BroadView - OpenNSL - SDK

JSON-RPC API

Bandaru Viswanath

19th September, 2016

Introduction

This document describes the proposed API for BroadView[1] to interact with OpenNSL & SDK using JSON-RPC messaging.

BroadView Instrumentation Agent will have a Southbound plugin which uses this JSON-RPC API, thus BroadView Instrumentation Agent is a Client of these API. Both SDK and the OpenNSL may implement these API. In other words, SDK and OpenNSL are the servers of these API. The server implementations must be able to support more than one client.

This document doesn't specify any transport or security / authentication mechanisms for this messaging. It is possible to easily add-on these mechanisms on the top of the proposed API.

This document is intended to be a Broadcom Internal Document and doesn't exercise any restraint in discussing internal / implementation details.

The following are the design goals for the API.

1. It must be possible to easily map current C-language API to the JSON-RPC API.
2. The JSON-RPC messaging should not introduce a large performance overhead.

To satisfy the first goal, this document ensures a mapping between the existing C API and the proposed JSON-RPC API. Such a mapping is documented along with the JSON-RPC API description.

To satisfy the second goal, the JSON-RPC API are designed to minimise the number of API invocations for carrying out any given task. The API can support multiple requests in a single invocation. The API also retrieves aggregated data where possible. Some of the error checking that is traditionally done on the Client is now mandated on the server, and therefore can be avoided on the Client.

All notifications sent by the Server include a timestamp in the form of a string. Typically this timestamp indicates the time at which the event has occurred.

[1] Only BST feature is supported currently.

This document doesn't describe any functionality of the ASIC. Rather, the mapping between the JSON API and the Existing C API is provided. Reader is also expected to be familiar with the JSON-RPC specification version 2.0.

API

Overview

API	Existing C API Reference	Comments
`configure-buffer-tracking`	opennsl_switch_control_set	This Json API configures only BST subset of the properties as supported by opennsl_switch_control_set().
`get-buffer-tracking-configuration`	opennsl_switch_control_get	This Json API retrieves only the BST properties as supported by opennsl_switch_control_get().
`get-unit-info`	opennsl_info_get opennsl_attach_check	Use of opennsl_attach_check() is discouraged. Server returns error if unit is invalid.
`get-max-units`	opennsl_attach_max	
`get-port-config`	opennsl_port_config_get	
`notify-switch-event`	opennsl_switch_event_register opennsl_switch_event_unregister	
`get-global-portid`	opennsl_port_gport_get	Use of opennsl_port_gport_get() is discouraged. All JSON API accept local port number as input, and conversion to gport is done at Server.
`get-queue-counters`	opennsl_cosq_stat_get	Can return multiple counters in an aggregated form
`clear-queue-counters`	opennsl_cosq_stat_set	Can clear a selected counter, or all counters.
`get-buffer-statistics`	opennsl_cosq_bst_stat_get	Can return multiple statistics in an aggregated form
`clear-buffer-statistics`	opennsl_cosq_bst_stat_clear	
`sync-buffer-statistics`	opennsl_cosq_bst_stat_sync	Use of this API can be avoided by specifying `sync` option for the `get-`

		buffer-statistics API.
configure-buffer-thresholds	opennsl_cosq_bst_profile_set	Can configure threshold values for different buffers in an aggregated form
get-buffer-thresholds	opennsl_cosq_bst_profile_get	Can return threshold values for different buffers in an aggregated form
clear-buffer-thresholds	opennsl_cosq_bst_profile_set	

configure-buffer-tracking

The **configure-buffer-tracking** API configures the Buffer Tracking and the Tracking Mode on the ASIC.

There are three parameters[2] associated with this API, described below. At least one of the three parameters must be part of the configuration request.

Parameter	Type	Description
enable-buffer-tracking	Boolean	opennslSwitchBstEnable Optional Parameter.
buffer-tracking-mode	String Enum	opennslSwitchBstTrackingMode Optional parameter. Accepts either "peak" or "current" as the values
enable-snapshots	Boolean	opennslSwitchBstSnapshotEnable Optional parameter.

