endace
accelerated

Configuration and Status API Programming Guide

EDM04-08

www.endace.com

Published by:

Endace Measurement Systems® Ltd

Building 7
17 Lambie Drive

PO Box 76802
Manukau City 1702
New Zealand

Phone: +64 9 262 7260

Fax: +64 9 262 7261

support@endace.com

www.endace.com

International Locations

New Zealand	**Americas**	**Europe, Middle East & Africa**
Endace Technology® Ltd	Endace USA® Ltd	Endace Europe® Ltd
Level 9 85 Alexandra Street PO Box 19246 Hamilton 2001 New Zealand	Suite 220 11495 Sunset Hill Road Reston Virginia 20190 United States of America	Sheraton House Castle Park Cambridge CB3 0AX United Kingdom
Phone: +64 7 839 0540 Fax: +64 7 839 0543	Phone: ++1 703 382 0155 Fax: ++1 703 382 0155	Phone: ++44 1223 370 176 Fax: ++44 1223 370 040

Table of Contents

Chapter 1: Introduction

Overview

The Endace range of DAG cards provide the means to transfer data at the full speed of a network into the memory of a host PC, with zero packet loss guaranteed in even the worst-case conditions.

The present range of Endace Accelerated™ card products provide the benefits of Endace technology across the broad array of contemporary network standards, ranging from legacy copper T1/E1 through to modern high-speed optical OC192/STM-64 and 10G Ethernet.

The Endace Configuration and Status Application Programming Interface (API) enables developers to configure the varied range of components and associated attributes of a DAG card, previously only available through the individual DAG command line utilities.

It allows allow third-party developers to perform the following tasks from within their own application software:

- Resetting a DAG card.
- Loading firmware images onto a DAG card.
- Setting and retrieving the hardware configuration.
- Retrieving status and statistics information.

Supported Cards

The Configuration & Status API supports the following DAG cards.

- DAG 3.7D
- DAG 3.7GE
- DAG 3.7T
- DAG 3.8S
- DAG 4.3S
- DAG 4.3GE
- DAG 4.5G
- DAG 6.2SE
- DAG 7.1S

Please refer to the appropriate DAG Card User Guide available from the Endace Customer Support website at support@endace.com for detailed information on the features and functionality of each DAG card.

Purpose

The purpose of this Programming Guide is to:

- Provide general information about the Configuration and Status API
- Describe DAG Card components and attributes
- Define function definitions
- Describe data structures and constants

Thread Safety

! Please note that the routines described in this Programming Guide are not thread safe or reentrant.

If you are using multiple threads Endace strongly recommended that you use wrapper functions to serialize access to the Endace supplied routines.

Support

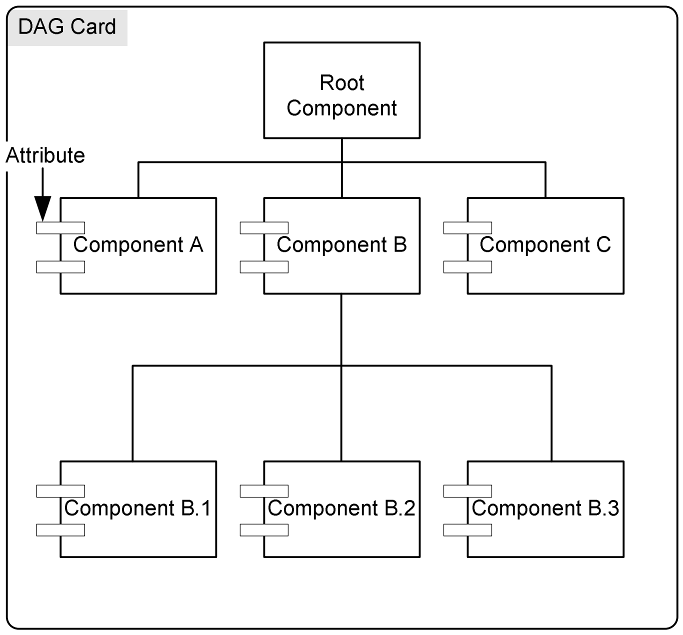
If you encounter any problems either installing or using Endace supplied hardware, firmware or software and you are unable to resolve them, please contact Endace Customer Support at support@endace.com.for further assistance

Note: To enable Endace Customer Support to respond to your problem as quickly and efficiently as possible please supply as much detailed information as possible when reporting your problem.

Chapter 2: Theory of Operation

Mental Model

The mental model of a DAG card implied by the API is a hierarchical tree of components with each of those components having a set of attributes associated with it as shown below:



The Configuration & Status API divides a component into several distinct attributes. It then allows you to configure the behaviour of those attributes by reading and writing values to and from the DAG card.

Note: Not all components and attributes are common to all DAG Cards.

Card Configuration

Attribute Reference

Before you can change a card’s configuration you must first obtain a reference to the card, then a reference to the desired component, and finally a reference to the component’s attribute that you wish to change.

Once you have the attribute reference you can use it to retrieve and modify the attribute value.

For example, to see if a particular port is active, you would first obtain a reference to the card, then a reference to the port component, and finally a reference to the port component `'active'` attribute.

Attribute Value

Reading the value returned by the attribute reference provides information about the port status. Writing a value to the attribute reference would configure the port status.

A sample program that displays the `'active'` attribute for all parts on a DAG card is shown in Example Program later in this chapter.

Attribute Type

There are two types of attributes associated with components on DAG cards. There are those used to represent status and statistics information, and those used to represent configuration information.

You can use the `dag_config_get_attribute_config_status` function to check if an attribute is marked as a status or configuration attribute.

Configuration Attribute

Configuration attributes represent properties of the card that can be modified. They include such items as:

- POS or ATM mode for SONET cards
- Auto-negotiation mode on/off for Ethernet cards
- Variable or fixed-length packet capture
- Snap length for packet capture
- Amount of memory allocated to each receive and transmit stream

Status Attribute

Status attributes represent the card properties that are read-only and can not be modified. They include such items as:

- Physical layer error indicators.
- PCI bus speed.
- Number of frames that failed the Frame Checksum.
- Number of receive and transmit streams supported by the firmware.

Note: The precise set of attributes and components presented by the API depends on the model of DAG card and the capabilities of the loaded firmware image(s).

The API provides functions to deal with attributes depending on whether they are configuration or status specific.

Component Definitions

The different types of components present on DAG cards and their associated definitions are described below:

Root	The root component is a special component that has no attributes. All other components are children of the root component.
GPP	The generic packet processor component captures the packet. It can be configured to capture, using the `snaplen` attribute which defines a fixed number of bytes to capture from each packet.
Pbm	The PCI Burst Manager component handles the transfer of captured packets to the receive memory stream and from the transmit stream back to the card for transmitting. This component can be used to check the size of the memory buffer allocated, and to count the number of transmit and receive streams present. On some DAG cards you can set the overlap attribute to enable inline forwarding of packets.
Stream	The stream component represents the receive stream or transmit stream. The number of stream components differs depending on the firmware image loaded. This component can be used to allocate memory to the receive or transmit streams.
Port	The port component is generally used to configure and read attributes specific to the line. The specific attributes differ widely between cards, although there is some commonality depending on the protocol that the card is designed for. For example, all Ethernet cards have similar attributes associated with their port component. However, a SONET card port component will not have many attributes in common with an Ethernet card port component.
Framer	Represents the framer component. A Framer encapsulates data within a SONET Frame for transmit
Deframer	Represents a deframer component. A deframer breaks down a SONET Frame when received and extracts the data.
E1T1	**Represents the E1T1 deframer/framer**
Demapper	Represents a demapper component. Demapper components are used to provide a higher level of functionality over the base framer.
Mapper	Represents a mapper component. Mapper components are used to provide a higher level of functionality over the base framer.
LED Controller	Represents the LED controller for the pod
MiniMac Statistics	Represents the statistics module for each port

Component Definitions (cont.)

Mux	Represents the mux component. This component can be used to merge or split the receive streams on the card.
Phy	Represents the physical layer on a card.
Optics	Represents the optics component on the card.
Terf	Represents the terf register on cards that have the appropriate firmware loaded
Hardware Monitor	The hardware monitor (temperature, fan, voltage etc.)
Sonic	Controls attributes of the SONET/SDH deframer.

Example Programme

The following program illustrates how the Configuration & Status API is used. It queries the `'active'` attributes of all ports on the card and displays the result. For the sake of clarity the error-handling code has been omitted from this example.

```
#include "dag_component.h"

#include <stdio.h>
#include <stdlib.h>

int
main(int argc, const char* argv[])
{
      dag_card_ref_t card_ref = NULL;
      dag_component_t root_component = NULL;
      uint32_t count;
      uint32_t i;

      /* Get a reference to the card. */
      card_ref = dag_config_init("/dev/dag0");

      /* Get a reference to the root component. */
      root_component = dag_config_get_root_component(card_ref);

      /* Find out how many ports the card has. */
      count = dag_component_get_subcomponent_count_of_type(root_component,
            kComponentPort);

      for (i = 0; i < count; i++)
      {
            dag_component_t port = NULL;
            attr_uuid_t active_uuid = 0;
            uint8_t val = 0;

            /* Get a reference to the port. */
            port = dag_component_get_subcomponent(root_component,
            kComponentPort, i);

            /* Get a reference to the active attribute of the port. */
            active_uuid = dag_component_get_config_attribute_uuid(port,
            kBooleanAttributeActive);

            /* Read and display the value of the attribute. */
            val = (uint8_t) dag_config_get_boolean_attribute(card_ref,
            active_uuid);
            printf("Port %u active = %u\n", i, val);
      }
      /* Dispose of the card. */
      dag_config_dispose(card_ref);

      return EXIT_SUCCESS;
}
```

Chapter 3: Using the API

Pre-requisites

To use the Configuration and Status API you require a certain level of development expertise. This Programming Guide assumes you are competent in programming in C and are familiar with the Linux operating system.

Header Files

When using the Configuration & Status API you must include the following header files:

- **`dag_config.h`** Contains routines that relate to the card as a whole e.g. getting an initial reference to the card, loading firmware, finding a component by name, as well as routines that retrieve and set values on attributes.
- **`dag_component.h`** Contains routines that operate on components, e.g. getting the root component, getting subcomponents, getting attributes of a component.
- **`dag_component_codes.h`** Contains the codes e.g. `kComponentStream` used to refer to components.
- **`dag_attribute_codes.h`** Contains the codes e.g. `kBooleanAttributeVarlen` used to refer to attributes and enumerated types for attributes that have a restricted range of valid values.

