FDMS Development



**FIRST**

**DATA**

**CORP.**



Event Logger Subsystem

Version 1.0

Author : Erico Bueno

Contents

[FIRST 1](#_Toc433433320)

[1. Introduction 2](#_Toc433433321)

[2. Architecture 2](#_Toc433433322)

[2](#_Toc433433323)

[3. Files and Directories 3](#_Toc433433324)

[4. Configuration Files 3](#_Toc433433325)

[4.2.1. The Config file: 4](#_Toc433433326)

[4.2.2. The LBS file 5](#_Toc433433327)

[5. Utilities 6](#_Toc433433328)

[5.1. Event file dump utility – evt\_dump 6](#_Toc433433329)

[5.2. Event logger SFF read utility – evt\_read\_sff 6](#_Toc433433330)

[5.3. Disposition logs dump utility – evt\_tool 6](#_Toc433433331)

[6. Control 8](#_Toc433433332)

[7. Alarms 8](#_Toc433433333)

1. Introduction

The event logger subsystem, aka, the Authorization Capture system, is a system designed to capture, store and forward event records to the HDC mainframe. At the time of this writing, the only event record planned is the authorization billing record, or authorization record.

Events are variable length records with a type and are formatted using the ISO+ format. Each event type has it own set of fields that have meaning only to the applications sending and receiving it. The event logger does not interpret event records, but only stores and forwards them. The job of the event logger is to store and deliver the records reliably to the HDC mainframe.

Events are sent by client applications to the event logger. The event logger (evtlog process) stores the event on an event file and queues an event header into a store and forward queue (SFF).

The event dispatcher (ai\_evt process) dequeues event headers from the SFF, retrieves the event records from the event files, bundles the event records into packages and forwards the packages to the HDC main frame.

1. Architecture

The figure below shows the basics of the event logger subsystem.

The event logger (evtlog) process receives the event records from multiple client applications like cnss and others. Currently, cnss will be the only client utilizing the event logger to deliver billing records to the HDC mainframe.

EVTLOG

AI\_EVT

CNSS

HDC

Mainframe

SFF

Some

Client

Authorization Events

Event files

EVTLOG

AI\_EVT

CNSS

SFF

Some other

Client

Event files

Disposition logs

good.events.xx.xx.xx

bad.events.xx.xx.xx

evtfile.xx.xx.xx.xx.xx

evt.sff.0.04.07.08.00.00.02

The event dispatcher (ai\_evt) process accepts TCP/IP connections from the HDC mainframe via two sets of lines. The primary lines are the preferred option to deliver the events. If no primary lines are available, then the ai\_evt process will deliver via the secondary lines. When there are multiple lines of the same type (primary, secondary), the ai\_evt process will alternate sending events via all lines of the same type (Round Robin).

1. Files and Directories
2. Event Files

Event files are stored in the primary billing directory and contain variable length event data which is formatted in the ISO+ format. They are also stored in the secondary billing directory if the mirror option is on for the evtlog process. Event files are named as **evtfile.04.07.07.09.33.24.00**.

1. Disposition logs ( Bad Event, Good Event )

The disposition log contains the event header loaded with the final processing status obtained during the delivery of the event. Event status are separated by good and bad status and recorded in the bad or good disposition logs. Disposition logs are named as: **good.events.0.04.07.10.13.23.15.0** and **bad.events.0.04.07.10.13.23.15.0.**

1. Store and forward file (SFF)

The SFF file contains event headers queued by the evtlog process. The ai\_evt process dequeue event headers from this file and initiates the process of delivering the correspondent events to the HDC mainframe. The SFF file resides in the billing directory, the control file for the SFF queue resides in the ctrl subdirectory. The names are as follows: **evt.sff.0.04.07.10.00.00.00** and **evt.ctrl**

1. Configuration Files
   1. Event Logger Process (evtlog)

The configuration file for the evtlog process is named evtlog.cfg and is located in the config directory. The parameters are as follows:

# evtlog.cfg

#

# 06/10/2004 NSH061011 EB Initial Release

#

# Processor id identifies from which system the event came from

processor\_id 01

#

# Maximum Event file size in Kilobytes

max\_evt\_file\_size 2048

#

# Mirror on creates copy of event files in secondary directory

mirror off

* 1. Event Dispatcher Process (ai\_evt)

The event dispatcher process utilizes one configuration file located in the config directory named ai\_evt.cfg and an lbs file located in the data directory named ai\_evt.lbs. The contents are as follows:

* + 1. The Config file:

