

# TRACE V2.0

OpenVMS Console Monitoring Tool

## Trap and Act on Console Events

### Quick Links

Error messages .....	37
Filters .....	8
Installation instructions .....	40
Overview .....	4
Qualifier reference .....	19

## **Saiga Systems Software**

#215, 801 - 6th Street SW  
Calgary, Alberta, Canada  
T2P 3V8

Voice: 1-800-561-8876

Fax (403) 263-0744  
Telephone (403) 263-1151  
Internet sales@saiga.com  
support@saiga.com  
WWW <http://www.saiga.com/>  
FTP ftp.saiga.com  
Listserver trace@saiga.com

April 2000

The information in this document is subject to change without notice and should not be construed as a commitment by Saiga Systems Inc. Saiga Systems Inc. assumes no responsibility for any errors or omissions that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Copyright © 1999, 2000 by Saiga Systems Inc.  
All rights reserved

DECNET, VAX, OpenVMS, Alpha AXP and VAXCLUSTER are trademarks of:  
Compaq

## Table of Contents

Overview .....	4
How Trace Works .....	5
Getting Started .....	6
Filters .....	8
Default Record .....	13
Reference Section .....	14
Logical Names .....	14
Command procedures .....	16
Files .....	18
Qualifier Reference .....	19
/ACTION .....	20
/HISTORY .....	27
/LIST .....	28
/LOG .....	29
/OUTPUT .....	30
/RESEQUENCE .....	31
/START .....	32
/STOP .....	33
/STATISTICS .....	34
/VERSION .....	35
Appendix .....	36
A. Trace Account; recommended settings .....	36
B. Error Messages .....	37
C. Installing Trace .....	40

## Overview

This section gives a brief description of the Trace utility. Trace runs as a detached process on your system and monitors a logical LAT terminal that has been enabled to receive one or more classes of operator messages. Each time a message is received Trace wakes up and checks the message against its parameter file of message filters to determine if it should take one of two special actions; ignore the message or submit a batch job. Trace is easily configured for either of these two actions. It is also possible to control Trace by enabling the LAT terminal to receive only specific operator classes; for example Trace on NodeA in a cluster can watch for tape and central (operator) messages while Trace on NodeB is watching for security messages.

Trace is easy to maintain with a standard DCL interface and the detailed examples in this manual should enable you to quickly configure Trace to meet your sites specific requirements. Up to 256 filters may be added to the Trace parameter file; with the last one normally being set up to either ignore all messages that have not yet been filtered or to send all messages that have not been filtered, via the OpenVMS mail utility, to a specific VMS user who will review these messages to determine if additional filters should be added to the Trace parameter file.

Care must be taken when setting up Trace if any action procedures you create are used to send additional console messages. During testing we were able to create an endless loop by sending a console message that Trace trapped and submitted a batch job for; this batch job sent a console message that was then trapped by the same filter and the whole process repeated indefinitely. To avoid this problem console messages should be sent by any batch jobs submitted by Trace to operator classes that Trace is not monitoring. Trace will automatically shut down if it believes it has entered an endless loop (if it responds to more than a defined number of console messages within 1 minute; the default is 15 but this can be configured using a logical name).

Trace can also be used in a Decnet network that is not clustered to help centrally manage console events. A master node is designated and all other nodes that Trace is running on have a logical name defined to designate this node. The filters on the remote nodes are configured to take action or ignore any local messages that don't need to be forwarded and for console messages that should be forwarded use the `send_task_message` action procedure. This procedure forwards the message to a special procedure on the master node that rebroadcasts the message on that node. Trace's filters on the master node are configured to trap and take action on these critical events.

## How Trace Works

When console messages are sent in the OpenVMS environment they are broadcast to all the terminals on the system, or cluster, that have been enabled as operator terminals. Messages are usually sent to a specific class of operator so that the volume of messages received by a particular operator terminal can be controlled by enabling it to receive only specific classes.

Trace creates a logical terminal, A LAT device, and then enables it for the specific class(es) that you want it to monitor. A detached process is started that is linked to this terminal. Trace detects each time an operator messages is receive at the LAT terminal, it captures the text of the message and then compares that text to the user defined filters. These filters contain a match string to compare to the text of the message and instructions on what to do if a match is detected. For each match Trace will take one of two possible actions; it will ignore the message or act in response to it by submitting a batch job. If a message does not match any of the filters it is ignored.

