endace
accelerated

IMA Host API User Guide

EDM04-18

Protection Against Harmful Interference

When present on equipment this manual pertains to, the statement "This device complies with part 15 of the FCC rules" specifies the equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the Federal Communications Commission [FCC] Rules.

These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment.

This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.

Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at their own expense.

Extra Components and Materials

The product that this manual pertains to may include extra components and materials that are not essential to its basic operation, but are necessary to ensure compliance to the product standards required by the United States Federal Communications Commission, and the European EMC Directive. Modification or removal of these components and/or materials, is liable to cause non compliance to these standards, and in doing so invalidate the user's right to operate this equipment in a Class A industrial environment.

Disclaimer

Whilst every effort has been made to ensure accuracy, neither Endace Technology Limited nor any employee of the company, shall be liable on any ground whatsoever to any party in respect of decisions or actions they may make as a result of using this information.

Endace Technology Limited has taken great effort to verify the accuracy of this manual, but nothing herein should be construed as a warranty and Endace shall not be liable for technical or editorial errors or omissions contained herein.

In accordance with the Endace Technology Limited policy of continuing development, the information contained herein is subject to change without notice.

Published by:

Endace Technology® Ltd
Level 9
85 Alexandra Street

PO Box 19246
Hamilton 3244
New Zealand

Phone: +64 7 839 0540
Fax: +64 7 839 0543
support@endace.com
http://www.endace.com

International Locations

New Zealand
Endace Technology Ltd
Building 7, Lambie Drive
PO Box 76802
Manukau City 2241
New Zealand

Phone: +64 9 262 7260
Fax: +64 9 262 7261

Americas
Endace USA® Ltd
Suite 220
11495 Sunset Hill Road
Reston, Virginia 20190
United States of America

Phone: +1 703 382 0155
Fax: +1 703 382 0155

Europe, Middle East & Africa
Endace Europe® Ltd
Sheraton House
Castle Park
Cambridge CB3 0AX
United Kingdom

Phone: +44 1223 370 176
Fax: +44 1223 370 040

Copyright 2007 - 2008 Endace Technology Ltd. All rights reserved.

Contents

Introduction

Overview

The current implementation of IMA is intended for wire-tap or monitoring mode. It is therefore not intended to be used to terminate an IMA connection. However this will be corrected in a future release.

The IMA implementation is based on the IMA standard, AF-PHY-0086.001. Where "Section" references are used in this Programming Guide they refer to sections within that standard.

This document describes the functionality of the IMA Host API used on the DAG 3.7T card.

Naming Convention

All functions in the API are prefixed with `dagima_` . The arguments for the functions are fixed to the C99 standards (`uint_t`, `uint16` etc). The one exception to the UNIX file descriptor for the DAG card (returned by `dag_open`) is a standard `int` type. This is to maintain consistency with existing Endace APIs

Support

In the event that you experience problems with any Endace supplied hardware, or software, it is recommended that you visit the Endace website at www.endace.com. This website includes a *Support* page which offers a range of online assistance options including a Public Knowledge Base. It also allows you to submit a problem report online via the *Online Case Submission* link.

If you have a support contract with Endace you can login using your support username and password which provides access to the secure area of the website. This contains the latest versions of software, device drivers, firmware, user manuals, and release notes.

For more information about the Endace Support Package, or how to obtain (or change) your secure support website login details, please contact sales@endace.com

If you are unable to resolve a problem using the information on the website, you can email Endace Technical Support at support@endace.com for further assistance. .

Reporting Problems

When reporting a problem please supply as much information as possible. The more information you supply the quicker Endace Technical Support will be able to effectively respond to you. Although the exact information available may be limited by the type of problem you are experiencing, you should try to supply the following:

- DAG card model and serial number.
- DAG software version in use as returned by `rpm -q dag-base`
- System log messages generated when DAG device driver is loaded. These can be collected from command `dmesg`, or from log file `/var/log/syslog`.
- Output of `daginf`.
- Firmware versions from `dagrom -x`
- Card configuration as reported by: `dagconfig`
- Network link statistics reported by: `dagconfig -ei`
- Network link configuration from the router where available.
- Contents of any scripts in use.
- Complete output of session where error occurred including any error messages from DAG tools. The `typescript` Unix utility may be useful for this.
- A small section of captured packet trace illustrating the problem.

