Freescale Semiconductor
Authors: Authors: Subhashini Venkataramanan, Srini Addepalli 6/2015

Virtio-IPSec Accelerator

g-API

Revision History

Date	Version	Author	Reason
07/07/2015	1	Freescale Semiconductor	Initial version
07/25/2015	2	Freescale Semiconductor	Minor changes in data structures, naming

Table of Contents

Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font
Formatted: Default Paragraph Font

1 Introduction

This document introduces g-APIs for IPsec. These APIs are defined that enable VNF applications can use to access the underlying IPsec h/w accelerator.

2 References

Virtio Specifications

http://docs.oasis-open.org/virtio/virtio/v1.0/virtio-v1.0.pdf

http://ozlabs.org/~rusty/virtio-spec/virtio-0.9.5.pdf

Virtio-net, Vhost-net, Vhost-user implementations in Linux 3.19, Qemu 2.3.0

3 Scope

This document identifies the necessary generic apis or g-apis for ipsec, that may be required for application to use to underlying ipsec accelerator. The g-apis cover the management and lifecycle APIs as well as the command and data processing APIs.

4 IPsec Device Definition

A default IPsec Device is expected to provide the following functionality:

1. IPv4 Support
2. Tunnel and Transport Mode
3. ESP (Encapsulating Security Protocol)
4. Checksum to be calculated for Tunnel packets

An IPSec Device may exhibit other capabilities such as AH processing etc. Applications can learn about the same by invoking appropriate g-APIs.

5 System Overview (Virtio IPSec device)

The Virtio IPsec device is emulated as a Virtio PCI based device. The high level picture of device and drivers as projected to the Guest is shown in Figure 1

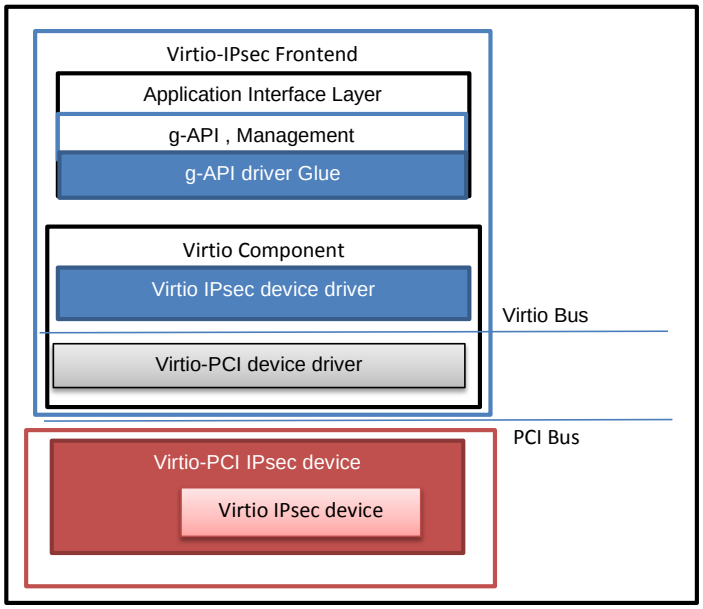

Figure 1 Virtio IPsec Device and Driver

The Virtio IPsec Frontend driver contains two main components,

1. Virtio Component
    1. The Virtio Component interfaces with the underlying Virtio registering a driver to drive the Virtio IPsec device.
    2. It comprises of two components, namely
        1. Virtio PCI component
            1. This is a Virtio Generic Module that acknowledges the Virtio Device before publishing the Virtio IPsec device on the Virtio Bus. (Part of existing code.)
        2. Virtio Bus IPsec Component
            1. This registers with the Virtio Bus so that the virtio-ipsec driver can drive the virtio

ipsec device.

3. The driver provides an API interface that can be used by the g-API glue layer on behalf of the Application to communicate to the underlying accelerator

2. Application Interface Component

1. The g-apis provide APIs for application and management layers to talk to the underlying accelerator. They include

1. Application interface

2. Management interface

2. The g-api glue layer will glue the g-apis to the underlying virtio-driver APIs.

5.1 Lifecycle – Virtual Accelerator detection, programming and removal

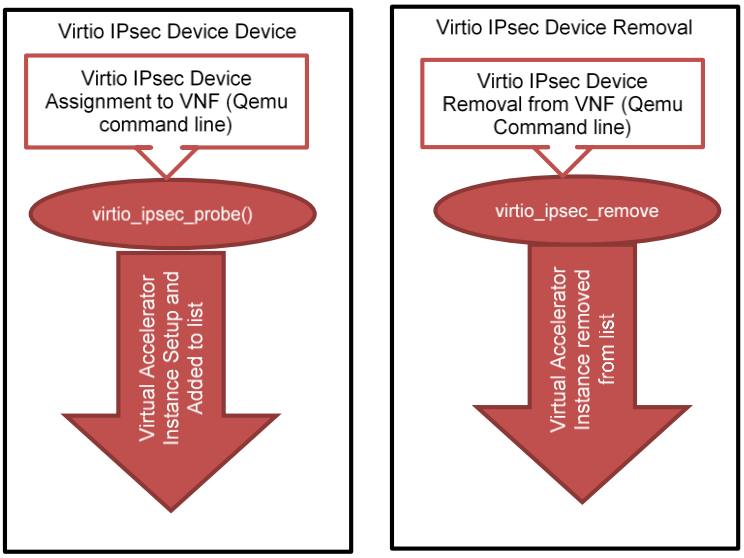

Figure 2 Lifecycle - Virtio Device Assignment and Removal to VNF

Figure 2 shows the discovery and removal of Virtio-IPsec Look aside accelerator device.

5.1.1 Discovery:

3. Upon Qemu command line or similar invocation, a virtio-ipsec device is discovered on the Virtio-PCI bus

4. The device is configured: Queues and Interrupt (virtio_ipsec_probe())

5. The virtual accelerator device is added to the device list.

5.1.2 Removal

6. Upon Qemu command line or similar invocation, a virtio-ipsec device is removed
7. The device is removed from the virtual IPsec instance list (virtio_ipsec_remove())

6 Application Usage

VNF Applications may use an IPsec Accelerator instance in an exclusive mode or shared mode. In some cases a single application may make use of several accelerators

