



## Tip of the Week :

- How to use the Local and Central Alarm Servers

# [TINE Alarms

(a short review)

- An alarm belongs to a 'device'
- An alarm has a timestamp
  - Last alarm signal
- An alarm has a start time
  - First alarm signal
- An alarm has a code
  - Identifies and defines the alarm
- An alarm can have data
  - Up to 64 total bytes (any format)

# [ TINE Alarms ]

- An alarm can be persistent
  - Always there (until someone takes care of it)
  - e.g. hardware readout error
- An alarm can be transient
  - A change of state from okay to not okay
  - e.g. beam dump, quench, RF trip
- An alarm can be oscillating
  - There for a while then not there then back again.
  - e.g. intermittent hardware error (a flickering sedac error).

# [ TINE Alarm Definition ]

- Alarm Code cross references static alarm information
  - Severity
    - (Release 4.0: can override dynamically !)
  - System ID
    - Usually = 0 (=> let the CAS decide which system)
  - Tag
    - Short description of alarm
  - Alarm Text, Device Text, Data Text, url
  - Data Type and Size

# [ TINE Alarm Severities ]

- Range of 0 to 15
  - 0 = test alarm
  - 15 = you can't possibly have beam unless you fix this
  - Typically:
    - 0 => not really an alarm (not handled at CAS)
    - 1 – 6 => information (not archived)
    - 7 – 12 => warning
    - > 12 => fatal

# The Old Alarm Viewer :

**DESY2 Alarm Viewer: Archived Data**

Printing Display Compact! View Options Machine Alarm List to File Extra Forms

Info (Sev < 7) Warnings (7 <= Sev < 12) Fatal Alarms (Sev >= 12)

- Not Archived 6 96

Alarm Display: Live  Archive

Archived Alarms Severity >= 12 from Thu Mar 27 to Thu Mar 27 2008

System	Alarms	System	Alarms	System	Alarms
Magnet		SeKi		System	4
HCorr		Peak-Strip		Hardware	47
VCorr		Zyklus Gen		Radio	
HF		Trigger Mod		Tim. Mon	
AM-Gen		Timing		Bunche	
Vac		Schirme		I-Hist	
Per.Interlock	49			Profile	

Severity: 12

The number of alarms with Severity >= 12: 100

refresh

March 2008

Mon	Tue	Wed	Thu	Fri	Sat	Sun
25	26	27	28	29	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Today: 27-Mar-08

Alarm View

Show All Alarm-Events

Extra Info in Grid

Collect arch data ONLY for

Subsystem: SUB

Alm-Subsystem: Per.Interlock

Archived Alarm list for all alarm subsystems : 100 alarms.  Show Terminated Alarms

SubSystem	Loc.	Error	Severity	Alarm Time	Duration / Info
System	PIEventsRem	Not Responding, Thu	12	Mar 27 21:39:42	Persistent ?
Per.Interlock	Platz-501	SEDAC-Err, 0	11!	Mar 27 20:09:54	Persistent ?
Hardware	Platz-501	SEDAC-Err, 0	11!	Mar 27 20:09:54	Persistent ?
System	IEAlarm	Not Responding, Thu	12	Mar 27 20:03:51	Persistent ?
Per.Interlock	Platz-501	SEDAC-Err, 0	11!	Mar 27 15:49:41	
Hardware	Platz-501	SEDAC-Err, 0	11!	Mar 27 15:49:41	
Per.Interlock	Platz-282	Elektr Stroer, 0	11!	Mar 27 15:09:41	39 sec
Hardware	Platz-282	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Hardware	Platz-232	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Per.Interlock	Platz-232	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Hardware	Platz-229N	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Per.Interlock	Platz-229N	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Hardware	Platz-228N	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Per.Interlock	Platz-228N	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Per.Interlock	Platz-224	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Per.Interlock	Platz-281	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Hardware	Platz-224	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Hardware	Platz-343	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr
Per.Interlock	Platz-282	SEDAC-Err, 0	11!	Mar 27 08:16:41	6.8 hr

Alarm Description

Elektr. Stoerung.  
Dev. info: MessPlatz  
Device : Platz-282

The alarm is Terminated.