Note: All functions in the API are declared in: **`dagima.h`**

dagima_atmrx_set-mode

Purpose

Sets the ATM receiver mode for IMA. The mode determines how the IMA will handle incoming ATM cells

Prototype

```
int32_t dagima_atmrx_set_mode (int dagfd, atmrx_mode_t mode )t
```

Parameters

`dagfd`	dag file descriptor
`mode`	ATM Rx mode.

Returns

`0`	success
`1`	failure

Comments

The mode can be any one of the following:

`ATMRX_AUTO`	Configure Rx groups by checking for ICP cells
`ATMRX_AUTOBEST`	Configure Rx groups by checking for ICP cells. This mode is similar to the `ATMRX_AUTO`, but divides IFCs into a logical grouping of 8 top and 8 bottom IFCs, defined by Pod layout. It allows the auto configuration of 2 groups with the same IMA ID. This can be obtained by using IFCs 0-7 for Group 1 (ID: 0) and IFCs 8-15 for Group 2 (ID: 0).
`ATMRX_MANUAL`	Groups must constructed explicitly by the user using the appropriate function. ATM cells will be dropped, while in this mode, if there are no groups configured on the appropriate IFCs.
`ATMRX_FREEFLOW`	Packets will be forwarded to the host

dagima_atmrx_get_mode

Purpose

Retrieve the `ATM RX` mode

Prototype

```
void dagema_atmrx_get_mode (int dagfd )
```

Parameters

`dagfd`	dag file descriptor

Returns

Returns the ATMRX mode

Comments

None

dagima_atmrx_get_stats

Purpose

Returns statistics for the ATM Rx module

Prototype

```
int32_t dagima_atmrx_get_stats (int dagfd, uint32_t cell_stats )
```

Parameters

`Dagfd`	dag file descriptor
`cell_stats`	[in/out] specific statistics

Returns

`> 0`	cell statistic
`-1`	failure

Comments

Select the specific cell statistics from the following:

`DAG_IMA_ALL_CELLS`	Return cumulative statistics for all cell types
`DAG_IMA_DATA_CELLS`	Return statistics for IMA cells
`DAG_IMA_IDLE_CELLS`	Return statistics for OAM idle cells
`DAG_IMA_ICP_CELLS`	Return statistics for OAM ICP sells

dagima_ifc_set_mode

Purpose

Sets IFC mode for a specific interface

Prototype

```
int32_t dagima_ifc_set_mode (int dagfd, uint32_t ifc, ifc_mode_t ifcmode )
```

Parameters

`dagfd`	dag file descriptor
`ifcmode`	[in/out] mode to set the IFC to.

Returns

`0`	success
`1`	failure

Comments

Set the IFC mode to one of the following

`IFC_AUTO`	Set IFC to simple auto-configure
`IFC_AUTOBEST`	Set the IFC to auto-configure using IFC top and bottom separation
`IFC_MANUAL`	Set the IFC
`IFC_FREEFLOW`	Set the IFC to let ATM cells through

Note: You cannot change the mode of an IFC if it is set to `IFC_GROUP`. The only way to change from `IFC_GROUP` is to either change the ATM Rx mode, which releases all groups, or release the group holding the IFC explicitly.

dagima_ifc_get_mode

Purpose

Retreives the current mode of the IFC

Prototype

```
ifc_mode_t dagima_ifc_get_mode (int dagfd, uint32_t ifc)
```

Parameters

`dagfd`	dag file descriptor
`ifc`	IFC to retrieve the mode from.