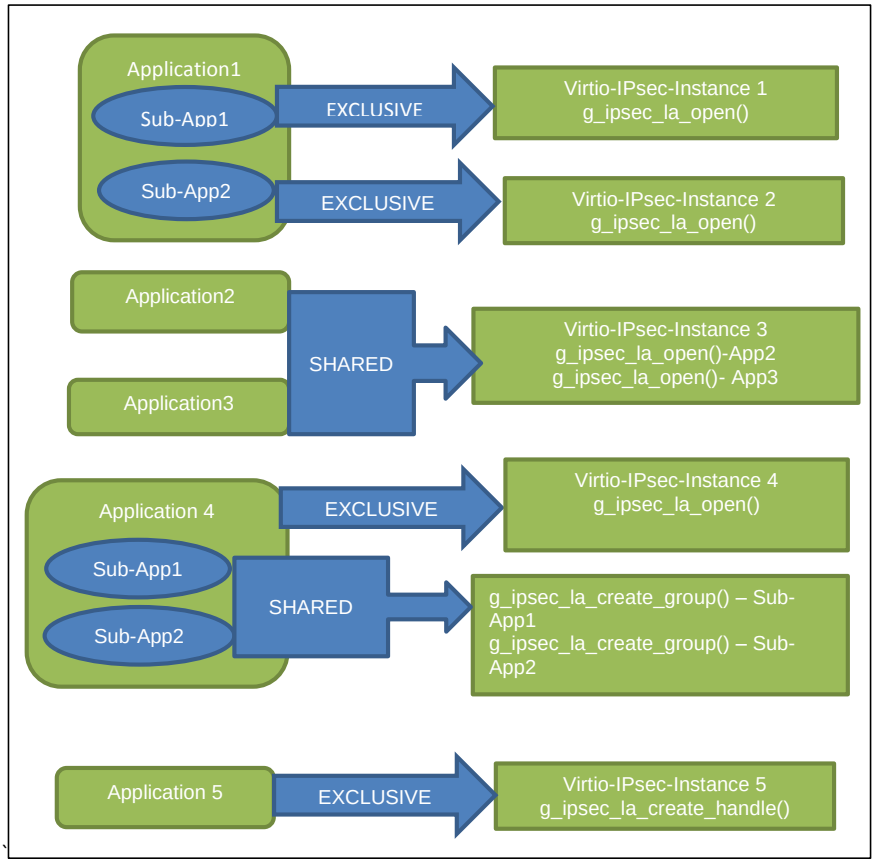

Figure 3 Application Usage Model

Figure 3 shows possible application models on how VNF may make use of Virtual accelerators. From a VNF perspective there may be several applications that may need to make use of the Virtual-IPsec Accelerator. Each application may have several groups as well. Here are examples for applications and groups

1. Linux IPsec supporting multiple namespaces. In this case, the linux application in the root namespace opens

a virtual accelerator instance and for each new namespace, a group can be created.

~~1.~~a. Ideally for this case, for fair arbitration, each namespace or group should have exclusive access to a virtual accelerator. (e.g: Application 1 in Figure 3)

~~1.~~b. Alternatively, one virtual accelerator can be assigned to a namespace and several applications within the namespace can make use of the accelerator in a shared mode. (e.g.: Application 2 and Application 3 in Figure 2.)

~~2.~~c. Alternatively, if application is by itself able to provide fair arbitration to its various users, then, application can use one virtual accelerator for all its users, by establishing several groups- one for each sub-application. (e.g.: Application 4 in Figure 3)

2. Different Applications can have exclusive virtual accelerator instances assigned to them. (e.g. : Application 5 in Figure 3)

6.1.1 Modes

Typically applications can access the underlying virtual IPsec accelerator in two modes –namely Exclusive mode or Shared Mode

~~1.~~• Exclusive Mode – The application or sub-application has exclusive access to the Virtual Accelerator Instance. The Virtqueues and any hardware resources allocated to that virtual accelerator are available for the application or sub-application in an exclusive mode.

~~1.~~• Shared Mode – The application or sub-application may share a virtual accelerator with other applications or sub-applications. The Virtqueues and any hardware resources allocated for the virtual accelerator are shared across several applications.

6.1.2 Virtual Accelerator Assignment

Though now shown in the figure, it is possible that a single application/sub-application may use several virtual accelerators in shared mode or exclusive mode.

The Application or sub-application that has access to several virtual accelerators potentially could distribute the load across the virtual accelerators. The load distribution is entirely up to the application.

7 g-APIs

The application Interface APIs (g-APIs) have two components, namely the Accelerator Management APIs and the functional APIs.

7.1 Accelerator Management APIs

The following APIs shall be supported for Accelerator Management.

1. g_ipsec_la_get_api_version
2. g_ipsec_la_open
3. g_ipsec_la_group_create~~create_group~~
4. g_ipsec_la_group_delete~~la_delete_group~~
5. g_ipsec_la_close
6. g_ipsec_la_get_available_list
7. g_ipsec_la_get_active_list

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Indent at: 0.5", Tab stops: Not at 0.5"

7.2 Functional APIs

The functional APIs are in turn classified to control or setup APIs and data processing APIs. Each API requires an accelerator handle, which the application must have obtained by calling g_ipsec_la_open() function.

7.2.1 Control or setup APIs

1. g_ipsec_la_capabilities_get
2. g_ipsec_la_notification_hooks_register
3. g_ipsec_la_notifications_hook_deregister
4. g_ipsec_la_sa_add
5. g_ipsec_la_sa_mod
6. g_ipsec_la_sa_del
7. g_ipsec_la_sa_flush
8. g_ipsec_la_sa_get

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Indent at: 0.5", Tab stops: Not at 0.5"

7.2.2 Data Processing APIs

1. g_ipsec_la_packet_encap
2. g_ipsec_la_packet_decap
3. g_ipsec_la_multi_packet_encap
4. g_ipsec_la_multi_packet_decap

Formatted: Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Indent at: 0.5", Tab stops: Not at 0.5"

8 g-API definitions

8.1 g_ipsec_la_get_api_version

```
int32_t g_ipsec_la_get_api_version(char *version)

/* Function Name:  g_ipsec_la_get_api_version
 * Input/Output: a variable to hold the version
 * Return value: SUCCESS (0) or FAILURE (-ve value)
 * Description : Application to use this api to get the API version
 */
```

Application can use this API to get the underlying API version.

8.2 g_ipsec_la_open

```
int32_t g_ipsec_la_open(
      enum g_ipsec_la_mode mode, /* Mode = EXCLUSIVE OR SHARED */
      struct g_ipsec_la_open_inargs *in,
      struct g_ipsec_la_open_outargs *out);

/* Function Name: g_ipsec_la_open
 * Input/Output  : mode = EXCLUSIVE or SHARES, in : application
                identity, callback function to invoke when the
                underlying accelerator connection is broken, callback
                argument and length of the same. out: handle to the
                accelerator
 * Return Value : SUCCESS(0) or FAILURE (-ve value)
 * Description  : Get a handle to an IPsec Look Aside Accelerator
                Instance.
 */
```

An Application shall use this API to open a virtual accelerator in either a shared mode or exclusive mode. When exclusive mode is requested, every attempt would be made to assign a virtual accelerator exclusively for usage by that application. When shared mode is requested, a shared virtual accelerator may be assigned to the application. In case the suggested mode is unavailable (due to non-available virtual accelerator instances,) a failure would be returned.

The application registers a callback function to be invoked, if the underlying virtual accelerator association is broken. The application is expected to take corrective action such as closing the current handle and opening a new handle if required.

8.3 g_ipsec_la_~~create_group~~group_create

```
int32_t g_ipsec_la_~~create_group~~group_create(
      struct g_ipsec_la_handle *handle;
            /* handle should be valid one */
      struct g_ipsec_la_~~create_group~~group_create_inargs *in,
      enum g_ipsec_la_control_flags flags,
      struct g_ipsec_la_~~create_group~~group_create_outargs *out,
```

```
      struct g_ipsec_la_resp_args resp);

/* Function Name: g_ipsec_la_~~create_group~~group_create
 * Input        :
                : g_ipsec_la_handle: handle, char *group_identity, a
                  name that identifies an application group,
 * Output       : g_ipsec_la_handle;
 * Return Value : SUCCESS(0) or FAILURE (-ve value)
 * Description  : Get a group handle to an IPsec Look Aside
                 Accelerator Instance.
 */
```

An Application can use this API to create a group within an accelerator handle. The group would use the same virtual accelerator instance as the one that was assigned as per the application's g_ipsec_la_open(). Depending on the mode used at the time of g_ipsec_la_open(), the group may be sharing the virtual accelerator instance across several other groups (g_ipsec_la_open() invoked with G_IPSEC_LA_INSTANCE_EXCLUSIVE), or may be sharing the virtual accelerator across other applications and other groups. (g_ipsec_la_open invoked with G_IPSEC_LA_INSTANCE_SHARED).

