



# ENH Records Specification

## Enhancing the Flex™ Debug Log

### Introduction

The Enhanced Logging Initiative is the outcome of several meetings convened by License Tracker Inc. These meetings were held to discuss issues related to software usage, capturing usage information and post-analysis of this data, with a view to achieving these goals in the most cost effective and efficient manner. Participants in these meetings included application software vendors, enterprise software users, usage analysis tool vendors and vendors of license management software.

The first project of the Enhanced Logging Initiative is the development of this ENH Records Specification. This includes the syntax for the enhanced records to be inserted into the Flex debug log, as well as the provision of source code for use in modified vendor daemons.

All of the ENH records are prepended with the standard debug log time and the vendor daemon name, *hh:mm:ss (daemon\_name)*, to achieve a visual consistency with the rest of the file.

### Basic Usage Logging

Basic usage logging makes use of existing callbacks within the Flex vendor daemon API to insert enhanced information records into the debug log. Proper interpretation of these records requires processing of subsequent standard debug log messages (OUT, DENIED, QUEUED, etc.).

Basic usage consists of two records:

- ENH\_OUT
- ENH\_IN

#### Mandatory Fields - ENH\_OUT and ENH\_IN

Both the ENH\_OUT and ENH\_IN records have the following mandatory fields:

Field	Type	Description
feature	string	feature name
user	string	user logon ID
host	string	host name
timestamp	string	milliseconds since 00:00:00 Jan. 1, 1970 (UTC) formatted as <seconds>.mmm
num_licenses	int	number of licenses requested/returned
session_handle	int	handle assigned by the license server
display	string	display details
addr	string	IP address of user



### Additional Mandatory Fields - ENH\_OUT/ENH\_IN

Both the ENH\_OUT and ENH\_IN records have the following additional mandatory fields (note: prior to ver1.0.1 these ENH\_OUT only):

Field	Type	Description
dup_group	string	duplicate grouping N - none S - site UHDV - user, host, display, vendor if V, then V:<vendor_code>
project	string	value of LM_PROJECT
sw_version	string	version of software checking out the feature

### Token System Fields - ENH\_OUT and ENH\_IN

The ENH\_OUT and ENH\_IN records can provide additional information for token based systems. These fields are used when the feature being checked out is a token feature and the user will be running some different actual feature.

In non-token systems these fields would be empty strings.

Field	Type	Description
actual_feature	string	name of actual software feature used for token feature checked out
actual_feature_count	int	if checkout is denied, name of next token pool to be tried for checkout attempt

### Additional Mandatory Fields - ENH\_IN

The ENH\_IN records have the following additional mandatory fields:

Field	Type	Description
linger	int	linger period, in seconds

### Server Initialization Callback : ls\_user\_init3()

The Flex API provides several callbacks to the vendor daemon during server initialization. Of these, the ls\_user\_init3() callback is provided at a time after the processing of the license files both at startup and on re-read.

The ENH function enh\_init() should be called during this callback. If the vendor is currently using this callback then their existing function should call enh\_init(), and if the callback is not in use then the enh\_init() function can be assigned directly to the callback.

The enh\_init() function determines whether the dup\_sel value has been set for each FEATURE in the license file, and if so what its value is. This information is passed on the ENH\_OUT records.



During the `enh_init()` routine the `ENH_VERSION` record is output as is the optional `ENH_SECURITY` record. Both of these are described later in this document.

### **Pre-Checkout Callback : `Is_outfilter()`**

The Flex API provides a callback to the vendor daemon prior to execution of the checkout (it is referred to as a pre-checkout filter).

Within this callback it is possible to log the intention to attempt a checkout, providing enhanced information about the `lc_checkout` call instance. Processing of subsequent standard entries in the debug log file by usage analysis software will determine if the attempt was successful (a checkout), unsuccessful (a denial) or is pending (queued).

