

Process for Adding SAI Extensions

|  |  |
| --- | --- |
| **Title** | **Process for Adding SAI Extensions** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **11/28/2016** |
| **Modified** | **02/23/2017** |
| **SAI-Version** | **V1.0** |

# SAI Extensions

SAI Extensions can be used for introducing custom attributes and experimental or extension modules.

* Custom attributes – specific to a particular SAI implementation
* Experimental/Extension SAI Modules

## Custom Attributes

Normally reserve a numeric range of 128 values for each SAI implementation that defines custom SAI attributes. Custom SAI attributes are defined in the header file of the module they belong to. For instance:

**saiport.h**

typedef enum \_sai\_port\_attr\_t

{

…

/\*\* Custom range base value \*/

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_SAI\_IMPL\_X\_START = SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_START,

SAI\_PORT\_ATTR\_CAPABILITY\_X,

…

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_SAI\_IMPL\_X\_END = SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_START + 0x7F

…

/\*\* End of custom range base \*/

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_END

} sai\_port\_attr\_t;

Do NOT use the word CUSTOM in the attribute name. It will make it easier to move it to the main stream attribute list if the OCP community agrees on it.

A file called saimodule\_extensions.h should contain enum values specific to extension attributes, e.g. saiport\_extensions.h. Please refer to usage notes below. Note that no new complex types are allowed as extensions, only new enum values.

## Experimental/Extension SAI Modules

Each experimental SAI Module must define its own header file, as any regular module.

The module **must** contain a doxygen warning, to state that it is an extension, and thus may not be supported by all SAI implementations.

The module description must state what SAI implementation(s) support this module.

/\*\*

…

\* @file saiextensionmodule.h

\*

\* @brief This module defines a …

\* **@description Supported only by Acme Corp. SAI Implementation**

\* **@warning This module is a SAI extension module**

/

Extension module APIs are added to sai.h file. The documentation of the module name must state that the module is an extension. However, do not use any indication that the module is an extension in the enum itself. It will make it easier to move it to the main stream API list if the OCP community agrees on it.

typedef enum \_sai\_api\_t

{

SAI\_API\_UNSPECIFIED = 0, /\*\*< unspecified api \*/

SAI\_API\_SWITCH = 1, /\*\*< sai\_switch\_api\_t \*/

…

**SAI\_API\_CUSTOM\_RANGE\_START = 0x1000,**

**/\*\* Extension module provided by Acme Corp. \*/**

**SAI\_API\_ACME\_MODULE = SAI\_API\_CUSTOM\_RANGE\_START + 1,**

} sai\_api\_t;

There is no requirement that module extension APIs be published (contents of saiextensionmodule.h). The only recommendation is that module extension API’s use the SAI API approach (create/get/set/delete, with key /value pairs for attributes).

## Usage Notes for Custom/Extension Attributes

Assume a vendor (vendor A) implements a port attribute that controls a port LED state. In this case, the ‘saiport.h’ has an LED State attribute

**saiport.h**

typedef enum \_sai\_port\_attr\_t

{

…

/\*\* Custom range base value \*/

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_START = 0x10000000,

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_SAI\_IMPL\_X\_START = SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_START,

**SAI\_PORT\_ATTR\_LED\_STATE**,

…

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_SAI\_IMPL\_X\_END = SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_START + 0x7F

…

/\*\* End of custom range base \*/

SAI\_PORT\_ATTR\_CUSTOM\_RANGE\_END

} sai\_port\_attr\_t;

If vendor B desires to implement a similar functionality, that vendor B should reuse the same attribute.

The attribute does not impact in any way the code provided by a Vendor C which may not support the attribute. However, the Host Adapter implementation needs to be able to distinguish cases where the SAI implementation does not support the new extension attribute.

**saiport\_extensions.h**

/\*\*

\* @brief Attribute data for #SAI\_PORT\_ATTR\_LED\_STATE

\*/

typedef enum \_sai\_port\_led\_state\_t

{

/\*\* Unknown \*/

SAI\_PORT\_LED\_STATE\_UNKNOWN,

/\*\* ON \*/

SAI\_PORT\_LED\_ON,

/\*\* Down \*/

SAI\_PORT\_LED\_OFF,

} sai\_port\_led\_state\_t;

**#include <saiport.h>**

**#include <saiport\_extensions.h>**

sai\_attribute\_t attr;

attr.id = **SAI\_PORT\_ATTR\_LED\_STATE;**

attr.u32.value = **SAI\_PORT\_LED\_ON**;

sai\_status\_t status = sai\_set\_port\_attribute(port\_id, &attr);

Note. Including the “extensions” header file is intended. Source files which refer extensions can be easily identified, e.g. grep –r extensions.h . !

## Usage Notes for Extension Modules

Vendor A may provide an MPLS API specific to vendor A, and Vendor B may provide a different MPLS API (of course, unless they agree on the same API!).

In this case, sai.h contains:

typedef enum \_sai\_api\_t

{

SAI\_API\_UNSPECIFIED = 0, /\*\*< unspecified api \*/

SAI\_API\_SWITCH = 1, /\*\*< sai\_switch\_api\_t \*/

…

**SAI\_API\_CUSTOM\_RANGE\_START = 0x1000,**

**/\*\* MPLS Extension module provided by Acme-A Corp. \*/**

**SAI\_API\_ACME\_A\_MPLS = SAI\_API\_CUSTOM\_RANGE\_START + 1,**

**/\*\* MPLS Extension module provided by Acme-B Corp. \*/**

**SAI\_API\_ACME\_B\_MPLS = SAI\_API\_CUSTOM\_RANGE\_START + 2,**

} sai\_api\_t;

Each vendor provides its own MPLS API, say:

sai\_acme\_a\_mpls\_extension.h

sai\_acme\_b\_mpls\_extension.h

Since the two API’s are mutually exclusive, the Host Adapter needs to provide two separate implementations for MPLS, one for Vendor A, the other one for vendor B. The two extensions cannot be used at the same time at run time – this is not a serious limitation, since it is unlikely that NPU’s from different vendors are used for the same platform.

This is an expected - and desired outcome - since it allows vendor differentiation. Vendor A and Vendor B can define very different API’s for MPLS – or any other extension modules.