# Getting Started

This section explains, in general terms, how to get Trace up and running. Appendix A `Installing Trace` on page 36 gives a detailed example of a Trace installation.

1. Install Trace on a system. If you plan to use Trace to filter messages from remote nodes on a central node you should install it on the central node first.
2. Install your license key for Trace. Cohort customers do not need to install a separate Trace license.
3. Do some initial configuration. This should include:
  1. Edit the `trace_com:define_opcom_class.com` procedure and define which operator classes should be monitored. The default will be to monitor all operator classes.
  2. Edit the `trace_com:create_lat_device.com` procedure and change the device name from `LAT1000:` to a unique device. This step only needs to be done IF you are currently using the `LAT1000:` device for another purpose. To check this:  
MCR LATCP  
LATCP> Show port lat1000:  
LATCP> exit  
If there is no `lat1000` port no change is necessary. If there is change the port to a different value after checking to make sure it is not used already.
  3. It is recommended that an initial filter be created that will match all console messages (`match="*"`) and run the `trace_com:sample_command.com` procedure to e-mail these messages to the user `SYSTEM`.
4. If you are installing on the “Master” node you must define proxies for the trace account for each remote node that will be forwarding message to this node. These proxies must be created with the `/default` qualifier to work properly.
5. Start the trace monitor detached process:  
`@sys$manager:start_trace.com`
6. Check VMS mail under the username that will be receiving the forwarded console messages. For each message determine if this message is significant enough to require a filter. It is also important to ask yourself if the message will occur often enough to require a filter. To decide how to treat this message answer these questions.

Is this message significant enough to require a filter?

1. If “Yes” then is the message something that should be acted on?
  1. If “Yes” then add a filter with the appropriate actions
  2. If “No” then add an ignore filter
2. If “No” then does this message occur on a regular basis?
  1. If “Yes” then add an ignore filter to help limit the amount of VMS

- mail generated by trace
3. If “No” then is the message safe to ignore?
    1. If “Yes” then add an ignore filter
    2. If “No” then do nothing. If a final event has been added that forwards all messages not yet filtered Trace will forward the message to a user via mail.
  7. Repeat steps 4 and 5 on a regular basis until the only mail being received by the Trace administration account from the detached monitor is unusual messages that warrant review.
  8. Periodically check the statistics from the detached monitor. This will tell you how many matches have occurred against each filter and how many messages, if any, did not match any filter and were ignored. If there are filters that have had no matches it is important to review them and either delete them or update their match string so that the filter will once again start responding to the appropriate console message(s).

# Filters

At the heart of Trace is a parameter file that contains up to 256 filters. Each console message that Trace detects is compared, sequentially, to each of these filters until a match is found for all the filters have been checked. A default filter exists that can be modified; when adding new filters all values that are not specified in the actual DCL command will be taken from this default filter.

A filter contains the following:

ID number	A user assigned ID number for the filter. Trace compares against the filters in sequential order by this number.
Match string	The string to compare the console message against when checking for a match.
Action type	The action type for a filter will be either Action (respond to a match by submitted a batch job) or Ignore (ignore this console message).
Action procedure	The batch job that is to be submitted if a message matches this filter and the filter has not been configured to ignore the message.
Range	A numeric range. The action will be taken if Traces`counter for how many messages have matched this filter falls within this range.
Reset after	A delta time that tells Trace how often it should reset the counter for matches against this filter back to 0. In combination with range this controls how often Trace will respond to a particular filter.
History	This controls whether Trace logs all matches against a filter, all batch job submissions in response to these matches or both matches and submissions.
Keep, Print	Together these control whether a log file is created for an action procedure that has been submitted to batch and whether or not the log file will be printed or kept.
Node	If a node is listed Trace will only respond to a match against this filter if it is running on the specified node. If node is blank Trace will respond to any match.