A sample JSON-RPC request is shown below.

```
{
      "jsonrpc": "2.0",
      "method": "configure-buffer-tracking",
      "unit": 1,
      "params": {
            "enable-buffer-tracking" : 1,
            "buffer-tracking-mode" : "peak"
      },
      "id": 1
}
```

[2] The unit number parameter, "unit", is part of all API and is not described / counted separately.

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

get-buffer-tracking-configuration

The **get-buffer-tracking-configuration** API retrieves the Buffer Tracking configuration on the ASIC. The configuration is in terms of the three parameters described as part of **configure-buffer-tracking** API. All of the three parameters are returned as part of the response.

A sample JSON-RPC request is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-buffer-tracking-configuration",
	"unit": 1,
	"params": { },
	"id": 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns the configuration as the response. A Sample response is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-buffer-tracking-configuration",
	"unit": 1,
	"result": {
		"enable-buffer-tracking" : 1,
		"buffer-tracking-mode" : "peak",
		"enable-snapshots" : 1
	},
	"id": 1
}
```

get-unit-info

The **get-unit-info** API retrieves the Device details for the ASIC. There are two parameters returned as part of the response, shown below. All of the parameters are returned as part of the response.

Parameter	Type	Description
device	Integer	opennsl_info_t::device

revision	Integer	opennsl_info_t::revision

A sample JSON-RPC request is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-unit-info",
	"unit": 1,
	"params": { },
	"id": 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns the parameters as the response. A Sample response is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-unit-info",
	"unit": 1,
	"result": {
		"device" : 56780,
		"revision" : 2
	},
	"id": 1
}
```

get-max-units

The **`get-max-units`** API retrieves the highest currently attached unit number. A single parameter is returned as part of the response, shown below.

Parameter	Type	Description
max-unit	Integer	opennsl_attach_max

A sample JSON-RPC request is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-max-units",
	"unit": 1,
	"params": { },
	"id": 1
}
```

The "unit" parameter sent as part of the JSON is ignored. Server may return an error if either the validation or execution of the API fails. Otherwise, it returns the parameter as the response. A Sample response is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-max-units",
	"unit": 1,
	"result": {
		"max-unit" : 2
	},
	"id": 1
}
```

get-port-config

The **get-port-config** API retrieves the port configuration for the specified device. Port configuration is specified in terms of bitmap arrays for various ports.

There are six parameters returned as part of the response, shown below. All of the parameters are returned as part of the response.

Parameter	Type	Description
ge-bmp	Integer Array	opennsl_port_config_t :: ge An array of 8 bytes indicating the bitmap for all GE ports.
xe-bmp	Integer Array	opennsl_port_config_t :: xe An array of 8 bytes indicating the bitmap for all 10G ports.
ce-bmp	Integer Array	opennsl_port_config_t :: ce An array of 8 bytes indicating the bitmap for all 100G ports.
port-bmp	Integer Array	opennsl_port_config_t :: port An array of 8 bytes indicating the bitmap for all front-panel ports.
cpu-bmp	Integer Array	opennsl_port_config_t :: cpu An array of 8 bytes indicating the bitmap for all CPU ports.
all-bmp	Integer Array	opennsl_port_config_t :: all An array of 8 bytes indicating the bitmap for all ports.

A sample JSON-RPC request is shown below.

```
{
      "jsonrpc": "2.0",
      "method": "get-port-config",
      "unit": 1,
      "params": { },
      "id": 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns the parameters as the response. A Sample response is shown below.

```
{
  "jsonrpc" : "2.0",
  "method" : "get-port-config",
  "unit" : 1,
  "result" : {
       "ge-bmp" : [4564, 0, 0, 0, 0, 45, 0, 0],
       "xe-bmp" : [0, 4564, 0, 0, 0, 0, 0, 23],
       "ce-bmp" : [0, 0, 1234, 0, 0, 0, 0, 0],
       "port-bmp" : [4564, 4564, 1234, 0, 0, 45, 0, 23],
       "cpu-bmp" : [0, 0, 0, 0, 0, 0, 1, 0],
       "all-bmp" : [4564, 4564, 1234, 0, 0, 45, 1, 23]
       },
  "id" : 1,
}
```

get-global-portid

The **`get-global-portid`** API retrieves the GPORT Id corresponding to the supplied local port number.

For performance reasons, Clients are encouraged not to use this API. Where there is need for client to specify port number as a parameter, the JSON-API takes either the local port number or a GPORT ID as an input. Server handles the conversion to GPORT Id. This API is provided for backward compatibility purposes only.

The local port number is the single input parameters for the API.

Parameter	Type	Description
`local-port`	Integer	Input for `opennsl_port_gport_get()`

A sample JSON-RPC request is shown below.

