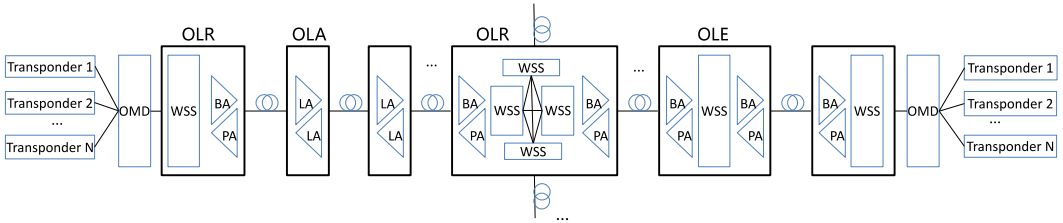
Introduction to Line-card Abstraction Interface (LAI)

Alibaba Cloud
Xiaodong Gui (guixiaodong.gxd@alibaba-inc.com)

What is the problem?

- A typical white-box optical transport equipment has a main board and several line-cards.
- Each line-card has some optical modules, such as transponders, optical amplifiers, etc.
- Different line-cards and modules may be produced by different vendors.
- How to describe these line-cards and modules?
  - northbound interfaces used today:
    OpenConfig, OpenROADM
  - southbound interfaces used today:
    Transponder Abstraction Interface (TAI), Switch Abstraction Interface (SAI)

