TAI Me Up!

A Detailed Look at the Transponder Abstraction Interface

Scott Emery

Terminology

- Optical transceiver
  - A hardware component which converts electrical signals to/from light
  - Sometimes called a transponder
  - May or may not adhere to some form of the MSA MIS
- TAI Adapter
  - The hardware-specific component of TAI
  - A user-mode device driver
  - Implemented as a shared object library: libtai.so
  - Sometimes called a driver or device-specific code
  - Not thread-safe, non-reentrant
- TAI Adapter Host
  - The code which interfaces to the TAI adapter
  - Sometimes called a TAI host, application, or device-independent code
  - Typically a daemon or systemd service

Terminology (cont.)

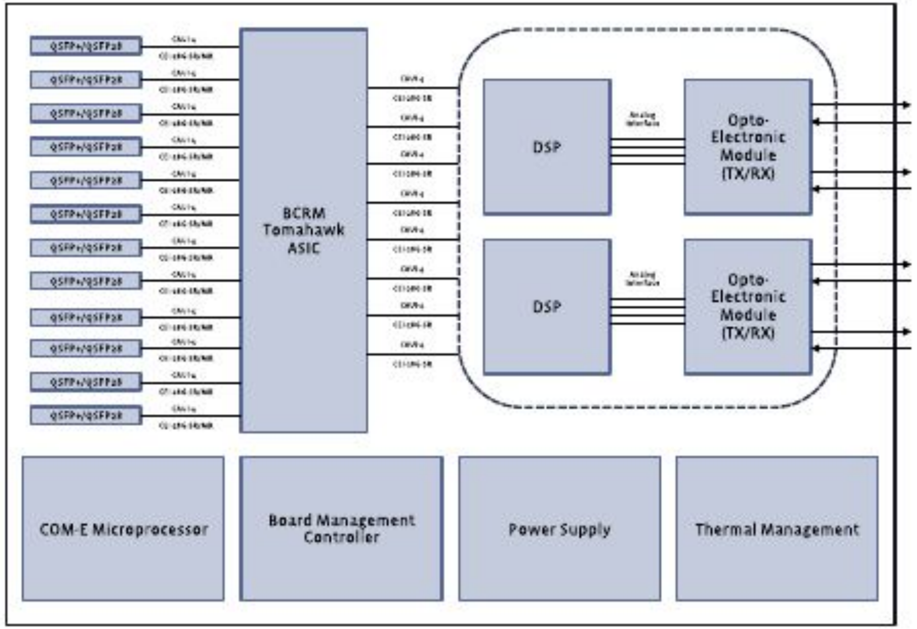
- TAI Module
  - The combination of a DSP and optical transceiver
    - CFP2 DCO modules
    - CFP2 ACO modules plus the external DSP
    - An AC400 5x7 module
- Network Interface
  - An optical interface, which connects externally to some type of optical line system
  - The “external” optical port(s) of an optical module
- Host Interface
  - The datapath electrical interface(s) of the TAI module
  - In Voyager and Cassini are connected to the Broadcom Tomahawk ASIC

Terminology (cont.)

- TAI Object
  - Currently one of these three types: Module, Host Interface, Network Interface
    - Also called a TAI API
    - Each object type has a set of attributes
  - Created for each instance in the system, e.g. a network interface object is created for each network interface in the system
    - Each object instance is given an object ID
- TAI Attribute
  - A property of an object instance which can be read and possibly written
  - For example:
    - Module objects have a serial number attribute which can be read
    - Network interface objects have an output power attribute which can be read/written
  - Types are defined in taitypes.h
  - Object attributes defined in enums in taimodule.h, tainetworkif.h, and taihostif.h
  - Custom attributes are allowed

Voyager High Level

- Tomahawk
  - 12 “Client” ports (swpX, QSFP28)
  - 8 “Host” ports to AC400s (swpLX)
- COM-e Microprocessor
  - 4-core Atom E3845 @ 1.9GHz
  - 8G DRAM
  - 128G SSD
- BMC
  - AST2520
  - OpenBMC (not upstreamed)
  - Random eth0 MAC address
- 2 x AC400
  - Acacia
  - 2 x 200G DWDM each