```
{
       "jsonrpc": "2.0",
       "method": "get-global-portid",
       "unit": 1,
       "params": {
              "local-port": 2
       },
       "id": 1
}
```

Server returns the GPORT ID as part of the response. Both of these parameters are shown below.

Parameter	Type	Description
global-port-id	Integer	Output from opennsl_port_gport_get()

A Sample response is shown below.

```
{
  "jsonrpc" : "2.0",
  "method" : "get-global-portid",
  "unit" : 1,
  "result" : {
      "global-port-id" : 56432
   },
  "id" : 1,
}
```

Server may return an error if either the validation or execution of the API fails.

get-queue-counters

The **get-queue-counters** API retrieves counters associated with specified queues. This API supports multiple sources to be specified for retrieving counters with in the same API invocation. Each source specifies the set of ports/queues from which the counters are requested.

The counter values returned by the server are cumulative in nature.

There are multiple ways of specifying the the sources.

1. Counters for specific queues, for a given port.
2. Counters for all (each of the) queues, for a given port.
3. Port level counters - Sum of counters of all queues for the specified port.

The API input parameters are shown below.

Parameter	Type	Description
`port`	Integer	Optional Parameter, indicating the Local port number. If this parameter as well as `gport` (below) is not specified, data is returned for all ports.
`gport`	Integer	Global port Id as derived from `opennsl_port_gport_get()`. Only one - either `port` or `gport` - can be specified in a given source. If this parameter as well as `port` (above) is not specified, data is returned for all ports.
`counter`	String Enumeration	Indicates the counter of interest. Maps to entries of the Enum opennsl_cosq_stat_t. Only "`discard-counters`" and "`out-packets`" are supported.
`counter-options`	Array of Strings from a String Enumeration	Optional parameters. It specifies the options for the returned data. Valid values are "`cumulative`". "`unicast`" and "`multicast`". - "`cumulative`" indicates that the cumulative counter data must be returned for the specified port. If this option is specified, the parameter `queue` ignored. If this option is unspecified, individual queue counters for the specified queues are returned. - "`unicast`" indicates that only unicast counters need to be returned. - "`multicast`" indicates that only multicast counters need to be returned. - If neither of the "`unicast`" and "`multicast`" options are specified, both the types of counters are returned.
`queue`	Integer Array	Optional element. If this param is specified as an integer array, each of the array elements must be valid queue numbers, and Server returns the values for the specified queues.

		If this parameter is not specified, data is returned for all queues.

A sample JSON-RPC request is shown below.

```
{
	"jsonrpc": "2.0",
	"method": "get-queue-counters",
	"unit": 1,
	"params": {
	    "counter" : "discard-counters",
	    "counter-options" : ["unicast"],
	    "sourcelist : [
	        { "port" : 2, "queue" : [2,3,4] },
	    ]
	},

	"id": 1
}
```

A Sample response for the above request is shown below.