A whitebox optical transport equipment

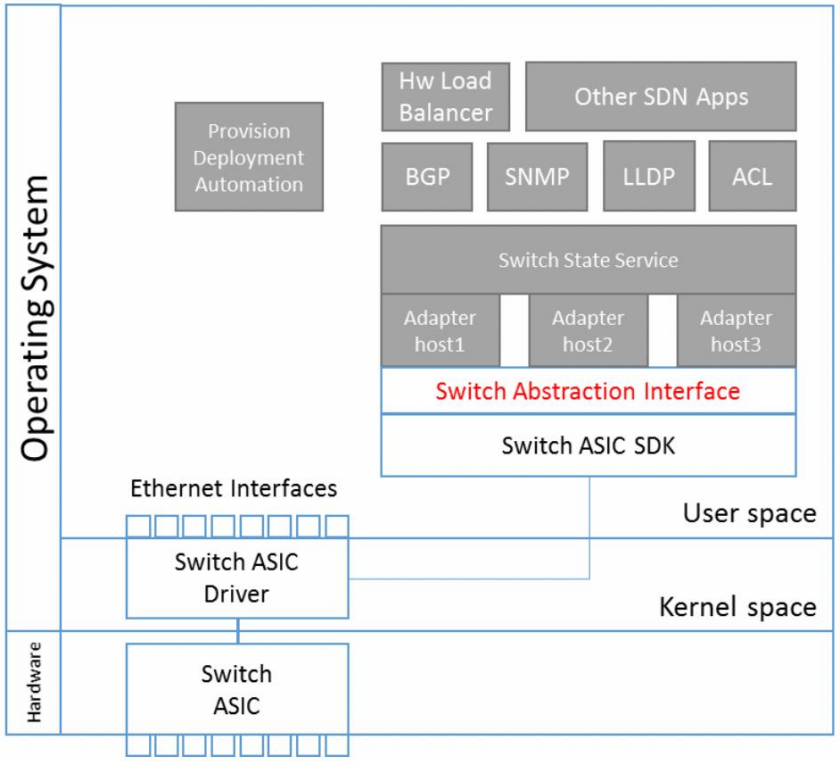


Optical modules in an optical network

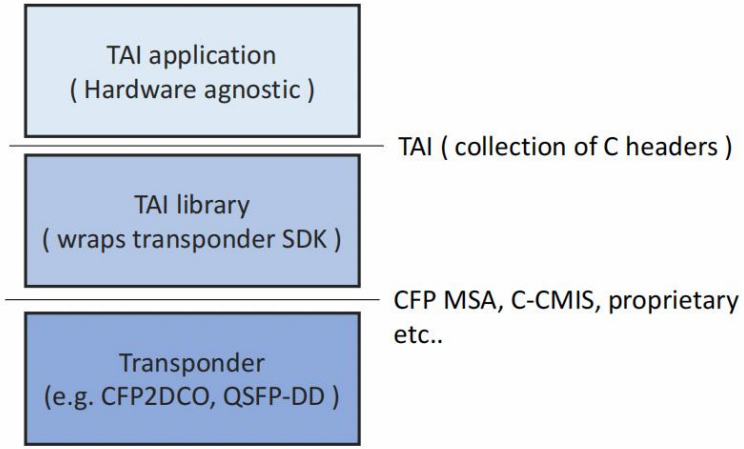
- OMD: optical multiplexer and demultiplexer
- WSS: wavelength selective switch
- BA: pre-amplifier
- PA: booster amplifier

Why not SAI or TAI ?

- SAI is designed for switch ASICs and NPUs.
- TAI is designed for transponders and transceivers.



SAI Architecture



TAI Architecture

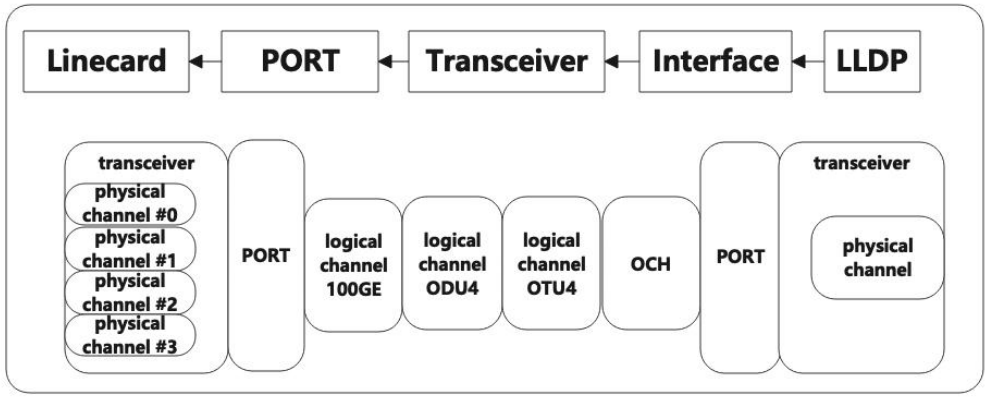
The idea of LAI

- SAI Framework
  - object-oriented, object id, CRUD API (Create/Read/Update/Delete) ...

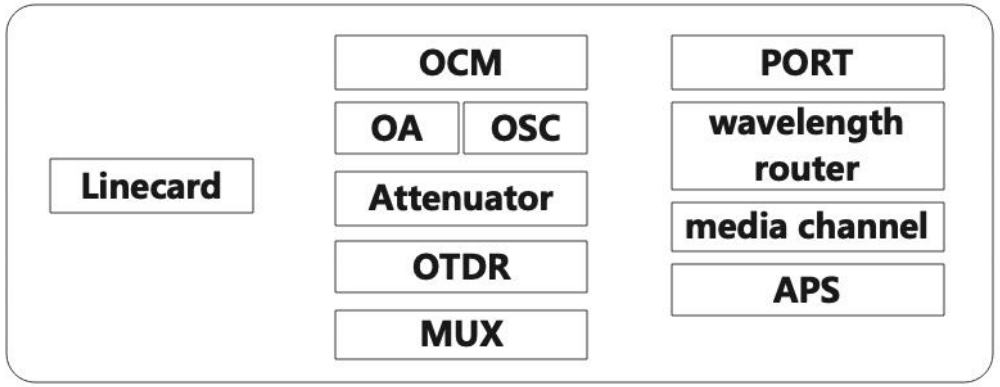
- OpenConfig Models:
  - line-card, port, transceiver, interface, logical channel, physical channel ...
  - ocm, oa, osc, attenuator, wavelength router, aps ...

- LAI inherits the framework from SAI, and its objects from OpenConfig.
  It defines a set of APIs to control and monitor different optical line-cards in a unified manner.