8.4 g_ipsec_la_~~delete_group~~group_delete

```
int32_t g_ipsec_la_~~delete_group~~group_delete(
      struct g_ipsec_la_handle *handle,
      enum g_ipsec_la_control_flags flags,
      struct g_ipsec_la_group_delete_outargs *out,
      struct g_ipsec_la_resp_args resp
      )

/* Function Name: g_ipsec_la_~~delete_group~~group_delete
 * Input        : accelerator handle and group handle
 * Output       : None
 * Return Value : Success(0) or Failure (-ve value)
 * Description  : Given a handle, close the group
 */
```

Application should use this API to delete a group. Any data structures that were created using this group would be deleted at that point. Application must exercise the g_ipsec_la_sa_flush API to flush any SAs created with this group, before exercising this call. Application may no longer use the group handle for subsequent calls.

8.5 g_ipsec_la_close

```
int32_t g_ipsec_la_close(struct g_ipsec_la_handle *handle)

/* Function Name: g_ipsec_la_close
 * Input        : g_ipsec_la_handle; handle
 * Output       : None
 * Return Value : Success(0) or Failure (-ve value)
```

```
 * Description  : Given a handle, close the virtual accelerator
instance */
```

Application should use this API to close the handle of the previously opened accelerator instance. If any groups were created under this handle, then the Application should delete them, before making this call. Application must flush all SAs created using the accelerator handle/groups before making this call. Application may no longer access the underlying accelerator.

8.6 g_ipsec_la_avail_devices_get_num

Prototype:

```
int32 g_ipsec_la_avail_devices_get_num(u32 *nr_devices)

/*
 * Function Name: g_ipsec_la_avai_devices_get_num
 * Input/Output: u32 pointer to hold the value
   Return Value: Success or Failure
       Description: Return the number of avail virtual accelerator
devices.

 */
```

8.7 g_ipsec_la_avail~~able~~_devices~~list~~_get_info

Prototype:

```
int32 g_ipsec_la_avail_devices_get_info(
      struct g_ipsec_la_avail_devices_get_inargs *in,
            struct g_ipsec_la_avail_devices_get_outargs *out)

      /*
       * Function Name: g_ipsec_la_avail_devices_get_info
       * Input: g_ipsec_la_avail_devices_get_in_args, number of devices
for which information has to be retrieved.
       * Ouput: For each available device, get the name, its mode
(available or already shared and if shared, the number of apps sharing
the device
       Description: Application can call this API to find out the list
of available devices that it can use.
~~8.6~~ */
```

~~8. for each virtual accelerator~~

~~8. Name~~

~~9. The accelerator instance Identifier~~

Formatted: Bulleted + Level: 1 + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Bulleted + Level: 2 + Aligned at: 0.75" + Indent at: 1", Tab stops: Not at 1"

8.108.8 g_ipsec_la_active_list_get

1. for each virtual accelerator
    1. Name
    1. the accelerator instance Identifier
    2. Application owner details (Application Identity, Group-identity)
    3. the mode (shared/exclusive)
    4. Displays the statistics
        1. Bytes In/Bytes Out
        1. Packets In/Packets Out
        2. SAs created, SAs Deleted, SAs Modified
        3. Current Number of Active SAs

8.118.9 g_ipsec_la_get_capabilities_get

```
int32_t g_ipsec_la_capabilities_get(
     struct g_ipsec_la_handle *handle,
     struct g_ipsec_la_control_flags flags,
     struct g_ipsec_la_cap_get_outargs *out,
     struct g_ipsec_la_resp_args *resp)

/*
 * Function Name: g_ipsec_la_capabilities_get
 * Input: handle – accelerator handle with optional group handle;
          subflags indicating SYNC or ASYNC, Response required
          or not; In this case response is required. Out – Pointer to
          the output parameter structure (Capabilities); resp –
          Response callback function and details in case ASYNC
response
          is requested
   Output: Success or Failure
   Description: Returns the capabilities of the underlying
accelerator.
              In the case of synchronous response, the out parameter
              has the capabilities, otherwise, the resp callback
              function is invoked with the capabilities

*/
Description: Application can call this API to find out the
capabilities offered by the underlying virtual IPSec accelerator. The
response may be returned synchronously or asynchronously based on the
Application's preference as set by the flags argument. When returned
synchronously, the capabilities are returned by the out parameter.
```

When returned asynchronously, the capabilities are passed as type struct g_ipsec_la_cap_get_outargs through the response callback function.

~~8.12~~8.10 g_ipsec_la_notification_hooks_register

```
int32_t g_ipsec_la_notification_hooks_register(
      struct g_ipsec_la_handle handle, /* Accelerator Handle */
      const struct g_ipsec_la_notification_hooks *in
      );

/* Function Name: g_ipsec_la_notification_hooks_register
 * Input: Virtual Accelerator Instance Handle, Notification hook
          functions
 * Output: Success or Failure
 * Description: Registers hook function to be called for notifications
               from underlying accelerator; Notifications if supported
               by underlying Virtual IPsec accelerator include
               Periodic Sequence Number Announce, Sequence Number
               Overflow and Soft Lifetime in bytes expiry
 */
```

~~8.13~~8.11 g_ipsec_la_notifications_hook_deregister

```
g_ipsec_la_notifications_hook_deregister(
      struct g_ipsec_la_handle ,  /* Accelerator Handle */ )

/* Function Name: g_ipsec_la_notifications_hook_deregister
 * Input: Accelerator handle, group handle if applicable
 * Ouput: None
 * Description : The notification callback function hooks get de-
registered
 */
```

Application can call this API to de-register previously registered callback functions.