Returns

`0`	success
`1`	failure

Comments

None

dagima_add_rx_group

Purpose

Add an Rx group to the IMA mux

Prototype

```
void dagima_add_rx_group (int dagfd, uint32_t ifc_mask )
```

Parameters

`dagfd`	dag file descriptor
`ifc_mask`	IFC mask specifying the IFCs that will be adopted by the group

Returns

`> 0`	handle of the group
`-1`	failure

Comments

If the IFC specified by the IFC mask is in use by another group then this function will fail.

dagima_add_txrx_group

Purpose

Remove a group from the IMA mux

Prototype

```
int dagima_add_txrx_group (int dagfd, int32_t handle)
```

Parameters

`dagfd`	DAG file descriptor
`ID`	IMA ID of the group to add
`Framelen`	Group frame length
`TxTRLID`	ID of the RL link
`exp_txactive`	Number of active Tx links before group is considered `GRP_OPERATIONAL`
`exp_rxactive`	Number of active Rx links before group is considered `GRP_OPERATIONAL`
`symconfig`	group configuration
`clock`	group clock mode
`plldescs`	pointer to a link descriptor structure

Returns

`>0`	success, group handle
`<0`	failure
`-1`	Invalid TX ID
`-2`	UM is in the wrong mode, must be in ATMRX_Manual
`-3`	Failed to receive response to message
`-4`	Failed to add links to group
`-5`	Invalid TRL TxID, it does not match any of the link descriptors
`-6`	Failed to transmit
`-7`	Message ID returned is invalid
`-8`	Conflicting IFC numbers
`-9`	Invalid ICP position for the link, it is outside the valid range
`-10`	Conflicting ICP positions, positions must be distributed throughout the link
`-11`	Conflicting connection numbers
`-12`	Invalid Rx ID
`-13`	Invalid frame length
`-14`	Invalid connection number
`-15`	Failed to un register group, once link addition failed
`-16`	Conflicting Tx IDs in link descriptor

Comments

The ID refers to the IMA group ID that is transmitted to the FE. It does not have to be unique within the set of created groups. The handle returned after creating the group will be unique and is used to reference the group via the other IMA API functions.

The `framelen` is the size of the IMA frame transmitted over the Tx links. At time of writing a framelen of 128 is supported.

The `TxTRLID` specifies, by link ID, the Tx link that will be designated as carrying the reference timing information. If the link referred to by Tx ID does not exist, then an arbitrary Tx link will be selected, the user has no control over which link is selected.

The `exp_txactive` and `exp_rxactive` define how many links must be `TX_ACTIVE` and `RX_ACTIVE` before we can go to `GRP_OPERATIONAL`.

A link descriptor describes a bidirectional link over a given interface. As such, it differs slightly, from the definition in the standard, where the use of "`link`" refers to either a Tx or Rx direction. The link descriptor can describe the configuration information for both the Tx and Rx link over a particular interface at one time. Thus, it could also be though of as a interface descriptor, describing the Tx and Rx links over it.

The definition of a link descriptor is given below:

```
ypedef struct link_descriptors_s
{
     uint32_t num_entries;
     link_descriptor_t lldescs[0];
} link_descriptors_t, *plink_descriptors_t;
```

The link descriptors structure contains a list of link descriptors, defined in the following way:

```
typedef struct link_descriptor_s
{
     uint32_t RxID;
     uint32_t TxID;
     uint32_t ICPpos;
     uint32_t ifcnum;
     uint32_t connum;
     uint32_t delay;
} link_descriptor_t, *plink_descriptor_t;
```

A typical link descriptor is defined in the following way:

```
pldescs = malloc( sizeof( link_descriptors_t ) + ( num_links * sizeof(
link_descriptor_t )));

pldescs->lldescs[0].RxID = DEFAULT_LINK_ID;
pldescs->lldescs[0].TxID = 2;
pldescs->lldescs[0].ICPpos = 32;
pldescs->lldescs[0].ifcnum = 1;
pldescs->lldescs[0].connum = 17;
pldescs->lldescs[0].delay = 0;

pldescs->num_entries = 1;
```

The variable `num_links` is equal to the number of links defined in the group.