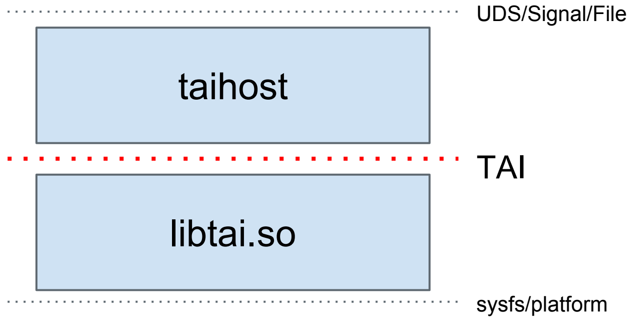

QSFP+/QSFP28
BCRM Tomahawk ASIC
DSP
DSP
Opto-Electronic Module (TX/RX)
Opto-Electronic Module (TX/RX)
COM-E Microprocessor
Board Management Controller
Power Supply
Thermal Management

Cassini High Level

- Tomahawk+
  - 16 “Host” ports (QSFP28)
  - 8 module slots (2 CAUI-4 each)
- CPU Subsystem
  - 4-core Broadwell-DE 1.6GHz
  - 8G DRAM
  - 32G Flash
- Module slots accept different cards
  - LCDCO-1
    - CFP2 DCO module slot
    - PHY for MACsec
  - LCQSFP-1
    - 2 QSPF28 modules
    - PHY for MACsec
  - LCACO-1
    - CFP2 ACO module slot
    - DSP

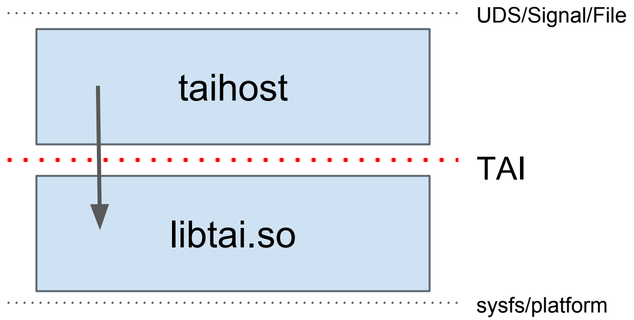
The TAI Interface

- Only 7 public symbols are exposed by TAI adapter
    - tai_api_initialize(), tai_api_query(), tai_api_uninitialize()
    - tai_log_set(), tai_object_type_query(), tai_module_id_query(), tai_dbg_generate_dump()
- Prototypes defines in tai.h
- But... several of these functions pass additional function pointers



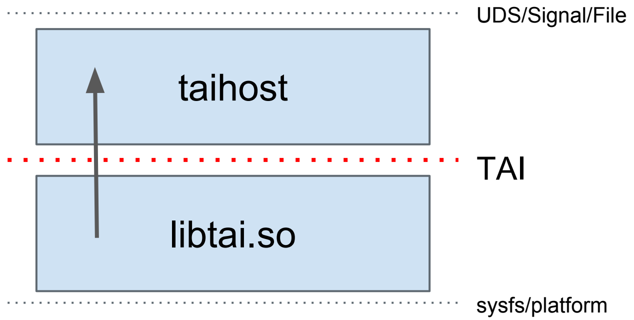
tai_api_initialize()

- Provides initial entry point for TAI adapter code
  - Initialize variables
  - Allocate memory
  - Start threads/hook ISRs
- Does NOT touch the hardware
- Receives flags (unused) and pointer to a Service table (function pointers in TAI host)
  - Currently only one function pointer: module_presence()
- Not intended to take a long time, should quickly initialize and return



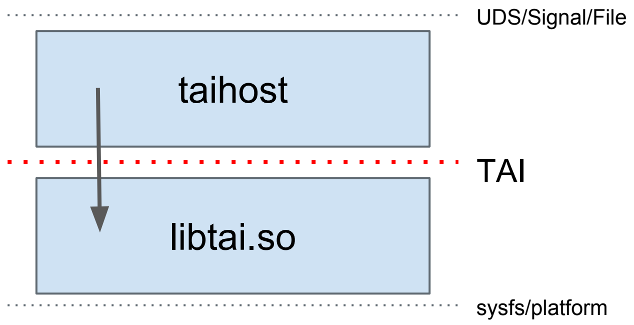
module_presence()