The ENH function `enh_out()` should be called during this callback. If the vendor is currently using this callback then their existing function should call `enh_out()`, and if the callback is not in use then the `enh_out()` function can be assigned directly to the callback.

```
hh:mm:ss (daemon_name) ENH_OUT: "feature" user@host timestamp num_licenses
session_handle "display" addr "dup_group" "project" "sw_version"
"actual_feature" "actual_feature_count"
```

### **Pre-Checkin Callback : `Is_infilter()`**

The Flex API provides callbacks to the vendor daemon both before and after execution of the `lc_checkin`. The ENH Records specification makes use of the pre-checkin callback, `Is_infilter()` as it is the only one guaranteed to have access to the session handle (if lingering is used the post checkin callback, `Is_incallback()`, does not have the session handle).

The ENH function `enh_in()` should be called during this callback. If the vendor is currently using this callback then their existing function should call `enh_in()`, and if the callback is not in use then the `enh_in()` function can be assigned directly to the callback.

```
hh:mm:ss (daemon_name) ENH_IN: "feature" user@host timestamp num_licenses
session_handle "display" addr linger "dup_group" "project" "sw_version"
"actual_feature" "next_token_pool"
```

## **ENH Record Integrity**

As the debug log is an ASCII file, and ENH records may be used for both compliance verification and pay-per-use analysis by software vendors, a mechanism to validate log file integrity is provided. The use of this mechanism is optional, an ENH Record enhanced debug log file containing no security records is considered complete and is absolutely adequate for enterprise user usage analysis needs as well as vendor needs in trusted environments.

The security mechanism is accomplished with two additional ENH records:

- `ENH_SECURITY`
- `ENH_DIGEST`

The `ENH_SECURITY` record indicates that security is being used, which digesting technique is employed. The `ENH_SECURITY` record shall be output in the `enh_init()` routine within the `lm_user_init3()` callback. If security is not being used, it will not be output.

```
hh:mm:ss (daemon_name) ENH_SECURITY: "digest_method"
```



The ENH\_DIGEST record provides an obfuscated digest of all ENH records since the previous ENH\_SECURITY or ENH\_DIGEST record. Log file validation is achieved by creating a validation digest and comparing its contents against the decrypted digest.

The ENH\_DIGEST records are output on one of the following three conditions:

- once a specified amount of data exists in the enh\_digest\_buffer
- once a specified number of ENH records have been written since the previous digest
- during enh\_init() if there is data in the enh\_digest\_buffer

```
hh:mm:ss (daemon_name) ENH_DIGEST: "digest"
```

The data in a digest is obfuscated by the concatenation of a vendor specific pass phrase (selectable to be placed either at the front or the end of the buffer). A validation tool will be built that will accept as input a log file, the pass phrase and a flag indicating whether prepend or append of the pass phrase is used..

## Additional Logging

### Mandatory - ENH Version

In the enh\_init() function during vendor daemon initialization an ENH\_VERSION record shall be output indicating the specification version being adhered to for all other ENH records.

Field	Type	Description
enh_version	string	version of the ENH Records specification

```
hh:mm:ss (daemon_name) ENH_VERSION: enh_version
```

### Optional - Vendor Specific Data

Software vendors may choose to include additional information for other purposes in the debug log file. A generic ENH record, ENH\_VENDOR, is provided to include such items in a common name:value pair format.

Field	Type	Description
name	string	the name of a vendor specific data item
value	string	the associated value

```
hh:mm:ss (daemon_name) ENH_VENDOR: "name" "value"
```

## Source Code Integration

### C Language File Descriptions

The ENH Records source package contains all of the C source files necessary to integrate the ENH Records functionality into a Flex-based vendor daemon.



The package consists of three groups of files:

- ENH Records base functionality

These files contain the base functionality of generating ENH records. These will be updated with each new release of the ENH Records Specification.

```
enh_records.c  
enh_records.h  
get_time_milli.c
```

- Vendor Customizable files

These two files contain all of the variables and functions that must (or can) be modified by the software vendor for: vendor identification (required), use of ENH security records (optional), use of token pools (optional).

```
enh_vendor.c  
enh_vendor.h
```

- MD5 digest generation

The ENH Records package makes use of the md5 software developed by L. Peter Deutsch and copyright Aladdin Enterprises.

```
md5.c  
md5.h
```