```
{
	"id" : 1,
	"jsonrpc" : "2.0",
	"method" : "get-queue-counters",
	"result" : {
		"counter" : "discard-counters",
		"sources" : [
			{
				"port" : 2,
				"queue" : [
					{
					"queue" : 2,
					"unicast" : 1234
					},
					{
					"queue" : 3,
					"unicast" : 34
					},
					{
					"queue" : 4,
					"unicast" : 41234
					}
				]
			}
		]
	},
	"unit" : 1
}
```

Server may return an error if either the validation or execution of the API fails.

clear-queue-counters

The **`clear-queue-counters`** API clears counters associated with specified queues. This API mimics the **`get-queue-counters`** API for specifying different counters to be cleared.

The API input parameters are shown below.

Parameter	Type	Description
`port`	Integer	Optional Parameter, indicating the Local port number. If this parameter and the `gport` (below) is not specified, counters are cleared for all ports.
`gport`	Integer	Global port Id as derived from `opennsl_port_gport_get()`. Only one - either `port` or `gport` - can be specified in a given source. If this parameter and the `port` (above) is not specified, counters are cleared for all ports
`counter`	String Enumeration	Indicates the counter of interest. Maps to entries of the Enum opennsl_cosq_stat_t. Only "`discard-counters`" and "`out-packets`" are supported.
`queue`	Integer Array	Optional element. - If this param is not specified, Server clears all queue counters for the port. - If this param is specified as an integer array, each of the array elements must be valid queue numbers, and Server clears the counters for the specified queues.

A sample JSON-RPC request is shown below.

```
{
       "jsonrpc": "2.0",
       "method": "clear-queue-counters",
       "unit": 1,
       "params": {
           "counter" : "discard-counters",
```

```
            "sources" : [
                { "port" : 2, "queue" : [2,3,4] },
                { "port" : 34 }
            ]
        },

        "id": 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

get-buffer-statistics

The **`get-buffer-statistics`** API retrieves buffer usage statistics from the MMU[3]. MMU supports a huge number of statistics, divided into different categories, called **`realms`**. Thus, the buffer statistics are available on a per-realm basis. Further, each buffer in a realm is qualified by a set of parameters. These parameters act as indices for the given buffer in a realm. For example, to access a buffer in the *Egress Unicast Queue* realm, the parameter is the corresponding queue-id. Similarly, to access a buffer in I*ngress Port-Priority-group* realm, the parameters are the port-id and the priority-group-id. The *Device* realm has no indices. No realm has more than two indices.

Each buffer, given its parameters, may have one or more than one statistic. For example, the *Ingress Port-Service-Pool* realm buffer, for a given port & service pool, offers a single statistic: **`um-share`**. The *Egress Port-Service-Pool* realm buffer, for a given port & service pool id, offers multiple statistics: **`uc-share`**, **`um-share,`** and, **`mc-share`**.

This API allows buffer usage statistics retrieval in the following methods.

1. All statistics - a complete snapshot.
2. All statistics for a given realm(s).

The current 'C' API works at an individual statistic level, and expects users of the API to loop through all possible realms and indices to acquire a complete snapshot. Given the ASIC typically supports 8000 statistics, such an approach over RPC will result in worst performance. Considering that the most often executed use case is taking a complete

[3] Some introduction to provide context for this API and parameters. More details are to be obtained elsewhere.

snapshot, this API is designed to provide larger set of statistics in a single go. BroadView never reads a single statistic.

The API input parameters are shown below.

Parameter	Type	Description
`realms`	Array of Strings from a String Enumeration	Optional parameter. This indicates the List of realms, for which the statistics are to be included in the response. If this parameter is unspecified, it indicates that a complete snapshot is required. The list is from among the following strings : `"device", "ingress-port-priority-group", "ingress-port-service-pool", "ingress-service-pool", "egress-port-service-pool", "egress-service-pool", "egress-uc-queue", "egress-uc-queue-group", "egress-mc-queue", "egress-cpu-queue", "egress-rqe-queue"`
`options`	Array of Strings from a String Enumeration	An optional element indicating the Options for Statistics reading. Supported options are - `"clear-on-read"` - `OPENNSL_COSQ_STAT_CLEAR` `"sync"` - if specified, `opennsl_cosq_bst_stat_sync` will be invoked before reading the statistics.

A sample JSON-RPC request for statistics from selected realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "get-buffer-statistics",
  "params" : {
       "realms" : [
            "device",
            "ingress-port-priority-group",
            "egress-uc-queue"
       ] },
  "unit" : 1
}
```

A sample JSON-RPC request for statistics from all realms, which an option specified, is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "get-buffer-statistics",
  "params" : { "options" : ["sync"] },
  "unit" : 1
}
```

Responses for the above requests are described in Statistics and Threshold Reports

clear-buffer-statistics

The **clear-buffer-statistics** API clears the buffer usage statistics from the MMU. This API mimics the **get-buffer-statistics** API for invocation and specifying different statistics to be cleared.