Alternatively you may use the files **`dag_config_api.h`**. This is provided simply for convenience as its only function is to include the four essential files listed above.

FreeBSD/Linux:

Header files are installed in `/usr/local/include` by default. However you can change this location when running the `'configure'` script.

Library files are installed in `/usr/local/lib` by default and can also .be changed when running the `'configure'` script.

Windows:

Header files are installed in `%Program Files%\Endace\dag-x.y.z\include` . Stub library files are installed in `%Program Files%\Endace\dag-x.y.z \lib\windows\VCproject\Release`

Runtime library files are installed in `%System%`.

Note: The phrases in `%%` are standard system locations and may vary from machine to machine.

Chapter 4
DAG Card Components & Attributes

DAG 3.7D

Components

Enum	Description	Instances
kComponentGpp	Generic packet processor	2
kComponentPbm	PCI burst manager	1
kComponentStream	Rx/tx streams on the card	3
kComponentPort	Ports on the card	2
kcomponentSC256	Optional coprocessor on the card	1

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the record is padded up to the number of bytes specified by the snap length attribute.	config
kBooleanAttributeAlign64	Turns 64-bit alignment ON/OFF. If on the ERF records captured will be 64 bit aligned	config
kUint32AttributeSnaplength	The number of bytes to capture per packet.	config

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBusSpeed	The PCI bus speed. See pci_bus_speed_t.	status
kUint32AttributeTxStreamCount	The number of transmit streams	status
kUint32AttributeRxStreamCount	The number of receive streams.	status
kUint32AttributeBufferSize	The size of the buffer allocated to the DAG card.	status
kBooleanAttributeOverlap	Shares the memory hole between the receive and transmit streams to support inline forwarding.	config

DAG 3.7D (cont.)

kComponentStream

Enum	Description	Access
kUint32AttributeMem	The memory allocated to a receive or transmit stream. Can be used to allocate different amounts of memory from the buffer to a stream. The size of the buffer can be read using the attribute `kUint32AttributeBufferSize`. The value is returned in mebibytes.	config
kUint32AttributeMemBytes	Same as above except the unit of measurement is bytes.	config

kComponentPort

Enum	Description	Access
kBooleanAttributeEquipmentLoopback	Enables/disables EQL. Useful for testing. Normally be disabled.	config
kbooleanAttributeLinkDiscard	When unset, packets/ cells with checksum errors are passed through as if having no error. If set, the errored packets are dropped.	config
kUint32AttributeFramingMode	Indicates the type of framing to be used. These modes are listed in the enumeration framing_mode_t.	config
kUint32AttributePayloadMapping	Determines the type of payload mapping see section payload_mapping_t	config
kBooleanAttributeRXMonitorMode	Enable or disable RX Monitoring. This is used to enable the receive LIU monitor mode pre-amplifier. Enabling the pre-amplifier adds about 20 dB of linear amplification for use in monitor applications where the signal has been reduced 20 dB using resistive attenuator circuits.	config

kBooleanAttributeDescramble	Enable or disable SONET frame scrambling	config
kBooleanAttributeReceiveLock Error	Indicates a failure in clock recovery from the received signal.	status
kBooleanAttributeLossOfSignal	Indicates if link is experiencing LOS	status
kBooleanAttributeLossOfFrame	Indicates if link is experiencing LOF.	status
kBooleanAttributeRemoteDefect Indication	Indicates if the Line Remote Defect Indication Signal is set.	status
kBooleanAttributeAlarmIndication Signal	Indicates when the receive frame processor has in Alarm Indication Signal	status
kBooleanAttributeOutOfFrame	Indicates if link is experiencing OOF.	status

kComponentSC256

Enum	**Description**	**Access**
kStructAttributeSC25672BitData	Use this attribute to read/write data to the TCAM	config
kUint32AttributeSC256Data Address	Use this attribute to set the address of the data space on the CAM to read or write.	config
kStructAttributeSC25672BitSearch	Use this attribute to perform 72-bit searches	config
kNullAttributeSC256Init	Initialize the SC256 Coprocessor	config
kStructAttributeSC25672BitMask	Use this attribute to read/write mask values to the TCAM	config
kUint32AttributeSC256Mask Address	Use this attribute to set the address of the mask space on the CAM to read or write.	config
kUint32AttributeSC256Search Length	Use this attribute to set the search length. /sa SC256SearchLength.	config

DAG 3.7GE

Components

Enum	Description	Instances
kComponentGpp	Generic packet processor.	2
kComponentPbm	PCI burst manager	1
kComponentStream	Rx/tx streams on the card	3
kComponentPort	Ports on the card	2
kComponentMux	Multiplexer allows you to merge or split the streams from the ports	1

kComponentPort

Enum	Description	Access
kBooleanAttributeNic	Enable or disable Ethernet auto-negotiation mode. By default this is disabled; the disabled mode is intended for use with optical fibre splitters, and in the disabled mode Ethernet auto-negotiation is not performed.	config
kBooleanAttributeLink	Indicates whether the link is OK. In general if there is synchronization then the link will be OK.	status
kBooleanAttributeAutoNegotiationComplete	When nic is enabled this indicates if Ethernet auto-negotiation has completed.	status
kBooleanAttributeRemoteFault	Indicates a fault at the remote end of the link.	status
kBooleanAttributeJabber	In an Ethernet network, jabber is traffic from a device that is always sending -bringing the network effectively to a halt. This attribute indicates whether jabber is being detected.	status
kBooleanAttributeMaster	Indicates if the card is resolved to master or slave mode.	status

DAG 3.7GE (cont.)

kBooleanAttributeFullDuplex	Indicates if the link is full duplex.	status
kUint32AttributeForceLineRate	Force the DAG card to operate at the given line rate. See line_rate_t.	status
kUint32AttributeLineRate	The line rate.	status
kUint32AttributeErrorCounter	Number of bad packets received	status

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the record is padded up to the number of bytes specified by the snap length attribute.	config
kBooleanAttributeAlign64	Turns 64-bit alignment ON/OFF. If on the ERF records captured will be 64 bit aligned	config
kUint32AttributeSnaplength	The number of bytes to capture per packet.	config

kComponentMux

Enum	Description	Access
kBooleanAttributeSplit	Splits data from the ports to the two receive streams.	config
kBooleanAttributeMerge	Merges data from the two ports into one receive stream.	config
kBooleanAttributeSwap	Used when transmitting. Swaps the interface on which the packet transmits. By default the packets are sent to the port marked in the ERF header. Swaps the ports so that packets intended for port 0 go to port 1 and vice versa.	config

DAG 3.7GE (cont.)

kComponentStream

Enum	Description	Access
kUint32AttributeMem	The memory allocated to a receive or transmit stream. Can be used to allocate different amounts of memory from the buffer to a stream. The size of the buffer can be read using the attribute `kUint32AttributeBufferSize`. The value is returned in mebibytes.	config
kUint32AttributeMemBytes	Same as above except the unit of measurement is bytes.	config

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBusSpeed	The PCI bus speed. See pci_bus_speed_t.	status
kUint32AttributeTxStreamCount	The number of transmit streams	status
kUint32AttributeRxStreamCount	The number of receive streams.	status
kUint32AttributeBufferSize	The size of the buffer allocated to the DAG card.	status
kBooleanAttributeOverlap	Shares the memory hole between the receive and transmit streams to support inline forwarding.	config

DAG 3.7T

Components

Enum	Description	Instances
kComponentDemapper	HDLC or ATM demapper	1
kComponentPort	Ports on the card.	16
kComponentStream	Receive/transmit streams on the card.	1 or 2 (depends) on image loaded).
kComponentPbm	PCI burst manager.	1
kComponentLEDController	Controller for LEDs on the DAG 3.7T pod.	1
kComponentFramer	Sonic E1/T1 framer	1

kComponentDemapper

Enum	Description	Access
kUint32AttributeDropCount	Number of packets dropped	status
kUint32AttributeLossOfCell DelineationCount	Indicates the number of LCD instances.	config
kBooleanAttributeTimeStampEnd	Indicates when the timestamp is to be added to the record.	config

kComponentPort

Enum	Description	Access
kUint32AttributeMode	For more information see Mode Table	config
kUint32AttributeLineType	Set the line type. For valid values see `line_type_t`.	
kUint32AttributeTermination	The termination strength. For valid values see termination_t	config
kBooleanAttributeFacility Loopback	Enables/disables FCL. Useful for testing.	config
kBooleanAttributeEquipment Loopback	Enables/disables EQL. Useful for testing. Normally this should be disabled.	config
kUint32AttributeZeroCode Suppress	For valid values see zero_code_suppress_t	config

DAG 3.7T (cont.)

kBooleanAttributeReceiveLossOfSignal	This is set to indicate that the receive input signal is lost	status
kBooleanAttributeLineCodeViolation	This is set to indicate that the receiver channel is currently detecting a Line Code Violation. or an excessive number of zeros in the B8ZS or HDB3 modes.	status
kBooleanAttributeFIFOLimitStatus	This is set to indicate that the jitter attenuator read/write FIFO pointers are within +/- 3 bits.	status
kBooleanAttributeDriverMonitorOutput	This is set to indicate when a transmit driver failure is detected	status
kBooleanAttributeAlarmSignal	Indicates if link is experiencing AIS	status
kUint32AttributeCableLoss	Represents the cable attenuation indication within ±1dB.	status
kBooleanAttributeE1T1Rx0	If this is set it means that nothing is being processed by the SONIC E1/T1 framer. Causes of this are related to faulty hardware.	status
kBooleanAttributeE1T1Rx1	If this is set it means that nothing is being processed by the SONIC E1/T1 framer. Causes of this are related to faulty hardware.	status
kBooleanAttributeE1T1Tx0	If this is set it means that nothing is being processed by the SONIC E1/T1 framer. Causes of this are related to faulty hardware.	status
kBooleanAttributeE1T1Tx1	When set it means nothing is being processed by the SONIC E1/T1 framer. Causes of this are related to faulty hardware.	status
kBooleanAttributeE1T1FramerError	Indicates if there was a framer error	status

