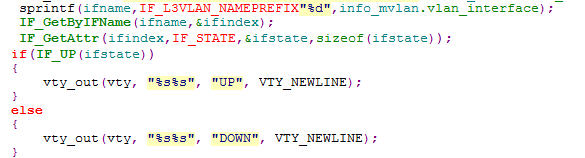


接口中存在都是实际生效的数据，例如网口可以配置为静态，也可以配置为动态获取

，其中的IP就是实际生效的IP信息

通过此接口可以获取接口状态

例如：



## 接口属性设置函数

IF\_RET\_E IF\_SetAttr( IN ifindex\_t uiIfindex, IN IF\_ATTR\_E iAttrId, IN VOID \* pValue,IN UINT uiLen );

参数说明：

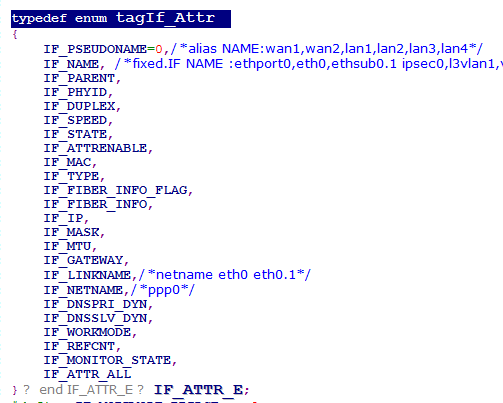
uiIfindex 接口索引

enAttrId 要获取的属性ID

pValue 设置值的BUFFER

uiLen BUFFER 的大小

可以设置的属性ID如下：



注意：

此接口只能在MASTER中调用，普通用户一般不要去设置。让master根据系统状态去设置。

## 接口遍历函数

#define IF\_GetFirstIfindex(enSubType,puiIfindex) IF\_GetFirstIfindexByDomain(enSubType,IF\_DOMAIN\_NORMAL,puiIfindex)

获取指定类型接口的首个接口

参数说明：

enSubType 要遍历的接口类型

puiIfindex 返回的接口索引

IF\_RET\_E IF\_GetNextIfindex( IN IF\_SUB\_TYPE\_E enSubType,INOUT ifindex\_t \* puiIfindex );

获取指定类型接口的下一个接口，

enSubType 要遍历的接口类型

puiIfindex 入参时，表示当前接口，出参时，表示获取的下一个接口

如果出参 puiIfindex 为IF\_INVALID\_IFINDEX，表示没有接口存在

举例：

static int timer\_handle\_vlanif\_monitor\_check(timer\_element\_t \*timer, void \*data)

{

INT iRet = 0;

ifindex\_t uiIfindex = 0;

UINT32 uiVid = 0;

UINT32 uiStatus=0;

IF\_STATUS\_E logstate = IFM\_MAX;

CHAR szBuf[BUF\_SIZE128]={0};

//CHAR szName[IFNAMESIZE];

CHAR szIfname[IFNAMESIZE];

CHAR szPseduname[BUF\_SIZE128];

iRet = IF\_GetFirstIfindex(IF\_SUB\_L3VLAN,&uiIfindex);

while(IF\_OK == iRet)

{

(VOID)cfg\_getval(uiIfindex,CONFIG\_IF\_ENABLE,szBuf,NULL,sizeof(szBuf));

if(strcmp(szBuf,ENUMTOSTR(IF\_MOD\_ENABLE))==0)

{

memset(szIfname,0,sizeof(szIfname));

if(IF\_OK!=IF\_GetAttr(uiIfindex,IF\_NAME,szIfname,sizeof(szIfname)))

{

iRet = IF\_GetNextIfindex(IF\_SUB\_L3VLAN,&uiIfindex);

continue;

}

sscanf(szIfname,IF\_L3VLAN\_NAMEPREFIX"%u",&uiVid);

#ifdef ONU\_STYLE

Drv\_GetVlanStatus((vlan\_id\_t)0, &uiStatus);

#else

Drv\_GetVlanStatus((vlan\_id\_t)uiVid, &uiStatus);

#endif

//printf("%s is %d.. \n",szPseduname,uiStatus);

if(UP == uiStatus)

{

/\*查看虚接口状态\*/

IF\_GetAttr(uiIfindex, IF\_STATE, &logstate, sizeof(IF\_STATUS\_E));

/\*虚接口UP，不做处理\*/

if(IF\_UP(logstate))

{

iRet = IF\_GetNextIfindex(IF\_SUB\_L3VLAN,&uiIfindex);

continue;

}

logstate = IFM\_NOIPGET;

IF\_SetAttr(uiIfindex, IF\_STATE, &logstate, sizeof(logstate));

memset(szPseduname,0,sizeof(szPseduname));

IF\_GetCliName(uiIfindex,szPseduname,sizeof(szPseduname));

ifm\_call\_notifiers(uiIfindex,IF\_NOTIFY\_PHYUP,NULL);

printf("%s is up.. \n",szPseduname);

}

else

{

/\*查看虚接口状态\*/

(VOID)IF\_GetAttr(uiIfindex, IF\_STATE, &logstate, sizeof(IF\_STATUS\_E));

/\*虚接口DOWN，不做处理\*/

if(!IF\_UP(logstate))

{

iRet = IF\_GetNextIfindex(IF\_SUB\_L3VLAN,&uiIfindex);

continue;

}

logstate = IFM\_PHYLINKDOWN;

