

# Acnet Alarms via DI

## *Alarm reporting protocol*

Fri, Apr 2, 2004

Acnet alarms reporting to AEOLUS has always been keyed by an EMC (Error Message Code) for the convenience of front ends that do not have to know about an Acnet device index (DI), the main key to Acnet device database records. Of course, AEOLUS must convert the EMC to a DI before it can meaningfully process the alarm message. In recent years, there was talk about changing the alarm reporting protocol to use a DI rather than an EMC. Now it seems that such a scheme has not only been designed, it has been in use by all other Acnet front end designs. This note shows how to implement support for the new scheme.

### *DI capturing*

Since DI values are not known to the front end, some means of capturing them must be done before it is possible to use them in the new alarm message protocol. A new system table #23, called ADEVX, with 4-byte entries indexed by analog channel number, can be used for the purpose. When a RETDAT request is received for an alarm block property for which the listype specified in the SSDN is 2, the channel number from the SSDN can be used to copy the accompanying 20-bit DI to the ADEVX entry. It is likely that a Big Save, which occurs once or twice a day, will include requests for all the device alarm block properties in the database, so we can easily capture all the DI values that we need. As a diagnostic, it may be useful to look for times in which the DI for a given channel# is modified. There should not be any case for which two different DI's are used to address the same alarm block. This is just as bad as having two different setting properties accessing the same channel. But it can be possible, even if unusual, to have two different DI's access the same reading property.

One can imagine implementing the new diagnostic via a data stream or fixed buffer, but such errors are not expected, and it must be done for every node. Instead, simply add it to the ACReq variables block. There can be a count of the number of times a change in ADEVX entry occurs, plus a record of the most recent of such changes. This is only an internal diagnostic.

### *Alarm message protocol*

The document that describes the Acnet alarms reporting protocol can be found at

[http://www-bd.fnal.gov/controls/micro\\_p/mooc\\_project/alarms.html](http://www-bd.fnal.gov/controls/micro_p/mooc_project/alarms.html)

Without repeating everything here, the ERM message format includes a typecode that was 1. A new option allows a typecode value of 14 to indicate that each ERP that follows uses a DIEMC rather than the usual (older) EMC format. The DIEMC is also 8 bytes in length and is formatted as follows:

<i>Field</i>	<i>Size</i>	<i>Meaning</i>
nodeT	2	node*256 + trunk, such as 7209 for node 0972 (node0611)
spare	2	not used
di	4	device DI, presumably word-swapped

An implementation of this new format should be easy, assuming that the ADEVX table is suitably populated with the DI value for each channel. It would be helpful to not only examine RETDAT references to alarm blocks, but also examine SETDAT references, since a new device placed into the database will result in an initial SETDAT to establish the nominal and tolerance values. In this way, one will not have to await the next Big Save for a new device to be able to generate an alarm message.

### Detailed implementation

The system changes are made in the ACReq task, for both the NSERVER and SETNSERV routines. The former supports non-server RETDAT requests; the latter supports non-server SETDAT requests. Check for an alarm block property that uses listype 2. Reference the new ADEVX system table with the channel number obtained from the SSDN. Store the low 20 bits of the PIDI in the normal word order, even though communications with the Acnet clients is done with the word-swapped version. To make it easy to get at the ADEVX table, allocate a new low memory global variable to hold a pointer to the new table, if it exists and has a size large enough to refer to all ADATA entries. The pointer variable should be set up during the initialization of ACReq. The name of the new low memory variable is ADEVXPTR. A routine that takes a pointer to a request/setting packet and captures the DI for deposit into the ADEVX table is ADEVXCAP. From the request/setting packet it can find the listype, channel number, and DI. In case of conflict, where the ADEVX table entry is altered from one nonzero value to a different one, it implies something may be wrong in the database, so log such occurrences internally, just so we have something to look at. One way to log conflicts internally is to simply add a small circular buffer ADEVXLOG to the ACReq task variables. The entry format for this log is as follows:

<i>Field</i>	<i>Size</i>	<i>Meaning</i>
ADEVXSP	2	spare
ADEVXCH	2	channel#
ADEVXDI	4	new device index
ADEVXTI	8	BCD time of day

The new ACReq task variables are:

<i>Field</i>	<i>Size</i>	<i>Meaning</i>
ADEVXKEY	2	visual diagnostic key = 0x4343
ADEVXINX	2	index into ADEVXLOG entries
ADEVXSPR	2	(spare)
ADEVXCHN	2	ADEVX channel# last seen
ADEVXCOC	4	conflict count
ADEVXENC	4	DI encounter count
ADEVXLOG	15*16	conflict log circular buffer of 15 entries, 16 bytes each

### Comment alarms

An alarm message that announces that a node has been rebooted is delivered as a comment alarm. For Acnet, this is either an analog or digital alarm block in which the listype is 56, implying a reference to the CDATA table, which has 32-byte entries for each possible comment alarm. We also need to cache the DIs associated with these alarm blocks, which are not indexed by analog channel number but by CDATA entry number. An entry in the CDATA table consists of an alarm flags word, a trip count, and 28 characters of text. If we can assume that 24 characters is enough text, then the last 4 bytes of the entry can house a DI.