The API input parameters are shown below.

The API input parameters are shown below.

Parameter	Type	Description
realms	Array of Strings from a String Enumeration	Optional Parameter. It provides the List of realms, for which the statistics are to be cleared. If this parameter is not specified, it indicates that all statistics are t to be cleared. The list is from among the following strings : "device", "ingress-port-priority-group", "ingress-port-service-pool", "ingress-service-pool", "egress-port-service-pool", "egress-service-pool", "egress-uc-queue", "egress-uc-queue-group", "egress-mc-queue", "egress-cpu-queue", "egress-rqe-queue"

A sample JSON-RPC request for clearing statistics from selected realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "clear-buffer-statistics",
  "params" : {
```

```
      "realms" : [
          "device",
          "ingress-port-priority-group",
          "egress-uc-queue"
      ] },
  "unit" : 1
}
```

A sample JSON-RPC request for clearing statistics from all realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "clear-buffer-statistics",
  "params" : { },
  "unit" : 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

sync-buffer-statistics

The **`sync-buffer-statistics`** API synchronizes the buffer usage statistics from the MMU with the SDK buffer. The corresponding C API is `opennsl_cosq_bst_stat_sync()` . Typically, this API is invoked before **`get-buffer-statistics`** API. However, use of this API can be avoided by specifying the `"sync"` option for the **`get-buffer-statistics`** API.

The API input parameters are shown below.

Parameter	Type	Description
`realms`	Array of Strings from a String Enumeration	Optional Parameter. The List of realms, for which the statistics are to be synchronized. If the parameter is not specified, all statistics are t to be synchronized. The list is from among the following strings : `"device", "ingress-port-priority-group", "ingress-port-service-pool", "ingress-service-pool", "egress-port-service-`

		pool", "egress-service-pool", "egress-uc-queue", "egress-uc-queue-group", "egress-mc-queue", "egress-cpu-queue", "egress-rqe-queue"

A sample JSON-RPC request for synchronizing the statistics for selected realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "sync-buffer-statistics",
  "params" : {
       "realms" : [
            "device",
            "ingress-port-priority-group",
            "egress-uc-queue"
       ] },
  "unit" : 1
}
```

A sample JSON-RPC request synchronizing the statistics for all realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "sync-buffer-statistics",
  "params" : { },
  "unit" : 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

clear-buffer-thresholds

The **`clear-buffer-thresholds`** API resets all the thresholds for the specified MMU buffers to the power-on defaults. There is no corresponding C API. However, this can be implemented on the server by repeatedly invoking the `opennsl_cosq_bst_profile_set` API.

The API input parameters are shown below.

Parameter	Type	Description
realms	Array of Strings from a String Enumeration	Optional Parameter The List of realms, for which the thresholds are to be cleared. If not specified, it indicates that all thresholds are to be cleared. The list is from among the following strings : "device", "ingress-port-priority-group", "ingress-port-service-pool", "ingress-service-pool", "egress-port-service-pool", "egress-service-pool", "egress-uc-queue", "egress-uc-queue-group", "egress-mc-queue", "egress-cpu-queue", "egress-rqe-queue"

A sample JSON-RPC request for clearing the thresholds for selected realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "clear-buffer-thresholds",
  "params" : {
       "realms" : [
            "device",
            "ingress-port-priority-group",
            "egress-uc-queue"
       ] },
  "unit" : 1
}
```

A sample JSON-RPC request clearing the thresholds for all realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "clear-buffer-thresholds",
  "params" : { },
  "unit" : 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

get-buffer-thresholds

The **`get-buffer-thresholds`** API retrieves all the thresholds for the specified MMU buffers. The corresponding C API is `opennsl_cosq_bst_profile_get`.

The API input parameters are shown below.

Parameter	Type	Description
`realms`	Array of Strings from a String Enumeration	Optional Parameter The List of realms, for which the thresholds are to be retrieved. If not specified, it indicates that all thresholds are to be retrieved. The list is from among the following strings : `"device", "ingress-port-priority-group", "ingress-port-service-pool", "ingress-service-pool", "egress-port-service-pool", "egress-service-pool", "egress-uc-queue", "egress-uc-queue-group", "egress-mc-queue", "egress-cpu-queue", "egress-rqe-queue"`

A sample JSON-RPC request for retrieving the thresholds for selected realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "get-buffer-thresholds",
  "params" : {
       "realms" : [
           "device",
           "ingress-port-priority-group",
           "egress-uc-queue"
       ] },
  "unit" : 1
}
```