In this example, we have defined a group with one link, it has a transmit ID of 2, transmits ICP cells at position 32 within the IMA frame, cells are transmitted over connection number 17, which runs over IFC 1. At time of writing the use of the delay parameter is not supported and should be set to 0. It will allow the insertion of link delay on the link, a useful feature for testing. Delay is measured in "cells", thus, if running over a E1 link, to insert a link delay of 25ms, mandated as being the maximum tolerated by the IMA standard, the delay would be 200 cells. The recommended delay maximum is around 1000 cells, or 125 ms.

Note: The connection number relates the number that is assigned when the channels are configured, and will be application specific. The IFC must be the same over which the connection number applies. Failure to ensure the validity of these values will result in undefined behavior.

Setting the `Rx ID` to `DEFAULT_LINK_ID` means the `Rx ID` will be assigned once the FE group has begun transmitting ICP cells. It will be set to the ID of the FE Tx link.

A number of checks are performed by the group addition process over the link descriptors, failing, if the link descriptors contain any duplicate information, i.e., two transmit links over the same IFC, or with the same ID.

dagima_remove_group

Purpose

Remove a group from the IMA Mux

Prototype

```
int dagima_remove_group (int dagfd, int32_t handle)
```

Parameters

`dagfd`	dag file descriptor
`handle`	handle of the group to remove.

Returns

`0`	success
`1`	failure

Commentst

Fails if the group specified does not exist.

dagima_group_add_links

Purpose

Add a link(s) to a group reference by its group handle

Prototype

```
int32_t dagima_group_add_links( int dagfd, uint32_t handle,
link_descriptors_t* plldescs )
```

Parameters

`dagfd`	dag file descriptor
`group_handle`	unique group handle
`plldescs`	pointer to a link descriptors structure

Returns

`0`	success
`<0`	failure
`-1`	Invalid Tx ID.
`-2`	UM is in the wrong mode, must be in ATMRX_MANUAL.
`-3`	Failed to receive response to the message.
`-4`	Failed to add links to the group
`-5`	Invalid TRL Tx ID, it does match any of the link descriptors.
`-6`	Failed to transmit message
`-7`	Message ID returned is invalid
`-8`	Conflicting IFC numbers
`-9`	Invalid ICP position for the link, it is outside the valid range.
`-10`	Conflicting ICP positions, positions must be distributed throughout the link
`-11`	Conflicting connection numbers
`-12`	Invalid Rx ID
`-13`	Invalid frame length
`-14`	Invalid connection number
`-15`	Failed to unregister group, once link addition failed.
`-16`	Conflicting Tx ID's in link descriptors
`-17`	Failed to find group, group handle refers to a group that does not exist

Comments

We describe links using the same structure used when adding a group. See `dagima_group_add_txrx_links` function for a discussion of the link descriptors structure.

dagima_group_remove _rx_link

Purpose

Remove an Rx link from a a group

Prototype

```
int dagima_group_remove_rx_link(int dagfd, uint32_t handle, uint32_t RxID )
```

Parameters

`dagfd`	dag file descriptor
`handle`	handle of the group to remove.
`RxID`	receive ID

Returns

`0`	success
`<0`	failure
`-1`	Failed to find group, group handle refers to a group that does not exist
`-2`	Cannot remove Rx link, group handle refers to a monitor mode group
`-3`	Failed to find Rx link
`-4`	Failed to detach symmetrical Tx link
`-5`	Failed to detach Rx link.

Comments

Keep in mind that removing a Rx link will also remove the associated Tx link over the same IFC. This is because all links are run symmetrically in this version of IMA.

dagima_group_remove_tx_link

Purpose

Remove a Tx link froma group

Prototype

```
int dagima_group_remove_tx_link (int dagfd, uint32_t handle, uint32_t TxID
)
```

Parameters

`dagfd`	dag file descriptor
`handle`	handle of the group to remove.
`TxID`	transmit ID

Returns

`0`	success
`<0`	failure
`-1`	Failed to find group, group handle refers to a group that does not exist
`-2`	Cannot remove Tx link, group handle refers to a monitor mode group
`-3`	Failed to remove Tx link
`-4`	Failed to detach Tx/Rx link

Comments

Keep in mind that removing a Rx link will also remove the associated Tx link over the same IFC. This is because all links are run symmetrically in this version of IMA.

dagima_group_get_tx_transmit

Purpose

Return the number of cells that can be transmitte to the group from the host and subsequently to theline.