DAG 3.7T (cont.)

kBooleanAttributeE1T1AISError	Indicates if there was an Alarm Indication Signal Error	status
kBooleanAttributeE1T1CRCError	Indicates if there was a CRC error	status
kBooleanAttributeE1T1Link	Indicates if the link is up	status
kBooleanAttributeRxPkts	Enables or disable receive packets	config
kBooleanAttributeTxPkts	Enables or disable transmit packets	config

kComponentStream

Enum	**Description**	**Access**
kUint32AttributeMem	Represents the memory allocated to a receive or transmit stream. Can be used to allocate different amounts of memory from the buffer to a stream. Size of the buffer can be read using the attribute kUint32 AttributeBufferSize. The value is returned in mebibytes.	config
kUint32AttributeMemBytes	Same as above except the unit of measurement is bytes	config

kComponentPbm

Enum	**Description**	**Access**
kUint32AttributePCIBusSpeed	A number representing the PCI bus speed. See `pci_bus_speed_t`	status
kUint32AttributeBufferSize	Size of the buffer allocated to the DAG card.	status
kUint32AttributeTxStreamCount	Count of the number of transmit streams.	status
kUint32AttributeRxStreamCount	Count of the number of receive streams.	status
kBooleanAttributeOverlap	Shares the memory hole between receive and transmit streams.	config

DAG 3.7T (cont.)

kComponentLEDController

Enum	Description	Access
kUint32AttributePeriod	Use to set the period/frequency that the LED will blink at. The period is specified in 100ths of a second. Note you cannot assign a different period per LED. This is due to hardware limitations of the PCA9552 chip that controls the LEDs. What this implies is one frequency is assigned for all the LED so all LEDs, depending on their status, will blink at that frequency.	config
kUint32AttributeLEDStatus	Use to set the status of an LED. There are 32 of these attributes in this component that represent the 32 LEDs on the pod. Each LED can be asssigned a status. For valid values see led_status_t .	config

kComponentFramer

Enum	Description	Access
kBooleanAttributeClear	Use to clear the framer. This will clear the statistics counters in the framer.	config
kBooleanAttributeReset	Use to reset the framer.	config

DAG 3.7T (cont.)

Mode Table

Value	E1/T1 Mode and Receive Sensitivity	Transmit Line Build Out	Cable	Coding
8	T1 Short Haul/15dB	0-133 ft./0.6dB	100Ω/TP	B8ZS
9	T1 Short Haul/15dB	133-266 ft./1.2dB	100Ω/TP	B8ZS
10	T1 Short Haul/15dB	266-399 ft./1.8dB	100Ω/TP	B8ZS
11	T1 Short Haul/15dB	399-533 ft./2.4dB	100Ω/TP	B8ZS
12	T1 Short Haul/15dB	533-655 ft./3.0dB	100Ω/TP	B8ZS
13	T1 Short Haul/15dB	Arbitrary Pulse	100Ω/TP	B8ZS
14	T1 Gain Mode/29dB	0-133 ft./ 0.6dB	100Ω/TP	B8ZS
15	T1 Gain Mode/29dB	133-266 ft./ 1.2dB	100Ω/TP	B8ZS
16	T1 Gain Mode/29dB	266-399 ft./ 1.8dB	100Ω/TP	B8ZS
17	T1 Gain Mode/29dB	399-533 ft./ 2.4dB	100Ω/TP	B8ZS
18	T1 Gain Mode/29dB	533-655 ft./ 3.0dB	100Ω/TP	B8ZS
19	T1 Gain Mode/29dB	Arbitrary Pulse	100Ω/TP	B8ZS
28	E1 Short Haul	ITU G.703/Arbitrary	75Ω Coax	HDB3
29	E1 Short Haul	ITU G.703/Arbitrary	120Ω/TP	HDB3
30	E1 Short Haul	ITU G.703/Arbitrary	75Ω Coax	HDB3
31	E1 Short Haul	ITU G.703/Arbitrary	120Ω/TP	HDB3

DAG 3.8S

Components

Enum	Description	Instances
kComponentPort	The ports on the card	2
kComponentPbm	The PCI burst manager.	1
kComponentStream	The receive transmit streams on the card	2
kComponentGPP	The generic packet processor	2

kComponentPort

Enum	Description	Access
kBooleanAttributeScramble	Enables/disables SONET frame scrambling.	config
kUint32AttributeLineRate	Changes the card line rate The card can operate at OC-3 or OC-12.	config
kBooleanAttributeEquipmentLoopback	Enables/disables EQL. Useful for testing. Normally this should be disabled.	config
kBooleanAttributeFacilityLoopback	Enables/disables FCL. Useful for testing.	config
kUint32AttributeNetworkMode	Sets the port to POS or ATM mode. See network_mode_t	config
kBooleanAttributePayloadScramble	Enables/disables payload scrambling	config
kUint32AttributeCrcSelect	Select the CRC to use. For valid CRC types crc_t.	config
kBooleanAttributeLossOfSignal	Indicates if link is experiencing LOS.	status
kBooleanAttributeLossOfFrame	Indicates if link is experiencing LOF.	status
kBooleanAttributeOutOfFrame	Indicates if link is experiencing OOF.	status
kBooleanAttributeB1Error	Indicates if link is experiencing B1 errors.	status
kBooleanAttributeB2Error	Indicates if link is experiencing B2 errors.	status

DAG 3.8S (cont.)

kBooleanAttributeB3Error	Indicates if link is experiencing B3 errors.	status
kBooleanAttributeREIError	Indicates if link is experiencing a remote error.	status
kBooleanAttributeRDIError	Indicates if link is experiencing a remote data error.	status
kBooleanAttributeAlarmSignal	Indicates if link is experiencing AIS	status
kUint32AttributeC2PathLabel	Read the path label of the C2 byte in the SONET frame.	status
kBooleanAttributeReset	Hold the framer in reset/release the framer from reset	config
kBooleanLossOfPointer	Indicates if the link is experiencing Loss Of Pointer	status

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the record is padded up to the number of bytes specified by the snaplen attribute.	config
kUint32AttributeSnaplength	The number of bytes to capture per packet.	config
kBooleanAttributeAlign64	Turns 64-bit alignment ON/OFF. If on the ERF records captured will be 64-bit aligned.	config,

DAG 3.8S (cont.)

kComponentStream

Enum	Description	Access
kUint32AttributeMem	Represents the memory allocated to a receive or transmit stream. Can be used to allocate different amounts of memory from the buffer to a stream. Size of the buffer can be read using the attribute `kUint32AttributeBufferSize`. The value is returned in mebibytes.	config
kUint32AttributeMemBytes	Same as above except the unit of measurement is bytes	config

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBusSpeed	A number representing the PCI bus speed. See `pci_bus_speed_t`	status
kUint32AttributeTxStreamCount	Count of the number of transmit streams.	status
kUint32AttributeRxStreamCount	Count of the number of receive streams.	status
kUint32AttributeBufferSize	Size of the buffer allocated to the DAG card.	status
kBooleanAttributeOverlap	Shares the memory hole between receive and transmit streams.	config

DAG 4.3S

Components

Enum	Description	Instances
kComponentGpp	Generic packet processor.	1
kComponentPort	Ports on card.	2
kComponentPbm	PCI burst manager	1
kComponentTerf	Transmit ERF Firmware Component	1

DAG 4.3S (cont.)

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the record is padded up to the number of bytes specified by the snaplen attribute.	config
kBooleanAttributeAlign64	Turns 64-bit alignment ON/OFF. If on the ERF records captured will be 64 bit aligned	config
kUint32AttributeSnaplength	Number of bytes to capture per packet.	config

kComponentTerf

Enum	Description	Access
kUint32AttributeTerf StripCrc	Used to set the option on the terf component to strip the crc. See terf_strip_t	config

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBusSpeed	Number representing the PCI bus speed. See pci_bus_speed_t	status
kUint32AttributeTxStreamCount	The number of transmit streams.	status
kUint32AttributeRxStreamCount	The number of receive streams.	status
kUint32AttributeBufferSize	Size of the buffer allocated to the DAG card.	status
kBooleanAttributeOverlap	Shares the memory hole between receive and transmit streams.	config

kComponentPort

Enum	Description	Access
kBooleanAttributeEquipmentLoop back	Used to set or disable equipment loopback. Used mainly for testing.	config
kBooleanAttributeActive	Represents the active value of a port. Allows you to set or get a ports active value.	config

DAG 4.3S (cont.)

kUint64AttributeRxBytes	Number of bytes successfully received.	status
kUint64AttributeTxBytes	Count of bytes successfully transmitted.	status
kbooleanAttributeCounterLatch	This attribute must be set before reading statistics. It latches the statistic and counters to allow the values to be read.	config
kBooleanAttributeCrcStrip	Enable or disable CRC Stripping.	config
kBooleanAttributeFacilityLoop back	Used to set or disable facility loopback. Used mainly for testing.	config
kBooleanAttributeLinkDiscard	When unset, packets/ cells with checksum errors are passed through as if having no error. If set, the errored packets are dropped.	config
kBooleanAttributePayload Scramble	Enable or disable payload scrambling	config
kBooleanAttributeReset	Holds/releases the framer in reset.	config
kBooleanAttributeScramble	Enable or disable SONET frame scrambling.	config
kUint32AttributeCrcSelect	Select the CRC to use. For valid CRC types crc_t.	config
kUint32AttributeMasterSlave	Set the SONET clock master/slave status. For valid values see master_slave_t.	config
kUint32AttributeMaxPktLen	Maximum expected packet length.	config
kUint32AttributeMinPktLen	Mininum expected packet length.	config
kUint32AttributeNetworkMode	Configures PoS or ATM mode.	config
kUint32AttributeSteer	Set the ERF record steering mode. See steer_t.	config
kBooleanAttributeDataOutOfLock	Indicates a Data Out Of Lock error condition.	status

DAG 4.3S (cont.)