A sample JSON-RPC request retrieving the thresholds for all realms is shown below.

```
{
  "id" : 1,
  "jsonrpc" : "2.0",
  "method" : "get-buffer-thresholds",
  "params" : { },
  "unit" : 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns the thresholds as requested.

Responses for the above requests are described in Statistics and Threshold Reports

configure-buffer-thresholds

The **`configure-buffer-thresholds`** API configures the thresholds for specified buffers with the supplied values. The corresponding C API is `opennsl_cosq_bst_profile_set`. This API supports multiple buffers to be specified for configuring thresholds within the same API invocation. Each source specifies the set of realms and buffers for which the thresholds are to be configured.

The API specifies the threshold in the JSON in a structured way described below. In this paragraph, the text in the brackets identifies the json parameter name. The threshold is identified by a realm (`realm`), and the two indices (`indices`) that point to the threshold in the realm. Each of the two indices has a name (`index-name`) and a value for the index (`index_value`). The realms (bound by the indices) support multiple buffer thresholds (`thresholds`). Each of the threshold has a name (`threshold_name`), and a value to be configured for the threshold (`threshold_value`).

The API input parameters are shown below.

Parameter	Type	Description
`realm`	String Enumeration	The realm which contains the buffer, for which the threshold is to be configured. Valid values are - `"device", "ingress-port-priority-group", "ingress-port-service-pool", "ingress-service-pool", "egress-port-service-`

		pool", "egress-service-pool", "egress-uc-queue", "egress-uc-queue-group", "egress-mc-queue", "egress-cpu-queue", "egress-rqe-queue"
index-name	String Enumeration	Optional Parameter (only for "device" realm) Specifies the index for the threshold. Valid values are - "port", "pg", "sp", "q", "qgrp"
index-value	Integer	Optional Parameter (only for "device" realm) Value for the index identified by index-name.
threshold-name	String Enumeration	Specifies the name of the threshold. Valid values are - "threshold","uc","mc","um-share"
threshold-value	Integer	Value for the threshold identified by threshold-name, to be configured in the Silicon.

A sample JSON-RPC request is shown below.

```
{
  "jsonrpc" : "2.0",
  "method" : "set-buffer-thresholds",
  "unit" : 1,
  "params" : {
      "data": [
           {
               "realm": "dev",
               "data": [ { "threshold-name" : "threshold", "threshold-value" : 10
} ]
           },
           {
               "realm": "eucq",
               "indices": [ { "index-name" : "queue", "index-value" : 3 } ],
               "data": [ { "threshold-name" : "uc", "threshold-value" : 45 } ]
           },
           {
               "realm": "ippg",
               "indices": [ { "index-name" : "port", "index-value" : 3 },
                            { "index-name" : "pg", "index-value" : 2 } ],
               "data": [ { "threshold-name" : "umshare", "threshold-value" : 4523
} ]
           }
       ]
   },
  "id" : 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

notify-switch-event

The **`notify-switch-event`** API allows the Client to register for specified event notifications with the Server. Upon successful registration, the Server notifies the Client, every time the event occurs on the Server. Registrations are not cumulative. The Client must provide the complete list of events it is interested in for each registration. If an event previously registered for is not specified in a subsequent registration, it is considered as de-registration.

The API input parameters are shown below.