- Is called whenever a module is detected or removed
- Can be called in any context
  - Different thread
  - ISR
  - Before tai_api_initialize() returns
- Since TAI adapter code is not reentrant care must be taken to not call TAI adapter recursively
- Passes a presence/absence flag and module location (string)
  - Module location string is used in subsequent module object creation function call



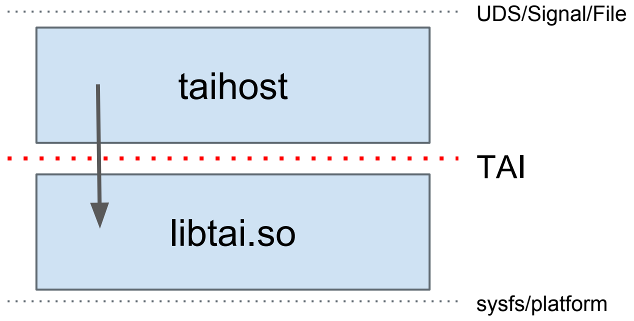
tai_api_query()

- Retrieve a method table (function entry points) for an object type
- Allows TAI host to get the TAI adapter methods for an object type
    - Module
    - Host interface
    - Network Interface
- Returns a pointer to a structure in TAI adapter with function entry points
    - Structure must remain valid until tai_api_uninitialize() is called



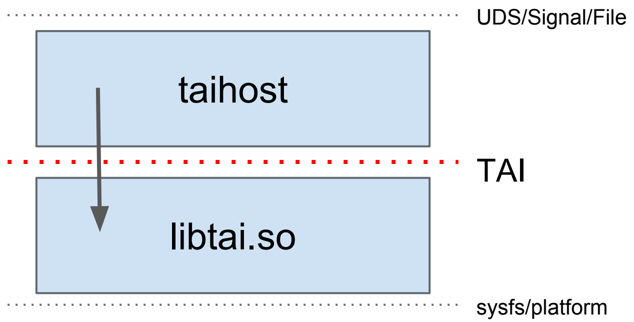
tai_api_query() - cont.

- Currently the methods of all objects are similar:
    - create_module()
    - remove_module()
    - set_module_attribute()
    - set_module_attributes()
    - get_module_attribute()
    - get_module_attributes()
        - Replace "module" with "network_interface" or "host_interface"
- But this may change with additional capabilities
    - Statistics, Performance monitoring, FAWS, etc.
- More on these functions later



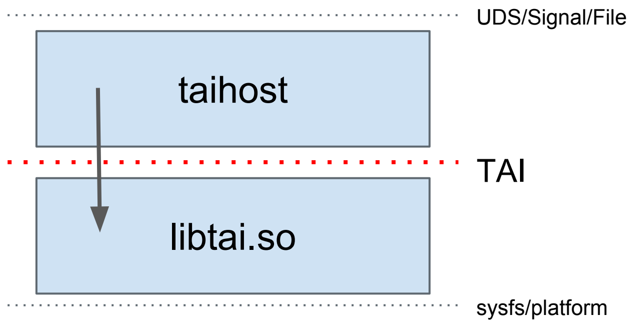
tai_api_uninitialize()

- Provides for TAI adapter code to clean up
  - Free memory
  - Stop threads/un-hook ISRs
- No parameters
- Typically called when TAI Host exits



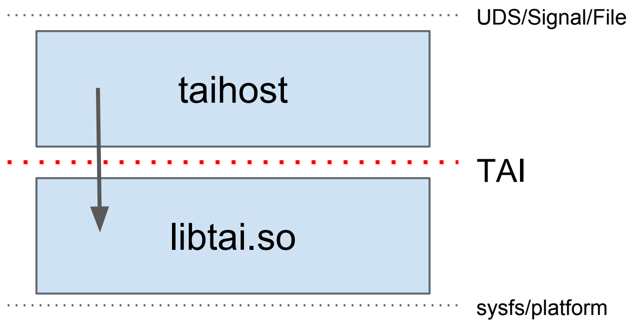
tai_log_set()

- Sets the logging level in the TAI adapter
- Two parameters
  - TAI API (object type)
    - Different logging levels can be set for each object type
  - Logging level
    - Follows standard syslog logging levels
    - Debug, Info, Notice, Warn, Error, Critical
- Enables logging above and including the selected level
  - E.g. Selecting Warn enables Warn, Error, and Critical messages
- Default logging level is Warn



