**相关命令行**

**vlan创建和删除**

**create vlan {1-4094} active**

**vlan <1-4094>**

**no vlan all**

**no vlan {1-4094}**

vlan的创建有两个命令行可以实现。用create方式可以创建单个vlan，也可以批量创建，用vlan <1-4094>命令只能创建单个vlan。

vlan参数输入时注意命令行定义的格式，如果是大括号“{ }”定义的vlan值，说明该参数可以输入多个vlan，格式例如：

create vlan 100,200,300 active

用来批量创建vlan 100，200和300三个vlan。如下格式可以批量创建vlan 101到110十个vlan：

create vlan 101-110 active

vlan的删除使用no命令：

no vlan { all | vlan-list }

可以批量，或者删除单个vlan。例如：

no vlan 100 //删除单个vlan 100

no vlan 101-110 //删除vlan 101到110

**接口切换二三层模式**

**portswitch**

**no portswitch**

接口空配置时，默认为三层口模式，配置二层相关配置时前需要将物理接口配置为二层接口模式，接口下使用命令portswitch。切回到三层模式：no portswitch。

**接口配置access和trunk模式**

**switchport mode access**

**switchport mode trunk**

使用portswitch命令将接口切到二层模式后，默认是access模式，可以不用配置switchport mode来配置access模式，如果接口本来是trunk模式，可以用这条命令切到access模式。

**接口配置access vlan**

**switchport access vlan <1-4094>**

**no switchport access vlan**

接口需要首先配置为access模式，才能配置access vlan。配置后，接口对收到的报文进行vlan检查，如果是untag报文，在入接口打上access vlan，然后在access vlan内转发；如果报文是带着tag的，则检查报文的tag与access vlan是否一致，如果一致，则允许报文通过正常转发，否则在入接口处丢弃报文。

其他接口转发过来的需要从本接口出去的报文，如果携带的vlan与access vlan一致，则剥掉vlan后从本接口发出去。

删除access vlan使用no命令。

**access模式下配置允许通过的vlan**

**switchport access egress-allowed vlan {1-4094} confirm**

**switchport access egress-allowed vlan add {1-4094}**

**switchport access egress-allowed vlan remove {1-4094}**

**switchport access egress-allowed vlan all confirm**

如果是首次配置允许通过的vlan，使用第一条命令就行。

如果是已经配置过允许通过的vlan列表，需要增加或者删除一些vlan，使用第二条或者第三条命令。

如果是允许所有vlan通过，使用第四条命令。一般没有必要用这条命令。

配置允许通过的vlan，可以与access vlan相同，不影响接口的vlan检查。如果，配置access vlan为100，允许通过的vlan为200，那么接口收到vlan为200的报文后正常转发。其他接口转发过来的需要从本接口出去的报文，如果携带的vlan为200，则剥掉vlan 200然后从本接口发出去。

接口可以不配置access vlan，只配置本条命令。实现的效果就是，接口只接收在允许通过的vlan列表中的报文，untag报文直接丢弃。出接口剥掉vlan转发。