The following pages provide several examples of Trace filters with detailed explanations and the DCL commands that would be used if you were adding that filter.

```
Sort ID      :      5
Type        : IGNORE
Match       : *hitman*warned user*
Case sensitive : NO
Description  : Ignore Hitman warning messages
Node
History
  Log Submits: NO
  Log Matches: NO
```

*This particular filter would trap all console messages from Hitman that indicate a user has been warned that they are idle. The action in this case is to ignore these messages. Since it is not important to know when these messages have been sent to the console logging matches has been turned off by setting history to NO. Since there are a fairly significant amount of these messages it has been given a low Sort ID, meaning the match will occur quickly after Trace receives the message and begins filtering; ordering filters from most to least commonly matched will reduce Traces` processing time significantly.*

DCL command

```
$ optrace /action=(id=5, ignore, match="*hitman*warned user*", -
  description="Ignore Hitman warning messages", -
  history=(nosubmit, nomatch))
```

Nocase\_sensitive is the default and did not need to be specified. The node name has not been specified so that regardless of what node Trace is running on messages matching “\*hitman\*warned user\*” will be ignored.

```

Sort ID      :      20
Type        : Action
Match       : *PLEASE INSERT THE DAILY BACKUP TAPE*
Case Sensitive : YES
Description  : The backup tape was not inserted at the proper time
Node        : NodeA
Command     : trace_com:notify_senior_operator.com
Queue       : NodeA$Batch
User        : Oper
Keep Log    : Yes
Print Log   : No
Range       : 10:10
Reset After  : 0 01:00:00.00
History
  Log Submits: Yes
  Log Matches: No

```

This filter will detect the tenth time a message is sent to the console requesting that the daily backup tape be inserted. Since the reset time is one hour Trace will act on this match once per hour until the console message stops being broadcast. Trace will only respond to this filter if it is running on NodeA. An action job to notify the senior operator is submitted under the username OPER to Nodea\$batch with the log being kept but no printed.

#### DCL Command

```

$ optrace /action=(id=20, match="*PLEASE INSERT THE DAILY BACKUP TAPE*", -
  case_sensitive, node=NodeA, queue=nodea$batch, user=oper, keep, noprint,-
  range=10:10, command=trace_com:notify_senior_operator.com, -
  description="The backup tape was not inserted at the proper time")

```

Note: It is not necessary to specify the reset time since 1 hour is the default.

```

Sort ID      :      300
Type        : ACTION
Match       : *
Case Sensitive : NO
Description  : An unfiltered console message
Node        :
Command     : trace_com:sample_command.com
Queue       : sys$batch
User        :
Keep Log    : NO
Print Log   : NO
Range       : 1:2000000
Reset After : 0 01:00:00.00
History
  Log Submits: NO
  Log Matches: NO

```

This filter is the last one in an extensive Trace parameter file. If a message has not matched any filters this final filter will submit the sample\_command.com that turns the console message into an OpenVMS Mail message and sends it to system. The system user can then read the resulting mail messages and decide if ignore or action filters should be added for this message. Some sites set up this same filter but make it an Ignore filter since the message has not matched any of their critical messages that filters have already been set up for.

#### DCL Command

```

$ optrace /action=(id=300, match="*", queue=sys$batch, nokeep,-
  noprint, command=trace_com:sample_command.com, -
  description="An unfiltered console message", -
  history=(nosubmit, nomatch))

```

Note: In this filter sys\$batch is actually a logical name that points to a node specific queue. With this setup Trace can respond to console messages regardless of the node it is running on. The match="\*" guarantees that all console messages that have not yet been filtered will be acted on by this filter.

```
Sort ID      :      300
Type        : ACTION
Match       : *
Case Sensitive : NO
Description  : Forward unfiltered messages to master node
Node        :
Command     : trace_com:send_task_message.com
Queue       : sys$batch
User        : TRACE
Keep Log    : yes
Print Log   : no
Range       : 1:2000000
Reset After  : 0 01:00:00.00
History
  Log Submits: YES
  Log Matches: YES
```

This filter was created as the last filter in an extensive Trace parameter file. Unlike the previous example this filter submits `send_task_message.com` to forward the console message to the master node. With a blank node and logical name for the queue this filter will work on any node. The username must be set to Trace because proxy access to the master node has been set up for that account; the `reissue_opcom.com` procedure has been copied to `sys$login` for the Trace account on the master node.