SAI Framework + OpenConfig Model => LAI



OpenConfig models in an optical terminal system card



OpenConfig models in an optical line system card

LAI Architecture

Terminology

- Optical Line-card State Service (OLSS)

The OLSS is a collection of software that provides a database interface for communication between NBI and line-card.

- Adapter Host

The adapter host is a component that loads adapter and exposes its functionalities to the OLSS .

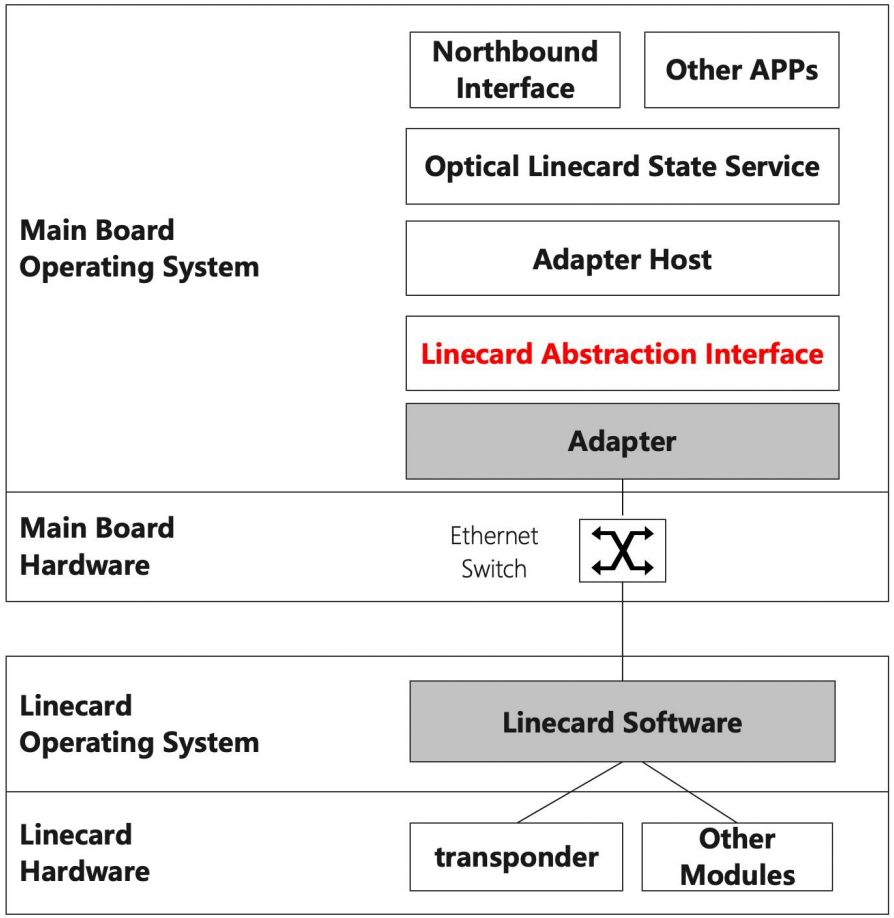
- Adapter

The adapter is a shared software library that implements LAI specifications. It is supplied by line-card vendors.

- LAI

LAI is an interface specification implemented as a collection of C-language header files.