~~8.14~~8.12 g_ipsec_la_sa_add

```
int32_t g_ipsec_la_sa_add(
       struct g_ipsec_la_handle *handle,
         const struct g_ipsec_la_sa_add_inargs *in,
         enum g_ipsec_la_control_flags flags,
         struct g_ipsec_la_sa_add_outargs *out,
         struct g_ipsec_la_resp_args resp);
/*
 * Function Name: g_ipsec_la_sa_add
 * Parameters:  handle – Accelerator handle,
 *         group; Input Arguments = {flags, sa parameters}; flags:
```

```
 *        Synchronous or asynchronous, Response required or not; Out
 *        Argument: Result and SA Handle; resp: Response callback
 *        function and callback argument in case ASYNC response is
 *        requested
 * Return Value: Success or Failure (< 0)
 * Description: Application uses this API to create an Inbound or
 *              Outbound SA
*/
```

Application can call this API to create an Inbound or Outbound SA. This API returns SUCCESS when the SA has been successfully created by the Virtual Accelerator. A SA Handle is returned by this API. Application is expected to use the SA Handle in subsequent calls such as g_ipsec_la_sa_modify, g_ipsec_la_sa_delete, or one of the Read SA commands

~~8.15~~8.13 g_ipsec_la_sa_mod

```
int32_t g_ipsec_la_sa_mod(
      struct g_ipsec_la_hanlde *handle, /* Accelerator Handle */
      const struct g_ipsec_la_sa_mod_inargs *in, /* Input Arguments */
        g_ipsec_la_control_flags flags, /* Control flags: sync/async,
response required or not */
        struct g_ipsec_la_sa_mod_outargs *out, /* Output Arguments */
        struct g_ipsec_la~~api~~_resp_args resp /* Response data structure
with callback function information and arguments with ASYNC response
is requested);

/* Function Name: g_ipseC_la_sa_mod
 * Input/Out: Accelerator Handle, SA Handle, SA Modification
parameters, API Control flags, Output arguments, Response callback
function and arguments, in case ASYNC mode is chosen
 * Return Value: SUCCESS or FAILURE
 * Description: Application uses this API to modify SA parameters such
as Local Gateway IP Address/Port, Remote Gateway IP Address/Port and
Sequence number information */
```

Application can call this API to modify SA parameters. When the Local gateway IP Address has been updated or the remote Gateway IP Address has been changed or when sequence number related information has to be updated, Application can call this API to update the SA maintained by the underlying virtual accelerator.

~~8.16~~8.14 g_ipsec_la_sa_del

```
int32_t g_ipsec_la_sa_del(
     struct g_ipsec_la_handle *handle,
       const struct g_ipsec_la_sa_del_inargs *in,
       g_api_control_flags flags,
       struct g_ipsec_la_sa_del_outargs *out,
       struct g_ipsec_la_resp_args resp);
```

```
/* Function Name: g_ipsec_la_sa_del
 * Input: Accelerator Handle, SA Direction, SA Handle
 * Input/Output: Success or error code
 * Description: Given the virtual accelerator handle and the SA
handle, delete the SA
 */
```

Application calls this API to delete the SA.

~~8.17~~8.15 g_ipsec_la_sa_flush

```
Prototype:
int32_t g_ipsec_la_sa_flush(
      struct g_ipsec_la_handle *handle,
      g_ipsec_la_control_flags flags,
      struct g_ipsec_la_sa_flush_outargs *out,
      struct g_ipsec_la_resp_args *resp)

/* Function Name: g_ipsec_la_sa_flush
 * Input: Virtual Accelerator Handle and optional group handle
 *        information, flags : Async/sync, Response required or not;
 *        Response Callback function and argument in case async
 *        response is requested
 * Return Value: Success or Failure
 * Description: Application can use this API, to flush the SAs that
 *              were created given a handle/group
 */
```

Application/sub-application can call this API to flush SAs. If an application has several groups, the application has to flush SAs for each group individually.

~~8.18~~8.16 g_ipsec_la_sa_get

```
int32_t g_ipsec_la_sa_get(
      struct g_ipsec_la_handle *handle,
      const struct g_ipsec_la_sa_get_inargs *in,
      g_ipsec_la_~~api~~_control_flags flags,
      struct g_ipsec_sa_get_outargs *out,
      struct g_ipsec_la_~~api~~_resp_args *resp);
/* Function Name: g_ipsec_la_sa_get
 * Input: Virtual Accelerator Handle (handle/group handle), Input
 *        arguments that include direction (inbound or outbound)
 *        sa_handle (valid for get exact or get next calls), Operation
 *        Get First/Get First N/Get Next/Get Next N/Get Exact/, number
 *        of SAs to read (for Get First, Get Next and Get Exact, it
 *        would be 1; flags: API control flags, out: contains required
 *        memory to hold the output information (statistics or SA),
```

```
 *        result: SUCCESS or error code; resp: Optional response
 *        callback function and arguments, in case ASYNC flag is set.
 * Return Value: Success or Error
 * Description: Application/Sub-application can call this API to read
 *              SA Information or statistics
 */
```

Application can use this API to retrieve SAs or SA statistics. For convenience several flags are available, such as 'get first', get first n number of SAs', get next, get next n number of SAs and get_exact. Application has the flexibility to get either the SA information or the SA statistics.

~~8.19~~8.17 g_ipsec_la_packet_encap

```
Prototype:
int32_t g_ipsec_la_packet_encap(
           struct g_ipsec_la_handle *handle,
           struct g_ipsec_la_control_flags flags,
           struct g_ipsec_la_sa_handle *handle,~~,~~ /* SA Handle */
           uint32_t num_sg_elem,~~,~~ /* num of Scatter Gather elements */
           struct g_ipsec_la_data in_data[],~~,~~
                /* Array of data blocks */
           struct g_ipsec_la_data out_data[],~~,~~
                /* Array of output data blocks */
           struct g_ipsec~~api~~_la resp_args resp)
/*
 * Function Name: g_ipsec_la_encap_packet
 * Arguments: Accelerator handle, Control Flags, SA Handle, Input
data-
           length segments, Output data-length segments, result
 *         Success or error code, Response callback and args, in case
 *         async response is requested.
 * Return Value : Success or Failure
 */
```

Application calls this API for Outbound Packet processing. When the application submits the SA Handle, and the set of input buffers to the virtual accelerator (using handle and optional group), the application expects the virtual accelerator to IPSec outbound process the buffers as per the Security Association and return the processed buffers.

~~8.20~~8.18 g_ipsec_la_packet_decap

```
Prototype:
int32_t    g_ipsec_la_decap_packet(
           struct g_ipsec_la_handle *handle,
           struct g_ipsec_la_control_flags flags,
           struct g_ipsec_la_sa_handle *handle,~~,~~ /* SA Handle */
```

```
            uint32_t num_sg_elem,~~;~~      /* number of Scatter Gather
elements */
            struct g_ipsec_la_data in_data[],~~;~~/* Array of data blocks
*/
            struct g_ipsec_la_data out_data[], /* Array of out data
blocks*/
            struct g_ipsec_la_~~api~~_resp_args resp)
/*
 * Function Name: g_ipsec_la_decap_packet
 * Arguments: Accelerator handle, Control Flags, SA,
 *         Handle, Input data-length segments
 *         Success or error code, array of data blocks to hold the
 *         output data, Response callback and args, in case async
 *         response is requested.
 * Return Value: Success or Failure
 */
```

Application calls this API for Inbound Packet processing. When the application submits the SA Handle, and the set of input buffers to the virtual accelerator (using handle – and optional group), the application expects the virtual accelerator to IPSec inbound process(decapsulation and decryption) the buffers as per the Security Association and return the processed buffers.

~~8.21~~8.19 g_ipsec_la_multi_packet_encap

```
Prototype:
int32_t g_ipsec_la_multi_packet_encap(
            struct g_ipsec_la_handle *handle,
            struct g_ipsec_la_control_flags flags,
            struct g_ipsec_la_sa_handle *handle; /* SA Handle */
            uint32_t num_packets; /* num of Scatter Gather elements */
            struct g_ipsec_la_packet in_packets[];
                  /* Array of data blocks */
            struct g_ipsec_la_packet out_packets[];
                  /* Array of output data blocks */
            struct g_api_resp_args *resp)
/*
 * Function Name: g_ipsec_la_encap_packet
 * Arguments: Accelerator handle, Control Flags, SA Handle, Input
*         packets, Output packets, result
 *         Success or error code, Response callback and args, in case
 *         async response is requested.
 * Return Value : Success or Failure
 */
```

This function is similar to g_ipsec_la_packet_encap. However multiple packets can be submitted by application in one API invocation.