### **Integrating the Source with Your Vendor Daemon Source**

- 1) Copy the ENH source files to the machind directory in your FLEXIm development directory.
- 2) Review the section at the top of `enh_vendor.c` and `enh_vendor.h` labelled 'Customizations Required by ISVs', and make any necessary changes.
- 3) Existing vendor daemon source changes
  - A. If you are not currently using callbacks in your vendor daemon

Edit the file `machind/lsvendor.c`

- i) Add the following line just below the last `#include` line at the top of the file:  
`#include "enh_records.h"`

- ii) Make these changes:

```
From: void (*ls_user_init3)() = 0;  
To: void (*ls_user_init3)() = enh_init;
```

```
From: int (*ls_outfilter)() = 0;  
To: int (*ls_outfilter)() = enh_out;
```

```
From: int (*ls_infilter)() = 0;  
To: int (*ls_infilter)() = enh_in;
```



B. If you already are using callbacks

- i) In each of your current callback source files add the following below the last #include line

```
#include "enh_records.h"
```

4) Edit the makefile supplied by Acreso (in each platform directory)

A. Change the following line:

From: XTRAOBJS =

To: XTRAOBJS = enh\_records.o get\_time\_milli.o enh\_vendor.o md5.o (UNIX)

or: XTRAOBJS = enh\_records.obj get\_time\_milli.obj enh\_vendor.obj obj md5.obj (Windows)

B. Add these lines at the bottom of the makefile:

UNIX

```
enh_records.o: $(SRCDIR)/enh_records.c $(SRCDIR)/enh_records.h  
$(CC) -c $(CFLAGS) $(SRCDIR)/enh_records.c
```

```
get_time_milli.o: $(SRCDIR)/get_time_milli.c  
$(CC) -c $(CFLAGS) $(SRCDIR)/get_time_milli.c
```

```
enh_vendor.o: $(SRCDIR)/enh_vendor.c $(SRCDIR)/enh_records.h $(SRCDIR)/enh_vendor.h  
$(CC) -c $(CFLAGS) $(SRCDIR)/enh_vendor.c
```

```
md5.o: $(SRCDIR)/md5.c $(SRCDIR)/md5.h  
$(CC) -c $(CFLAGS) $(SRCDIR)/md5.c
```

WINDOWS:

```
enh_records.obj: $(SRCDIR)/enh_records.c $(SRCDIR)/enh_records.h  
$(CC) -c $(CFLAGS) $(SRCDIR)/enh_records.c
```

```
get_time_milli.obj: $(SRCDIR)/get_time_milli.c  
$(CC) -c $(CFLAGS) $(SRCDIR)/get_time_milli.c
```

```
enh_vendor.obj: $(SRCDIR)/enh_vendor.c $(SRCDIR)/enh_records.h $(SRCDIR)/enh_vendor.h  
$(CC) -c $(CFLAGS) $(SRCDIR)/enh_vendor.c
```

```
md5.obj: $(SRCDIR)/md5.c $(SRCDIR)/md5.h  
$(CC) -c $(CFLAGS) $(SRCDIR)/md5.c
```

### Customizations Required by ISVs

- 1) In the "record prefixes" section [enh\_vendor.h]
  - set the value of daemon
- 2) Token systems only [enh\_vendor.c]
  - update the code in the set\_token\_fields() function
- 3) If security records are to be used
  - in the "security digest variables and definitions" [enh\_vendor.h]



set do\_security to TRUE  
set the value of obfuscate\_end to OBF\_FRONT or OBF\_BACK  
set the value of digest\_type (currently Adler32 and MD5 are provided)  
set the values of MAX\_ENH\_DIGEST\_LENGTH and  
MAX\_ENH\_DIGEST\_RECORDS

- update the create\_pass\_phrase() function in enh\_vendor.c
- 4) If vendor code duplicate grouping is used [enh\_vendor.c]
- modify the function append\_vendor\_string() if you want the code to be obfuscated in the ENH records, otherwise it will be output in plain text

#### **Enabling the output of ENH records**

Prior to starting the vendor daemon on the license server, set the environment variable <daemon\_name>\_ENH\_RECORDS to the value 1. If this environment variable is not present or set to a value other than 1 then ENH records will not be output.