LAI-Github-URL

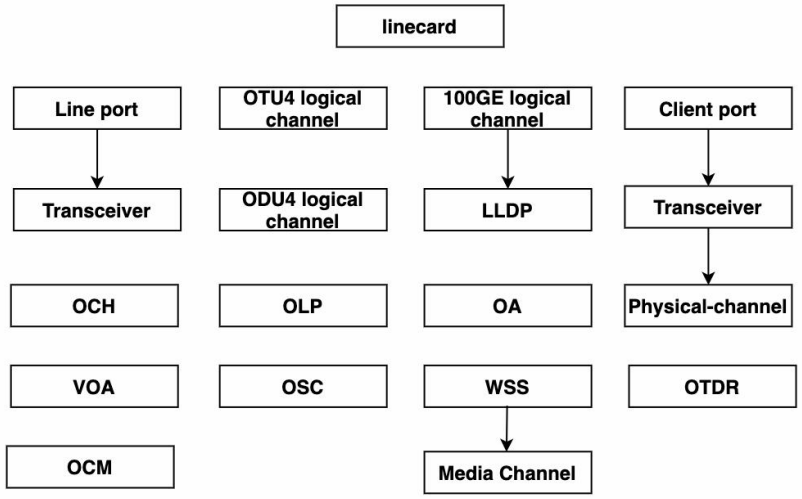


Tip. The grey blocks are developed by vendors

LAI in an optical transport white-box system

LAI object

- Top-level object:
  - line-card
- Objects on an optical terminal system card:
  - port, transceiver, logical-channel, otn, ethernet, physical-channel, och, lldp ...
- Objects on an optical line system card:
  - oa, osc, aps, attenuator, wss, media-channel, ocm, otdr, port ...
- Each object owns a unique object ID (oid) which is assigned by LAI library.
- Each object has a set of attributes and statistics.
- Each object has a set of CRUD(Create/Read/Update/Delete) APIs.



LAI object model and hierarchy

Acronyms:

oa - optical amplifier
osc - optical supervisory channel
aps - automatic protection switching
wss - wavelength selective switch
ocm - optical channel monitor
otdr - optical time domain reflectometer

LAI attribute

- Each LAI object has a set of attributes.
- One or more attributes describe a certain feature of this LAI object.
- The annotations of an attribute indicate its type, read-write property, etc.

```
/**
 * @brief Enabled
 *
 * @type bool
 * @flags CREATE_AND_SET
 */
LAI_OA_ATTR_ENABLED,
```

boolean type

settable

An example of a pre-defined attribute

```
typedef union _lai_attribute_value_t
{
    bool booldata;
    char chardata[512];
    lai_uint8_t u8;
    lai_int8_t s8;
    lai_uint16_t u16;
    lai_int16_t s16;
    lai_uint32_t u32;
    lai_int32_t s32;
    lai_uint64_t u64;
    lai_int64_t s64;
    lai_double_t d64;
    lai_pointer_t ptr;
    lai_object_id_t oid;
    lai_object_list_t objlist;
    lai_u8_list_t u8list;
    lai_s8_list_t s8list;
    lai_u16_list_t u16list;
    lai_s16_list_t s16list;
    lai_u32_list_t u32list;
    lai_s32_list_t s32list;
    lai_u32_range_t u32range;
    lai_s32_range_t s32range;
} lai_attribute_value_t;

typedef struct _lai_attribute_t
{
    lai_attr_id_t id;
    lai_attribute_value_t value;
} lai_attribute_t;
```

The definition of lai_attribute_t

LAI statistics

- The LAI statistics contain counter and gauge values. For instance, the total number of bad frames of a transceiver is a counter value, while the output power of a transceiver is a gauge value.
- The annotations of a statistic indicate its type, is counters or not, unit, precision, etc.

```
typedef union _lai_stat_value_t
{
    lai_int32_t s32;
    lai_uint32_t u32;
    lai_int64_t s64;
    lai_uint64_t u64;
    lai_double_t d64;
} lai_stat_value_t;
```

The definition of lai_stat_value_t

```
/**
 * @brief Tx bad frame
 *
 * @type lai_uint64_t
 * @iscounter true
 */
LAI_ETHERNET_STAT_TX_BAD_FRAME,
```

unsigned int type

is a counter

An example of a pre-defined statistics

```
/**
 * @brief Output power
 *
 * @type lai_double_t
 * @unit dBm
 * @precision precision2
 * @iscounter false
 */
LAI_TRANSCEIVER_STAT_OUTPUT_POWER,
```

double type

unit is dBm

is a gauge

Another example of a pre-defined statistics

LAI API

- ***lai_api_initialize***
  - This API allows the adapter to initialize various data structures for subsequent use.
- ***lai_link_check***
  - Adapter host calls this API to check the link status between adapter and line-card software.
- ***lai_api_query***
  - This API is used for retrieval of various method tables for LAI functionalities.
- ***lai_api_uninitialize***
  - This API is the inverse function to lai_api_initialize.