Start: Mar 27 15:09:41  
Stop: Mar 27 15:10:20  
Duration: 39 sec

Duration: 39 sec

Alarm from server:  
/DESY2/D3Strahlung

Further Information:  
not supplied

# [ Alarm Configuration ]

- Static information from either
  - 'alarms.csv' (relative to the equipment module)
  - 'fec.xml' (the <ALARM\_DEFINITION> tag)
  - API Calls (java device server wizard)

# Automatic Alarm Generation

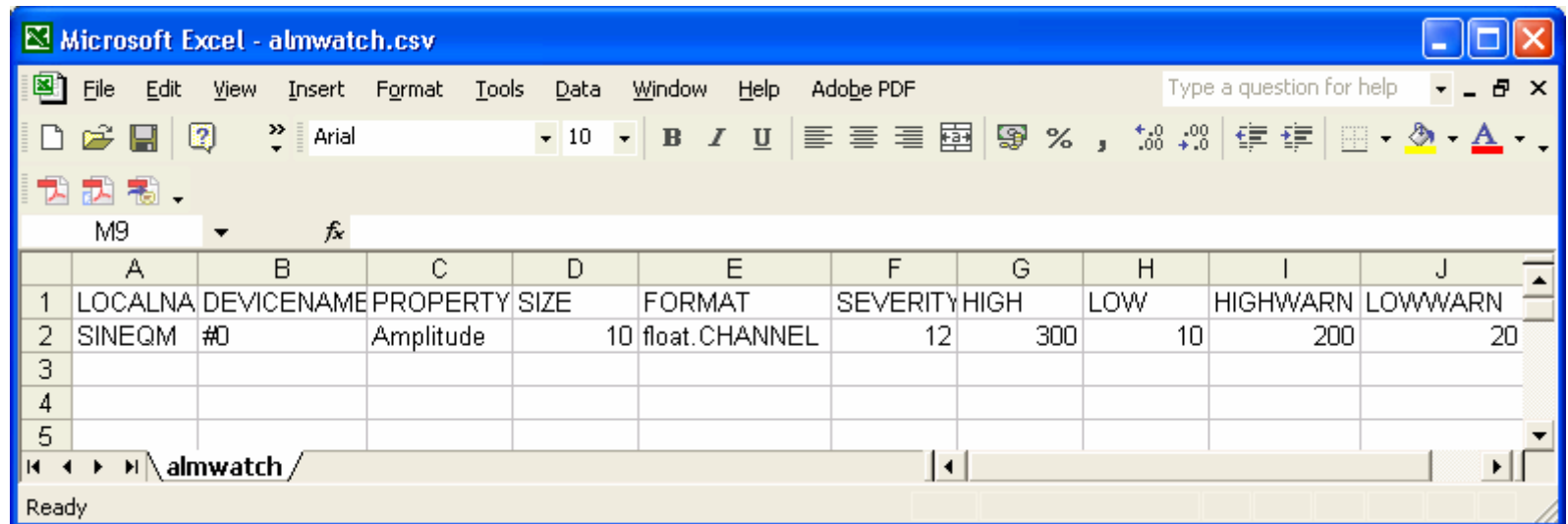
- An Alarm Watch Table
  - Threshold alarms
    - readback is out of bounds !
    - value\_too\_high, value\_too\_low, warn\_too\_high, warn\_too\_low
  - Value Mask alarms
    - masked readback does not match the ! 'normal' value
    - invalid\_data
- Configure via either
  - 'almwatch.csv' (relative to the equipment module) or
  - 'fec.xml' (the <ALARM> tag) or
  - API Calls
- You do NOT have to Set/Clear these alarms yourself!



# Alarm Watch Table

almwatch.csv

Instruct the Local Alarm Server which properties should be monitored and where the thresholds are :