~~8.22~~8.20 g_ipsec_la_multi_packet_decap

```
Prototype:
int32_t    g_ipsec_la_decap_packet(
           struct g_ipsec_la_handle *handle,
           struct g_ipsec_la_control_flags flags,
           struct g_ipsec_la_sa_handle *handle; /* SA Handle */
           uint32_t num_packets; /* number of Scatter Gather elements
*/
           struct g_ipsec_la_data in_packets[];/* Array of in packets
*/
           struct g_ipsec_la_data out_packets[] /* Array of out
packets*/
           struct g_api_resp_args resp)
/*
 * Function Name: g_ipsec_la_decap_packet
 * Arguments: Accelerator handle, Control Flags, SA,
 *        Handle, Input packets
 *        array of packets to hold the
 *        processed data, Response callback and args, in case async
 *        response is requested.
 * Return Value: Success or Failure
 */
```

This function is similar to g_ipsec_la_packet_decap. However multiple packets can be submitted by application in one API invocation.

9 Data Structures

9.1 g_ipsec_la_create_group_inargs

```
struct g_ipsec_la_create_group_inargs {
     char *group_identity; /* Group identity */
}
```

9.2 g_ipsec_la_create_group_outargs

```
struct g_ipsec_la_create_group_outargs {
     int32_t result;
     uint32_t g_ipsec_la_group_handle[G_IPSEC_LA_GROUP_HANDLE_SIZE];
/* Group handle holder */
};
```

9.3 g_ipsec_la_group_delete_outargs

```
struct g_ipsec_la_group_delete_outargs {
     int32_t result;
};
```

~~9.3~~9.4 g_ipsec_la_instance_broken_cbk_fn

```
typedef void (*g_ipsec_la_instance_broken_cbk_fn)(struct
g_ipsec_la_handle *handle,  void *cb_arg);
```

The above application registered callback function will be invoked, when underlying accelerator instance to which the handle is attached is removed.

~~9.4~~9.5 g_ipsec_la_open_inargs

```
struct g_ipsec_la_open_inargs {
      uint16_t pci_vendor_id; /* 0x1AF4 */
      uint16_t device_id;   /* Device Id for IPsec */
      char *accl_name; /* Optional */
      char *app_identity;   /* Application identity */
      g_ipsec_la_instance_broken_cbk_fn,    /* Callback function to be
called when the connection to the underlying accelerator is broken */
      void *cb_arg;    /* Callback argument */
      int32_t cb_arg_len;   /* Callback argument length */
};
```

~~9.5~~9.6 g_ipsec_la_open _outargs

```
struct g_ipsec_la_open_outargs{
       g_ipsec_la_handle *handle /* handle */
};
```

~~9.6~~9.7 g_ipsec_la_resp_args

```
struct g_ipsec_la_resp_args
{
      struct g_ipsec_la_resp_cbfn cb_fn;
            /* Callback function if
               ASYNC flag is chosen */
      void *cb_arg;
      int32_t cb_arg_len; /* Callback argument length */
}
```

The above structure can be used by applications to provide callback function, arguments, that can be subsequently invoked by virtio-ipsec

~~9.7~~9.8 g_ipsec_la_handle

```
struct g_ipsec_la_handle {
      uint32_t handle[G_IPSEC_LA_HANDLE_SIZE]; /* Accelerator handle */
      uint32_t group_handle[G_IPSEC_LA_GROUP_HANDLE_SIZE]; /* Group
handle */
};
```

9.9 g_ipsec_la_avail_devices_get_inargs

```
struct
```

```
{
      uint32 num_devices;
      char *last_device_read; /* NULL if this is the first time this
call is invoked;  Subsequent calls will have a valid value here */

};
```

9.10 g_ipsec_la_device_info

Formatted: Heading 2

```
struct g_ipsec_la_device_info
{
      char device_name[IPSEC_IFNAMESIZ];
      u8 mode; /* Shared or Available */
      u32 num_apps; /* If shared */
};
```

9.11 g_ipsec_la_avail_devices_get_outargs

Formatted: Heading 2

```
struct g_ipsec_la_avail_devices_get_outargs
{
      uint32 num_devices; /* filled by API */
      /* Array of pointers, where each points to
        device specific information */
      struct g_ipsec_la_device_info *dev_info;

      char *last_device_read; /* Send a value that the application can
use and  invoke for the next set of devices */
      bool b_more_devices;
      };
```

~~9.8~~9.12 g_ipsec_la_sa_handle

```
struct g_ipsec_la_sa_handle {
      uint32_t ipsec_sa_handle[G_IPSEC_LA_SA_HANDLE_SIZE];
};
```

The above structure would be used by the application for all functions once accelerator handle/group handle have been established

~~9.9~~9.13 g_ipsec_la_auth_algo_cap

```
/* Authentication Algorithm capabilities */
struct g_ipsec_la_auth_algo_cap {
      uint32_t          md5:1,
                  sha1:1,
                  sha2:1,
                  aes_xcbc:1,
                  none:1
```

```
                des:1;
};
```

9.109.14 g_ipsec_la_cipher_algo_cap

```
/* Cipher Algorithm Capabilities */
struct g_ipsec_la_cipher_algo_cap {
     uint32_t        des:1,
                des_c:1,
                aes:1,
                aes_ctr:1,
                null:1;
};
```

9.119.15 g_ipsec_la_comb_algo_cap

```
/* Combined mode algorithm capabilities */
struct g_ipsec_la_comb_algo_cap {
     uint32_t        aes_ccm:1,
                aes_gcm:1,
                aes_gmac:1;
};
```

9.129.16 g_ipsec_la_capabilities

```
/* Accelerator capabilities */
struct g_ipsec_la_capabilities
{
     uint32_t sg_features:1, /* Scatter-Gather Support for I/O */
          ah_protocol:1,  /* AH Protocol */
          esp_protocol:1, /* ESP protocol */
          wesp_protocol:1,/* WESP Protocol */
          ipcomp_protocol:1,    /* IP Compression */
          multi_sec_protocol:1, /* SA Bundle support */
          udp_encap:1,    /* UDP Encapsulation */
          esn:1,     /* Extended Sequence Number support */
          tfc:1,     /* Traffic Flow Confidentiality */
           ecn:1,    /* Extended Congestion Notification */
          df:1,      /* Fragment bit handling */
          anti_replay_check:1,  /* Anti Replay check */
          ipv6_support:1, /* IPv6 Support */
          soft_lifetime_bytes_notify:1,    /* Soft Lifetime Notify
Support */
          seqnum_overflow_notify:1,  /* Seq Num Overflow notify */
          seqnum_periodic_notify:1;  /* Seq Num Periodic Notify */
     uint8_t num_dscp_based_queues;   /* Number of DSCP based queues
*/
```

```
      struct g_ipsec_la_autho_algo_cap auth_algo_caps;
      struct g_ipsec_la_cipher_algo_cap cipher_algo_caps;
      struct g_ipsec_la_comb_algo_cap comb_algo_caps;
}
```