- ***lai_create_xxx_fn*** - create an object
- ***lai_remove_xxx_fn*** - delete an object
- ***lai_set_xxx_attribute_fn*** - set an attribute of an object
- ***lai_get_xxx_attribute_fn*** - get an attribute value of an object
- ***lai_get_xxx_stats_fn*** - get a statistics value of an object

LAI line-card creation

- LAI line-card is a mandatory object.
- Calling the line-card create function with an array of attributes.
- The line-card type①, the alarm notification function② and the line-card state notification function③ are passed.
- LAI adapter assigns an object id for this line-card.

```
lai_object_id_t oid;
lai_attribute_t attrs[3];
lai_status_t rv;

attrs[0].id = LAI_LINECARD_ATTR_LINECARD_TYPE;                    ①
attrs[0].value.s32 = LAI_LINECARD_TYPE_P230C;

attrs[1].id = LAI_LINECARD_ATTR_LINECARD_ALARM_NOTIFY;            ②
attrs[1].value.ptr = linecard_alarm_notify;

attrs[2].id = LAI_LINECARD_ATTR_LINECARD_STATE_CHANGE_NOTIFY;     ③
attrs[2].value.ptr = linecard_state_change_notify;

rv = lai_linecard_api->create_linecard(&oid, 3, attrs);
```

An example of LAI linecard creation

LAI line-card operation

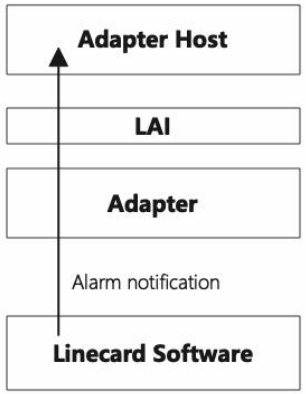
- Calling the line-card set attribute function to warm reset the line-card.
- Calling the line-card get attribute function to get the line-card software version.
- Calling the line-card get statistics function to get the line-card temperature value.

```
// warm reset linecard
attr.id = LAI_LINECARD_ATTR_RESET;
attr.value.s32 = LAI_LINECARD_RESET_WARM;
rv = lai_linecard_api->set_linecard_attribute(oid, &attr);

// get linecard software version
attr.id = LAI_LINECARD_ATTR_SOFTWARE_VERSION;
rv = lai_linecard_api->get_linecard_attribute(oid, 1, &attr);

// get linecard temperature
lai_stat_id_t stat_id = LAI_LINECARD_STAT_TEMPERATURE;
lai_stat_value_t stat_value;
rv = lai_linecard_api->get_linecard_stats(oid,
                                          1,
                                          &stat_id,
                                          &stat_value);
```

Examples of LAI line-card operations

Alarm notification

- When an alarm occurs (or disappears) on the line-card, the adapter calls the alarm notification function to notify upper application.
- The *lai_alarm_type_t* enumeration contains all types of alarms.
- The *lai_alarm_info_t* structure contains detailed information, such as severity, source, created time, active or inactive.



Alarm notification process

```
typedef struct _lai_alarm_info_t
{
    lai_alarm_status_t status;
    uint64_t time_created;
    lai_s8_list_t text;
    lai_s8_list_t resource;
    lai_alarm_severity_t severity;
} lai_alarm_info_t;
```

The definition of alarm info

```
typedef enum _lai_alarm_type_t
{
    LAI_ALARM_TYPE_BOARD_INIT,
    LAI_ALARM_TYPE_BOARD_LOAD_FILE_FAILED,
    LAI_ALARM_TYPE_BOARD_LOADING,
    LAI_ALARM_TYPE_BOARD_LOAD_FAILED,
    LAI_ALARM_TYPE_BOARD_LOAD_ACTIVE,
    ...
} lai_alarm_type_t;
```

The definition of alarm type

```
typedef void (*lai_linecard_alarm_notification_fn)(
        _In_ lai_object_id_t linecard_id,
        _In_ lai_alarm_type_t alarm_type,
        _In_ lai_alarm_info_t alarm_info);
```

