NAME

auplugin_init, auplugin_stop, auplugin_event_loop, auplugin_event_feed − plugin event processing helpers

SYNOPSIS

#include <auplugin.h>

int auplugin_init(int *inbound_fd***, unsigned** *queue_size***, int** *q_flags***, const char ****path***);**
void auplugin_stop(void);
void auplugin_event_loop(auplugin_callback_ptr *callback***);**
int auplugin_event_feed(auparse_callback_ptr *callback***, unsigned** *timer_interval***, auplugin_timer_callback_ptr** *timer_cb***);**

DESCRIPTION

auplugin_init initializes the plugin framework. The *inbound_fd* parameter specifies the file descriptor that will provide audit messages, typically standard input. The *queue_size* argument controls the maximum number of events that may be queued for processing. The *q_flags* parameter selects in-memory or file-backed storage using the **AUPLUGIN_Q_*** constants defined in **auplugin.h**. If*q_fla gs* includes **AUPLUGIN_Q_IN_FILE,** *path* specifies the backing file. Any events already present in the file are queued on startup so plugins resume processing previously unhandled records. The library maintains global state for its queue and worker threads. Only one plugin instance is supported, so callers must not invoke auplugin_init() concurrently from multiple threads. The function returns 0 on success or −1 if initialization fails.

auplugin_stop signals the framework to terminate. It is normally called from a SIGTERM handler or other shutdown logic.

auplugin_event_loop starts a worker thread to deliver queued events to the supplied *callback* function one record at a time. The function blocks in the caller until **auplugin_stop** is invoked.

auplugin_event_feed behaves like **auplugin_event_loop**, except that queued events are fed to libauparse. The provided *callback* must match the **auparse_callback_ptr** type. The *timer_interval* argument specifies how many seconds the worker thread will wait for new records. A value of 0 disables the timer logic. When the interval elapses, **auparse_feed_age_events** is called to flush aged events. If *timer_cb* is not **NULL,** it is invoked with the interval before the flush. Passing a *timer_cb* of **NULL** keeps the default behaviour of calling **auparse_feed_age_events** only. The function returns 0 on success or −1 if libauparse could not be initialized.

Plugins can query queue statistics with **auplugin_queue_depth**, **auplugin_queue_max_depth**, and **auplugin_queue_overflow**. Register a callback with **auplugin_register_stats_callback**, and invoke it using **auplugin_report_stats**.

SIGNAL HANDLING

Plugins should establish signal handlers with sigaction(2) before entering the event loop. The SIGTERM handler should call auplugin_stop() to shut down the worker thread. Handlers for other signals, such as SIGHUP or SIGUSR1, should set global flags that are processed in the event or timer callbacks.

Example:

```
static volatile sig_atomic_t reload;
static void handler(int sig)
{
   if (sig == SIGTERM)
      auplugin_stop();
   else if (sig == SIGHUP)
      reload = 1;
}
```

SEE ALSO

auplugin_fgets(3), **auparse_feed**(3)

AUTHOR

Steve Grubb