Parameter	Type	Description
`events`	Array of Strings from a String Enumeration	The event list of interest to the Client. An empty string is also allowed (de-register all events for the client) Valid values are - `"buffer-threshold-breach"` The Client must provide the complete list of events it is interested in for each registration. If an event previously registered for is not specified in a subsequent registration, it is considered as de-registration. An empty list here will deregister all previous registrations. During the notification, the Server may choose to report multiple events in a single notification.

A sample JSON-RPC request is shown below.

```
{
  "jsonrpc" : "2.0",
  "method" : "notify-switch-event",
  "unit" : 1,
  "params" : {
      "event" : "buffer-threshold-breach"
   },
  "id" : 1
}
```

Server may return an error if either the validation or execution of the API fails. Otherwise, it returns success.

The event notification data format depends on the specific event. For the "`buffer-threshold-breach`" event, the data format is similar to the "`configure-buffer-thresholds`" command. A sample response is provided below.

```
{
    "jsonrpc": "2.0",
    "method": "notify-switch-event",
    "unit": 1,
    "report": {
        "event-count" : 1,
        "event-type" : "buffer-threshold-breach",
        "event-data" : [ {
                "realm": "eucq",
                "indices": [ { "name" : "queue", "value" : 3 } ],
                "data": [ { "name" : "uc", "value" : 45 } ]
     } ]

}
```

Error Messaging

The error messages sent by the Server to the Client conform to the JSON-RPC specified behaviour. The error responses are described at http://www.jsonrpc.org/specification#error_object.

The error code parameter `code` is populated as follows -

- For all JSON related errors - such as invalid request - the `code` parameter has the standard specified values.
- For all errors returned by internal functions of the server - such as the errors returned by OpenNSL/SDK API, the same return value is populated in the `code` parameter.

Statistics and Threshold Reports

The following table provides the list of realms and the associated indices for each realm. It also lists all available statistics and thresholds for each realm.

Realm	Index # 1	Index # 2	Statistic(s)	Threshold(s)
device			data	threshold
ingress-port-priority-group	port	priority-group	um-share-buffer-count, um-headroom-buffer-count	um-share-threshold, um-headroom-threshold
ingress-port-service-pool	port	service-pool	um-share-buffer-count	um-share-threshold
ingress-service-pool	service-pool		um-share-buffer-count	um-share-threshold
egress-port-service-pool	port	service-pool	uc-share-buffer-count, um-share-buffer-count, mc-share-buffer-count, mc-share-queue-entries	uc-share-threshold, um-share-threshold, mc-share-threshold, mc-share-queue-entries- threshold
egress-service-pool	service-pool		um-share-buffer-count, mc-share-buffer-count, mc-share-queue-entries	um-share-threshold, mc-share-threshold, mc-share-queue-entries- threshold
egress-uc-queue	queue		uc-buffer-count	uc-threshold
egress-uc-queue-group	queue-group		uc-buffer-count	uc-threshold
egress-mc-queue	queue		mc-buffer-count, mc-queue-entries	mc-threshold, mc-queue-entries-threshold
egress-cpu-queue	queue		cpu-buffer-count	cpu-threshold
egress-rqe-queue	queue		rqe-buffer-count	rqe-threshold

While reporting statistics /thresholds for a given realm, the statistics names are not included in the report. A JSON positional parameter method is used. In addition, one (and

not more than one) of the indices is included as the first element in the array. This scheme is used to reduce the JSON message size for report messages. This scheme is applicable only for BST reports and threshold reports. This method is illustrated below with a few examples.

Example 1

Consider the following report snippet:

```
{
            "realm": "ingress-service-pool",
            "data": [[1, 3240], [2, 3660]]
}
```

The `ingress-service-pool` has a single index (`service pool`), and it is mentioned in the JSON array.

This report indicates the following.

- The realm is the `ingress-service-pool`.
- The service pool with ID 1 has the `uc-share-buffer-count` value of 3240.
- The service pool with ID 2 has the `uc-share-buffer-count` value of 3660.