**接口配置native vlan**

**switchport trunk native vlan <1-4094>**

**no switchport trunk native vlan**

native vlan在access模式下也能配置，且不报错，但是需要接口切到trunk模式下才会生效。这个为什么不加一个限制，不让在access模式下配置？

接口的native vlan实现的效果与access模式下的access vlan一致。接收的报文如果是untag的，则打上native vlan进行转发；

如果收到的tag报文与native vlan一致，正常在native vlan内转发；

如果收到的tag报文与native vlan不一致，在入接口丢弃；

其他接口转发过来的需要从本接口出去的报文，如果携带的vlan与native vlan一致，则剥掉vlan后从本接口发出去。

在88-E设备上敲命令，除了vlan id参数，还有两个参数可以选择：

switchport trunk native vlan tagged

switchport trunk native vlan untagged

打开ssp的debug打印，配置这两条命令没有打印输出，需要看下平台的逻辑，后面有时间再确认一下。

删除配置的native vlan使用no命令。

**trunk模式下配置允许通过的vlan**

**switchport trunk allowed vlan {1-4094} confirm**

**switchport trunk allowed vlan add {1-4094}**

**switchport trunk allowed vlan remove {1-4094}**

**switchport trunk allowed vlan all confirm**

**no switchport trunk allowed vlan**

如果是首次配置允许通过的vlan，使用第一条命令就行。

如果是已经配置过允许通过的vlan列表，需要增加或者删除一些vlan，使用第二条或者第三条命令。

如果是允许所有vlan通过，使用第四条命令。一般没有必要用这条命令。

删除所有配置的allowed vlan使用no命令即可。

**trunk模式下配置允许通过的untagged vlan**

**switchport trunk untagged vlan {1-4094} confirm**

**switchport trunk untagged vlan add {1-4094}**

**switchport trunk untagged vlan remove {1-4094}**

**switchport trunk untagged vlan all confirm**

**no switchport trunk untagged vlan**

该条命令实现的效果是接口入方向允许配置的vlan报文正常转发，出方向如果报文的tag在配置的untagged vlan列表中，则会剥掉vlan以untagged方式从本接口转发出去。

为了实现入方向正常检查vlan并转发，在配置本条命令时会下发一条allowed vlan配置。比如配置允许untagged vlan 200命令，查看接口配置会出现一条allowed vlan：

switchport trunk allowed vlan 200

switchport trunk untagged vlan 200

删除所有允许通过的untagged vlan使用no命令，但是注意，还要删除自动添加的allowed vlan。如果不想多敲一条删除allowed vlan的命令，可以直接使用删除allowed vlan的no命令。比如删除上面配置的允许untagged vlan 200，使用untagged vlan命令对应的no命令删除后，allowed vlan配置仍然存在：

switchport trunk allowed vlan 200

如果直接使用allowed vlan命令对应的no命令删除的话，allowed和untagged配置均被删除：

no switchport trunk allowed vlan

**接口禁止tag或untagged类型的报文**

**switchport reject-frame tagged**

**switchport reject-frame untagged**

**no switchport reject-frame**

接口配置该命令后，会直接丢弃所有该类型的报文，即使接口配置了allowed vlan或者其他vlan。比如，trunk模式下配置允许vlan 100通过，再配置：

switchport reject-frame tagged

来禁止所有tag报文通过。最后的效果就是接口收到的vlan 100报文也会在接口丢弃，无法正常转发。

删除配置使用no命令即可。

**配置框图**

//TODO

**主要数据结构**

**ROS\_hwApiModuleReg\_t**

typedef struct

{

ROS\_HWSTRUCT\_GET\_FN funcStructGet; //查询的函数指针

ROS\_HWSTRUCT\_SET\_FN funcStructSet; //配置的函数指针

ROS\_HWSTRUCT\_DEL\_FN funcStructDel; //删除的函数指针

} ROS\_hwApiModuleReg\_t;

三个函数指针声明：

typedef STATUS (\*ROS\_HWSTRUCT\_GET\_FN)(int iCmd, uint\_t uiIfindex, void\* pStruct)

typedef STATUS (\*ROS\_HWSTRUCT\_SET\_FN)(int iCmd, uint\_t uiIfindex, void\* pStruct)

typedef STATUS (\*ROS\_HWSTRUCT\_DEL\_FN)(int iCmd, uint\_t uiIfindex, void\* pStruct)