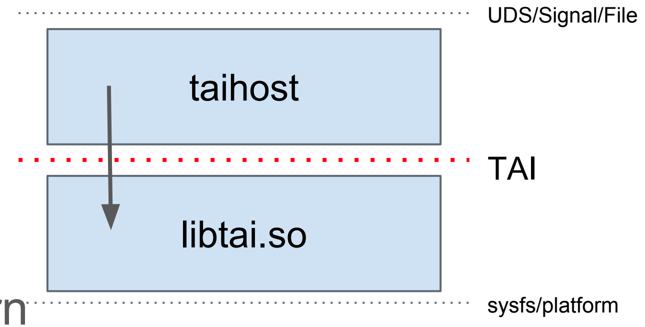
tai_object_type_query()

- All object IDs are 64-bit opaque values provided by the TAI Adapter to the TAI Host
    - Each TAI Adapter defines the format of the object IDs used by that TAI Adapter
- Given an object ID, return the object type
- For example, passing in an object ID for a network interface object, will return that the object is a network interface type



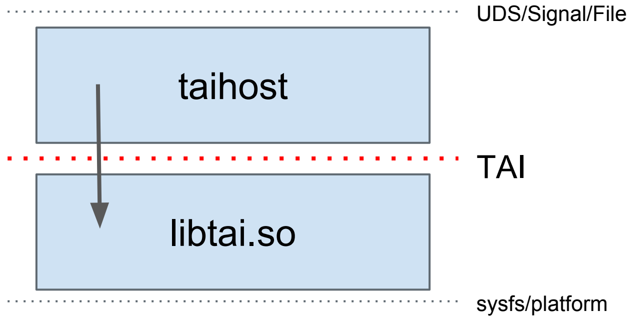
tai_module_id_query()

- Given any object ID, return the module ID to which that object belongs
- For example, network interface and host interface objects “belong” to a module. Calling this function with a network interface object ID will return the module object ID to which that network interface belongs.
- Calling with a module object ID will simply return that same module object ID



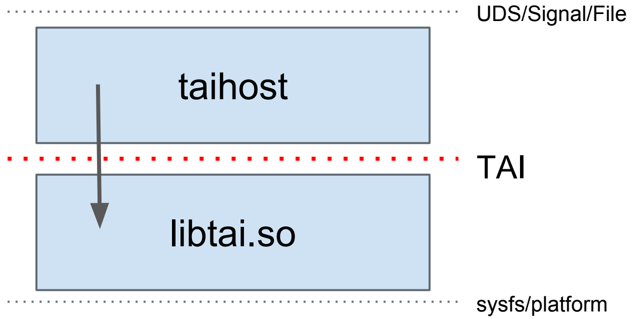
tai_dbg_generate_dump()

- Requests that the TAI adapter dump its state to a file for debugging purposes
- A filename (char *) is passed
- The format of the dump file is not specified by TAI
  - Specific for each TAI adapter



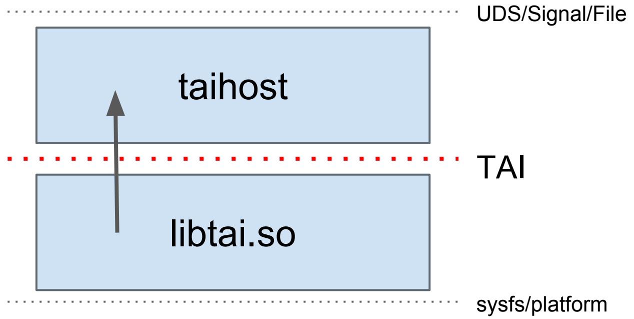
create_module()

- Called to create a module instance
  - Typically invoked as a result of a module_presence() call with the present flag set
- TAI host supplies
  - A list of attributes to set
    - Must include the module location (value provided by the module_presence() function
  - Pointer to a function notification table
    - Allows TAI adapter to notify TAI host
      - Shutdown request
      - State change
- TAI adapter
  - Initializes module to default state and applies attributes
  - Returns module object ID



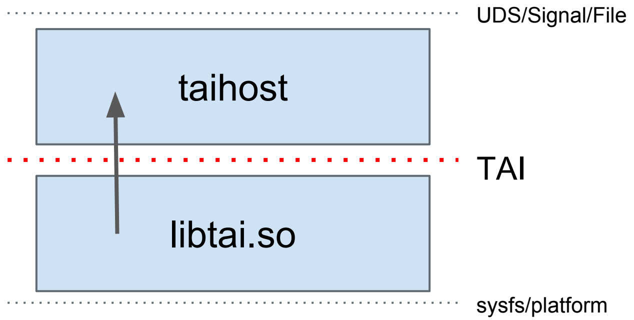
shutdown_request()

