Tip of the Week :

•Local Alarm and Archive Filtering

Tip of the Week

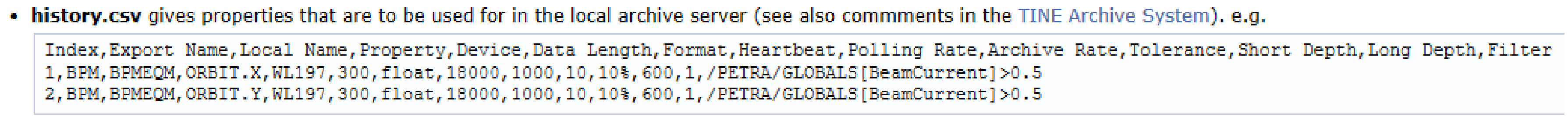
- Problem:
  - Need a local history of property P
  - But this takes up a lot of disk space especially when there is no beam
  - e.g. BPM positions
- Solution:
  - Trap ‘access & CA_HIST’ in your equipment module handler OR
    - still need to decide if there’s beam in the machine
  - Apply a filter !

Tip of the Week

- Problem:
  - Fatal alarm if property P exceeds some value
  - But only the machine state is ‘Running’
  - e.g. Magnets, RF stations, etc.
- Solution:
  - Trap ‘access & CA_ALARM’ in your equipment module handler OR
    - still need to look at the machine state
  - Apply a filter !

Local History Filters

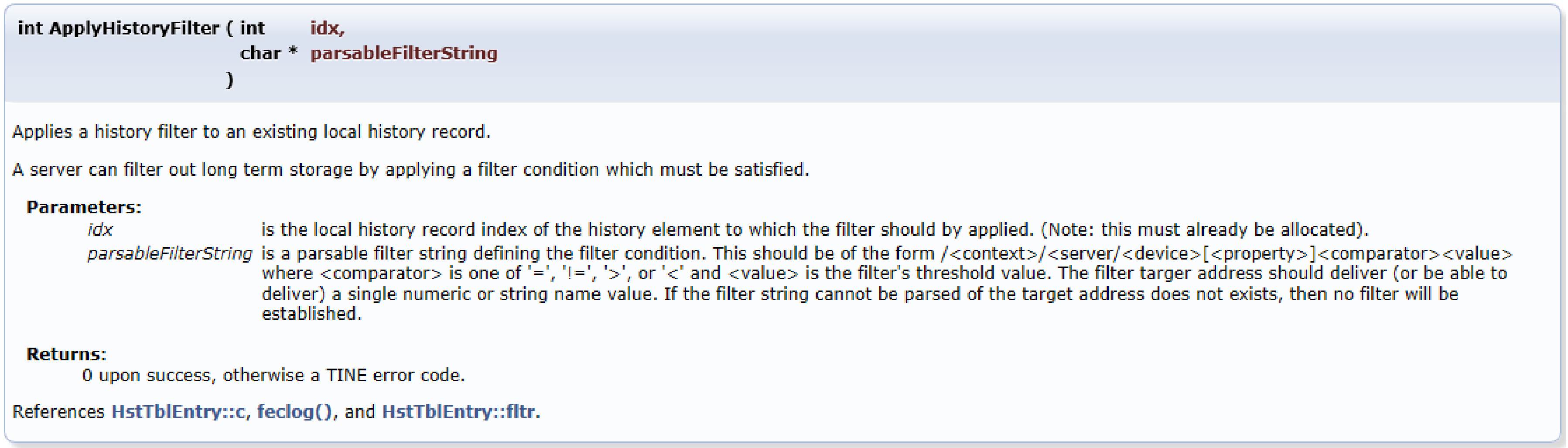
Configuration : history.csv



- **history.csv** gives properties that are to be used for in the local archive server (see also commments in the TINE Archive System). e.g.

```
Index,Export Name,Local Name,Property,Device,Data Length,Format,Heartbeat,Polling Rate,Archive Rate,Tolerance,Short Depth,Long Depth,Filter
1,BPM,BPMEQM,ORBIT.X,WL197,300,float,18000,1000,10,10%,600,1,/PETRA/GLOBALS[BeamCurrent]>0.5
2,BPM,BPMEQM,ORBIT.Y,WL197,300,float,18000,1000,10,10%,600,1,/PETRA/GLOBALS[BeamCurrent]>0.5
```

C API : ApplyHistoryFilter()



```
int ApplyHistoryFilter ( int      idx,
                         char *   parsableFilterString
                       )
```

Applies a history filter to an existing local history record.

A server can filter out long term storage by applying a filter condition which must be satisfied.

Parameters:

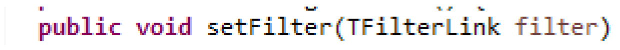
idx	is the local history record index of the history element to which the filter should by applied. (Note: this must already be allocated).
parsableFilterString	is a parsable filter string defining the filter condition. This should be of the form /<context>/<server/<device>[<property>]<comparator><value> where <comparator> is one of '=', '!=', '>', or '<' and <value> is the filter's threshold value. The filter targer address should deliver (or be able to deliver) a single numeric or string name value. If the filter string cannot be parsed of the target address does not exists, then no filter will be established.

Returns:

0 upon success, otherwise a TINE error code.

References **HstTblEntry::c**, **feclog()**, and **HstTblEntry::fltr**.

java API : THistoryRecord class



```
public void setFilter(TFilterLink filter)
```

Local Alarm Server Filters

Configuration : almwatch.csv

- **almwatch.csv** gives a list of properties and value thresholds for setting value_too_high and value_too_low alarms. A 'warning' threshold can also be supplied to set warn_too_high and warn_too_low alarms.

```
LOCALNAME,DEVICENAME,PROPERTY,SIZE,FORMAT,SEVERITY,HIGH,LOW,HIGHWARN,LOWWARN,FILTER
SINEQM,#0,SINE,10,Float,15,500,0,400,10,/DESY2/GLOBALS[ParticleType]=1
```

C API : ApplyAlarmWatchFilter()

```
int ApplyAlarmWatchFilter ( char *  eqm,
                            char *  prp,
                            char *  dev,
                            char *  parsableFilterString
                          )
```

Applies a filter to an existing local alarm server's Watch Table.

A server can filter out automatic watch table alarms by applying a filter condition which must be satisfied.

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
parsableFilterString	is a parsable filter string defining the filter condition. This should be of the form /<context>/<server/<device>[<property>]<comparator><value> where <comparator> is one of '=', '!=', '>', or '<' and <value> is the filter's threshold value. The filter targer address should deliver (or be able to deliver) a single numeric or string name value. If the filter string cannot be parsed of the target address does not exists, then no filter will be established.

Returns:
0 upon success, otherwise a TINE error code.

References **feclog()**.

java API : TAlarmWatchEntry class

```
public void setFilter(TFilterLink filter)
```