ssp各个模块注册平台消息的处理函数时，指向对应的配置、查询和删除函数。

**HWAPI\_VLAN\_PORT\_VALUE\_CMD**

平台与ssp处理消息的命令字，ssp用来区分具体的vlan消息类型，是配置、删除或者其他消息类型。比如：

ROS\_HW\_SS\_VLAN\_CFG //用来创建、删除vlan

ROS\_HW\_SS\_VLAN\_MEMBER //用来配置接口加入vlan，或者从vlan中删除

ROS\_HW\_PORT\_VLAN\_ID //用来配置接口默认vlan

ROS\_HW\_VLAN\_MAC\_LEARN\_ENABLE //用来设置基于vlan的mac学习使能

ROS\_HW\_VLAN\_PORT\_REJECT\_FRAME //用来设置接口禁止通过的报文类型

ROS\_HW\_PV\_VLAN\_MAC\_BASED\_VLAN\_ENABLE //用来设置接口基于mac划分vlan的功能

ROS\_HW\_SS\_VLAN\_MAC\_ASSOCIATE\_OPERATION //设置mac与vlan的关联关系

ROS\_HW\_PV\_VLAN\_SUBNET\_BASED\_VLAN\_ENABLE //用来设置接口基于IP子网划分vlan的功能

ROS\_HW\_SS\_VLAN\_SUBNET\_ASSOCIATE\_OPERATION //设置mac与IP子网的关联关系

ROS\_HW\_SS\_VLAN\_PROTOCOL\_ASSOCIATE\_OPERATION //设置mac与协议的关联关系

ROS\_HW\_SS\_VLAN\_PVLAN\_PORT\_CFG //设置接口的pvlan模式

**rosvlanlst\_t**

暂时没搜到定义，从使用上来看是一个数组，用bit位是否置位来表示对应vlan是否需要设置。

**vlanInfoCfgHw\_t**

typedef struct

{

uint32\_t ulVlan; //vlan id

ushort\_t usMemberCount; //成员端口数

ushort\_t usUntagCount; //untagged端口数

uint32\_t \*pulMembers; //链表，成员端口

uint32\_t \*pulUntags; //链表，untagged端口

} vlanInfoCfgHw\_t;

数据结构用来保存单个vlan接口信息，有哪些端口加入该vlan，有哪些端口是以untag方式加入该vlan。

vlanMacLearnHw\_t

typedef struct

{

uint32\_t ulVlan;

rosEnable\_e eMacLearnEnable;

}vlanMacLearnHw\_t;

设置基于vlan的mac学习使能时使用，其中rosEnable\_e是枚举类型，用来传入使能或非使能。

**VlanPortInfo\_t（ssp定义使用）**

typedef struct

{

uchar\_t \*pucTagPortList;

uchar\_t \*pucUnTagPortList;

}VlanPortInfo\_t;

ssp层定义使用的数据接口，平台通过vlanInfoCfgHw\_t传入要配置的vlan和接口，ssp配置完成后将vlan有哪些接口保存到本地的VlanPortInfo\_t数据结构。

**pVlanPortCfgHw\_t**

typedef struct pVlanPortCfgHw\_s

{

ulong\_t ulPriVlan; //主vlan

ulong\_t ulSecVlan; //辅助vlan

ulong\_t ulMode; //端口模式

} pVlanPortCfgHw\_t;

平台设置接口pvlan使用该数据结构传入参数，端口混杂模式，隔离模式或者团体模式等，到ssp后，ssp使用本层定义的数据结构VlanPortCfg\_t里面的pvlan\_port\_t保存pvlan信息。

**VlanPortCfg\_t（ssp定义使用）**

typedef struct

{

uint\_t pvid;

uint\_t tagDiscard;

BOOL macVlanEna;

BOOL ipVlanEna;

BOOL pvlanEna;

pvlan\_port\_t pvlan;

} VlanPortCfg\_t

数据结构用来保存接口下的pvlan设置，比如：基于mac/ip划分的vlan功能是否使能，pvlan的主vlan，辅助vlan，端口模式等。在ssp层通过该数据结构类型的全局变量g\_pVlanPortCfg保存。

**pvlan\_port\_t（ssp定义使用）**

typedef struct pvlan\_port\_s

{

lstNode\_t stNode;

int port;

int fid\_vlan; //主vlan id

int sec\_vlan; //辅助vlan id

int mode; //端口模式：混杂、隔离、团体

int isolate\_id; //共32个，ssp本地根据辅助vlan分配

} pvlan\_port\_t;

数据结构保存平台使用pVlanPortCfgHw\_t传到ssp的配置信息。

如果端口配置团体模式，isolate id在0-15内分配；

如果端口配置为隔离模式，isolate id在0-31内分配，0表示disable。

**vlan\_mac\_hw\_t**

typedef struct vlan\_mac\_hw\_s

{

uint\_t ulVlan;

uint\_t ulPriority;

uchar\_t aucMac[6];

} vlan\_mac\_hw\_t;

基于mac划分vlan时平台使用

**vlan\_subnet\_hw\_t**

typedef struct vlan\_subnet\_hw\_s

{

uint\_t ulVlan;

uint\_t ulPriority;

rosIpAddrType\_e eIpType; //ip地址类型，ipv4，ipv6或者dns等

rosIpAddr\_t ipAddress;

rosIpAddr\_t ipMask;

} vlan\_subnet\_hw\_t;

基于IP子网划分vlan时平台使用

**vlan\_protocol\_hw\_t**

typedef struct vlan\_protocol\_hw\_s

{

uint32\_t ulVlan;

uint32\_t ulEtherType; //协议类型

uint32\_t ulIfindex;

} vlan\_protocol\_hw\_t;