- Called by the TAI Adapter to request the module be removed
  - An unrecoverable error
  - For maintenance
- TAI adapter supplies the module ID
- Will typically result in a module_remove()
  - Again, be careful to avoid re-entrant code



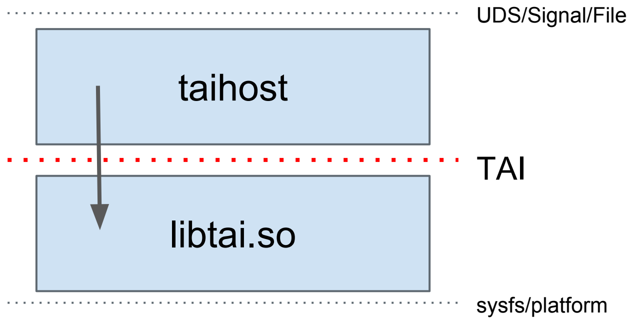
state_change_notification()

- Called by the TAI Adapter to provide notification of a change of state in the module
  - E.g. Initialize or Ready
- TAI adapter supplies the module ID and the new state
- Adapter host can react (or ignore) these changes



remove_module()

- Called to remove a module instance
  - Typically invoked as a result of a module_presence() call with the present flag clear, or when exiting
- TAI host supplies the module object ID
- TAI adapter
  - May or may not reset module to default state
  - Cleans up any resources allocated to the module
    - Question: Should it “remove” any network interface and host interface objects that are part of the module and not yet removed?

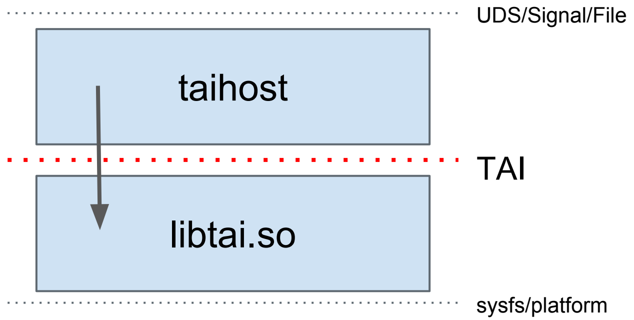


Module Attributes

- Most attributes for a module are read-only. Two are **read-write**
- Location - Somewhat opaque character string. Required on create, can only be supplied in module_create() attribute list
- Vendor name, part number, serial number, firmware versions
- Operational status, **administrative status**
- Temperature, input voltage
- Number of network interfaces, host interfaces
- **Tributary mapping**

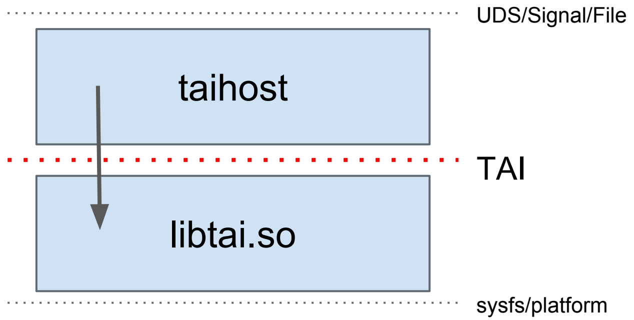
set_module_attribute()

- Called to set the value of a single attribute
- TAI host supplies
  - module object ID
  - Pointer to an attribute
- TAI adapter
  - Performs whatever operations are necessary to modify the hardware to the supplied attribute value



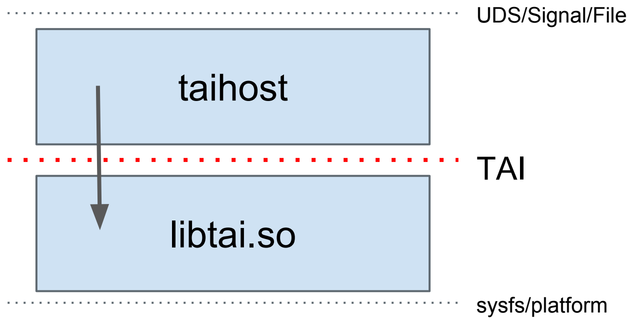
set_module_attributes()