#######################################################

# ai\_evt.cfg

#

# 06/10/04 NSH061011 EB Initial Release

#

#######################################################

read\_lblcfg ai\_evt.lbs

cics\_tran\_id PRI DCA3

system\_id PRI SND

ip PRI 000.000.000.000

port PRI 00000

pwd PRI AUTH

cics\_tran\_id SEC DCA3

system\_id SEC SND

ip SEC 000.000.000.000

port SEC 00000

pwd SEC AUTH

route\_id EB01

settle\_type RISCAUTH

# Check for connections to HDC host every

check\_connections 5

# ai\_evt sends ruthere anytime there is no activity in a HDC line for :

max\_inactivity 120

# When we do send a ruthere, we expect a reply within:

ruthere\_timeout 10

# ai\_evt disconnects the line and starts connection over if this many ruthere

# goes without a reply from HDC:

outstanding\_ruthere 2

# Check for queued events in the SFF this often:

fsm\_interval 4

# ai\_evt expects a reply from HDC within the event\_timeout interval when it

# sends an event window to HDC. If a reply is not received, the event window is

# is marked as RETRAN and ai\_evt reattempts the delivery up to event\_retry.

# The ai\_evt send a challenge to HDC before it resends the window

# The window is marked as TIMEOUT after it is reattempted event\_retry times

window\_timeout 30

max\_window\_retry 3

# When a window timeouts, ai\_evt sends a challenge to HDC. If ai\_evt receives a

# positive reply to the challenge, it will resend the window. If HDC does respond

# to the challenge, ai\_evt will resend up to max\_challenges to HDC seeking a reply.

# After the max number of challenges has been attempted unsuccessfully, ai\_evt will

# hang up the TCP/IP session and re-route the window to the next available line, and

# again send a challenge to HDC, repeating the process. The window is marked as

# TIMEOUT if the max number of challenges is expired or the max\_retry is exhausted.

challenge\_timeout 10

max\_challenge 2

suspect\_trace off

object\_logging parse on

object\_logging format on

#######################################################

* + 1. The LBS file

#----------------------------------------------

# ai\_evt.lbs

#

# 02/25/04 NSH061011 EB Initial Release

#

#----------------------------------------------

# Primary System - Support multiple Lines

system EVP

device EVP1 EVT\_PRI\_LINE1 TCP/IP \

rule Standard \

length ebcdic5

client default

server default

system EVS

device EVS1 EVT\_SEC\_LINE1 TCP/IP \

rule Standard \

length ebcdic5

client default

server default

#----------------------------------------------

1. Utilities
   1. Event file dump utility – evt\_dump

This utility is used to view the event data in the event file. Its usage is as follows:

Usage: evt\_dump [-d] [-h] evtfile

-h Print Header Only (Default)

-d Print Header and Details

* 1. Event logger SFF read utility – evt\_read\_sff

This utility is used to read the event headers queued in the SFF. Its usage is as follows:

evt\_read\_sff [-h] [-n] [-s] [-v] [-z] [ -c | <sff>]

[-c] Use evt.ctrl file to locate SFF file and current record offset

[-h] Display help

[-n] Do not display statistics (def=off)

[-s] Display statistics only (def=off)

[-d] Print details of each header record (def=off)

[-z] Read SFF file from zero offset. Only valid with '-c' option

<sff> Store-and-Forward Queue file to open

***Note: Command line arguments for this routine are likely to change in Release 4.***

* 1. Disposition logs dump utility – evt\_tool

This utility is used to read the final disposition status of events sent to the HDC Mainframe. It is also used to create a replay file used to resubmit failed events back into the system.

The evt\_tool utility has a built-in query interpreter that allows the user to specify query criteria to select records to be viewed or replayed. If no query file is specified or the query file is empty, the tool will select all records on the given disposition file.

The usage is a follows:

Usage: -----------------------------------------------------------------------

Queries and performs action on selected events.

All events are selected if query option not entered or query file is empty.

All options OFF by default. Print Stats Only.

evt\_tool [-d] [-h] [-p] [ q query\_file] [-r] disposition\_file

-d -> Print Headers and Details

-h -> Print Headers

-p -> Print query sample and exit

-q file -> Execute query

-r -> Extract Replay File

------------------------------------------------------------------------------

The query file sample can be printed, in case you forgot the syntax, by utilizing the –p argument. The query file looks as follows:

$evt\_tool –p

type AUTHORIZATION

type DOWNLOAD

type PARAM

type TRACKER

type TYPE4

type TYPE5

header ORIGINAL

header REPLAYED

source cnss

disposition EVT\_LOGGED

disposition EVT\_CAPTURED

disposition EVT\_TIMEOUT

disposition EVT\_DECLINE

disposition EVT\_ERROR\_SENDING

refno 0110851480910012

response\_code 390

response\_code 660

output\_label EVP1

version 4 # GE than 4

start\_date 10/12/2003 15:31:04

end\_date 10/13/2003

1. Control

A new entry for the evtlog process was added to the main control menu under EventLo**g,** with the letter “g” highlighted.

Commands for the ai\_evt process follow the same rules as other ai\_applications. To enter a command, select “**f**” for application\_interface and then select 7 for the ai\_evt process.

1. Alarms

The ai\_evt process will generate an external MARS alarm for each event dispatched to HDC that is declined. Since this should be a very unusual situation, intervention from RISCFES, FLHDC and RISCAPPL groups may be required.

The text of the alarm is as follows:

***ai\_evt: Too many Declines. Page RISCFES***

All other abnormal conditions such as lines down, queue depth to high and others will still be detected by chief and alarms will be fired accordingly.