### Missing ADEVX entry

What should happen if there is no entry in the ADEVX indexed by a given channel for which an alarm message must be sent? We may have to ignore it, though one is reluctant to omit sending an alarm message, for which operators are unlikely to feel sympathetic. We must try to ensure that such failures to send alarms are unusual. This means we must be sure that we populate ADEVX entries for all analog channels for which an alarm block property is defined in the Acnet database. In the beginning, we need a program to request alarm blocks from every device that we support in the database. This RETDAT action should serve to

completely populate the ADEVX table, assuming that all nodes have such tables and that all nodes are running a system that includes this new support.

### *RETDAT/SETDAT test page*

The page application PAGEACRQ uses a dummy DI in forming its requests, counting on the fact that the front ends ignore the device index values. But this new scheme is hardly going to ignore it. The page application should be modified to use a distinctively-different device index value, so it will not conflict with a valid device index number. The value now in use is 0x23456. Perhaps a new value to use might be 0xFFFFF, outside any valid range.

### *AERS processing—Have it both ways*

The Acnet alarm message protocol sends an ERM message that consists of a header followed by a number of ERP packets, one for each Acnet device for which an alarm message is being sent. The ERM header identifies which of two formats is to be used in the following ERPs. Both formats are the same but for an 8-byte EMC, which can be of two different formats. The first includes three bytes to specify the source node and some flags, plus 5 bytes of information that is completely determined by the front end. This is the original format that requires AEOLUS to do the extra step of performing a database lookup to obtain a device index from the EMC before further alarm message processing.

The second format of EMC consists of the source node and the 20-bit device index, thus saving AEOLUS one step in alarm message processing. The wrinkle is that this format requires that a front end knows the device index for each device for which it sends an alarm message. A device index has meaning only in the Acnet database, so it is necessary for this information to be somehow obtained from Acnet. It is planned to do this by monitoring alarm block traffic via the RETDAT and SETDAT protocols, in which the device index is included. A special program will be used at first to request all alarm blocks relevant to a given node, thus allowing the front end to capture all of the associated device indexes.

Here are the two 8-byte formats, in big-endian byte order:

Original format

40	node
type	trunk
chan	
0000	

New format with DI

node	trunk
device index	

In the original format, the `type` byte usually has the value 00, except for the comment alarm case, in which its value is 02 and the value of `chan` is the comment number. (Comment alarms are unusual. Only two have ever been defined. One announces that a node has been reset; the other announces that alarms have been reset. The latter is nearly always disabled.)

The `trunk` and `node` bytes refer to the Acnet node number, such as 0x0987, for which the

trunk byte is 0x09 and the node byte is 0x87.

The 20-bit device index is specified in word-swapped Vax-compatible byte order. For example, a value of 0x23456 would be represented in the above word order as 0x3456 followed by 0x0002. The values stored in the new ADEVX system table are in the normal order, so that the above example would be stored simply as the long word 0x00023456.

The new AERS implementation may first construct the original format. Then, when it is time to send the ERM to AEOLUS, it can first review whether it can use the new device index format of EMC. Alternatively, when it is building each ERP, it can check whether a DI exists and save it in a parallel array, counting how many it finds. When it is time to send the EMC, it can ask whether the count matches the number of ERPs. If it does, it can do the conversion to the new format. In this way, all the checking is done as each ERP is generated, so that no additional searches are needed before sending the ERM.

Another plan is to build the ERPs with the new format, optimistically thinking that this will be the format that works. In this case, the second word of the 4-word EMC can hold the channel number in case it is needed to revert to the original EMC format. (It may also hold a flag that indicates a comment alarm.) As a diagnostic, the second word could remain even in the new format, as it is so far ignored by AEOLUS. If conversion to the original format is needed, overwrite the device index field with a channel number followed by a zero word, set the third byte to 02 if a comment, else to 00, and set the 1st, 2nd, and 4th bytes to the flags value 0x40, the local Acnet node number byte, and the local Acnet trunk number.

### *Post-implementation*

The scheme used in LOOPAERS is the following. At the time that the `Extract` routine is called to pull out and format the ERP packets from the cache, set a global flag that indicates to `Extract` that it should use the DI-based form of EMC inside the ERP. If `Extract`, when it seeks to find the required DI via the `LookupDI` function, cannot find one for the given channel index (or comment index), it resets the global flag and formats the current ERP in the original way. The caller to `Extract` notices such a reset of the global flag, and if it occurs, it reviews all the previous ERPs already assembled in the ERM buffer and modifies them to use the original EMC form. In this way, when the ERM buffer is full, or the cache has been emptied, the ERM message consists of ERPs all using a common EMC format. So, if all devices being alarmed have available DIs, LOOPAERS sends AEOLUS the new DI-based form that saves it an extra database lookup step; otherwise, the old tried-and-true format is used that requires each device to have an EMC in the database. In the latter case, should there be no EMC in the database for that device, we will get a `BADEMC` message on the alarm screen that should alert us to the problem.

Suppose there is a `BADEMC` message. What DI is missing? The `BADEMC` message includes the EMC code that was sent from the front-end, and that will tell us which channel was lacking a captured DI.

### *Upgrading all nodes*

Two components of software are required to support sending DI-based alarm messages. One is in the `ACReq` task in the system code, and the other is in `LOOPAERS`. Although it would be best to upgrade all nodes together, we may install either component first in any node. Acnet DI capture can work without an updated `LOOPAERS`, and `LOOPAERS` can operate without an updated system.