kBooleanAttributeLineAlarm IndicationSignal	Indicates if the Line Alarm Indication Signal is set.	status
kBooleanAttributeLineRemote DefectIndicationSignal	Indicates if the Line Remote Defect Indication Signal is set.	status
kBooleanAttributeLossOfCell Delineation	Iindicates if the demapper is in LCD (loss of cell delineation) state.	status
kBooleanAttributeLossOfPointer	Indicates that the card is experiencing Loss Of Pointer (cannot lock to the SONET/SDH framer pointers).	status
kBooleanAttributeLossOfSignal	Indicates that the card is experiencing Loss Of Signal.	status
kBooleanAttributeOutOfFrame	Indicates that the card is in an Out Of Frame condition.	status
kBooleanAttributeReferenceOutOf Lock	Indicates a Reference Out of Lock error condition	status
kUint32AttributeAborts	Number of PoS frames aborted since last reading	status
kUint32AttributeHECCount	The number of cells with HEC errors since this attribute was last read.	status
kUint32AttributeMaxPktLenError	Number of packets rejected because they were too large since last reading.	status
kUint32AttributePathBIPError	Number of Path Bit Interleaved Parity Errors seen	status

DAG 4.3S (cont.)

kUint32AttributeC2PathLabel	Used to read the SONET/SDH C2 path label byte (Path Signal Label). Typical settings are 0x16 for PoS and 0xCF for Cisco HDLC. On cards that support virtual containers the path label will be read from the virtual container specified by the `kUint32 AttributeVCIndex` attribute.	status
kUint32AttributeMinPktLenError	Number of packets rejected because they were too small since last reading.	status
kUint32AttributePathREIError	Number of Path Remote Error Indications seen.	status
kUint32AttributeRxFDrop	Number of frames dropped since last reading.	status
kUint32AttributeRxFrames	Number of valid frames received since last reading.	status
kUint32AttributeTxFDrop	Number of frames dropped in transmission since last reading.	status
kUint32AttributeTxFrames	Number of frames transmitted since last reading.	status

DAG 4.3GE

Components

Attribute	Description	Instances
kComponentGpp	Generic packet processor.	1
kComponentPbm	PCI burst manager	1
kComponentPort	Ports on card.	2
kComponentTerf	Transmit ERF firmware component	1

kComponentGPP

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the length is padded up to the number specified by the Snaplen attribute.	config
kBooleanAttributeAlign64	Turns 64bit alignment on or off. If ON the ERF records captured will be 64 bit aligned.	config
kUint32AttributeSnaplength	Number of bytes to capture per packet	config

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBus Speed	The PCI bus speed.	status
kUint32AttributeTXStreamCount	The number of transmit streams	status
kUnit32AttributeRxStreamCount	The number of receive streams	status
kUintt32AttributeBufferSize	Size of the buffer allocated to the DAG card	status
kBooleanAttributeOverlap	Shares the memory hole between receive and transmit streams	config

DAG 4.3GE (cont.)

kComponentTerf

Enum	Description	Access
kUint32AttributeTerfStripCrc	Used to set the CRCstripping functionality see section terf_strip_t	config

kComponentPort

Enum	Description	Access
kBooleanAttributeActive	Represents the active value of a port. Allows you to set or get a port's active value.	config
kBooleanAttributeEquipment Loopback	Enables/disables EQL. Used mainly for testing.	config
kBooleanAttributeNic	Enables/disables Ethernet auto-negotiation . By default this is disabled. Disabled mode is intended for use with optical splitters. In this mode auto-negotiation is not performed.	config
kBooleanAttributeRxPkts	Enable or disable packet reception	config
kBooleanAttributeTxPkts	Enable or disable packet transmission	config
kBooleanAttributeAutoNegotiation Complete	When nic is enabled this indicates if Ethernet auto-negotiation has completed.	status
kBooleanAttributeLink	Indicates link is OK. In general if there is sync then the link will be OK.	status
kBooleanAttributeRemoteFault	Indicates a fault at the remote end of the link.	status
kBooleanAttributeSync	Indicates the synchronisation status	status
kBooleanAttributeByteCount	The maximum number of bytes allowed per packet.	status

DAG 4.3GE (cont.)

kBooleanAttributeCounterLatch	Used to latch the counter attributes on the card. Must be set to 1 before reading values from the following: kUint64AttributeBad Symbol, kUint64AttributeCrcFail kUint64AttributeInterna lMACError, kUint64AttributeRx Bytes, kUint64AttributeRx Frames, kUint64AttributeTransm itSystemError kUint64AttributeTx Bytes, kUint64AttributeTx Frames.	Config
kUint32AttributeDropCount	Count of the packets dropped on a port.	status
kUint64AttributeBadSymbol	Count of the number of times a valid length frame was received at the port and during which time there was at least one of an event that causes the PHY to indicate a "Data Reception Error" or invalid "Data Symbol Error"	status
kUint64AttributeCrcFail	Count of frames received that do not pass the Frame Checksum [FCS] check.	status
kUint64AttributeInternalMACErro r	Count of the frames that could not be sent correctly due to various errors.	status
kUint64AttributeRxBytes	Number of bytes successfully received	status

kUint64AttributeRxFrames	Count of valid frames received	status
kUint64AttributeTransmitSystemError	Count of frames that could not be sent correctly due to various errors. Frames already counted by `kUint64AttributeInternalMACError` are not included in this count.	status
kUint64AttributeTxBytes	Count of bytes successfully transmitted.	status
kUint64AttributeTxFrames	Count of the frames successfully transmitted	status

DAG 4.5G

Components

Attribute	Description	Instances
kComponentGpp	Generic packet processor.	1
kComponentPbm	PCI burst manager	1
kComponentPort	Ports on card.	2
kComponentMiniMacStatistics	Statistics Module for each port	2
kComponentHardwareMonitor	Monitor temperature and voltage	1
kComponentTerf	Transmit ERF firmware component	

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the length is padded up to the number specified by the Snaplen attribute.	config
kBooleanAttributeAlign64	Turns 64bit alignment on or off. If ON the ERF records captured will be 64 bit aligned.	config
kUint32AttributeSnaplength	Number of bytes to capture per packet	config

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBusSpeed	Number representing the PCI bus speed. See pci_bus_speed_t	status
kUint32AttributeTxStreamCount	The number of transmit streams.	status
kUint32AttributeRxStreamCount	The number of receive streams.	status
kUint32AttributeBufferSize	Size of the buffer allocated to the DAG card.	status
kBooleanAttributeOverlap	Shares the memory hole between receive and transmit streams.	config

DAG 4.5G (cont.)

kComponentPort

Enum	Description	Access
kBooleanAttributeActive	Indicates whether a physical interface is active or inactive.	config
kBooleanAttributeEquipment Loopback	Enable or disable the equipment loopback (EQL) capability. Used for testing, should normally be disabled.	config
kBooleanAttributeLaser	Configures the transmit laser to be on or off.	config
kBooleanAttributeRocketIOPower	Enable or disable Rocket IO	config
kBooleanAttributeSfpPwr	Enable or disable optics transmit power.	config
KbooleanAttributeNic	Enable or disable Ethernet auto-negotiation mode. By default this is disabled; the disabled mode is intended for use with optical fibre splitters, and in the disabled mode Ethernet auto-negotiation is not performed.	config
kBooleanAttributeLink	Indicates whether the link is OK. In general if there is synchronization then the link will be OK.	status
kBooleanAttributeLossOfFrame	Indicates that the card is experiencing Loss Of Frame.	status
kBooleanAttributePeerLink	Indicates that the peer link is up	status
kBooleanAttributeRemoteFault	Indicates a fault at the remote end of the link.	status
kBooleanAttributeSFPDetect	Indicates if the SFP is present	status
kUint32AttributeDropCount		status

DAG 4.5G (cont.)

kComponentMiniMacStatistics

Enum	Description	Access
kBooleanAttributeCounterLatch	Set this attribute before reading statistics. It latches the statistics counters so they can be read in a consistent state.	config
kBooleanAttributeRefreshCache	It is necessary to cache the statistics values before reading them as they are cleared once any of the values are read from the component.	config
kBooleanAttributeCrcErrorEverHi	Indicates if a CRC error was ever seen since last read	status
kBooleanAttributeCrcErrorEverLo	Indicates if a CRC error was ever set to 0 since last read.	status
kBooleanAttributeLinkCurrent	Indicates if there is a current Link error.	status
kBooleanAttributeLinkEverHi	Indicates if a Link error was ever seen since last read	status
kBooleanAttributeLinkEverLo	Indicates if a Link error was ever set to 0 since last read.	status
kBooleanAttributeLossOfFraming Current	Indicates if there is a current Loss of Framing error.	status
kBooleanAttributeLossOFFraming EverHi	Indicates if a Loss of Framing error was ever seen since last read.	status
kBooleanAttributeLossOfFraming EverLo	Indicates if a Loss of Framing error was ever set to 0 since last read.	status
kBooleanAttributeLossOfSignal Current	Indicates if a there is a current LOS.	status
kBooleanAttributeLossOfSignalEver Hi	Indicates if a Loss Of Signal error was ever seen since last read	status
kBooleanAttributeLossOfSignalEver Lo	Indicates if a Loss of Signal error was ever set to 0 since last read.	status

DAG 4.5G (cont.)

kBooleanAttributeMiniMacLostSync	Indicates if the Mini Mac has lost Synchronization	status
kBooleanAttributePeerLinkCurrent	Indicates if there is a current Peer Link error	status
kBooleanAttributePeerLinkEverHi	Indicates if there ever was a Peer Link error since last read	status
kBooleanAttributePeerLinkEverLo	Indicates if there ever was a Peer Link error set to 0 since last read	status
kBooleanAttributeRemoteErrorCurrent	Indicates if there is a current Remote Error	status
kBooleanAttributeRemoteErrorEverHi	Indicates if a there was a Remote Error since last read	status
kBooleanAttributeRemoteErrorEverLo	Indicates if there ever was a Remote error set to 0 since last read	status
kBooleanAttributeSFPTxFault Current	Indicates if there is a current SFP Tx Fault	status
kBooleanAttributeSFPTxFaultEverHi	Indicates if there ever was a SFP Tx Fault since last read	status
kBooleanAttributeSFPTxFaultEverLo	Indicates if there ever was a SFP Tx Fault set to 0 since last read	status
kUint32AttributeBadSymbols	Indicates the number of bad symbols since last read	status
kUint32AttributeConfigSequences	Indicates the number of configuration Sequences since last read	status
kUint32AttributeCrcErrors	Indicates the number of CRC Errors since last read	status
kUint32AttributeRemoteErrors	Indicates the number of remote errors since last read	status
kUint32AttributeRxFrames	Indicates the number of RX Frames since last read	status
kUint32AttributeTxFrames	Indicates the number of TX Frames since last read	status
kuint64AttributeRxBytes		status
kuint64AttributeTxBytes		status