IF\_SetAttr(uiIfindex, IF\_STATE, &logstate, sizeof(logstate));

/\*添加LOG信息\*/

memset(szPseduname,0,sizeof(szPseduname));

IF\_GetCliName(uiIfindex,szPseduname,sizeof(szPseduname));

ifm\_call\_notifiers(uiIfindex,IF\_NOTIFY\_PHYDOWN,NULL);

printf("%s is down.. \n",szPseduname);

}

}

iRet = IF\_GetNextIfindex(IF\_SUB\_L3VLAN,&uiIfindex);

}

return 0;

}

## 接口获取函数

#define IF\_GetByPseudoName(name,puiIfIndex) IF\_GetByPseudoNameDomain(name,IF\_DOMAIN\_NORMAL,puiIfIndex)

根据伪接口名获取接口索引

#define IF\_GetByIFName( name, puiIfIndex ) IF\_GetByIFNameDomain(name,IF\_DOMAIN\_NORMAL,puiIfIndex)

根据接口名获取索引

举例：



注意：

一般都是使用IF\_GetByIFName，如果用户自己定义了伪名，可以通过IF\_GetByPseudoName

获取

## 接口事件相关函数

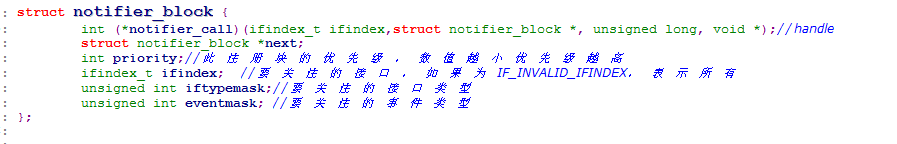
int ifm\_notifier\_register(struct notifier\_block \*nb);

注册关注事件

int ifm\_notifier\_unregister(struct notifier\_block \*nb);

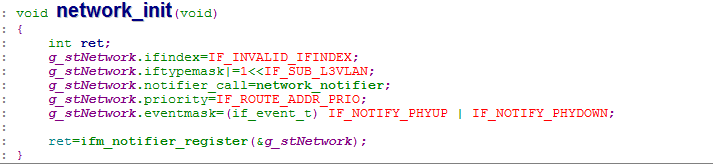
注销关注事件

事件注册块定义如下：



举例：

例如根据接口变化去配置网络地址



LEPEIWARE FAQ

1. 调用接口删除时会失败，是怎么回事？

请假查是否存在关联的子接口，如果存在，需要先删除子接口，然后再删除此接口

1. WEB 能否不通过RPC从master 获取数据？

可以，如果自己组织数据，可以直接通过cfg\_get 接口从cfg中获取原始配置

1. master中，想shutdown 一个接口怎么做？

直接调用 IF\_DISABLE ,如果要开启调用IF\_ENABLE

1. 对于ethernet port，逻辑端口是从几开始？

是从1，开始，如果要遍历逻辑端口，调用LgcPortFor(lport)

1. 怎么从web/cli 接口名转化成实际接口名？

调用IF\_Userstr2ifame

1. 怎么从获取接口的web或者CLI名称？

IF\_RET\_E IF\_GetCliName(IN ifindex\_t uiIfindex,INOUT CHAR \*pszName,IN UINT uiSize);

IF\_RET\_E IF\_GetWebName(IN ifindex\_t uiIfindex,INOUT CHAR \*pszName,IN UINT uiSize);

1. 不想使用目前的CLI/web 接口提示名称，怎么修改？

请修改lw\_if\_pub.h 中

#define CLI\_GIGAPORTPREFIX "gigabitethernet"

#define CLI\_FASTPORTPREFIX "ethernet"

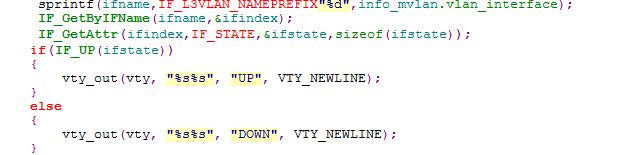
#define WEB\_GIGAPORTPREFIX "GE"

#define WEB\_FASTPORTPREFIX "FE"

1. 想查看一个接口是UP还是DOWN怎么做？

直接读取接口的IF\_STATE属性即可

举例：



1. 怎么修改出场配置文件？

只要在config/xxx/startup 中修改即可

Xxx 是产品名

1. 有没有cfg,ifm debug 接口？

命令\_debugshell 进入shell

输入cfg 或者ifm即有相关提示

1. 目前的用户进程没有控制终端，无法输出调试信息，该怎么做？

请修改 config/xxx/rc 启动脚本

把master d &

修改为 master & 即可

12 怎么判断当前接口类型，例如是千兆的还是百兆的？

如果知道逻辑端口号，PT\_SP((PORT\_TYPE((port\_num\_t)uiIfid)))==SP\_1000 进行判断

13 当前的ethport的物理属性有没有相关配置文件的？

有，在config/xxx/port\_config.c 中

14 每次make all ，感觉速度很慢，不想每次都全部编译，有没有快速的命令？

有，如果只编译某个模块，然后内存调试，只要执行make software-xxx;make rootfs;make ramfs;即可

15 怎么加载内存调试版本？

boot 下执行run ramdisk

16 boot 下烧image,会破坏产测数据，怎么恢复？

执行完 uumage 命令后，请立即执行saveenv