Example 2

Consider the following report snippet:

```
{
            "realm": "ingress-port-priority-group",
            "data": [{
                    "port": 2,
                    "data": [[5, 45500, 44450]]
                }, {
                    "port": 3,
                    "data": [[5, 6700, 250], [7, 12667, 13456]]
                }]
}
```

The `ingress-port-priority-group` has two indices: `port` and the `priority-group`. The top-level index (`port`) is mentioned explicitly, and the second index (`priority-group`) is included as the first element in the JSON array.

This report indicates the following:

- The realm is the `ingress-port-priority-group.`
- The port 2 report has the following entries:
  - `priority-group` id 5 has
    - The `um-share-buffer-count` value is 45500.
    - The `um-headroom-buffer-count` value is 44450.
- The port 3 report has the following entries:
  - `priority-group` id 5 has
    - The `um-share-buffer-count` value is 6700.
    - The `um-headroom-buffer-count` value is 250.
  - `priority-group` id 7 has
    - The `um-share-buffer-count` value is 12267.
    - The `um-headroom-buffer-count` value is 13456.

Example 3

Consider the following report snippet:

```
{
            "realm": "egress-mc-queue",
            "data": [[1, 1, 34, 89], [10, 2, 1244, 0]]
}
```

The `egress-mc-queue` has a single index (queue). It is included as the first element in the JSON array. The second element is *port*, to which queue is assigned.

This report indicates the following:

- The realm is the `egress-mc-queue`,
- The Queue #1 and port #1 has the following entries
  - The `mc-buffer-count` value is 34.
  - The `mc-queue-entries` value is 89.
- The Queue #10 and port #2 has the following entries
  - The `mc-buffer-count` value is 1244.
  - The `mc-queue-entries` value is 0.

Example 4

Consider the following report snippet:

```
{
            "realm": "egress-uc-queue",
            "data": [[1, 1, 34]]
}
```

The `egress-uc-queue` has a single index (queue). It is included as the first element in the JSON array. The second element is *port* to which queue is assigned.

This report indicates the following.

- The realm is the `egress-uc-queue`,
- The Queue #1 and port #1 has the following entries
  - The `uc-buffer-count` value is 34

The *port* is included as a second parameter for the buffer usage report and threshold reports for the `egress-uc-queue` and `egress-mc-queue` realms.

Sample Report

A sample BST report indicating various realms (categories) and the associated statistics is shown below. It is provided for illustrative purposes only and does not include all the realm/port/queue/pool data in it. Rather, the JSON array is used to indicate the multiplicity while providing data for a single element.

```
{
    "jsonrpc": "2.0",
    "method": "get-bst-report",
    "asic-id": "20",
    "version": "1",
    "time-stamp": "2014-11-18 - 00:15:04 ",
    "report": [{
            "realm": "device",
            "data": 46
```

```
        }, {
            "realm": "ingress-port-priority-group",
            "data": [{
                    "port": 2,
                    "data": [[5, 45500, 44450]]
                }, {
                    "port": 3,
                    "data": [[5, 45500, 44450]]
                }]
        }, {
            "realm": "ingress-port-service-pool",
            "data": [{
                    "port": 2,
                    "data": [[5, 324]]
                }, {
                    "port": 3,
                    "data": [[6, 366]]
                }]
        }, {
            "realm": "ingress-service-pool",
            "data": [[1, 3240], [2, 3660]]
        }, {
            "realm": "egress-cpu-queue",
            "data": [[3, 4566, 0]]
        }, {
            "realm": "egress-mc-queue",
            "data": [[1, 1, 34, 89], [2, 4, 1244, 0], [3, 5, 0, 3]]
        }, {
            "realm": "egress-port-service-pool",
            "data": [{
                    "port": 2,
                    "data": [[5, 0, 324, 0]]
                }, {
                    "port": 3,
                    "data": [[6, 0, 366, 0]]
                }]
        }, {
            "realm": "egress-rqe-queue",
            "data": [[2, 3333, 4444], [5, 25, 45]]
        }, {
            "realm": "egress-service-pool",
            "data": [[2, 0, 0, 3240], [3, 3660, 0, 0]]
        }, {
            "realm": "egress-uc-queue",
            "data": [[6, "0", 1111]]
        }, {
            "realm": "egress-uc-queue-group",
            "data": [[6, 2222]]
        }]

}
```