DAG 4.5G (cont.)

kComponentHardwareMonitor

Enum	Description	Access
kUint32AttributeTemperature	Indicates the current temperature value	status
kUint32AttributeVoltage	Indicates the current voltage.	status

kComponentTerf

Enum	Description	Access
kUint32AttributeTerfStripCrc	Represents the terf register on cards that have the appropriate firmware loaded	config

DAG 6.2SE

Components

Attribute	Description	Instances
kComponentGpp	Generic packet processor.	1
kComponentPbm	PCI burst manager	1
kComponentPort	Ports on card.	2
kComponentTerf	Represents the terf register on cards that have the appropriate firmware loaded	1

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the record is padded up to the number of bytes specified by the snaplen attribute.	config
kBooleanAttributeAlign64	Turns 64-bit alignment ON/OFF. If on the ERF records captured will be 64 bit aligned	config
kUint32AttributeSnaplength	Number of bytes to capture per packet.	config

DAG 6.2SE (cont.)

kComponentPbm

Enum	Description	Access
kUint32AttributePCIBusSpeed	Number representing the PCI bus speed. See_ pci_bus_speed_t	status
kUint32AttributeTxStreamCount	The number of transmit streams.	status
kUint32AttributeRxStreamCount	The number of receive streams.	status
kUint32AttributeBufferSize	Size of the buffer allocated to the DAG card.	status
kBooleanAttributeOverlap	Shares the memory hole between receive and transmit streams.	config

kComponentPort

Enum	Description	Access
kBooleanAttributeCounterLatch	Set this attribute before reading statistics. It latches the statistics counters so they can be read in a consistent state.	config
kBooleanAttributeCrcStrip	Enable or disable CRC stripping from received packets.	config
kBooleanAttributeEquipmentLoop back	Enable or disable the equipment loopback (EQL) capability. Used for testing, should normally be disabled.	config
kBooleanAttributeFacilityLoopback	Enable or disable the facility loopback (FCL) capability. Used for testing, should normally be disabled.	config
kBooleanAttributeLineSide EquipmentLoopback	Enables/disables Line Side EQL. Useful for testing. Normally this should be disabled.	config
kBooleanAttributeLineSideFacility Loopback	Enables/disables Line Side FCL. Useful for testing. Normally this should be disabled.	config

DAG 6.2SE (cont.)

kBooleanAttributePayloadScramble	Enable or disable payload scrambling for a concatenated POS demapper.	config
kBooleanAttributePMaxCheck	Enable or disable discard of packets larger than a predefined maximum size.	config
kBooleanAttributePMinCheck	Enable or disable discard of packets larger than a predefined minimum size.	config
kUint32AttributeMaxPktLen	Maximum packet size for the kBooleanAttributePMaxCheck	config
kUint32AttributeMinPktLen	Minimum packet size for the kBooleanAttributePMinCheck	config
kUint32AttributeCrcSelect	Select the CRC to use.	config
kUint32AttributeEthernetMode	Used to set the port to LAN or WAN mode.	config
kUint32AttributeNetworkMode	Used to set the port to PoS or ATM mode.	config
kUint32AttributeSteer	Set the ERF record steering mode. See `steer_t`.	config
kBooleanAttributeHighBitErrorRateDetected	High bit error rate detected, check optical level (Eth Only)	status
kBooleanAttributeLocalFault	Signal from peer is not being received correctly. (Eth Only)	status
kBooleanAttributeLossOfClock	The framer is not receiving a valid clock from the optics.	status
kBooleanAttributeLossOfFrame	Indicates that the card is experiencing Loss Of Frame.	status
kBooleanAttributeLossOfSignal	Indicates that the card is experiencing Loss Of Signal.	status
kBooleanAttributeOutOfFrame	Indicates if link is experiencing LOF.(PoS and WAN only).	status

DAG 6.2SE (cont.)

kBooleanAttributeReceiveAlarm Indication	Indicates a receive failure. Either /or both kBooleanAttributeReceice LockError and kBooleanAttributeReceiv ePowerAlarm will be set also.	status
kBooleanAttributeReceiveLockError	Indicates a failure in clock recovery from the received signal.	status
kBooleanAttributeReceivePower Alarm	Indicates insufficient optical input power (<-30dBm).	status
kBooleanAttributeRemoteFault	(Eth Only)	status
kUint32AttributeB1ErrorCount	Bit Interleaved Parity 1. SONET / SDH Section Parity error count. (PoS only)	status
kUint32AttributeB2ErrorCount	Bit Interleaved Parity 2. SONET / SDH Line Parity error count. (PoS only)	status
kUint32AttributeB3ErrorCount	Bit Interleaved Parity 3. SONET / SDH Path Parity error count. (PoS only)	status
kUint32AttributeC2PathLabel	Reflects content of SONET/SDH C2 overhead octect, or Path Signal Label. Typical settings are as follows: 16 PoS CF Cisco HDLC	status
kUint32AttributeDropCount	The number of packets dropped on a physical interface. FIXME: is this receive, transmit or both?	status
kUint32AttributeRxParityError	Receive parity error count between the framer and receive FPGA. (PoS Only)	status
kUint64AttributeBadPackets	Number of errored packets received since last reading. (Eth Only)	status
kUint64AttributeFCSErrors	Number of PoS/Ethernet FCS (CRC32) errors since last reading.	status
kUint64AttributeFIFOOverrunCount	Framer receive FIFO errors since last reading.	status

DAG 6.2SE (cont.)

kUint64AttributeGoodPackets	Number of correct frames/packets received since last reading.	status
kUint64AttributeRxBytes	(Eth Only)	status
kUint64AttributeRxBytesBad	Number of errored bytes received on the RX Stream. (Eth Only)	status
kbooleanAttributeLossOfPointer	The framer cannot lock to the SONET/SDH framer pointers (PoS and WAN only)	status

kComponentTerf

Enum	Description	Access
kUint32AttributeTerfStripCrc	The number of bytes to strip from the end of the ERF record when transmitting. Used to prevent a trailing CRC (e.g. on an ERF that has been captured and is now being retransmitted) being sent as part of a packet.	config

DAG 7.1S

Components

Attribute	Description	Instances
kComponentOptics	Controls the optical receivers on the card..	4
kComponentPort	Ports on the card	4
kComponentPbm	PCI burst manager	1
kComponentSonic	Controls attributes of the SONET/SDH deframer.	4
kComponentDemapper	The channelized demapper or the concatenated demapper. The API detects which is present and loads the appropropriate one.	4 or 1 (depends on firmware loaded) Channelized is 1. Concatenated is 4
kComponentE1T1	Represents the E1T1 deframer/framer.	4
kComponentStream	Receive/transmit streams on the card	1 or 2, (depends on firmware loaded.)
kComponentPhy	Physical Component	1
kComponentGpp	Generic Packet Processor	1
kComponentMapper	The concatenated HDLC mapper. The API detects whether this component should be added.	1

kComponentOptics

Enum	Description	Access
kBooleanAttributeLaser	Represents the status of the optics transmit laser. Allows the laser to be turned on and off.	config
kBooleanAttributeSfpPwr	Enable or disable optics transmit power	config
kBooleanAttributeDetect	Indicates if the optics module is present.	status
kBooleanAttributeSignal	Indicates if the optics is detecting input signal	status

DAG 7.1S (cont.)

kComponentPort

Enum	Description	Access
kUint32AttributeMasterSlave	Set the SONET clock master/slave status. For valid values see master_slave_t.	config
kBooleanAttributeCore	Indicates if the core is on	status
kBooleanAttributeFIFOError	Indicates a FIFO Error event	status
kBooleanAttributeLock	Indicates if the core is locked onto the datastream.	status
kUint32AttributeLineRate	The rate at which the line is currently operating. See line_rate_t.	status

kComponentPbm

Enum	Description	Access
kBooleanAttributeOverlap	Shares the memory hole between the receive and transmit streams to support inline forwarding	config
kUint32AttributePCIBusSpeed	Number representing the PCI bus speed. See pci_bus_speed_t	status
kUint32AttributeTxStreamCount	A count of the number of transmit streams.	status
kUint32AttributeRxStreamCount	A count of the number of receive streams.	status
kUint32AttributeBufferSize	The size of the buffer allocated to the DAG card.	status

kComponentSonic

Enum	Description	Access
kBooleanAttributeCounterLatch	This attribute must be set before reading statistics. It latches the statistics and counters to allow values to be read.	config
kUint32AttributeConnectionNumber	The connection number of the current configuration	config

DAG 7.1S (cont.)

kUint32AttributePayLoadMapping	Determines the type of payload mapping see section payload_mapping_t	config
kUint32AttributeTributaryUnit	Defines the type of payload to extract. See section tributary_unit_t	config
kBooleanAttributeOutOfFrame	Indicates if link is experiencing OOF.	status
kUint32AttributeREIErrorCount	Number of remote error indications seen.	status
kUint32AttributeVCSize	Use this attribute to set or get the size of the Virtual Containers. For more information on valid values. See `vc_size_t`	config
kBooleanAttributeScramble	Use to enable or disable SONET frame scrambling.	config
kBooleanAttributeLossOfSignal	Indicates that the card is experiencing Loss Of Signal.	status
kBooleanAttributeLossOfFrame	Indicates that the card is experiencing Loss Of Frame.	status
kBooleanAttributeB1Error	SONET B1 Error indication.	status
kBooleanAttributeB2Error	SONET B2 Error indication.	status
kBooleanAttributeB3Error	SONET B2 Error indication.	status
kBooleanAttributeREIError	Indicates that the SONET Remote Error Indication is set.	status
kBooleanAttributeRDIError	Indicates that the SONET Remote Data Indication is set.	status
kUint32AttributeSSM	The received synchronization status message.	status
kUint32AttributePointerState	The pointer state of the various virtual containers..	status
kUint32AttributeLineRate	The rate at which the line is currently operating. See line_rate_t.	config

