# EEM Utility Tcl Command Extensions

The following conventions are used for the syntax documented on the Tcl command extension pages:

- An optional argument is shown within square brackets, for example:

[type ?]

- A question mark ? represents a variable to be entered.
- Choices between arguments are represented by pipes, for example:

priority low|normal|high

**Note** For all EEM Tcl command extensions, if there is an error, the returned Tcl result string contains the error information.

**Note** Arguments for which no numeric range is specified take an integer from -2147483648 to 2147483647, inclusive.


- appl_read, on page 2
- appl_reqinfo, on page 2
- appl_setinfo, on page 3
- counter_modify, on page 4
- description, on page 5
- fts_get_stamp, on page 6
- register_counter, on page 7
- register_timer, on page 8
- timer_arm, on page 10
- timer_cancel, on page 11
- unregister_counter, on page 12

# appl_read

Reads Embedded Event Manager (EEM) application volatile data. This Tcl command extension provides support for reading EEM application volatile data. EEM application volatile data can be published by a Cisco software process that uses the EEM application publish API. EEM application volatile data cannot be published by an EEM policy.

**Note** Currently there are no Cisco software processes that publish application volatile data.

### Syntax

```
appl_read name ? length ?
```

### Arguments

| | |
|---|---|
| name | (Mandatory) Name of the application published string data. |
| length | (Mandatory) Length of the string data to read. Must be an integer number between 1 and 4294967295, inclusive. |

### Result String

```
data %s
```

Where data is the application published string data to be read.

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

# appl_reqinfo

Retrieves previously saved information from the Embedded Event Manager (EEM). This Tcl command extension provides support for retrieving information from EEM that has been previously saved with a unique

key, which must be specified in order to retrieve the information. Note that retrieving the information deletes it from EEM. It must be resaved if it is to be retrieved again.

### Syntax

```
appl_reqinfo key ?
```

### Arguments

| | |
|---|---|
| key | (Mandatory) The string key of the data. |

### Result String

```
data %s
```

Where data is the application string data to be retrieved.

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

# appl_setinfo

Saves information in the Embedded Event Manager (EEM). This Tcl command extension provides support for saving information in the Embedded Event Manager that can be retrieved later by the same policy or by another policy. A unique key must be specified. This key allows the information to be retrieved later.

### Syntax

```
appl_setinfo key ? data ?
```

### Arguments

| | |
|---|---|
| key | (Mandatory) The string key of the data. |
| data | (Mandatory) The application string data to save. |

### Result String

None

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  8)    FH_EDUPLICATEKEY  (duplicate appl info key)
```

This error means that the application event detector info key or other ID was a duplicate.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 34)    FH_EMAXLEN  (maximum length exceeded)
```

This error means that the object length or number exceeded the maximum.

```
(_cerr_sub_err = 43)    FH_EBADLENGTH  (bad API length)
```

This error means that the API message length was invalid.

# counter_modify

Modifies a counter value.

### Syntax

```
counter_modify event_id ? val ? op nop|set|inc|dec
```

### Arguments

| | |
|---|---|
| event_id | (Mandatory) The counter event ID returned by the **register_counter** Tcl command extension. Must be an integer between 0 and 4294967295, inclusive. |
| val | (Mandatory)<br>**Note** Mandatory except when the op nop argument value combination is specified.<br>• If op is set, this argument represents the counter value that is to be set.<br>• If op is inc, this argument is the value by which to increment the counter.<br>• If op is dec, this argument is the value by which to decrement the counter. |

| | |
|---|---|
| op | (Mandatory)<br>• nop--Retrieves the current counter value.<br>• set--Sets the counter value to the given value.<br>• inc--Increments the counter value by the given value.<br>• dec--Decrements the counter value by the given value. |

### Result String

```
val_remain %d
```

Where val_remain is the current value of the counter.

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 30)    FH_ECTBADOPER  (bad counter threshold operator)
```

This error means that the counter event detector set or modify operator was invalid.

# description

Provides a brief description of the registered policy.

### Syntax

```
description ?
```

### Arguments

| | |
|---|---|
| line | (Optional) Brief description of the policy consisting of 1 to 240 characters. |

### Result String

None

### Set _cerrno

Yes

### Sample Usage

The description statement is entered by the author of the policy. It can appear before or after any event registration statement in Tcl. The policy can have only one description.

**Note** Registration of a policy with more than one description statement will fail.

The following example shows how a brief description is provided for the **event_register_syslog** policy:

```
::cisco::eem::description "This Tcl command looks for the word count in syslog messages."
::cisco::eem::event_register_syslog tag 1 ...
::cisco::eem::event_register_snmp_object tag 2 ...
::cisco::eem::trigger {
       ::cisco::eem::correlate event 1 and event 2
       ::cisco::eem::attribute tag 1 occurs 1
       ::cisco::eem::attribute tag 2 occurs 1
}
```

# fts_get_stamp

Returns the time period elapsed since the last software boot. Use this Tcl command extension to return the number of nanoseconds since boot in an array "nsec nnnn" where nnnn is the number of nanoseconds.

### Syntax

```
fts_get_stamp
```

### Arguments

None

### Result String

```
nsec %d
```

Where nsec is the number of nanoseconds since boot.

### Set _cerrno

No

# register_counter

Registers a counter and returns a counter event ID. This Tcl command extension is used by a counter publisher to perform this registration before using the event ID to manipulate the counter.

### Syntax

```
register_counter name ?
```

### Arguments

| | |
|---|---|
| name | (Mandatory) The name of the counter to be manipulated. |

### Result String

```
event_id %d
event_spec_id %d
```

Where event_id is the counter event ID for the specified counter; it can be used to manipulate the counter by the **unregister_counter** or **counter_modify** Tcl command extensions. The event_spec_id argument is the event specification ID for the specified counter.

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

This error means that the request to register the specific event was made before the EEM event detector had completed its initialization.

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 10)    FH_ECORRUPT  (internal EEM API context is corrupt)
```

This error means that the internal EEM API context structure is corrupt.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 16)    FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

This error means that the context pointer that is used with each EEM API call is incorrect.