~~9.13~~9.17 g_ipsec_la_cap_get_outargs

```
struct g_ipsec_la_cap_get_outargs
{
      struct g_ipsec_la_capabilities caps; /* Capabilities */
}
```

~~9.14~~9.18 g_ipsec_la_resp_cbfn

```
typedef void(*g_ipsec_la_resp_cbfn) (void *cb_arg, int32_t cb_arg_len,
void *outargs);
```

~~9.15~~9.19 g_ipsec_seq_number_notification

```
struct g_ipsec_seq_number_notification {
      struct g_ipsec_la_handle *handle,
      struct g_ipsec_la_sa_handle *sa_handle; /* SA Handle */
      uint32_t seq_num;      /* Low Sequence Number */
      uint32_t hi_seq_num; /* High Sequence Number */
};
```

~~9.16~~9.20 g_ipsec_la_cbk_sa_seq_number_overflow_fn

```
/* Callback function prototype that application can provide to receive
sequence number overflow notifications from underlying accelerator */
typedef void (*g_ipsec_la_cbk_sa_seq_number_overflow_fn) (
      struct g_ipsec_la_handle handle,
      struct g_ipsec_seq_number_notification *in);
```

~~9.17~~9.21 g_ipsec_la_cbk_sa_seq_number_periodic_update_fn

```
/* Callback function prototype that application can provide to receive
sequence number periodic notifications from underlying accelerator */
typedef void (*g_ipsec_la_cbk_sa_seq_number_periodic_update_fn) (
      struct g_ipsec_la_handle handle,
      struct g_ipsec_seq_number_notification *in);
```

~~9.18~~9.22 g_ipsec_la_lifetime_in_bytes_notification

```
struct g_ipsec_la_lifetime_in_bytes_notification {
      struct g_ipsec_la_sa_handle sa_handle;/* SA Handle */
      uint32_t ipsec_lifetime_in_kbytes;    /* Lifetime in Kilobytes */
}
```

~~9.19~~9.23 g_ipsec_la_cbk_sa_soft_lifetimeout_expiry_fn

```
/* Callback function prototype that application can provide to receive
soft lifetime out expiry from underlying accelerator */
typedef void (*g_ipsec_la_cbk_sa_soft_lifetimeout_expiry_fn) (
      struct g_ipsec_la_handle handle,
      struct g_ipsec_la_lifetime_in_bytes_notification *in);
```

~~9.20~~9.24 g_ipsec_la_notification_hooks

```
struct g_ipsec_la_notification_hooks
{
      /* Sequence Number Overflow callback function */
      struct g_ipsec_la_cbk_sa_seq_number_overflow_fn
*seq_num_overflow_fn;
      /* Sequence Number periodic Update Callback function */
      struct g_ipsec_la_cbk_sa_seq_number_periodic_update_fn
*seq_num_periodic_update_fn;
      /* Soft lifetime in Kilobytes expiry function */
      struct g_ipsec_la_cbk_sa_soft_lifetimeout_expiry_fn
*soft_lifetimeout_expirty_fn;
      void *seqnum_overflow_cbarg;
      u32 seq_num_overflow_cbarg_len;

      void *seqnum_periodic_cbarg;
      u32 seq_num_periodic_cbarg_len;

      void *soft_lifetimeout_cbarg;
      u32 soft_lifetimeout_cbarg_len;

};
```

~~9.21~~9.25 g_ipsec_la_sa_crypto_params

```
struct g_ipsec_la_sa_crypto_params
{
      enum g_ipsec_la_auth_alg auth_algo;
      uint8_t *auth_key; /* Authentication Key */
      uint32_t auth_key_len_bits; /* Key Length in bits */
      enum g_ipsec_la_cipher_alg cipher_algo;     /* Cipher Algorithm
*/
      uint8_t *cipher_key;  /* Cipher Key */
      uint32_t cipher_key_len_bits;    /* Cipher Key Length in bits */
      uint8_t *iv;      /* IV Length */
      uint8_t iv_len_bits;  /* IV length in bits */
```

```
      enum g_ipsec_la_comb_alg comb_algo;    /* Combined Mode Algorithm
*/
      uint8_t *comb_key;     /* Combined Mode key */
      uint32_t comb_key_len_bits;/* Combined mode key length in bits;
It holds the sal
length followed by the key */
      uint8_t icv_len_bits; /* ICV - Integrity check value size in bits
*/
}
```

Formatted: Indent: Left: 0", First line: 0"

9.229.26 g_ipsec_la_ipcomp_info

```
struct g_ipsec_la_ipcomp_info
{
      enum g_ipsec_la_ipcomp_alg algo;
      uint32_t cpi;
}
```

9.239.27 g_ipsec_la_tunnel_end_addr

```
struct g_ipsec_la_tunnel_end_addr {
      struct g_ip_addr      src_ip;    /* Source Address */
      struct g_ip_addr      dest_ip; /* Destination Address */
};
```

9.249.28 g_ipsec_la_nat_traversal_info

```
struct g_ipsec_la_nat_traversal_info {
      uint16_t dest_port; /* Destination Port */
      uint16_t src_port; /* Source Port */
      struct g_ip_addr nat_oa_peer_addr; /* Original Peer Address;
valid if encapsulation Mode is transport */
};
```

9.259.29 g_ipsec_la_sa

```
struct g_ipsec_la_sa
{
      uint32_t spi; /* Security Parameter Index */
      uint8_t proto; /* ESP, AH or IPCOMP */
      enum g_ipsec_la_sa_flags cmn_flags;    /* Flags such as Anti-
replay check, ECN etc */
      uint8_t anti_replay_window_size;
      union {
            struct  {
                  uint8_t dscp; /* DSCP value  valid when dscp_handle is
set to "set" */
                  enum g_ipsec_la df_bit_handle df_bit_handle; /* DF
set, clear or propogate */
```

```
                enum g_ipsec_la_dscp_handle dscp_handle;   /* DSCP
handle set, clear etc. */
                uint8_t *iv;     /* IV Length */
                uint8_t iv_len_bits;  /* IV length in bits */
          }outb;
     struct {
          //enum g_ipsec_la_inb_sa_flags flags; /* Flags specific to
inbound SA */
          uint8_t anti_replay_window_size;
          }inb;
     }
     struct g_ipsec_la_sa_crypto_params crypto_params;  /* Crypto
Parameters */
     struct g_ipsec_la_ipcomp_info;   /* IP Compression Information */
     uint32_t soft_kilobytes_limit;
     uint32_t hard_kilobytes_limit;
     uint32_t seqnum_interval;
     struct g_api_ipsec_la_nat_traversal_info nat_info;
     struct g_api_ipsec_la_tunnel_end_addr te_addr;
}
```

9.269.30 g_ipsec_la_sa_add_inargs

```
struct g_ipsec_la_sa_add_inargs
{
     enum g_ipsec_la_sa_direction dir;
     uint8_t num_sas;
     struct g_ipsec_la_sa *sa_params;
};
```

