

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **Visibility** |
| **Authors** | **Mellanox** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **07/18/2016** |
| **SAI-Version** | **XXXX** |

1 Overview 3

2 Specification 3

3 Examples 3

4 References 3

# Overview

It becomes more and more complex to manage networks and network administrators need tools to understand a network behavior. It is needed to provide a basic information about network performance, identify a network bottlenecks and provide needed info for network optimization and future network planning.

Visibility mechanism provides following key components

* Resources that we want to make visible outside e.g. ports, queues, flows…
* Raw counters that we get from those objects: Rx counters, Tx Counters, Current Level, Max Level…
* Probes that can sample raw counter: histogram, thresholds…
* Collections – way we collect counters and probes: periodically, on-demad, triggered, snapshot…

Distribution mechanism – how keep and distribute collected information



# Specification

## Collector

typedef enum \_sai\_vis\_collector\_type\_attr\_t

{

SAI\_VIS\_COLLECTOR\_TYPE\_PERIODIC,

SAI\_VIS\_COLLECTOR\_TYPE\_ON\_DEMAND,

SAI\_VIS\_COLLECTOR\_TYPE\_TRIGGERED

}

typedef enum \_sai\_vis\_collector\_sampling\_type\_attr\_t

{

SAI\_VIS\_COLLECTOR\_SAMPLING\_SNAPSHOP,

SAI\_VIS\_COLLECTOR\_SAMPLING\_COARSE

}

typedef enum \_sai\_vis\_collector\_attr\_t

{

/\*\* collector type[sai\_vis\_collector\_type\_attr\_t] \*/

SAI\_VIS\_COLLECTOR\_TYPE,

/\*\* collector type[sai\_vis\_collector\_sampling\_attr\_t] \*/

SAI\_VIS\_COLLECTOR\_SAMPLING\_TYPE,

/\*\* collector period usec [uint64\_t] \*/

SAI\_VIS\_COLLECTOR\_PERIOD,

/\*\* collector period usec [sai\_object\_id] \*/

SAI\_VIS\_COLLECTOR\_TRIGGER,

/\*\* collector object list[sai\_object\_list\_t] \*/

SAI\_VIS\_COLLECTOR\_OBJECT\_LIST,

} sai\_ vis\_collector\_attr\_t ;

/\*\*

\* @brief Create Visibility collector

\*

\* @param[out] collector\_id Visibility collector Id

\* @param[in] attr\_count number of attributes

\* @param[in] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_vis\_ collector\_fn)(

\_Out\_ sai\_object\_id\_t\* collector\_id,

\_In\_ uint32\_t attr\_count,

\_In\_ const sai\_attribute\_t \*attr\_list

);

/\*\*

\* @brief Remove visibility collector

\*

\* @param[in] collector\_id visibility collector id to be removed.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_vis\_collector\_fn) (

\_In\_ sai\_object\_id\_t collector\_id

);

/\*\*

\* @brief Set attributes for visibility collector

\*

\* @param[in] vis\_distributor\_id Distributor Id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_vis­\_collector\_attribute\_fn)(

\_In\_ sai\_object\_id\_t vis\_collector\_id,

\_In\_ const sai\_attribute\_t \*attr

);

/\*\*

\* @brief Get attrbutes of collector

\*

\* @param[in] vis\_collector\_id collector id

\* @param[in] attr\_count number of attributes

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_ vis\_collector\_attribute\_fn)(

\_In\_ vis\_collector\_id\_t collector\_id ,

\_In\_ uint32\_t attr\_count,

\_Inout\_ sai\_attribute\_t \*attr\_list

);

/\*\*

\* @brief Sample collector objects

\*

\* @param[in] vis\_collector\_id collector id

\* @param[in] attr\_count number of attributes

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_sampel\_vis\_collector\_fn)(

\_In\_ vis\_collector\_id\_t collector\_id ,

);

### Collector API summary

typedef struct \_sai\_vis\_collector\_api\_t

{

sai\_create\_vis\_collector\_fn create\_vis\_collector;

sai\_remove\_vis\_collector\_fn remove\_vis\_collector;

sai\_set\_vis\_collector\_attribute\_fn set\_vis\_collector\_attribute;

sai\_get\_vis\_collector\_attribute\_fn get\_vis\_collector\_attribute;

sai\_sample\_vis\_collector\_attribute\_fn sample\_vis\_collector;

} sai\_vis\_collector\_api\_t;