Microsoft Excel - almwatch.csv

File Edit View Insert Format Tools Data Window Help Adobe PDF

Type a question for help

Arial 10 B I U

M9

	A	B	C	D	E	F	G	H	I	J
1	LOCALNA	DEVICENAME	PROPERTY	SIZE	FORMAT	SEVERITY	HIGH	LOW	HIGHWARN	LOWWARN
2	SINEQM	#0	Amplitude	10	float.CHANNEL	12	300	10	200	20
3										
4										
5										

almwatch

Ready

# Alarm Watch Table

fec.xml

```
</DEVICE>
- <PROPERTY>
  <NAME>Sine</NAME>
  <DEVICE_SET />
  <EGU>v</EGU>
  <XEGU>r</XEGU>
  <MAX>1000</MAX>
  <MIN>0</MIN>
  <XMAX>8092</XMAX>
  <XMIN>0</XMIN>
  <ID>0</ID>
  <DESCRIPTION>Sine Curve</DESCRIPTION>
  <SIZE_IN>0</SIZE_IN>
  <DTYPE_IN>null</DTYPE_IN>
  <SIZE_OUT>1024</SIZE_OUT>
  <DTYPE_OUT>float.SPECTRUM</DTYPE_OUT>
  <ACCESS>READ</ACCESS>
  <REDIRECTION />
- <ALARM>
  <DEVICE_NAME>SineGen0</DEVICE_NAME>
  <SEVERITY_HIGH>12</SEVERITY_HIGH>
  <SEVERITY_LOW>12</SEVERITY_LOW>
  <SEVERITY_HIGH_WARN>10</SEVERITY_HIGH_WARN>
  <SEVERITY_LOW_WARN>10</SEVERITY_LOW_WARN>
  <SYSTEM>100</SYSTEM>
  <VALUE_MASK />
  <VALUE_NORMAL />
  <COUNT_THRESHOLD>3</COUNT_THRESHOLD>
  <VALUE_HIGH>800</VALUE_HIGH>
  <VALUE_LOW>50</VALUE_LOW>
  <VALUE_HIGH_WARN>700</VALUE_HIGH_WARN>
  <VALUE_LOW_WARN>100</VALUE_LOW_WARN>
</ALARM>
- <HISTORY>
  <DEVICE_NAME>SineGen0</DEVICE_NAME>
  <TOLERANCE>10%</TOLERANCE>
```

Can also mask a readback value and compare versus a 'normal' value

# Alarm Watch Table

API :

```
int AppendAlarmWatchTable ( char *      eqm,  
                           char *      prp,  
                           char *      dev,  
                           int         siz,  
                           int         fmt,  
                           int         atyp,  
                           int         sev,  
                           int         sys,  
                           ALM_THRESHOLDS * thr  
                           )
```

Inserts a property to be monitored into the local alarm server's Watch Table.

Certain alarms are to be set whenever the value of a property exceeds a defineable threshold. Such alarms can be managed automatically by the local alarm server if the alarm criteria are entered into the alarm watch table. This can be achieved by calling this routine (or supplying a startup configuration file `almwatch.csv`).

#### Parameters:

*eqm* is the 6-character local equipment identifier name, which is internal to server.  
*prp* is the property which is to be called by the local alarm server  
*dev* is the device name associated with the property to be called by the local alarm server  
*siz* is the data array size to be called by the local alarm server  
*fmt* is the TINE data format to be called by the local alarm server  
*atyp* is the TINE data array type to be applied to the property called by the local alarm server  
*sev* is the severity of the alarm issued when the data returned by the call exceed the given thresholds.  
*sys* is the alarm system identifier to be associated with the alarm.  
*thr* is an **ALM\_THRESHOLDS** object specifying the threshold criteria for setting the alarm

**Note:** Information can also be entered into the alarm watch table by supplying the startup configuration file `almwatch.csv`, where the input parameters are given in the relevant columns. This is frequently the preferred way of inputting such alarm information, since it does not involve hard-coding such alarm criteria.

#### Returns:

0 upon success, otherwise a TINE error code.

# [ Alarm API : ]

- Make sure your alarms are defined !
  - alarms.csv (or fec.xml, or API)
- Make use of ClearAlarm()/SetAlarm() inside your I/O loop.
  - ClearAlarm() at the start of the loop
    - Increments the 'clear counter'
  - If the alarm is still active then SetAlarm()
    - Resets the 'clear counter'

# Alarm Definitions

alarms.csv :

	A	B	C	D	E	F	G	H	I	J
1	ALARMTAG	ALARMCODE	SEVERITY	DATAFOR	DATAARRAYSIZE	ALARMTEXT	DEVICETEXT	DATATEXTURL		ALARMSYST
2	hardware erro	34	7	short	3	BLM hardware c	Pforte Beam Los	Line - crate	http://mcs/petra3/servers/bl	0
3	dump threshc	512	12	float	1	Beam Loss cros	Pforte Beam Los	Beam Los:	http://mcs/petra3/procedure	0