9.279.31 g_ipsec_la_sa_add_outargs

```
struct g_ipsec_la_sa_add_outargs {
     int32_t result; /* Non zero value: Success, Otherwise failure */
     struct g_ipsec_la_handle handle;
}
```

9.289.32 g_ipsec_la_sa_modify_flags

```
struct g_ipsec_la_sa_modify_flags
{
     G_IPSEC_LA_SA_MODIFY_LOCAL_GW_INFO= 1, /* Modify the Local
Gateway Information */
     G_IPSEC_LA_SA_MODIFY_PEER_GW_INFO, /* Modify the Remote Gateway
Information */
     G_IPSEC_LA_SA_MODIFY_REPLAY_INFO, /* SA will be updated with
Sequence number, window bit map etc. */
};
```

~~9.29~~9.33 g_ipsec_la_sa_mod_inargs

```
struct g_ipsec_la_sa_mod_inargs
{
      enum g_ipsec_la_sa_direction; /* Inbound or Outbound */
      struct g_ipsec_la_sa_handle *handle; /* SA Handle */
      enum g_ipsec_la_sa_modify_flags flags; /* Flags that indicate
what needs to  be updated */
      union {
            struct {
                  uint16_t port; /* New Port */
                  struct g_ip_addr addr;  /* New IP Address */
            }addr_info; /* Valid when Local or Remote Gateway
Information is modified */
            struct  {
                  enum g_ipsec_la_sa_modify_replay_info_flags flags; /*
Flag indicates which parameters are being modified */
                  uint8_t anti_replay_window_size; /* Anti replay window
size is being modified */
                  uint32_t anti_replay_window_bit_map; /* Window bit map
array is being updated */
                  uint32_t seq_num; /* Sequence Number is being updated
*/
                  uint32_t hi_seq_num; /* Higher order Sequence number,
when Extended Sequence number is used */
            }; /* Valid when SA_MODIFY_REPLAY_INFO is set */
      }
};
```

~~9.30~~9.34 g_ipsec_la_sa_mod~~ify~~_outargs

```
struct g_ipsec_la_sa_mod~~ify~~_outargs
{
      int32_t result /* 0 Success; Non zero value: Error code
indicating failure */
}
```

~~9.31~~9.35 g_ipsec_la_sa_del_inargs

```
struct g_ipsec_la_sa_del_inargs
{
      enum g_ipsec_la_sa_direction  dir; /* Input or Output */
      struct g_ipsec_la_sa_handle *handle; /* SA Handle */
};
```

~~9.32~~9.36 g_ipsec_la_sa_del_outargs

```
struct g_ipsec_la_sa_del_outargs
{
```

```
      int32_t result; /* 0 success, Non-zero value: Error code
indicating failure */
};
```

9.37g_ipsec_la_sa_flush_outargs

Formatted: Heading 2

```
struct g_ipsec_la_sa_flush_outargs {
      int32_t result; /* 0 for success */
}
```

9.339.38 g_ipsec_la_sa_stats

```
struct g_ipsec_la_sa_stats {
      uint64_t packets_processed;/* Number of packets processed */
      uint64_t bytes_processed;  /* Number of bytes processed */
      struct {
            uint32_t invalid_ipsec_pkt; /* Number of invalid IPSec
Packets */
            uint32_t invalid_pad_length; /* Number of packets with
invalid padding length */
            uint32_t invalid_seq_num; /* Number of packets with invalid
sequence number */
            uint32_t anti_replay_late_pkt; /* Number of packets that
failed anti-replay check through late arrival */
            uint32_t anti_replay_replay_pkt; /* Number of replayed
packets */
            uint32_t invalid_icv; /* Number of packets with invalid ICV
*/
            uint32_t seq_num_over_flow; /* Number of packets with
sequence number overflow */
            uint32_t crypto_op_failed; /* Number of packets where
crypto operation failed */
      }protocol_violation_errors;
      struct {
            uint32_t no_tail_room; /* Number of packets with no tail
room required for padding */
            uint32_t submit_to_accl_failed; /* Number of packets where
submission to underlying hardware accelerator failed */
      }process_errors;
}
```

9.349.39 g_ipsec_la_sa_get_outargs

```
struct  g_ipsec_la_sa_get_outargs {
{
      int32_t result; /* 0: Success: Non zero value: Error code
indicating failure */
```

```
      struct g_ipsec_la_sa *sa_params; /* An array of sa_params[] to
hold 'num_sas' information */
      struct g_ipsec_la_sa_stats *stats; /* An array of stats[] to hold
the statistics */
      g_ipsec_la_sa_handle ** handle; /* handle returned to be used for
subsequent Get Next N call */
};
```

~~9.35~~9.40 g_ipsec_la_sa_get_inargs

```
struct g_ipsec_la_sa_get_inargs
{
      enum g_ipsec_la_sa_direction dir; /* Direction: Inbound or
Outbound */
      /* Following field is not applicable for get_first */
      struct g_ipsec_la_sa_handle *handle;
      enum g_ipsec_la_sa_get_op operation; /* Get First, Next or Exact
*/
      uint32_t num_sas; /* Number of SAs to read */
      uint32_t flags; /* flags indicate to get complete SA information
or only Statistics */
}
```

~~9.36~~9.41 g_ipsec_la_data

```
struct g_ipsec_la_data {
      uint8_t *buffer; /* Buffer pointer */
      uint32_t length; /* Buffer length */
}
```

~~9.37~~9.42 g_ipsec_la_packet

```
struct g_ipsec_la_packet{
      uint32_t num_sg; /* Number of scatter gather elements */
      struct *g_ipsec_la_data;  /* array of buffer segments */
}
```

9.43g_ipsec_la_ipv6_addr

Formatted: Heading 2

```
struct g_ipsec_la_ipv6_addr{
        /*!  U8 Addr Len. */
      #define G_IPSEC_LA_IPV6_ADDRU8_LEN 16
          /*!  U32 Addr len */
      #define G_IPSEC_LA_IPV6_ADDRU32_LEN 4
       union {
                uint8_t b_addr[G_IPSEC_LA_IPV6_ADDRU8_LEN];
                        /**< Byte addressable v6 address */
                uint32_t w_addr[G_IPSEC_LA_IPV6_ADDRU32_LEN];
                        /**< Word addressable v6 address */
       };
```

```
};
```

9.44 g_ipsec_la_ip_addr

Formatted: Heading 2

```
struct g_ipsec_la_ip_addr {
        enum g_ipsec_la_ip_version version; /**< IP Version */
      /*! Union details
      */
        union {
                uint32_t ipv4;
                        /**< IPv4 Address */
                struct g_ipsec_la_ipv6_addr ipv6;
                        /**< IPv6 Address */
        };
};
```

10 Macros

```
#define G_IPSEC_LA_HANDLE_SIZE   8

#define G_IPSEC_LA_GROUP_HANDLE_SIZE  8

#define G_IPSEC_LA_SA_HANDLE_SIZE     8
```

11 Enumerations

11.1 g_ipsec_la_mode

```
enum g_ipsec_la_mode {

        G_IPSEC_LA_INSTANCE_AVAILABLE=0, /* Not assigned to any mode */

        G_IPSEC_LA_INSTANCE_EXCLUSIVE=1, /* Exclusive Mode */

        G_IPSEC_LA_INSTANCE_SHARED          /* Shared Mode */

};
```

11.2 g_ipsec_la_control_flags

```
enum g_ipsec_la_control_flags
{
      G_IPSEC_LA_CTRL_FLAG_ASYNC, /* If Set, API call be asynchronous.
Otherwise, API call will be synchronous */
      G_IPSEC_LA_CTRL_FLAG_NO_RESP_EXPECTED, /* If set, no response is
expected for this API call */
};
```

Application shall use the above data structure to pass the response requested – async or sync and whether a response is required or not. This structure is a parameter in most of the APIs.