- Called to set the value of multiple attributes on a single module
- TAI host supplies
  - module object ID
  - The number of attributes in the list
  - Pointer to a list of attributes
- TAI adapter
  - Typically loops through the supplied attribute list calling set_module_attribute()



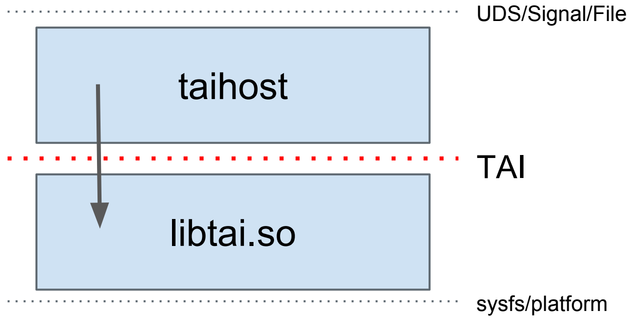
get_module_attribute()

- Called to retrieve the value of a single attribute
- TAI host supplies
  - module object ID
  - Pointer to an attribute with the ID to retrieve
- TAI adapter
  - Performs whatever operations are necessary to read the current value of the attribute
  - Put the retrieved value in the location provided



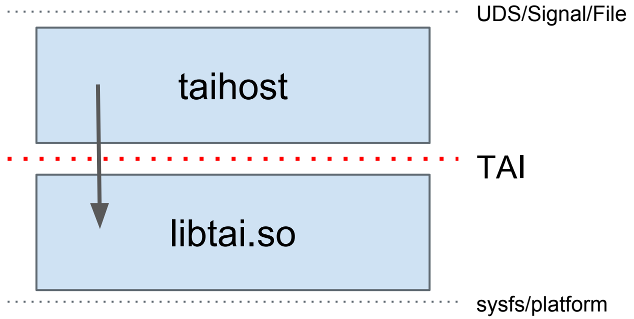
get_module_attributes()

- Called to retrieve the value of multiple attributes on a single module
- TAI host supplies
  - module object ID
  - The number of attributes in the list
  - Pointer to a list of attributes with attribute IDs
- TAI adapter
  - Typically loops through the supplied attribute list calling get_module_attribute()



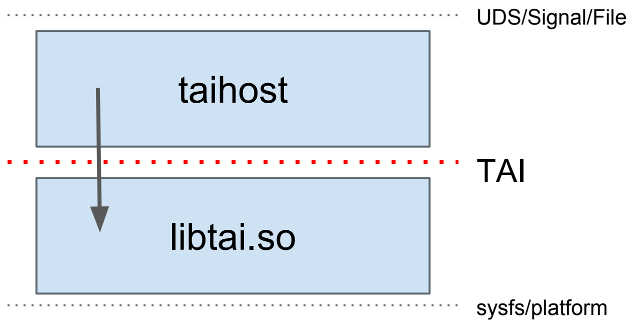
create_network_interface()

- Called to create a network interface instance
  - NUM_NETWORK_INTERFACES attribute of module object can be used to determine how many network interface instances to create
- TAI host supplies
  - A list of attributes to set
    - Must include the network interface index (zero-based) on the module
  - The module ID upon which the network interface exists
- TAI adapter
  - Initializes network interface to default state and applies attributes
  - Returns network interface object ID
- NOTE: There are no notification functions



remove_network_interface()

- Called to remove a network interface instance
  - Typically invoked prior to calling remove_module()
- TAI host supplies the network interface object ID
- TAI adapter
  - May or may not reset network interface to default state
  - Cleans up any resources allocated to the network interface