DAG 7.1S (cont.)

kUint32AttributeVCMaxIndex	The maximum number of active virtual containers in the SONET frame. This number depends on the hardware, loaded firmware and virtual container size.	status
kUint32AttributeVCIndex	Retrieve or specify the index of the virtual container to use. Any index written should be less than the result of reading `kUint32 AttributeVCMax Index.`	config
kUint32AttributeB1ErrorCount	The number of B1 errors since last read.	status
kUint32AttributeB2ErrorCount	The number of B2 errors since last read.	status
kUint32AttributeB3ErrorCount	The number of B3 errors since last read.	status
kUint32AttributeDataPointer	Which data byte to read for `kUint32 AttributeJ0Path Label` and `kUint32 AttributeJ1Path Label`	config
kUint32AttributeC2PathLabel	The C2 path label of the virtual container indicated by the `kUint32 AttributeVCIndex`	status
kUint32AttributeJ0PathLabel	The section trace value of the indicated virtual container and byte of the data pointer.	status
kUint32AttributeJ1PathLabel	The path trace value of the indicated virtual container and byte of the data pointer.	status

DAG 7.1S (cont.)

kComponentDemapper (channelized)

Enum	Description	Access
kUint32AttributeSonetType	Indicates whether the component is configured for channelized or concatenated See sonet_type_t	status
kStructAttributeAddConnection	Use to add a connection. For more information about the type to use with this attribute See connection_description_t	config
kUint32AttributeGetConnection Number	Use to get the connection number of the last connection added.	status
kUint32Attribute71sChannelized RevisionID	Use to retrieve the revision id of the ATM/HDLC demapper. The revision id can be used to determine what features the demapper supports.	status
kNullAttributeClearConnections	Use this attribute to clear the all connections on the card.	config

kComponentDemapper (concatenated)

Enum	Description	Access
kUint32AttributeSonetType	Indicates whether the component is configured for channelized or concatenated see section sonet_type_t	status
kBooleanAttributePayloadScramble	Enable/disable ATM cell or PoS frame scrambling.	config
kBooleanAttributeIdleCellMode	Set this to pass idle cells.	config
kUint32AttributeCrcSelect	Use to select CRC16, 32 or to turn off CRC checking, `crc_t`.	config
kUint32AttributeNetworkMode	Use this to switch between POS or ATM modes.	config
kBooleanAttributeLossOfCell Delineation	Indicates if the demapper is in LCD (loss of cell delineation) mode.	status

DAG 7.1S (cont.)

kUint32AttributeIdleCellCount	Count of the number of idle cells since this attribute was last read.	status
kUint32AttributeHECCount	Count of the cells with HEC errors.	status

kComponentE1T1

Enum	Description	Access
kUint32AttributeLineType	Set the line type. For valid values. See `line_type_t`	config
kUint32AttributeE1T1StreamNumber	Set the stream number to read the status data from.	config
kBooleanAttributeE1T1GenerateAlarmIndication	Turn on or off alarm indication.	config
kBooleanAttributeE1T1LinkSynchronized	Used to check if the stream number is synchronized to the framing information.	status
kBooleanAttributeE1T1LinkAIS	The stream was in AIS mode since last selected.	status
kBooleanAttributeE1T1LinkSynchronizedUp	The stream has synchronized to the framing information since last selected.	status
kBooleanAttributeE1T1LinkSynchronizedDown	The stream has lost framing lock since last selected.	status
kBooleanAttributeE1T1LinkCRCError	The stream has seen CRC error since last selected. Only valid if E1 with CRC is selected for that stream.	status
kBooleanAttributeE1T1LinkFramingError	The stream's framing experienced an error since last selected.	status

DAG 7.1S (cont.)

kComponentStream

Enum	Description	Access
kUint32AttributeMem	Represents the memory (in mebibytes) allocated to a receive or transmit stream. Writing to this attribute will allocate the specified amount of memory from the buffer to an individual stream. The size of the buffer can be read using the attribute `kUint32AttributeBufferSize`	config
kUint32AttributeMemBytes	Same as above except the unit of measure is bytes.	config

kComponentPhy

Enum	Description	Access
kBooleanAttributeEquipmentLoopback	Enables/disables EQL. Useful for testing. Normally this should be disabled.	config
kBooleanAttributeFacilityLoopback	Enables/disables FCL. Useful for testing. Normally this should be disabled.	config

kComponentGpp

Enum	Description	Access
kBooleanAttributeVarlen	Variable length capture. If disabled the length is padded up to the number specified by the Snaplen attribute.	config
kBooleanAttributeAlign64	Turns 64bit alignment on or off. If ON the ERF records captured will be 64 bit aligned.	config
kUint32AttributeSnaplength	Number of bytes to capture per packet	config

DAG 7.1S (cont.)

kComponentMapper

Enum	Description	Access
kBooleanAttributePayloadScramble	Enable/disable ATM cell or PoS frame scrambling.	config
kUint32AttributeCrcSelect	Use to select CRC16, 32 or to turn off CRC checking. See `crc_t`.	config
kUint32AttributeNetworkMode	Sets the port to POS or ATM mode.	config
kUint32AttributeSonetType	Indicates whether the component is configured for channelized or concatenated see section sonet_type_t	status

Chapter 5: Card Configuration Functions

Introduction

All of the functions defined in this chapter relate to functions which directly configure the DAG card. They are listed in alphabetical order.

Other functions relating to components, modifiers, accessors and firmware are contained in subsequent chapters.

Functions

dag_config_default

Description
Executes a card's default configuration routine.:

```
dag_err_t
dag_config_default
(
    dag_card_ref_t card
);
```

Note: `'card'` is a reference to a card

Return Value
Returns `'kDagErrInvalidCardRef'` if the card reference is invalid.

dag_config_dispose

Description
Cleans up when finished with a card reference.

```
void
dag_config_dispose
(
     dag_card_ref_t card
);
```

Note: `'card'` is a reference to a card.

Return Value
None

dag_config_get_attribute_code

Description

Retrieves the attribute code of a given attribute.

```
dag_attribute_code_t
dag_config_get_attribute_code
(
      attr_uuid_t uuid
);
```

Note: ‘`uuid`’ is an attribute identifier

Return Value

The attribute code

dag_config_get_attribute_config_status

Description

Determines whether the attribute is marked as configuration or status.

```
dag_attr_config_status_t
dag_config_get_attribute_config_status
(
      attr_uuid_t uuid
);
```

Note: ‘`uuid`’ is an attribute identifier

Return Value

See ‘`dag_attr_config_status_t`’ for more information..

dag_config_get_attribute_description

Description

Retrieves a human-readable description for an attribute.

```
const char*
dag_config_get_attribute_description
(
      attr_uuid_t uuid
);
```

Note: ‘`uuid`’ is an attribute identifier

Return Value

The description for the given attribute

dag_config_get_attribute_name

Description

Retrieves a human-readable name for an attribute.

```
const char*
dag_config_get_attribute_name
(
      attr_uuid_t uuid
);
```

Note: `'uuid'` is an attribute identifier

Return Value

The name of the given attribute

dag_config_get_attribute_to_string

Description

Retrieves the value of an attribute as a string.:

```
const char*
dag_config_get_attribute_to_string
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

Note: `'card'` is a reference to a card
`'uuid'` is an attribute identifier

Return Value

A string representing the value of the given attribute.

dag_config_get_attribute_valuetype

Description

Retrieves the type of an attribute's value.

```
const char*
dag_config_get_attribute_valuetype
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

Note: `'component'` is a reference to a component
`'uuid'` is an attribute identifier

Return Value

The type of the given attribute's value.

dag_config_get_card_type

Description

Returns the type of a card.

```
dag_card_t
dag_config_get_card_type
(
    dag_card_ref_t card
);
```

Note: `'card'` is a reference to a card.
The type code is shown later in this chapter in `'dag_card_t'`

Return Value

The type of card.

dag_config_get_component_count

Description

Retrieves the number of components in the card.

```
int
dag_config_get_component_count
(
      dag_card_ref_t card
);
```

Note: `'card'` is a reference to a card.

Return Value

The number of components.

dag_config_get_component_description

Description

Retrieves a human-readable description for a component.

```
const char*
dag_config_get_component_description
(
      dag_component_t component
);
```

Note: `'component'` is a reference to a component.

Return Value

The description for the given component.

dag_config_get_component_name

Description

Retrieves a human-readable name for a component.

```
const char*
dag_config_get_component_name
(
     dag_component_t component
);
```

Note: `'component'` is a reference to a component.

Return Value

The name of the given component

dag_config_get_root_component

Description

Retrieves the root component, from which all subcomponents descend, for a card.

```
dag_component_t
dag_config_get_root_component
(
dag_card_ref_t card
);
```

Note: `card'` is a reference to a card

Return Value

A reference to the root component.

dag_config_init

Description

Initialises the DAG card and returns a reference to the card for use with other functions in the API.

```
dag_card_ref_t
dag_config_init
(
const char* device_name
);
```

Note: 'device name' is the name of the card. On Linux this should look like `'/dev/dag0`' and on Windows like `'dag0'`

Once finished with the card use dag_config_dispose to deallocate memory used internally by the API

Return Value

A reference to a DAG card for use with other configuration functions. `'NULL'` is returned on failure

dag_config_reset

Description

Executes a card's reset configuration routine.

```
dag_err_t
dag_config_reset
(
dag_card_ref_t card
);
```

Note: `'card'` is a reference to a card

Return Value

Returns `'kDagErrInvalidCardRef'` if the card reference is invalid.

dag_config_set_attribute_from_string

Description

Sets the value for an attribute from a string.

```
const char*
dag_config_get_attribute_from_string
(
    dag_card_ref_t card,
    attr_uuid_t uuid
    const char* string
);
```

Note: `'card'` is a reference to a card
`'uuid'` is an attribute identifier
`'string'` is the value for the attribute in string form.

Return Value

Chapter 6: Component Functions

Introduction

All of the functions defined in this chapter relate to functions which configure or retrieve components on the DAG Card. They are liste in alphabetical order.

Other functions relating to functions that directly configure the DAG card, modifiers, accessors and firmware are contained in previous and subsequent chapters.