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

This error means that a control block address that was passed in the EEM API was incorrect.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED  (number of subscribers exceeded)
```

This error means that the number of timer or counter subscribers exceeded the maximum.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

This error means that the subscriber index was invalid.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# register_timer

Registers a timer and returns a timer event ID. This Tcl command extension is used by a timer publisher to perform this registration before using the event ID to manipulate the timer if it does not use the **event_register_timer** command extension to register as a publisher and subscriber.

### Syntax

```
register_timer watchdog|countdown|absolute|cron name ?
```

### Arguments

| | |
|---|---|
| name | (Mandatory) The name of the timer to be manipulated. |

### Result String

```
event_id %u
```

Where event_id is the timer event ID for the specified timer (can be used to manipulate the timer by the **timer_arm** or **timer_cancel** command extensions).

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

This error means that the request to register the specific event was made before the EEM event detector had completed its initialization.

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 10)    FH_ECORRUPT  (internal EEM API context is corrupt)
```

This error means that the internal EEM API context structure is corrupt.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 16)    FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

This error means that the context pointer that is used with each EEM API call is incorrect.

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

This error means that a control block address that was passed in the EEM API was incorrect.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED  (number of subscribers exceeded)
```

This error means that the number of timer or counter subscribers exceeded the maximum.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

This error means that the subscriber index was invalid.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# timer_arm

Arms a timer. The type could be CRON, watchdog, countdown, or absolute.

### Syntax

```
timer_arm event_id ? cron_entry ?|time ?
```

### Arguments

| | |
|---|---|
| event_id | (Mandatory) The timer event ID returned by the **register_timer** command extension. Must be an integer between 0 and 4294967295, inclusive. |
| cron_entry | (Mandatory) Must exist if the timer type is CRON. Must not exist for other types of timer. CRON timer specification uses the format of the CRON table entry. |
| time | (Mandatory) Must exist if the timer type is not CRON. Must not exist if the timer type is CRON. For watchdog and countdown timers, the number of seconds and milliseconds until the timer expires; for an absolute timer, the calendar time of the expiration time (specified in SSSSSSSSSS[.MMM] format, where SSSSSSSSSS must be an integer representing seconds between 0 and 4294967295, inclusive, and where MMM must be an integer representing milliseconds between 0 and 999). An absolute expiration date is the number of seconds and milliseconds since January 1, 1970. If the date specified has already passed, the timer expires immediately. |

### Result String

```
sec_remain %ld msec_remain %ld
```

Where sec_remain and msec_remain are the remaining time before the next expiration of the timer.

**Note** A value of 0 will be returned for the sec_remain and msec_remain arguments if the timer type is CRON.

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 27)    FH_ETMDELAYZR  (zero delay time)
```

This error means that the time specified to arm a timer was zero.

```
(_cerr_sub_err = 42)    FH_ENOTREGISTERED  (request for event spec that is unregistered)
```

This error means that the event was not registered.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# timer_cancel

Cancels a timer.

### Syntax

```
timer_cancel event_id ?
```

### Arguments

| | |
|---|---|
| event_id | (Mandatory) The timer event ID returned by the **register_timer** command extension. Must be an integer between 0 and 4294967295, inclusive. |

### Result String

```
sec_remain %ld msec_remain %ld
```

Where sec_remain and msec_remain are the remaining time before the next expiration of the timer.

**Note** A value of 0 will be returned for sec_remain and msec_remain if the timer type is CRON .

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

This error means that the event type specified in the internal event specification was invalid.

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

This error means that the application event detector info key or other ID was not found.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

This error means that the event ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.

# unregister_counter

Unregisters a counter. This Tcl command extension is used by a counter publisher to unregister a counter that was previously registered with the **register_counter** Tcl command extension.

### Syntax

```
unregister_counter event_id ? event_spec_id ?
```

### Arguments

| | |
|---|---|
| event_id | (Mandatory) Counter event ID returned by the **register_counter** command extension. Must be an integer between 0 and 4294967295, inclusive. |
| event_spec_id | (Mandatory) Counter event specification ID for the specified counter returned by the **register_counter**command extension. Must be an integer between 0 and 4294967295, inclusive. |

### Result String

None

### Set _cerrno

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

This error means that the operating system reported an error. The POSIX errno value that is reported with the error should be used to determine the cause of the operating system error.

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

This error means that an internal EEM request for memory failed.

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

This error means that the event specification ID could not be matched when the event was being registered or that an event detector internal event structure is corrupt.

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

This error means that an internal EEM event detector pointer was null when it should have contained a value.

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

This error means that the subscriber index was invalid.

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

This error means that the event detector was unavailable.

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

This error means that the EEM event detector that handles this request is not available.