Network Interface Attributes

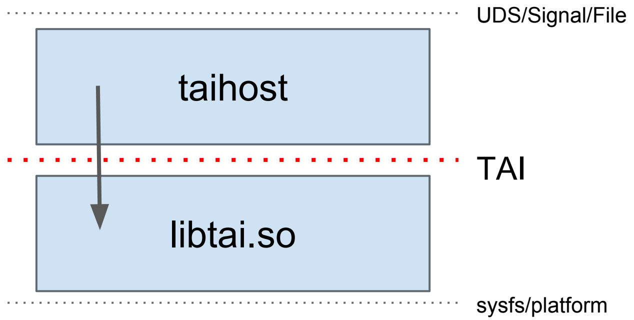
- Many attributes are **read-write**, some are read-only
- Index - Zero-based index of network interfaces on a module. Required on create, can only be supplied in network_interface_create() attribute list
- **TX Enable** - This is inversion of the well-known TX_DIS
- **Grid Spacing, Output power,** Measured output power
- **TX Channel number, Channel by frequency, channel by lambda** - Overlap each other, three ways to set the same thing
- **Modulation format, Differential encoding**
- **Operational Status**
- **Pulse shaping: RX, TX, RX Beta, TX Beta**
- **RX VOA attenuation**

Network Interface Attributes (cont.)

- Current Laser Freq, Fine-tune laser freq, Min/Max Laser freq, Grid support
- TX/RX Alignment status
- BER, BER Period

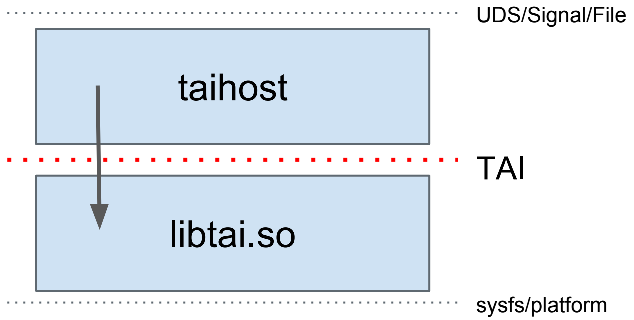
set/get_network_inteface_attribute/s()

- Same idea as the module object methods of the similar name



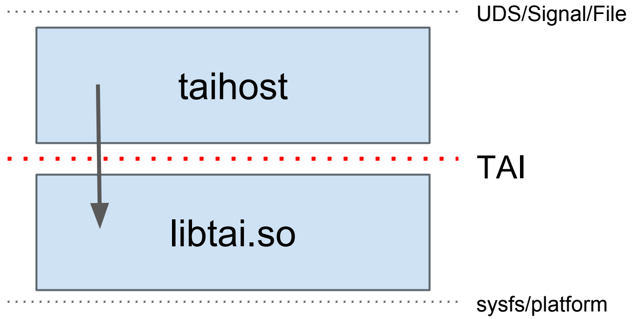
create_host_interface()

- Called to create a host interface instance
  - NUM_HOST_INTERFACES attribute of module object can be used to determine how many network interface instances to create
- TAI host supplies
  - A list of attributes to set
    - Must include the host interface index (zero-based) on the module
  - The module ID upon which the network interface exists
- TAI adapter
  - Initializes host interface to default state and applies attributes
  - Returns host interface object ID
- NOTE: There are no notification functions



remove_host_interface()

- Called to remove a host interface instance
  - Typically invoked prior to calling remove_module()
- TAI host supplies the host interface object ID
- TAI adapter
  - May or may not reset host interface to default state
  - Cleans up any resources allocated to the host interface

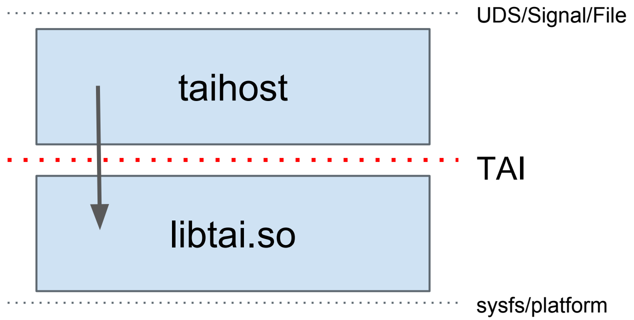


Host Interface Attributes

- Many attributes are **read-write**, some are read-only
- Index - Zero-based index of host interfaces on a module. Required on create, can only be supplied in host_interface_create() attribute list
- Lane Faults, TX Alignment status
- **FEC Type**
- There will typically be some custom attributes for SERDES configuraiton

set/get_network_inteface_attribute/s()

- Same idea as the module/network interface object methods of the similar name



CUMULUS

Thank you!

for staying awake

Visit us at cumulusnetworks.com or follow us @cumulusnetworks