Functions

dag_component_get_attribute_count

Description
Retrieve the number of attributes in a component.

```
int
dag_component_get_attribute_count
(
      dag_component_t
);
```

Note: `component` is a reference to a component

Returned Value
The number of attributes in that component.

dag_component_get_config_attribute_count

Description
Retrieves the number of config attributes in a given component.

```
int
dag_component_get_config_attribute_count
(
      dag_component_t component
);
```

Note: `component` is a reference to a component

Return Value
The count of the number of config attribute

dag_component_get_config_attribute_uuid

Description

Retrieves an attribute from a DAG component.

```
attr_uuid_t
dag_component_get_config_attribute_uuid
(
      dag_component_t component,
      dag_attribute_code_t attribute_code
);
```

Note: `'component` is a reference to a component
`'attribute'` is the code of the attribute to retrieve

Return Value

An identifier for the attribute if found. If the requested attribute cannot be found `'kNullAttributeUuid'` is returned.

dag_component_get_indexed_attribute_uuid

Description

Retrieves the DAG component attribute 'i' at the given index. :

```
attr_uuid_t
dag_component_get_indexed_attribute_uuid
(
      dag_component_t component,
      int index
);
```

Note: `'component` is a reference to a component
`'index'` is the index of the attribute to return

Return Value

The attribute at the given index.

dag_component_get_indexed_config_ attribute_uuid

Description

Retrieves a configuration attribute from a component by index.

```
attr_uuid_t
dag_component_get_indexed_config_attribute_uuid
(

    dag_component_t component,

    int index

);
```

Note: `'component` is a reference to a component
`'index'` is the index of the attribute to return

Return Value

The configuration attribute at the given index.

dag_component_get_indexed_status_ attribute_uuid

Description

Retrieves the status attribute at a given index.

```
attr_uuid_t
dag_component_get_indexed_status_attribute_uuid
(
      dag_component_t component,
      int index
);
```

Note: `‘component` is a reference to a component
`‘index’` is the index of the attribute to return

Return Value

The status attribute at the given index.

dag_component_get_indexed_subcomponent

Description

Retrieves a subcomponent at a given index.

```
attr_uuid_t
dag_component_get_indexed_subcomponent
(
dag_component_t component,
int index
);
```

Note: `‘component` is a reference to a component
`‘index’` is the index of the attribute to return

Return Value

The component at the given index.

dag_component_get_named_subcomponent

Description

Retrieves the component using the internal name of the component.

```
dag_component_t
dag_component_get_named_subcomponent
(
dag_component_t component,
const char* name
);
```

Note: `‘component` is a reference to a component
`‘name’` is the name of the subcomponent to return

Return Value

The component or NULL if not found

dag_component_get_status_attribute_count

Description

Retrieves the number of status attributes in a component.

```
int
dag_component_get_status_attribute_count
(
      dag_component_t component
);
```

Note: `'component` is a reference to a component

Return Value

The number of status attributes.

dag_component_get_subcomponent

Description

Retrieves a specific subcomponent of a given component.

```
dag_component_t
dag_component_get_subcomponent
(
      dag_component_t component,
      dag_component_code_t component_code,
int index
);
```

Note: `'component'` is the parent component
`'component code'` see Chapter 4 of this Guide for a list of valid component codes
`'index'` is the index of the component to retrieve. Useful when there is more than one component of a particular type.

Return Value

The component requested. If not found then NULL is returned

dag_component_get_subcomponent_count

Description

Retrieves the number of subcomponents of a given component.

```
int
dag_component_get_subcomponent_count
(
      dag_component_t component
);
```

Note: `'component'` is the parent component

Return Value

A count of the number of components.

dag_component_get_subcomponent_count_of_type

Description

Retrieves the number of components with a given component code.

```
int
dag_component_get_subcomponent_count_of_type
(
     dag_component_t component,
     dag_component_code_t code
);
```

Note: `component` is a reference to a component
`code` is the code for the desired subcomponent to count

Return Value

The number of components

Chapter 7: Attribute Accessor Functions

Introduction

Description

All of the following accessor functions retrieve the value of an attribute. The only difference is the type of the value returned. The are listed in alphabetical order.

Accessor functions retrieve the value of the given attribute depending upon the type of the attribute.

Note: `'card'` is the reference to the card
`'uuid'` is the attribute identifier.
`'component'` is a reference to a component

Return Value

The Return Value is the value of the attribute

Functions

dag_config_get_boolean_attribute

Description

```
uint8_t
dag_config_get_boolean_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_get_char_attribute

```
char
dag_config_get_char_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_get_int32_attribute

```
int32_t
dag_config_get_int32_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_get_int64_attribute

```
int64_t
dag_config_get_int64_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_get_string_attribute

```
const
char* dag_config_get_string_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_get_uint32_attribute

```
uint32_t
dag_config_get_uint32_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_get_uint64_attribute

```
uint64_t
dag_config_get_uint64_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

Chapter 8: Attribute Modifier Functions

Introduction

Description

All of the following functions assign a value to an attribute. The only difference is the type of the value assigned. They are listed in alphabetical order.

Modifier functions assign a value to the given attribute depending upon the type of the attribute.

> **Note:** `'card'` is the reference to the card
> `'uuid'` is the attribute identifier.
> `'value'` is the value to assign to the attribute

Return Values

`'kDagErrInvalidCardRef'` is returned if the card reference is invalid. `'kDagErrNone'` is returned on success.

Functions

dag_config_set_boolean_attribute

```
dag_err_t
dag_config_set_boolean_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
uint8_t value
);
```

dag_config_set_char_attribute

```
dag_err_t
dag_config_set_char_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      char value
);
```

dag_config_set_int32_attribute

```
dag_err_t
dag_config_set_int32_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      int32_t value
);
```

dag_config_set_int64_attribute

```
dag_err_t
dag_config_set_int64_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      int64_t value
);
```

dag_config_set_null_attribute

```
dag_err_t
dag_config_set_null_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid
);
```

dag_config_set_string_attribute

```
dag_err_t
dag_config_set_string_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      const char* value
);
```

dag_config_set_struct_attribute

```
dag_err_t dag_config_set_struct_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      void* value
);
```

dag_config_set_uint32_attribute

```
dag_err_t
dag_config_set_uint32_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      uint32_t value
);
```

dag_config_set_uint64_attribute

```
dag_err_t
dag_config_set_uint64_attribute
(
      dag_card_ref_t card,
      attr_uuid_t uuid,
      uint64_t value
);
```

Chapter 9: Firmware Functions

Intoduction

Overview

All of the following functions load or read firmware on a card. They are listed in alphabetical order.

Functions

`dag_firmware_load_pci`

Description

Loads a PCI firmware image onto a card.

```
dag_err_t
dag_firmware_load_pci
(
      const char* name,
      dag_card_ref_t* card_ref,
      const char* filename
);
```

Note: `'name'` is the name of the device
`'card ref'` is a valid pointer to a `dag_ref_t`
`'filename'` is the name of the image to load.

`card_ref` must be a valid. This function will destroy the card and build the object again, including all components and attributes. Therefore any reference to the card prior to using this function will be invalid by the time the function returns. Any reference to a component or attribute will also be invalid by the time the function returns. After returning, `card_ref` will be a reference to a valid card object.

See Chapter 6 for more information on error codes.

Return Value

`'kDagErrNone'` should be returned.

dag_firmware_load_pp

Description

Loads an image onto one of the packet processors.

```
dag_err_t
dag_firmware_load_pp
(
      const char* name,
      dag_card_ref_t* card_ref,
      const char* filename,
      nt which_pp
);
```

Note: `'name'` site name of the device
`'card_ref'` is a valid pointer to a `dag_ref_t`
`'filename'` is the name of the image to load.
`'which_pp'` is the index starting from 0 of the packet processor to load.

card_ref must be a valid. This function will destroy the card and build the object again, including all components and attributes. Therefore any reference to the card prior to using this function will be invalid by the time the function returns. Any reference to a component or attribute will also be invalid by the time the function returns. After returning, card_ref will be a reference to a valid card object.

Return Value

`'kDagErrNone'` should be returned.

dag_firmware_read_swid

Description

Reads a Software ID (SWID) from the card.

```
dag_err_t
dag_firmware_read_swid
(
      dag_card_ref_t card,
      uint8_t* buffer,
      int length
);
```

Note: `'card'` is a reference to a card
`'buffer'` is a buffer to hold the SWID read from the card. It should be at least 128 bytes.
`'length'` is the size of the buffer in bytes.

Return Value

`'kDagErrNone'` should be returned.

dag_firmware_write_swid

Description

Writes a Software ID (SWID) to the card.

```
dag_err_t
dag_firmware_write_swid
(
      dag_card_ref_t card,
      uint8_t* buffer,
      int length,
      uint32_t key
);
```

Note: `'card'` is a reference to a card
`'buffer'` is a buffer to hold the SWID write to the ROM It should be at least 128 bytes.
`'length'` is the size of the buffer in bytes
`'key'` is the key to match the key in the ROM. If this key does not match the SWID wrote will fail..

Return Value

`'kDagErrNone'` should be returned.

Chapter 10: Data Structures & Constants

Introduction

This Chapter describes the types used by the functions and the enumerated types that you can use when setting or getting attribute values. They are listed in alphabetical order.

attr_uuid_t

Description

An attribute identifier. This can be retrieved using the function dag_component_get_config_attribute_uuid.

Include

```
dag_config.h
```

connection_description_t

Description

The connection_description_t is a structure used to setup a connection on the DAG 7.1S card.