Prototype

```
int dagima_group_get_tx_transmit(int dagfd, uint32_t handle)
```

Parameters

`dagfd`	dag file descriptor
`handle`	handle of the group to remove.

Returns

`>0`	success, the number of cells to transmit.
`<0`	failure

Comments

This function is important to regulating transmit from the host. The host must query this function before transmitting a block of cells. The host must transmit a block of cells of that size. If it does not, it will not be able to transmit at full line rate. Once, a block of cells is transmitted, the host must not transmit any more cells until the function is called again and the size of the next block of cells can be ascertained.

dagima_group_get_ifcs

Purpose

Retrieve an IFC mask specifying the IFCs that the group uses to mux.

Prototype

```
int32_t dagima_group_get_ifcs (int dagfd, int32_t group_handle, uint32_t*
ifcmask)
```

Parameters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`ifcmask`	[in/out] IFC mask of the group.

Returns

`0`	success
`1`	failure

Comments

Fails if the group specified does not exist.

dagima_group_get_state

Purpose

Retrieves the state of the group

Prototype

```
int32_t dagima_group_get_state (int dagfd, int32_t handle, pstate_group_t
pstate_group )
```

Parameters

`dagfd`	dag file descriptor
`handle`	handle of the group
`pstate_group`	[in/out] current group state.

Returns

`0`	success
`1`	failure

Comments

Fails if the group specified does not exist. We distinguish between the two groups at either end of a IMA link by using the standard naming convention, FE for the furthest end, i.e the group we are talking to, and NE for the nearest end. More information about group transitions can be found in Section 10.1 Group Operation.

`GRP_NOT_CONFIGURED`	Group is not configured. This state is provided for completeness. In IMA Mux a group that is not configured no longer exists.
`GRP_START_UP`	Group has been configured and is checking FE group parameters.
`GRP_START_UP_ACK`	Group is waiting for `GRP_START_UP_ACK` from FE to move to `GRP_INSUFFICIENT_LINKS`
`GRP_CONFIG_ABORTED_ UNSUPPORTED`	A option is not supported by the NE.
`GRP_CONFIG_ABORTED_INCOMPAT_SYM`	Incompatible operation and configuration between the two groups. Refer to the IMA standard for more in formation
`GRP_CONFIG_ABORTED_UNSUPPORTED_ VERSION`	Unsupported IMA version provided by the FE
`GRP_CONFIG_ABORTED_ABORTED_1`	User specific 1
`GRP_CONFIG_ABORTED_ABORTED_2`	User specific 2
`GRP_CONFIG_ABORTED_ABORTED_3`	User specific 3
`GRP_INSUFFICIENT LINKS`	NE group is waiting for some x number of Rx/Tx links to become ACTIVE before moving to `GRP_OPERATIONAL.`
`GRP_BLOCKED`	The group has been blocked from moving to either the `GRP_INSUFFICIENT_LINKS` or `GRP_OPERATIONAL` state by the some internal inhibit.
`GRP_OPERATIONAL`	Group is in `GRP_OPERATIONAL` and is sending ATM cells to the host.

dagima_group_get_settings

Purpose

Retrieve group settings

Prototype

```
nt32_t dagima_group_get_settings (int dagfd, int32_t handle,
dag_ima_group_settings_t* pima_group_settings )
```

Parermaters

`dagfd`	dag file descriptor
`handle`	handle of the group
`pima_group_settings`	[in/out] selected settings of the group.

Returns

`0`	success
`1`	failure

Comments

The `dag_ima_group_`settings structure contains the following members.

`num_ifcs`	number of IFCs in IMA group
`frame_len`	IMA frame length for the group

dagima_avail_groups

Purpose

Returns a list of available groups that have been configured and are not in the group state `GRP_NOT_CONFIGURED`.