基于协议类型划分vlan时平台使用

**vlan\_class\_data\_t（ssp定义使用）**

typedef struct

{

lstNode\_t stNode; //ssp使用全局变量保存，这里作为链表头

uint\_t vlanClassType; //划分类型，基于mac、ip或者是协议

uint\_t snmp\_port; //接口id

uint\_t protocol; //协议类型，基于协议划分时使用

uchar\_t macSa[6]; //mac地址，基于mac地址划分时使用

uint\_t newSvlan; //vlan id

uint\_t newScos; //优先级

uint\_t ipSa; //ip地址，基于ip子网划分时使用

uint\_t ipSaMask; //ip子网掩码

uint\_t ulEid; //ssp分配的scl entry表项

uint\_t ulIpv6Eid; //暂时没有使用

} vlan\_class\_data\_t;

ssp使用全局链表m\_listVlanClass保存平台下发的配置。

vlan\_mac\_hw\_t： 基于mac划分vlan时平台使用

vlan\_subnet\_hw\_t： 基于IP子网划分vlan时平台使用

vlan\_protocol\_hw\_t：基于协议类型划分vlan时平台使用

最终下发的配置在ssp都以vlan\_class\_data\_t类型保存到全局链表m\_listVlanClass。

**ssp函数接口**

**bcmIfmVlanSwitchStructSet()：ssp配置实现接口函数**

bcmIfmVlanSwitchStructSet(int iCmd, uint\_t uiIfindex, void\* pSwitchStruct)

bcmIfmVlanSwitchStructGet(int iCmd, uint\_t uiIfindex, void\* pSwitchStruct)

bcmIfmVlanSwitchStructDelete(int iCmd, uint\_t uiIfindex, void\* pSwitchStruct)

这三个函数分别实现平台的配置下发、查询和配置删除，接口需要在ritp进程启动时注册消息的收发。

**bcmIfmVlanHwapiInit()：各个模块hwapi初始化接口函数**

STATUS bcmIfmVlanHwapiInit()

spp中每个功能模块都应该有各自的初始化函数，ritp进程启动初始化ssp的各个模块时调用该函数。这个函数的功能主要是定义数据结构：

ROS\_hwApiModuleReg\_t

的配置、查询和删除接口，通过库函数：

STATUS ROS\_hwapiModuleFuncReg(ROS\_hwApiModuleReg\_t \*phwReg, ROS\_MID eModId)

注册上面的配置、查询和删除接口。

**ROS\_hwapiModuleFuncReg：注册平台消息收发的接口函数**

STATUS ROS\_hwapiModuleFuncReg(ROS\_hwApiModuleReg\_t \*phwReg, ROS\_MID eModId)

结构体ROS\_hwApiModuleReg\_t声明了三个函数指针：配置、查询和删除，用来指向ssp各个模块的配置、查询和删除接口。

ROS\_MID则是模块ID，用来标识区分各个模块。比如ROS\_MID\_VLAN就是vlan功能模块，ROS\_MID\_MAC就是mac功能模块，等等。

这个函数属于平台部分的功能，其函数体内容就是将ROS\_hwApiModuleReg\_t传入的配置、查询、删除三个接口函数，以及ROS\_MID模块编号，赋值给一个全局变量。至于这个hwapi模块怎么通过这个全局变量实现平台和ssp层的交互，暂时没有深入阅读。

**bcm\_tms\_vlan\_init()：初始化配置函数**

void bcm\_tms\_vlan\_init(void \*pData)

ritp进程启动时调用函数对vlan相关配置进行初始化，参数pData暂时没有用。

初始化内容包括：

（1）创建所有4095个vlan

（2）接口配置缺省vlan

（3）创建消息队列

（4）ssp用到的全局变量初始化

（5）回调函数注册

（6）sdk的scl tcam相关表项初始化

ctc\_vlan\_trunk\_change\_call**()：trunk组成员变化的回调函数**

STATUS ctc\_vlan\_trunk\_change\_call(uint\_t trunkId, uint\_t portId, BOOL isAdd)

函数在bcm\_tms\_vlan\_init初始化时注册。功能是当trunk组新增端口时调用，用来将trunk组的vlan相关配置同步到新增端口。或者删除成员端口时，将删除的端口vlan相关配置恢复到默认状态。