11.3g_ipsec_la_auth_alg

```
enum g_ipsec_la_auth_alg {
      G_IPSEC_LA_AUTH_ALGO_NONE=1,     /* No Authentication */
      G_IPSEC_LA_AUTH_ALGO_MD5_HMAC   /* MD5 HMAC Authentication Algo.
*/
      G_IPSEC_LA_AUTH_ALGO_SHA1_HMAC  /* SHA1 HMAC Authentication Algo.
*
      G_IPSEC_LA_AUTH_AESXCBC,    /* AES-XCBC Authentication Algo. */
      G_IPSEC_LA_AUTH_ALG_SHA2_256_HMAC /* SHA2 HMAC Authentication
Algorithm; 256 bit key length */
      G_IPSEC_LA_AUTH_ALGO_SHA2_384_HMAC /* SHA2 HMAC Authentication
Algorithm with 384 bit key length */
      G_IPSEC_LA_AUTH_ALGO_SHA2_512_HMAC /* SHA2 HMAC Authentication
Algorithm with 512 bit key length */,
      G_IPSEC_LA_AUTH_ALGO_HMAC_SHA1_160
};
```

11.4g_ipsec_la_cipher_alg

```
enum g_ipsec_la_cipher_alg {
      G_IPSEC_LA_CIPHER_ALGO_NULL=1, /* NULL Encryption algorithm */
      G_IPSEC_LA_ALGO_DES_CBC,       /* DES-CBC Encryption Algorithm */
      G_IPSEC_LA_ALGO_3DES_CBC,
      G_IPSEC_LA_ALGO_AES_CBC,
      G_IPSEC_LA_ALGO_AES_CTR,
      G_IPSEC_LA_ALGO_COMB_AES_CCM, /* AES-CCM */
      G_IPSEC_LA_ALGO_COMB_AES_GCM, /* AES-GCM */
      G_IPSEC_LA_ALGO_COMB_AES_GMAC /* AES-GMAC */
```

~~11.4~~ ~~};~~

~~enum g_ipsec_la_cipher_alg {~~
~~G_IPSEC_LA_CIPHER_ALGO_NULL=1, /* NULL Encryption algorithm */~~
~~G_IPSEC_LA_ALG_DES_CBC, /* DES-CBC Encryption Algorithm */~~
~~G_IPSEC_LA_ALG_3DES_CBC,~~
~~G_IPSEC_LA_ALG_AES_CBC,~~
~~G_IPSEC_LA_ALG_AES_CTR~~
~~};~~

~~11.5g_ipsec_la_comb_alg~~

~~enum g_ipsec_la_comb_alg {~~
~~G_IPSEC_LA_COMB_AES_CCM=1, /* AES-CCM */~~
~~G_IPSEC_LA_COMB_AES_CCM, /* AES-CCM */~~
~~G_IPSEC_LA_COMB_AES_GMAC /* AES-GMAC */~~
~~};~~

~~11.6~~11.5 g_ipsec_la_ipcomp_alg

```
enum g_ipsec_la_ipcomp_alg {
      G_IPSEC_LA_IPCOMP_DEFLATE=1, /* Deflate IP Compression Algorithm
*/
      G_IPSEC_LA_IPCOMP_LZS /* LZS IP Compression Algorithm */
};
```

~~11.7~~11.6 g_ipsec_la_dscp_handle

```
enum g_ipsec_la_dscp_handle {
      G_IPSEC_LA_DSCP_COPY=1, /* copy from inner header to tunnel outer
header */
      G_IPSEC_LA_DSCP_CLEAR,      /* Clear the DSCP value in outer
header */
      G_IPSEC_LA_DSCP_SET,  /* Set the DSCP value in outer header to
specific value */
};
```

~~11.8~~11.7 g_ipsec_la_df_handle

```
enum g_ipsec_la_df_handle {
      G_IPSEC_LA_DF_COPY=1, /* Copy DF bit from inner to outer */
      G_IPSEC_LA_DF_CLEAR, /* Clear the DF bit in outer header */
      G_IPSEC_LA_DF_SET     /* Set the bit in the outer header */
};
```

~~11.9~~11.8 g_ipsec_la_sa_direction

```
enum g_ipsec_la_sa_direction {
      G_IPSEC_LA_~~IPSEC~~SA_INBOUND,
      G_IPSEC_LA_~~IPSEC~~SA_OUTBOUND
};
```

~~11.10~~11.9 g_ipsec_la_sa_flags

```
enum g_ipsec_la_sa_flags
{
      G_IPSEC_LA_SA_DO_UDP_ENCAP_FOR_NAT_TRAVERSAL = BIT(1),
      G_IPSEC_LA_SA_USE_ECN = BIT(2),
      G_IPSEC_LA_SA_LIFETIME_IN_KB = BIT(3),
      G_IPSEC_LA_SA_DO_ANTI_REPLAY_CHECK = BIT(4),
      G_IPSEC_LA_SA_ENCAP_TRANSPORT_MODE = BIT(5),
      G_IPSEC_LA_SA_USE_ESN=BIT(6),
      G_IPSEC_LA_SA_USE_IPv6=BIT(7),
      G_IPSEC_LA_NOTIFY_LIFETIME_KB_EXPIRY=BIT(8),
      G_IPSEC_LA_NOTIFY_SEQNUM_OVERFLOW=BIT(9),
      G_IPSEC_LA_NOTIFY_SEQNUM_PERIODIC=BIT(10)
};
```

~~11.11~~11.10 g_ipsec_la_inb_sa_flags

```
enum g_ipsec_la_inb_sa_flags {
      NF_IPSEC_INB_SA_PROPOGATE_ECN =1
      /* When set, ENC from outer tunnel packet will be propagated to
the decrypted packet */
};
```

~~11.12~~11.11 g_ipsec_la_sa_modify_replay_info_flags

```
enum g_ipsec_la_sa_modify_replay_info_flags {
      G_IPSEC_LA_SA_MODIFY_SEQ_NUM= BIT(1), /* Sequence number is being
updated */
      G_IPSEC_LA_SA_MODIFY_ANTI_REPLAY_WINDOW = BIT(2) /* Anti-replay
window is being updated */
};
```

~~11.13~~11.12 g_ipsec_la_sa_get_op

```
enum g_ipsec_la_sa_get_op {
      G_IPSEC_LA_SA_GET_FIRST_N = 0,
      G_IPSEC_LA_SET_GET_NEXT_N,
      G_IPSEC_LA_SA_GET_EXACT
};
```

11.13 g_ipsec_la_ip_version

Formatted: Heading 2

```
enum g_ipsec_la_ip_version {
        G_IPSEC_LA_IPV4 = 4, /**< IPv4 Version */
        G_IPSEC_LA_IPV6 = 6 /**< IPv6 Version */
};
```