#definition from errors.h

# Alarm Definitions

fec.xml :

```
- <EQM>
  <NAME>BLMEQM</NAME>
  <SERVER>BLM</SERVER>
  <CONTEXT>PETRA</CONTEXT>
  <SUBSYSTEM>DIAG</SUBSYSTEM>
  <GROUP />
  <GROUP_INDEX />
- <ALARM_DEFINITION>
  <TAG>hardware error</TAG>
  <DATA_FORMAT>short</DATA_FORMAT>
  <ALARM_TEXT>BLM hardware cassette readback error</ALARM_TEXT>
  <DEVICE_TEXT>Pforte Beam Loss Monitor</DEVICE_TEXT>
  <DATA_TEXT>Line - crate - subaddress</DATA_TEXT>
  <URL>http://mcs/petra3/servers/blm/hardware-trouble.html</URL>
  <ALARM_CODE>34</ALARM_CODE>
  <ALARM_MASK />
  <SEVERITY>7</SEVERITY>
  <ALARM_SYSTEM />
  <DATA_SIZE>3</DATA_SIZE>
</ALARM_DEFINITION>
- <ALARM_DEFINITION>
  <TAG>dump threshold reached</TAG>
  <DATA_FORMAT>flot</DATA_FORMAT>
  <ALARM_TEXT>Beam Loss crossed alarm threshold and triggered dump</ALARM_TEXT>
  <DEVICE_TEXT>Pforte Beam Loss Monitor</DEVICE_TEXT>
  <DATA_TEXT>Beam Loss (cnts/sec) that crossed the alarm threshold</DATA_TEXT>
  <URL>http://mcs/petra3/procedures/restart-following-dump.html</URL>
  <ALARM_CODE>512</ALARM_CODE>
  <ALARM_MASK />
  <SEVERITY>12</SEVERITY>
  <ALARM_SYSTEM />
  <DATA_SIZE>3</DATA_SIZE>
</ALARM_DEFINITION>
```

# [ Alarm Definitions ]

## API Call :

```
int AppendAlarmInfoTable ( char * eqm,  
                          ADS * ads  
                          )
```

Inserts an alarm definition into the alarm definition table.

As an alternative to the <local name>-alarms.csv configuration file, the front end server can make use of this API call in order to fill in the alarm definition table describing locally generated alarms. This is particularly useful for embedded platforms where there is no file system, or where a TINE server is used as a translation layer and needs to map a given alarm system onto the TINE alarm system.

### Parameters:

*eqm* is the 6-character local equipment identifier name, which is internal to server.

*ads* is a pointer to an Alarm Definition Structure (ADS) containing the alarm table information which is to be appended to the alarm definition table.

### Returns:

0 upon success, otherwise a TINE error code.

```
strncpy(ads.alarmTag,"Threshold exceeded",16);  
ads.alarmCode = 512;  
ads.alarmMask = 0xff;  
ads.severity = 7;  
ads.alarmDataFormat = BFMT(CF_FLOAT);  
ads.alarmDataArraySize = 1;  
ads.alarmSystem = 640;  
strncpy(ads.alarmText,"Take action immediately!",40);  
strncpy(ads.dataText,"Current threshold setting",40);  
strncpy(ads.url,"alarm512.html",40);  
if ((cc=appendAlarmInfoTable(EQPMODNAME,&ads)) != 0)  
{  
    feclog("appendAlarmInfoTable : %s",erlstr(cc));  
}
```

Java: from TEquipmentModule


```
public int addAlarmDefinition(int code, TAlarmDefinition adef)
```

# [ Alarm API Example : C ]


```
|
#define dump_threshold_reached 512
void hdwIo(void)
{
    int i;
    ClearAlarm(BLMEQM_TAG,-1); // clear them all

    for (i=0; i<gNumBlms; i++)
    {
        if ((cc=rdBlmLoss(i)) != 0)
        { // there was a readout problem
            SetAlarm(BLMEQM_TAG,i,hardware_error,blm[i].addr);
            continue;
        }
        if (blm[i].loss > blm[i].dumpThreshold)
        {
            SetAlarm(BLMEQM_TAG,i,dump_threshold_reached,blm[i].loss);
        }
    }
}
```