## Distributor

typedef enum \_sai\_vis\_distributor\_transport\_attr\_t

{

SAI\_VIS\_DISTRIBUTOR\_TRANSPORT\_GRPC,

SAI\_VIS\_DISTRIBUTOR\_TRANSPORT\_REST,

SAI\_VIS\_DISTRIBUTOR\_TRANSPORT\_DMA

} sai\_vis\_distributor\_transport\_attr\_t;

typedef enum \_sai\_vis\_distributor\_type\_attr\_t

{

SAI\_VIS\_DISTRIBUTOR\_TYPE\_LOCAL,

SAI\_VIS\_DISTRIBUTOR\_TYPE\_STREAMING,

} sai\_vis\_distributor\_type\_attr\_t ;

typedef enum \_sai\_vis\_distributor\_format\_attr\_t

{

SAI\_VIS\_DISTRIBUTOR\_FORMAT\_JSON,

SAI\_VIS\_DISTRIBUTOR\_FORMAT\_PROTOBUF,

SAI\_VIS\_DISTRIBUTOR\_FORMAT\_XML,

SAI\_VIS\_DISTRIBUTOR\_FORMAT\_RAW

} sai\_vis\_distributor\_format\_attr\_t ;

typedef enum \_sai\_vis\_distributor\_period\_attr\_t

{

SAI\_VIS\_DISTRIBUTOR\_PERIOD\_CONTINUOUS,

SAI\_VIS\_DISTRIBUTOR\_PERIOD\_SIZE\_BASED,

SAI\_VIS\_DISTRIBUTOR\_FORMAT\_ON\_DEMAND,

} sai\_vis\_distributor\_period\_attr\_t ;

typedef enum \_sai\_vis\_distributor\_attr\_t

{

/\*\* distributor type[sai\_vis\_distributor\_type\_attr\_t], Default SAI\_VIS\_DISTRIBUTOR\_LOCAL\*/

SAI\_VIS\_DISTRIBUTOR\_TYPE = SAI\_VIS\_DISTRIBUTOR\_LOCAL,

/\*\* distributor transport [sai\_vis\_distributor\_transport\_attr\_t] \*/

SAI\_VIS\_DISTRIBUTOR\_TRANSPORT,

/\*\* distributor transport [sai\_vis\_distributor\_format\_attr\_t] \*/

SAI\_VIS\_DISTRIBUTOR\_FORMAT,

/\*\* distributor destination [sai\_ip\_address\_t] \*/

SAI\_VIS\_DISTRIBUTOR\_REMOTE\_DESTINATION

/\*\* memory region for local distributor [sai\_object\_id] \*/

SAI\_VIS\_DISTRIBUTOR\_MEMORY\_REGION

/\*\* distribution period [sai\_vis\_distributor\_period\_attr\_t] \*/

SAI\_VIS\_DISTRIBUTOR\_PERIOD,

} sai\_ vis\_distributor\_attr\_t ;

/\*\*

\* @brief Create Visibility distributor

\*

\* @param[out] distributor\_id Visibility distributor Id

\* @param[in] attr\_count number of attributes

\* @param[in] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_create\_vis\_distributor\_fn)(

\_Out\_ sai\_object\_id\_t\* distributor\_id,

\_In\_ uint32\_t attr\_count,

\_In\_ const sai\_attribute\_t \*attr\_list

);

/\*\*

\* @brief Remove visibility distributor

\*

\* @param[in] distributor\_id visibility distributor id to be removed.

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_remove\_vis\_distributor\_fn) (

\_In\_ sai\_object\_id\_t distributor\_id

);

/\*\*

\* @brief Set attributes for visibility distributor

\*

\* @param[in] vis\_distributor\_id Distributor Id

\* @param[in] attr attribute to set

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_vis\_distributor\_attribute\_fn)(

\_In\_ sai\_object\_id\_t vis\_distributor\_id,

\_In\_ const sai\_attribute\_t \*attr

);

/\*\*

\* @brief Get attrbutes of distributor

\*

\* @param[in] vis\_distributor\_id distributor id

\* @param[in] attr\_count number of attributes

\* @param[inout] attr\_list array of attributes

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_ vis\_distributor\_attribute\_fn)(

\_In\_ vis\_distributor\_id\_t distributor\_id ,

\_In\_ uint32\_t attr\_count,

\_Inout\_ sai\_attribute\_t \*attr\_list

);

/\*\*

\* @brief Send collected information to distributor

\*

\* @param[in] vis\_distributor\_id distributor id

\*

\* @return SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_send\_vis\_distributor\_fn)(

\_In\_ vis\_distributor\_id\_t distributor\_id ,

);

### Distributor API summary

typedef struct \_sai\_vis\_distributor\_api\_t

{

sai\_create\_vis\_distributor\_fn create\_ vis\_distributor;

sai\_remove\_vis\_distributor\_fn remove\_ vis\_distributor;

sai\_set\_vis\_distributor\_attribute\_fn set\_ vis\_distributor \_attribute;

sai\_get\_vis\_distributor\_attribute\_fn get\_ vis\_distributor \_attribute;

sai\_send\_vis\_distributor\_fn send\_ vis\_distributor ;

} sai\_vis\_distributor\_api\_t;

# Examples

## Local periodic statistics collector



### Description

We have a periodic collector that during 1 second collects all ports' counters and keeps each read set.

Distributor is capable to deliver all captured sets on demand in raw format to a memory region.

## Triggered by congestion buffer level collector



### Description

The collector is triggered by threshold object to snapshot all current levels of all TC on all ports. Once snapshot is done continuous streaming distributor will deliver the snapshot in gRPC/protobuf format to defined network address.

# References