Command Logging in EntireX

Command logging is a feature to assist in debugging Broker ACI applications. A command in this context represents one user request sent to the Broker and the related response of Broker.

Command logging is a feature that writes the user requests and responses to file in a way it is already known with Broker trace and `TRACE-LEVEL=1`. But command logging works completely independent from trace, and data is written to a file only if defined command trace filters detect a match.

Broker stub applications send commands or requests to the Broker kernel, and the Broker kernel returns a response to the requesting application. Developers who need to resolve problems in an application need access to those request and response strings inside the Broker kernel. That’s where command logging comes in. With command logging, request and response strings from or to an application are written to a file that is separate from the Broker trace file.

This chapter covers the following topics:

- Introduction to Command Logging
- Command Log Filtering using System Management Hub
- Command Log Filtering using Command-line Interface ETBCMD
- ACI-driven Command Logging
- Dual Command Log Files

Introduction to Command Logging

This section provides an introduction to command logging in EntireX and offers examples of how command logging is implemented. It covers the following topics:

- Overview
- Command Log Files
- Defining Filters
- Programmatically Turning on Command Logging

Overview

Command logging is similar to a Broker trace that is generated when the Broker attribute `TRACE-LEVEL` is set to 1. Broker trace and command logging are independent of each other, and therefore the configuration of command logging is separate from Broker tracing.

The following Broker attributes are involved in command logging:

Attribute	Description
CMDLOG	Set this to "N" if command logging is not needed.
CMDLOG-FILE-SIZE	A numeric value indicating the maximum size of command log file in KB.
NUM-CMDLOG-FILTER	The maximum number of filters that can be set.

In addition to CMDLOG=YES, the Broker needs the assignment of the dual command logging files during startup. If these assignments are missing, Broker will set CMDLOG=NO. See also *Broker Attributes*.

Command Log Files

The Broker keeps a record of commands (request and response strings) in a command log file.

At Broker startup, you will need to supply two command log file names and paths. Only one file is open at a time, however, and the Broker writes commands (requests and responses) to this file.

Under UNIX and Windows, the startup options -y and -z are evaluated by executable etbnuc. These options are used to specify the command log file names. Startup script/service assign these files by default.

Under z/OS, the file requirements are two equally sized, physical sequential files defined with a record length of 121 bytes, i.e.
DCB=(LRECL=121,RECFM=FB,BLKSIZE=nnnn). We recommend you allocate files with a single (primary) extent only. For example SPACE=(CYL,(30,0)). The minimum file size is approximately 3 cylinders of 3390 device. Alternatively, the dual command log files can be allowed in USS HFS file system.

When the size of the active command log file reaches the KB limit set by CMDLOG-FILE-SIZE, the file is closed and the second file is opened and becomes active. When the second file also reaches the KB limit set by CMDLOG-FILE-SIZE, the first file is opened and second file is closed. Existing log data in a newly opened file will be lost.

Defining Filters

In command logging, a filter is used to store and identify a class, server, or service, as well as a topic name and user ID.

Use the System Management Hub to define a filter. Under UNIX and z/OS you can also use command line tool etbcmd. During processing, the Broker evaluates the class, server, service, topic, and user ID associated with each incoming request and compares them with the same parameters specified in the filters. If there is a match, the request string and response string of the request is printed out to the command log file.

Programmatically Turning on Command Logging

Applications using ACI version 9 or above have access to the new field LOG-COMMAND in the ACI control block.

If this field is set, the accompanying request and the Broker's response to this request is logged to the command log file.