```
typedef struct
{
    uint8_t mTUG3_ID;
    uint8_t mVC_ID;
    uint8_t mTUG2_ID;
    uint8_t mTU_ID;
    uint8_t mPortNumber;
    connection_type_t mConnectionType;
    payload_type_t mPayloadType;
    uint8_t mScramble;
    uint8_t mHECCorrection;
    uint8_t mIdleCellMode;
    uint32_t mTimeslotMask;
} connection_description_t;
```

The `connection_description_t` outputs are described in the following table.

Outputs	**Description**
`mTUG_ID`	The TUG3 id to use. Valid values are 0,1 and 2. This field is only valid if the card is using E1
`mVC_ID`	The VC id to use. Valid values are 0 when the cards line rate is configured to STM-1 and 0 to 3 when in STM-4 over E1. When using STM-1 over T1 valid values are 0 to 3 and when using STM-4 over T1 valid values are 0 to 11.
`mTUG2_ID`	When using E1 or T1 valid values are 0 to 6.
`mVC_ID`	The VC id to use. Valid values are 0 when the cards line rate is configured to STM-1 and 0 to 3 when in STM-4 over E1. When using STM-1 over T1 valid values are 0 to 3 and when using STM-4 over T1 valid values are 0 to 11
`mTUG2_ID`	When using E1 or T1, valid values are 0 to 6.
`mTU_ID`	When using E1, valid values are 0 to 2 and when using T1 valid values are 0 to 3.
`mPortNumber`	The DAG 7.1 has 4 ports. Use this field to set the port number of the connection to configure.
`mConnectionType`	The connection type to configure the channel for. See `conection_type_t` earlier in this chapter for valid values.
`mPayloadType`	The payload type to use for this connection. See `payload_type_t` later in this chapter for valid values
`mScramble`	Disable or enable SONET frame scrambling on this connection.
`mHECCorrection`	Disable or enable HEC correction on this connection.
`mIdleCellMode`	Enable or disable idle cell mode. When enabled, idle cells will be dropped.
mTimeslotMask	A bitmask used to configure the timeslots of the connection. The field `mConnectionType` must be set to `kUseTimeslotConfig` for this field to be used.

connection_type_t

Description

```
typedef enum
{
    kPCM31,
    kPCM30,
    kPCM24,
    kUseTimeslotConfig
} connection_type_t;
```

Include

`dag_attribute_codes.h`

crc_t

Different CRC checking modes that cards can be configured to use.

```
typedef enum
{
    kCrcInvalid = -1,
    kCrcOff,
    kCrc16,
    kCrc32
} crc_t;
```

Include

`dag_attribute_codes.h`

dag71s_channelized_rev_id_t

Description

To check what the firmware on the channelized DAG 7.1S card can support, use:

```
typedef enum
{
    kDag71sRevIdInvalid,
    kDag71sRevIdATM,
    kDag71sRevIdATMHDLC,
    kDag71sRevIdATMHDLCRAW,
    kDag71sRevIdHDLC,
    kDag71sRevIdHDLCRAW
} dag71s_channelized_rev_id_t;
```

Include

`dag_attribute_codes.h`

dag_attr_config_status_t

Description

```
typedef enum
{
    kDagAttrErr,
    kDagAttrStatus,
    kDagAttrConfig
} dag_attr_config_status_t;
```

Include

`dag_attribute_codes.h`

dag_card_ref_t

Description

A reference to a card. For example `dag_config` uses this type.

Include

dag_config.h

dag_card_t

Description

The type of DAG card.

```
typedef enum
{
    kDagUnknown
    kDag35e,
    kDag35,
    kDag36d,
    kDag36e,
    kDag36ge,
    kDag37ge,
    kDag37t,
    kDag38,
    kDag42ge,
    kDag423ge,
    kDag42,
    kDag423,
    kDag43ge,
    kDag43s,
    kDag60,
    kDag61,
    kDag62,
    kDag70s,
    kDag70ge,
    kDag71s,

    kFirstDagCard = kDag35e,
    kLastDagCard = kDag71s

} dag_card_t;
```

Include

`dag_config.h`

dag_component_t

Description

A reference to a component. For example `dag_component_get_ subcomponent` uses this type.

Include

`dag_config.h`

dag_err_t

Description

typedef enum

```
{
    kDagErrNone,
    kDagErrInvalidCardRef,
    kDagErrInvalidParameter,
    kDagErrNoSuchComponent,
    kDagErrNoSuchAttribute,
    kDagErrFirmwareVerifyFailed,
    kDagErrFirmwareLoadFailed,
    kDagErrSWIDerror,
    kDagErrSWIDInvalidBytes,
    kDagErrSWIDTimeout,
    kDagErrSWIDInvalidKey,
    kDagErrUnimplemented,
    kDagErrCardNotSupported
} dag_err_t;
```

The `dag_err_t` outputs are described in the following table.

Outputs	**Description**
`kDagErrNone`	No error occurred
`kDagErrInvalidCardRef`	The card referenced is invalid.
`kDagErrFirmwareLoadFailed`	Card failed to load the given firmware image.
`kDagErrSWIDerror`	A general SWID related error occurred.
`kDagErrSWIDInvalidBytes`	An invalid number of bytes was given when reading/writing the SWID.
kDagErrSWIDTimeout	Timeout when communicating with the Xscale. Valid for the 3.7t.
kDagErrSWIDInvalidKey	The given key was invalid and did not match the one in the ROM.

Include

`dag_config.h`

demapper_type_t

Description

To check the type of Demapper on the 3.7T card's firmware image, use:

```
typedef enum
{
    kDemapperTypeATM,
    kDemapperTypeHDLC
} demapper_type_t;
```

Include

`dag_attribute_codes.h`

led_status_t

Description
The status of the LED on the DAG 3.7t pod. Use with the attribute `kUint32AttributeLEDStatus` to change properties of an LED on the pod.

```
typedef enum
{
    kLEDOn
    kLEDOff
    kLEDAtBlinkRate0,
} led_status_t;
```

Include
`dag_attribute_codes.h`

line_rate_t

Description
Line rates that the cards can be configured to.

```
typedef enum
{
    kLineRateAuto,
    kLineRateOC3c,
    kLineRateOC12c,
    kLineRateOC48c,
    kLineRateOC192c,
    kLineRateEthernet10,
    kLineRateEthernet100,
    kLineRateEthernet1000

} line_rate_t;
```

Include
`dag_attribute_codes.h`

line_type_t

Description

An enumerated type denoting the various line types of the DAG 3.7T and DAG7.1s. For use with the attribute `kUint32AttributeLineType`.

```
typedef enum
{
    kLineTypeOff,
    kLineTypeE1,
    kLineTypeE1crc,
    kLineTypeE1unframed,
    kLineTypeT1,
    kLineTypeT1sf,
    kLineTypeT1esf
} line_type_t;
```

Include

`dag_attribute_codes.h`

master_slave_t

Description

To configure the card in master or slave mode, use:

```
typedef enum
{
    kMasterSlaveInvalid,
    kMaster,
    kSlave
} muster_slave_t;
```

Include

`dag_attribute_codes.h`

mux_t

Description

To configure the mux on a DAG 3.7 GP/GF card, use:

```
typedef enum
{
    kMuxMerge,
    kMuxSplit
} mux_t;
```

Include

`dag_attribute_codes.h`

`network_mode_t`

Description
To set the network mode, use:

```
typedef enum
{
    kNetworkModeInvalid,
    kNetworkModeATM,
    kNetworkModePoS,
    kNetworkModeRAW,
    kNetworkModeEth
} network_mode_t;
```

Include
`dag_attribute_codes.h`

`payload_mapping_t`

Description
Defines the payload mapping type. Used with the attribute `kUint32AttributePayloadMapping.`

```
typedef enum
{
   kPayloadMappingDisabled,
   kPayloadMappingAsync,
   kPayloadMappingBitSync,
   kPayloadMappingByteSync1,
   kPayloadMappingByteSync2
} payloadmapping_t
```

Include
dag_attribute_codes.h

`payload_type_t`

Description
typedef enum

```
{
    kPayloadTypeNotConfigured,
    kPayloadTypeATM
    kPayloadTypeHDLC,
    kPayloadTypeRAW
} payload_type_t;
```

Include
`dag_attribute_codes.h`

pci_bus_speed_t

Description

Speeds of the PCI bus. This can be detected using the `pbm` component.

```
typedef enum {
    kPCIBusSpeed33Mhz,
    kPCIBusSpeed66Mhz,
    kPCIBusSpeed100Mhz,
    kPCIBusSpeed133Mhz,
    kPCIBusSpeedUnknown,
    kPCIBusSpeedUnstable
} pci_bus_speed_t;
```

Include

`dag_attribute_codes.h`

sonet_type_t

Description

```
typedef enum
{
    kSonetTypeInvalid,
    kSonetTypeChannelized,
    kSonetTypeConcatenated
} sonet_type_t;
```

Include

`dag_attribute_codes.h`

steer_t

Description

typedef enum

```
{
   kSteerStream0,
   kSteerParity,
   kSteerCrc,
   kSteerIface
}
```

Include

`dag_attribute_codes.h`

terf_strip_t

Description

Used to set the CRC stripping functionality on cards with a Terf component, `kComponentTerf`. The Terf component requires transmit firmware installed.

```
typedef enum
{
    kTerfStripInvalid,
    kTerfNoStrip,
    kTerfStrip16,
    kTerfStrip32
} terf_strip_t;
```

Include

dag_attribute_codes.h

termination_t

Description

The termination mode.

```
typedef enum
{
    /* Both external. */
    kTerminationExternal,

    /* One internal, one external. */
    kTerminationRxExternalTx75ohm,
    kTerminationRxExternalTx100ohm,
    kTerminationRxExternalTx120ohm,
    kTerminationRx75ohmTxExternal,
    kTerminationRx100ohmTxExternal,
    kTerminationRx120ohmTxExternal,

    /* Both internal. */
    kTermination75ohm,
    kTermination100ohm,
    kTermination120ohm

} termination_t;
```

Include

`dag_attribute_codes.h`

tributary_unit_t

Description

The DAG 7.1S card is currently the only card that supports the `tributary_unit_t`. To set the tributary unit on the DAG 7.1S card, use:

```
typedef enum
{
    kTU11,
    kTU12
} tributary_unit_t;
```

Include

dag_attribute_codes.h

vc_pointer_state_t

Description

Different pointer states of virtual containers

```
 typedef enum
{
    kLossOfPointer,
    kAlarmSignalIndicator,
    kPointerValid,
    kConcatenationIndicator
} vc_pointer_state_t;
```

Include

`dag_attribute_codes.h`

vc_size_t

Description

Different virtual container sizes that the cards can be configured to.

```
typedef enum
{
   kVC3,
   kVC4,
   kVC4C
} vc_size_t;
```

Include

`dag_attribute_codes.h`

zero_code_suppress_t

Description

```
typedef enum
{
    kZeroCodeSuppressB8ZS,
    kZeroCodeSuppressAMI

} zero_code_suppress_t;
```

Include

```
dag_attribute_codes.h
```