Prototype

```
int32_t dagima_avail_groups( int dagfd, uint32_t* groupmask );
```

Parameters

`dagfd`	dag file descriptor
`groupmask`	[in/out] pointer to a 32bit value set by the callee

Returns

`0`	success
`1`	failure

Comments

The bit mask can be interpreted in the following way:

```
for( y = 0; y < MAX_GROUPS; y++ )
{
      if(( groupmask >> y ) & 1 )
      {
            printf( "Group Handle: %d\n", y );
      }
}
```

The constant `MAX_GROUPS` is defined in the `dag_ima.h` file.

dagima_ifc_get_grouphandle (DEPRECIATED)

Purpose

Get the group handle that the IFC uses to mux

Prototype

```
int32 dagima_ifc_get_grouphandle (int dagfd, uint32_t ifc )
```

Parameters

`dagfd`	dag file descriptor
`ifc`	IFC on which to retrieve the group handle.

Returns

`0`	success
`1`	failure

Comments

Fails if the group handle specified is invalid

dagima_um_set_icpctrl

Purpose

Sets the behaviour of the IMA when it encounters ICP cells

Prototype

```
int dagima_um_set_icpctrl (int dagfd, uint32_t icp_drop, uint32_t
icp_send_trl, uint32_t filler_drop )
```

Parameters

`dagfd`	dag file descriptor
`icp_drop`	Specify if ICP cells are dropped. By default ICP cells are set to drop. Set to 1 to drop ICP cells. Set to 0 to send ICP cells through
`icp_send_trl`	Specify if ICP cells with different SCCI are dropped from the TRL. By default ICP cells from the TRL are dropped. Set to 1 to send TRL ICP cells
`filler_drop`	Specify if filler cells are dropped, by default filler cells are dropped. Set to 0 to send filler cells.

Comments

This is useful if the host wishes to independently analyze ICP cells.

dagima_ifc_get_icpctrl

Purpose

Get the behaviour of the IMA when it encounters an ICP cell.

Prototype

```
int dagima_ifc_get_icptrl(int dagfd, dag_um_icpctrl *picpctrl )
```

Paramaters

`dagfd`	dag file descriptor
`picpctrl`	pointer to a control structure that is filled with the current control values.

Returns

`0`	success
`1`	failure

Comments

The `dag_um_icpctrl_t` structure is defined with the following members

`icp_drop`	specifies if ICP cells are to be dropped, set to 1 by default.
`icp_send_trl`	specifies if ICP cells from the TRL are to be sent to the host, set to 0 by default. Note, if set to 1, and icp_drop is set to 1, ICP cells from the TRL will be received.
`filler_drop`	specifies if filler cells are to be dropped, set to 1 by default

dagima_link_get_rx_state

Purpose

Returns the state of the Rx LSM for the link specified.

Prototype

```
int32_t dagima_link_get_rx_state (int dagfd, int32_t group_handle, uint32_t
linkID, state_rx_t* pstate_rx)
```

Parameters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`pstate_rx`	[in/out] state of the Rx LSM.

Returns

`0`	success
`1`	failure

Comments

More details can be found in Section 10.1.2.2 Receive Link States.

dagima_link_get_tx state

Purpose

Returns the state of the Rx LSM for the link specified

Prototype

```
int32_t dagima_link_get_tx_state_msg_handler (int dagfd, int32_t
group_handle, uint32_t linkID, state_tx_t* pstate_tx)
```

Paramaters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`pstate_tx`	[in/out] state of the Tx LSM.

Returns

`0`	success
`1`	failure

Comments

More details can be found in Section 10.1.2.1 Transmit Link States.

dagima_link_get_ne_alarms

Purpose

Return the alarms for the NE link specified.

Prototype

```
nt32_t dagima_link_get_ne_alarms (int dagfd, int32_t group_handle, uint32_t
linkID, dag_link_fe_alarms palarms )
```

Paramaters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`palarms`	[in/out] NE link alarms

Returns

`0`	success
`1`	failure

Comments

The current implementation defines the following Rx/Tx link alarms.

`LIF`	Rx, Loss of IMA frame
`LODS`	Rx, Loss of deleniation
`RFI_IMA`	Rx, Remove Failure Indicator detected at FE.
`tx_mis_connected`	Tx, link is misconnected, it is in the wrong group.
`rx_mis_connected`	Rx link is misconnected, it is in the wrong group.