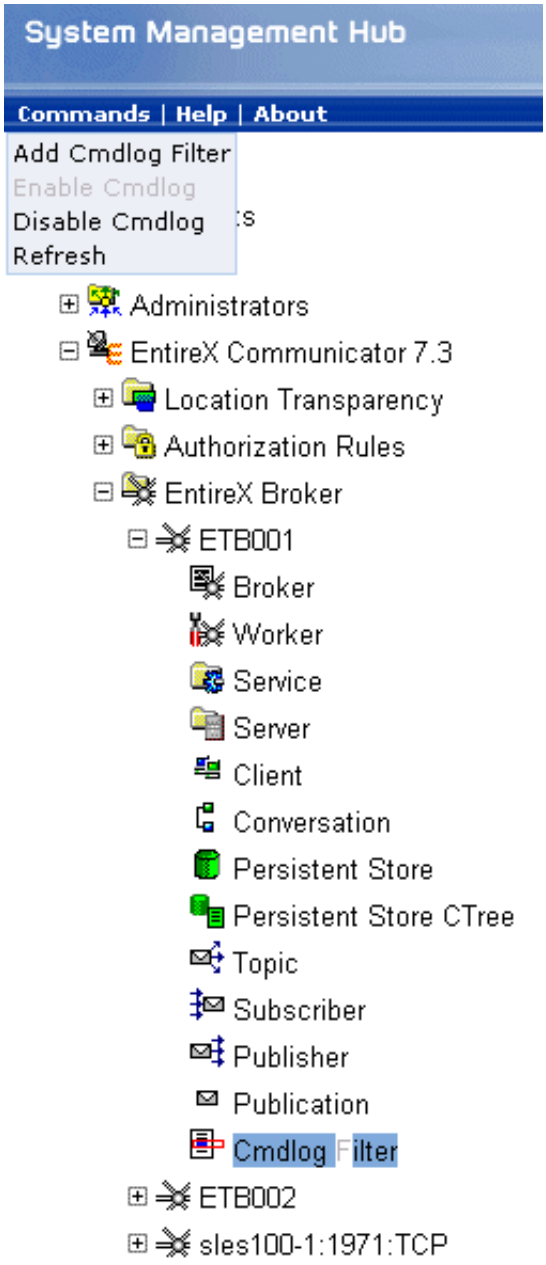
Note:
Programmatic command logging ignores any filters set in the kernel.

Command Log Filtering using System Management Hub

- Setting up your Environment
- Adding a Filter
- Managing Filters

Setting up your Environment

In order to process filters using System Management Hub, Broker attribute `CMDLOG` must be set to "YES" and the log files must be defined. See *Command Log Files* above. If this is the case, the **CmdlogFilter** node will be visible in the SMH tree.



Adding a Filter

To add a filter

1. In the SMH tree view, select the **CmdlogFilter** node and, with the context menu, choose **Add Cmdlog Filter**.
2. In the **Add Cmdlog Filter** screen, add values for User ID, Class/Server/Service or Topic. Confirm with **OK**.

Managing Filters

The following **Cmdlog Filter** screen shows four filters. Use this screen to

- delete a filter
- disable a filter
- enable a disabled filter

Cmdlog Filter (Global Cmdlog currently enabled)

20

Delete Button	Enable/Disable Button	User ID	Class/Server/Service	Topic	Enabled
Delete	Disable	USER_1	SAG/ETBCIS/SAGCCV5		Y
Delete	Disable	USER_1	SAG/ETBCIS/SAGCIV5		Y
Delete	Enable	USER_1	RCP/SAGCCV5/CALLNAT		N
Delete	Disable	USER_1	RPC/SAGCIV5/CALLNAT		Y

Items 1 to 4 of 4

Note:
You cannot change the values for User ID, Class/Server/Service or Topic in the **Cmdlog Filter** screen. Instead, delete the command log filter and add a new one with the required values.

Command Log Filtering using Command-line Interface ETBCMD

The examples assume that Broker has been started with the attribute `CMDLOG=Y`.

- Setting Filters
- Deleting Filters
- Disabling and Enabling a Filter

Setting Filters

Filters need to be set before running the stub applications whose commands are to be logged.

UNIX

Command	Description
`etbcmd -blocalhost:1970:TCP -cSET-CMDLOG-FILTER -dBROKER -xuser -nACLASS/ASERVER/ASERVICE`	This command sets filters on `ACLASS/ASERVER/ASERVICE`. All ACI calls issued by *all* users to this service will be logged.
`etbcmd -blocalhost:1970:TCP -cSET-CMDLOG-FILTER -dBROKER -xuser -nACLASS/ASERVER/ASERVICE -Usaguser1`	This command set filters on `ACLASS/ASERVER/ASERVICE` and user ID `saguser1`. All ACI calls to this service *as well as* those issued by `saguser1` will be logged.
`etbcmd -blocalhost:1970:TCP -cSET-CMDLOG-FILTER -dBROKER -xuser -TNYSE -Usaguser1`	This command set filters on topic `NYSE` and user ID `saguser1`. All ACI calls to this topic *as well as* those issued by `saguser1` will be logged.

z/OS

Command	Description

`//ETBCMD EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP -cSET-CMDLOG-FILTER -xuser ',`
`//  '-dBROKER -nACLASS/ASERVER/ASERVICE')`
	This command sets filters on `ACLASS/ASERVER/ASERVICE`. All ACI calls issued by *all* users to this service will be logged.