Pass the hardware address that caused the problem



Pass the readback value that crossed the threshold





# Alarm API : java

Many convenient constructors for setAlarm()

```
    }  
    /**  
     * \brief sets an alarm with the given data set  
     *  
     * @param code The alarm code to apply to the alarm  
     * @param data The alarm data associated with the alarm  
     * @return  
     */  
    public TAlarm setAlarm(int code, TDataType data)  
    {  
        return setAlarm(code, data.getDataBuffer(), TAlarmDescriptor.NEW);  
    }  
    /**  
     * \brief sets an alarm with the given data set  
     *  
     * @param code The alarm code to apply to the alarm  
     * @param data The alarm data associated with the alarm  
     * @return  
     */  
    public TAlarm setAlarm(int code, float data)  
    {  
        float[] fdata = new float[1];  
        fdata[0] = data;  
        TDataType td = new TDataType(fdata);  
        return setAlarm(code, td.getDataBuffer(), TAlarmDescriptor.NEW);  
    }  
    /**  
     * \brief sets an alarm with the given data set  
     *  
     */
```

# [ Alarm API : java ]

```
/**
 * Called in the background task.
 *
 * Note the calls to the local alarm server.
 * clearAlarm() increments the clear counter and is called when the routine is entered.
 * setAlarm() is called when an alarm state is detected.
 * This is just an example. Alarms for 'value too high', etc. are better handled
 * through the use of an 'almwatch.csv' configuration file.
 *
 */
public void update()
{
    clearAlarm(512);
    incrementPhase();
    for (int i = 0; i < 1024; i++)
        myData[i] = amplitude * Math.sin(phase + frequency * 6.28 * ((double) i / 1024.0);
    if (amplitude > 100) setAlarm(512); // amplitude too high !
}
public double getPhase()
{
    return phase;
}
```

# [ Some Notes ]

- ClearAlarm()
  - does not remove the alarm
  - Increments the clear counter
  - SetAlarm() resets the clear counter
  - If clear counter increase by more than 1 prior to the next SetAlarm() the alarm is marked as 'oscillating'
  - Clear counter > 8 => alarm has terminated !
- Transient Alarms need to call SetAlarmEx()
  - Can pass the alarm flag 'almINSTANT'
  - Immediately flagged as terminated
- Alarms stay in the local alarm table for the duration of the Alarm Termination Window (default = 4 seconds)
  - Longer if a configured CAS has not read the alarm
- CAS can react to (configured) alarm signals
  - Trigger events
  - Send email
  - Send to central logger

# [ Reading Alarms from a Server ]

- Best Practice:
  - Monitor the Alarms ‘Snapshot’
  - Stock Property “NALARMS”
    - 5 long integer values
      - Total number of alarms
      - UTC Timestamp of the most recent
      - Highest severity
      - Number at the most recent timestamp
      - Number at the highest severity
  - Can incrementally update an alarm cache using this snapshot (CAS)
    - Use DATACHANGE mode
    - If something’s different : get the most recent alarm set.
- Alarm Viewer gets all alarm information from the CAS !
- Java : TAlarmSystem query class with lots of static methods to get alarm information !

# [ Reading Alarms in java ]

You probably won't be doing this, but just for fun :

- TLink InkJalms =  
TAlarmSystem.monitorNumberOfAlarms("PETRA",null,"RF",12,cbNalms)
  - Monitors number of alarms from the PETRA CAS with severity  $\geq 12$
- TLink InkJalms =  
TAlarmSystem.monitorNumberOfAlarms("PETRA","ELWIS3",null,12,cbNalms)
  - Monitors number of alarms from server "ELWIS3" with severity  $\geq 12$  directly
- Inside cbNalms(Ink) :
  - `timeAlm = Ink.getLastTimeStamp();`
  - `TAlarmMessage[] almsNew = getAlarms("PETRA","RF",timeAlm,timeNow,12);`
  - Join or filter the 'almsNew' list to a cached alarm list.
    - e.g. remove anything older than 1 hour
    - e.g. don't include 'terminated' alarms in the list
    - etc.