NAME

pci_iov_schema, **pci_iov_schema_alloc_node**, **pci_iov_schema_add_bool**, **pci_iov_schema_add_string**, **pci_iov_schema_add_uint8**, **pci_iov_schema_add_uint16**, **pci_iov_schema_add_uint32**, **pci_iov_schema_add_uint64**, **pci_iov_schema_add_unicast_mac** - PCI SR-IOV config schema interface

SYNOPSIS

#include <machine/stdarg.h>
#include <sys/nv.h>
#include <sys/iov_schema.h>

*nvlist_t **
pci_iov_schema_alloc_node(*void*);

void
pci_iov_schema_add_bool(*nvlist_t *schema, const char *name, uint32_t flags, int defaultVal*);

void
pci_iov_schema_add_string(*nvlist_t *schema, const char *name, uint32_t flags, const char *defaultVal*);

void
pci_iov_schema_add_uint8(*nvlist_t *schema, const char *name, uint32_t flags, uint8_t defaultVal*);

void
pci_iov_schema_add_uint16(*nvlist_t *schema, const char *name, uint32_t flags, uint16_t defaultVal*);

void
pci_iov_schema_add_uint32(*nvlist_t *schema, const char *name, uint32_t flags, uint32_t defaultVal*);

void
pci_iov_schema_add_uint64(*nvlist_t *schema, const char *name, uint32_t flags, uint64_t defaultVal*);

void
pci_iov_schema_add_unicast_mac(*nvlist_t *schema, const char *name, uint32_t flags,*
*const uint8_t *defaultVal*);

DESCRIPTION

The PCI Single-Root I/O Virtualization (SR-IOV) configuration schema is a data structure that describes the device-specific configuration parameters that a PF driver will accept when SR-IOV is enabled on the PF device. Each PF driver defines two schema instances: the PF schema and the VF schema. The PF schema describes configuration that applies to the PF device as a whole. The VF schema describes

configuration that applies to an individual VF device. Different VF devices may have different configuration applied to them, as long as the configuration for each VF conforms to the VF schema.

A PF driver builds a configuration schema by first allocating a schema node and then adding configuration parameter specifications to the schema. The configuration parameter specification consists of a name and a value type.

Configuration parameter names are case-insensitive. It is an error to specify two or more configuration parameters with the same name. It is also an error to specific a configuration parameter that uses the same name as a configuration parameter used by the SR-IOV infrastructure. See iovctl.conf(5) for documentation of all configuration parameters used by the SR-IOV infrastructure.

The parameter type constrains the possible values that the configuration parameter may take.

A configuration parameter may be specified as a required parameter by setting the IOV_SCHEMA_REQUIRED flag in the *flags* argument. Required parameters must be specified by the user when SR-IOV is enabled. If the user does not specify a required parameter, the SR-IOV infrastructure will abort the request to enable SR-IOV and return an error to the user.

Alternatively, a configuration parameter may be given a default value by setting the IOV_SCHEMA_HASDEFAULT flag in the *flags* argument. If a configuration parameter has a default value but the user has not specified a value for that parameter, then the SR-IOV infrastructure will apply *defaultVal* for that parameter in the configuration before passing it to the PF driver. It is an error for the value of the *defaultVal* parameter to not conform to the restrictions of the specified type. If this flag is not specified then the *defaultVal* argument is ignored. This flag is not compatible with the IOV_SCHEMA_REQUIRED flag; it is an error to specify both on the same parameter.

The SR-IOV infrastructure guarantees that all configuration parameters that are either specified as required or given a default value will be present in the configuration passed to the PF driver. Configuration parameters that are neither specified as required nor given a default value are optional and may or may not be present in the configuration passed to the PF driver.

It is highly recommended that a PF driver reserve the use of optional parameters for configuration that is truly optional. For example, a Network Interface PF device might have the option to encapsulate all traffic to and from a VF device in a vlan tag. The PF driver could expose that option as a "vlan" parameter accepting an integer argument specifying the vlan tag. In this case, it would be appropriate to set the "vlan" parameter as an optional parameter as it would be legitimate for a VF to be configured to have no vlan tagging enabled at all.

Alternatively, if the PF device had an boolean option that controlled whether the VF was allowed to

change its MAC address, it would not be appropriate to set this parameter as optional. The PF driver must either allow the MAC to change or not, so it would be more appropriate for the PF driver to document the default behaviour by specifying a default value in the schema (or potentially force the user to make the choice by setting the parameter to be required).

Configuration parameters that have security implications must default to the most secure configuration possible.

All device-specific configuration parameters must be documented in the manual page for the PF driver, or in a separate manual page that is cross-referenced from the main driver manual page.

It is not necessary for a PF driver to check for failure from any of these functions. If an error occurs, it is flagged in the schema. The pci_iov_attach(9) function checks for this error and will fail to initialize SR-IOV on the PF device if an error is set in the schema. If this occurs, it is recommended that the PF driver still succeed in attaching and run with SR-IOV disabled on the device.

The **pci_iov_schema_alloc_node**() function is used to allocate an empty configuration schema. It is not necessary to check for failure from this function. The SR-IOV infrastructure will gracefully handle failure to allocate a schema and will simply not enable SR-IOV on the PF device.

The **pci_iov_schema_add_bool**() function is used to specify a configuration parameter in the given schema with the name *name* and having a boolean type. Boolean values can only take the value true or false (1 or 0, respectively).

The **pci_iov_schema_add_string**() function is used to specify a configuration parameter in the given schema with the name *name* and having a string type. String values are standard C strings.

The **pci_iov_schema_add_uint8**() function is used to specify a configuration parameter in the given schema with the name *name* and having a *uint8_t* type. Values of type *uint8_t* are unsigned integers in the range 0 to 255, inclusive.

The **pci_iov_schema_add_uint16**() function is used to specify a configuration parameter in the given schema with the name *name* and having a *uint16_t* type. Values of type *uint16_t* are unsigned integers in the range 0 to 65535, inclusive.

The **pci_iov_schema_add_uint32**() function is used to specify a configuration parameter in the given schema with the name *name* and having a *uint32_t* type. Values of type *uint32_t* are unsigned integers in the range 0 to (2**32 - 1), inclusive.

The **pci_iov_schema_add_uint64**() function is used to specify a configuration parameter in the given

schema with the name *name* and having a *uint64_t* type. Values of type *uint64_t* are unsigned integers in the range 0 to (2**64 - 1), inclusive.

The **pci_iov_schema_add_unicast_mac**() function is used to specify a configuration parameter in the given schema with the name *name* and having a unicast-mac type. Values of type unicast-mac are binary values exactly 6 bytes long. The MAC address is guaranteed to not be a multicast or broadcast address.

RETURN VALUES

The **pci_iov_schema_alloc_node**() function returns a pointer to the allocated schema, or NULL if a failure occurs.

SEE ALSO

pci(9), PCI_IOV_ADD_VF(9), PCI_IOV_INIT(9)

AUTHORS

This manual page was written by Ryan Stone <rstone@FreeBSD.org>.