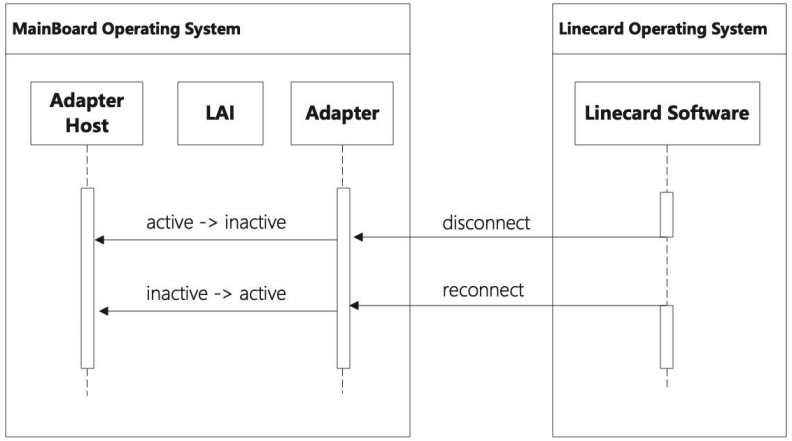
The definition of alarm notification function

Line-card oper-state change notification

- When the line-card operational state changes (e.g. rebooting, lose connection), the adapter calls the line-card state notification function to notify upper application.
- When the line-card operational state is inactive, the adapter host won’t operate the line-card until its state turn to active.
- When the line-card operational state change from inactive to active. The adapter host may re-config it.

```
typedef enum _lai_oper_status_t
{
    LAI_OPER_STATUS_ACTIVE,
    LAI_OPER_STATUS_INACTIVE,
    LAI_OPER_STATUS_DISABLED,
} lai_oper_status_t;

typedef void (*lai_linecard_state_change_notification_fn)(
        _In_ lai_object_id_t linecard_id,
        _In_ lai_oper_status_t linecard_oper_status);
```

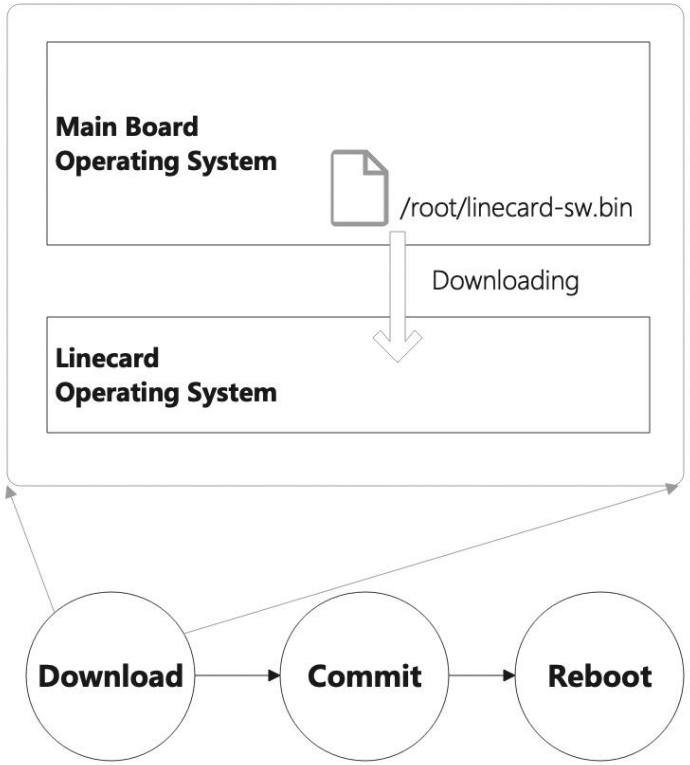
The definition of operational state change notification function



Line-card state change notification process

LAI line-card software upgrade

- The line-card software upgrade has three stages: downloading, committing and rebooting.
- Downloading the software image from a certain path in the main board.
- Installing the new software image. (committing)
- Rebooting the line-card to reload the new software image.
- They are executed individually via the following attributes:
  - *LAI_LINECARD_ATTR_UPGRADE_DOWNLOAD*
  - *LAI_LINECARD_ATTR_UPGRADE_COMMIT*
  - *LAI_LINECARD_ATTR_UPGRADE_REBOOT*



Linecard software upgrade process

Q & A