`//ETBCMD EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP -cSET-CMDLOG-FILTER -xuser ',`
`//  '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')`
	This command sets filters on `ACLASS/ASERVER/ASERVICE` and user ID `saguser1`. All ACI calls to this service *as well as* those issued by `saguser1` will be logged.

`//ETBCMD EXEC PGM=ETBCMD,`
`// PARM=('/-blocalhost:1970:TCP -cSET-CMDLOG-FILTER -xuser ',`
`//  '-dBROKER -TNYSE -Usaguser1')`
	This command sets filters on topic `NYSE` and user ID `saguser1`. All ACI calls to this topic *as well as* those issued by `saguser1` will be logged.

Note:
If more than one service or topic is set as a filter, all ACI calls sent to any of these services or topics will be logged. Identical filters cannot be set. Attempts to set a second filter that matches an existing filter will be rejected. Similarly, the maximum number of filters that can be added is defined in `NUM-CMDLOG-FILTER`. If the maximum number of filters is already being used, delete an existing filter to make room for a new filter.

Deleting Filters

The following provides an example of how to delete an existing filter on a service.

To delete a filter

- Enter the following command.

Under UNIX:

```
etbcmd -d BROKER -b localhost:1970:TCP -c CLEAR-CMDLOG-FILTER -nACLASS/ASERVER/ASERVICE -U saguser1
```

Under z/OS:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cCLEAR-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE')
```

If the filter does not exist, the command will return an error.

Disabling and Enabling a Filter

Filters can be set and still be disabled (made inactive).

To disable a filter

- Enter the following command.

  Under UNIX:

```
etbcmd -blocalhost:1970:TCP -cDISABLE-CMDLOG-FILTER -dBROKER -xuser -nACLASS/ASERVER/ASERVICE -Usaguser1
```

  Under z/OS:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cDISABLE-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')
```

  Note:
  A disabled filter will not bring down the count of filters in use.

To enable a filter

- Enter the following command to enable the disabled filter.

  Under UNIX:

```
etbcmd -blocalhost:1970:TCP -cENABLE-CMDLOG-FILTER -dBROKER -xuser -nACLASS/ASERVER/ASERVICE -Usaguser1
```

  Under z/OS:

```
//ETBCMD  EXEC PGM=ETBCMD,
// PARM=('/-blocalhost:1970:TCP -cENABLE-CMDLOG-FILTER -xuser ',
//       '-dBROKER -nACLASS/ASERVER/ASERVICE -Usaguser1')
```

ACI-driven Command Logging

EntireX components that communicate with Broker can trigger command logging by setting the field `LOG-COMMAND` in the ACI control block.

When handling ACI functions with command log turned on, Broker will not evaluate any filters. Application developers must remember to reset the `LOG-COMMAND` field if subsequent requests are not required to be logged.

Dual Command Log Files

Broker's use of two command log files prevents any one command log file from becoming too large.

When starting a Broker with command log support, you must therefore specify two file names and paths - one for each of the two command log files. The sample startup script installed with the product uses the variables `ETB_CMDLOG1` and `ETB_CMDLOG2` as the default command log file names.

Under UNIX, the startup script uses file names `CMDLOGR1` and `CMDLOGR2`.

Under Windows, the keys `ETB_CMDLOG1` and `ETB_CMDLOG2` are entered in the Registry with values `CMDLOGR1` and `CMDLOGR2`.

At startup, Broker initializes both files and keeps one of them open. Command log statements are printed to the open file until the size of this file reaches the value specified in the Broker attribute `CMDLOG-FILE-SIZE`. This value must be specified in KB.

When the size of the open file exceeds the value specified in the Broker attribute `CMDLOG-FILE-SIZE`, Broker closes this file and opens the other, dormant file. Because the Broker closes a log file only when unable to print out a complete log line, the size of a *full* file may be smaller than `CMDLOG-FILE-SIZE`.

To switch log files on demand, using `etbcmd` | `ETBCMD`

- An open command log file can be forcibly closed even before the size limit is reached. Enter the following command.

  Under UNIX:

  ```
  etbcmd -blocalhost:1970:TCP -cSWITCH-CMDLOG -dBROKER -xuser
  ```

  Under z/OS:

  ```
  //ETBCMD  EXEC PGM=ETBCMD,
  // PARM=('/-blocalhost:1970:TCP -cSWITCH-CMDLOG -xuser ',
  //       '-dBROKER')
  ```

  The command above will close the currently open file and open the one that has been dormant.