**ctc\_ssp\_vlanLst\_create()：vlan创建函数**

STATUS ctc\_ssp\_vlanLst\_create(rosvlanlst\_t vlanList)

命令行create vlan {1-4094} active和vlan <1-4094>创建vlan时，就是通过这个接口函数。参数vlanList是一个数组，用bit位形式传入需要创建的vlan id。单个vlan的创建通过下面的函数实现：

STATUS ctc\_ssp\_vlan\_create(uint\_t ulVlan)

创建单个vlan的函数，函数体调用SDK接口函数。

**ctc\_ssp\_vlanLst\_delete()：vlan删除函数**

STATUS ctc\_ssp\_vlanLst\_delete(rosvlanlst\_t vlanList)

STATUS ctc\_ssp\_vlan\_delete(uint\_t ulVlan)

参数定义与创建函数一致。

**ctc\_tms\_vlan\_learn\_enable\_set()：基于vlan的mac学习使能函数**

STATUS ctc\_tms\_vlan\_learn\_enable\_set(vlanMacLearnHw\_t \*pVlanMacLearn)

配置基于vlan的mac学习使能，参数的数据结构只有两个成员，一个是vlan id，另一个是动作，enable或者disable。

**ctc\_ssp\_vlan\_port\_add()：接口配置vlan函数**

STATUS ctc\_ssp\_vlan\_port\_add(vlanInfoCfgHw\_t \*stVlanEntry)

LOCAL STATUS ctc\_ssp\_vlan\_port\_add\_inner(uint\_t vlanId, uint\_t \*pIfindex, uint\_t portNum, VlanPortInfo\_t \*pVlanPortlist, BOOL btag)

**ctc\_ssp\_vlan\_port\_remove()：接口删除vlan函数**

STATUS ctc\_ssp\_vlan\_port\_remove(vlanInfoCfgHw\_t \*stVlanEntry)

LOCAL STATUS ctc\_ssp\_vlan\_port\_remove\_inner(uint\_t vlanId, uint\_t \*pIfindex, uint\_t portNum, VlanPortInfo\_t \*pVlanPortlist, BOOL btag)

**ctc\_vlan\_port\_qtag\_set()：接口配置默认vlan函数**

STATUS ctc\_vlan\_port\_qtag\_set(uint\_t ulSnmpPort, uint\_t ulVlanId)

接口配置access vlan和native vlan时调用。删除时传入ulVlanId值为0。

**ctc\_vlan\_port\_frame\_reject\_type\_set()：接口禁止tag或untag报文函数**

STATUS ctc\_vlan\_port\_frame\_reject\_type\_set(uint\_t ulSnmpPort, uint\_t ulMode)

**ctc\_ssp\_vlan\_mac\_class\_set()：基于mac划分vlan函数**

STATUS ctc\_ssp\_vlan\_mac\_class\_enable(uint\_t ulSnmpPort, uint\_t enable)

STATUS ctc\_ssp\_vlan\_mac\_class\_set(vlan\_mac\_hw\_t \*stVlanMacHwInfo, BOOL isAdd)

**ctc\_ssp\_vlan\_subnet\_class\_set()：基于IP划分vlan函数**

STATUS ctc\_ssp\_vlan\_subnet\_class\_enable(uint\_t ulSnmpPort, uint\_t enable)

STATUS ctc\_ssp\_vlan\_subnet\_class\_set(vlan\_subnet\_hw\_t \*stVlanIpHwInfo, BOOL isAdd)

**ctc\_ssp\_vlan\_protocol\_class\_set()：基于协议划分vlan函数**

STATUS ctc\_ssp\_vlan\_protocol\_class\_set(vlan\_protocol\_hw\_t \*stVlanProtcHwInfo, BOOL isAdd)

**ctc\_vlan\_pvlan\_port\_set()：接口配置pvlan函数**

STATUS ctc\_vlan\_pvlan\_port\_set(uint\_t ulSnmpPort, pVlanPortCfgHw\_t \*pVlanPortHw)

接口配置pvlan相关时调用。参数pVlanPortCfgHw\_t定义主vlan、辅助vlan和端口模式三个成员变量。

**ctc\_vlan\_pvlan\_port\_del()接口删除pvlan配置函数**

STATUS ctc\_vlan\_pvlan\_port\_del(uint\_t ulSnmpPort, pVlanPortCfgHw\_t \*pVlanPortHw)

**sdk函数接口**

//TODO

**基本功能测试报告**