More details can be found in Section 12.2.3 IMA Failure Alarms.

dagima_link_get_fe_alarms

Purpose

Returns the alarms for the FE link specified.

Prototype

```
int32_t dagima_link_get_fe_alarms (int dagfd, int32_t group_handle,
uint32_t linkID, dag_link_fe_alarms palarms )
```

Paramaters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`palarms`	[in/out] FE link alarms

Returns

`0`	success
`1`	failure

Comments

The current implementation defines the following Rx/Tx FE link alarms.

`tx_unusable_FE`	FE reports Tx is in state Unusable.
`rx_unusable_FE`	FE reports Rx is in state Unusable.

More details can be found in Section 12.2.3 IMA Failure Alarms.

dagima_link_get_ICP_statistics

Purpose

Returns the statistics for the ICP link specified.

Prototype

```
int32_t dagima_link_get_ICP_statistics (int dagfd, int32_t group_handle,
uint32_t linkID, dag_link_ICP_statistics* pstatistics )
```

Paramaters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`pstatistics`	[in/out] NE link statistics

Returns

`0`	success
`1`	failure

Comments

The current implementation defines the following NE link statistics:

`errored_ICP`	Errored ICP cells, as defined in Table 16, Section 11.
`invalid_ICP`	Invalid ICP cells, as defined in Table 16, Section 11.
`valid_ICP`	Valid ICP cells, as defined in Table 16, Section 11.
`missing_ICP`	Missing ICP cells, as defined in Table 16, Section 11.

dagima_link_get_ICP_statistics

Purpose

Returns the defect statistics for the NE link specified.

Prototype

```
int32_t dagima_link_get_ICP_statistics (int dagfd, int32_t group_handle,
uint32_t linkID, dag_link_ne_defect_statistics* pdefects )
```

Parameters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`pdefects`	[in/out] FE link alarms

Returns

`0`	success
`1`	failure

Comments

The implementation currently defines the following Rx/Tx defect statistics:

`AIS`	Rx, Interface Specific Transmission Convergence sub-layer defect.
`LCD`	Rx, Number of Loss of cell deleniation events.
`LIF`	Rx, Number of Loss of IMA frame events.
`tx_LODS`	Tx, Number of Link out of delay synchronization defects
`rx_LODS`	Rx, Number of Link out of delay synchronization defects.

dagima_link_get_fe_defect_statistics

Purpose

Returns the defect statistics for the FE link specified.

Prototype

```
int32_t dagima_link_get_fe_defect_statistics (int dagfd, int32_t
group_handle, uint32_t linkID, dag_link_fe_defect_statistics* pdefects )
```

Parameters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link
`pdefects`	[in/out] FE defect statistics.

Returns

`0`	success
`1`	failure

Comments

The following FE defect statistics are currently supported:

`Tx_FC_FE`	FE Tx link failure count,
`Rx_FC_FE`	FE Rx link failure count

dagima_link_get_OIF_toggle

Purpose

Provides a quick way of determining if there was a link OIF defect between the current call, and the last time this function was called.

Prototype

```
int32_t dagima_link_get_OIF_toggle( int dagfd, uint32_t group_handle,
uint32_t linkID );
```

Parameters

`dagfd`	dag file descriptor
`group_handle`	handle of the group
`linkID`	ID of the link

Returns

`0`	success (no defect)
`1`	success (defect)
`-1`	failure

Comments

The OIF toggle is a convenience function to set the Provides a quick way to determine if there was an OIF event, i.e. the link lost SYNCH from the last time this was queried.

Note: Querying the state of the OIF toggle clears the state. Querying it again will allow you to determine if a OIF event has occurred since the last time.
You can also determine if an OIF event has occurred by querying the links NE defect statistics.

Version History

Version	Date	Reason
0.1	11/05/06	Initial version – Jason Smith
0.2	28/06/06	Updated to reflect API changes
1	July 2007	First release
2	February 2007	Updated to new template