#### DCL Command

```
$ optrace /action=(id=300, queue=sys$batch, /match="*", user=trace, -
  keep, command=trace_com:send_task_message.com, noprint, -
  description="Forward unfiltered messages to master node",
  history=(submit,matches))
```

## Default Record

The following filter is the default filter for Trace. Anytime a filter is added any keywords that are not included in the action clause will be set to the values in this record. This record establishes a default batch job, queue, node that Trace will monitor on and that trace will log matches but not jobs submitted.

```
Sort ID      :          0
Type         : ACTION
Match        : *
Negated      : NO
Case Sensitive : NO
Description  :
Node         :
Command      : TRACE_COM:SAMPLE_COMMAND.COM
Queue        : SYS$BATCH
User         :
Keep Log     : NO
Print Log    : NO
Range        : 1:2000000
Reset After  : 0 01:00:00.00
History      :
  Log Submits : YES
  Log Matches  : NO
```

This default filter can be updated as often as desired. For example, to update the default record, so that the Trace username will be used for batch jobs and matches as well as submits will be logged:

```
$ optrace /action=(id=0, user=trace, history=(submits,matches))
```

Any filter can be updated by specifying the ID and the fields that need to be updated on the command line.

# Reference Section

## Logical Names

This section describes the logical names Trace requires to run as well as some additional logicals that can be defined to help control how Trace works. These logicals, if defined, must be in the system table.

Required (defined in sys\$manager:trace\_system\_logicals.com)

TRACE\_CDU - This logical points to the command definition directory. It is normally used to access the OPTRACE.CLD command definition

TRACE\_COM - This logical name points to the command procedure directory. It is used to access the Trace DCL command procedures.

TRACE\_DOC - This logical name points to the documentation directory. It is used to access the Trace help library if the library was not installed in the system or a user library.

TRACE\_EXE - This logical name points to the executable directory. It is used primarily by the command definition for trace and by the DCL command procedures to run the Trace rebroadcast utilities.

TRACE\_DAT - This logical name points to the data directory. The trace permanent data file, log file and error log file are all created and stored in this directory. This is also the directory that trace statistics reports will be written to.

TRACE\_TERMINAL - This required logical names specifies the LTA device that Trace is monitoring for console messages. This terminal is created during system startup by the trace\_com:create\_lat\_terminal.com procedure.

TRACE\_OPERATOR\_CLASS - This logical name defines which operator classes Trace will monitor for console messages. The operator classes which are being monitored can be defined by editing the trace\_com:define\_opcom\_class.com procedure which is run by the trace\_startup command procedure. If this logical name is not defined Trace will monitor all operator classes by default.

### Optional

TRACE\$DISABLE\_LOG - If this logical name is defined to be anything but itself Trace will not open a log file and log its actions, regardless of how logging is set in the individual filters. By default Trace does log its activities according to how the global history and individual filter history entries are defined.

TRACE\$ERROR\_HANDLER - This logical name is used to control how Trace will respond in the event of an internal error. If this logical name is defined to be anything but itself Trace will stop running if it encounters an internal error. If this logical name is not defined Trace will attempt to continue after an internal error.

TRACE\$LOOP\_LIMIT - It is possible for Trace to enter an endless loop if it detects a console message whose filter submits an action procedure that sends a console message that gets trapped by the same filter. A mechanism for attempting to detect and halt these conditions has been added to Trace. If this logical name is defined to be a numeric value Trace will halt if it responds to that number, or more, console messages in a 60 second period. If the logical name is not defined the default value of 15 will be used.

TRACE\$MASTER\_NODE - This logical must be defined on every node that will be using the send\_task\_message.com procedure to forward console messages from the current node to a central operations node. The send\_task\_message.com will attempt to establish a connection to the master node via DECNET and run the reissue\_opcom.com procedure on that node to copy and then rebroadcast the current console message on the master node. Since console messages are automatically broadcast to all enabled operators in a cluster this logical name is not necessary for multiple nodes if they are clustered.

## Command procedures

Related to startup:

Trace\_startup - This procedure gets automatically copied to sys\$manager during installation. Running this command procedure from within the SYSTARTUP\_VMS command procedure will:

- automatically define the trace logical names (@sys\$manager:trace\_system\_logicals)
- define any site specific Trace logicals (@sys\$manager:trace\_site\_logicals)
- load the Trace or Cohort license (@sys\$manager:cohort3axp\_pkms\_start or @sys\$manager:trace2axp\_pkms\_start)
- create the lat device for the monitor to use (@trace\_com:create\_lat\_terminal),
- define the operator class(es) to monitor (@trace\_com:define\_opcom\_class)
- load the OPTRACE command verb into the process command table  
(set command trace\_cdu:optrace)
- start the trace detached process (optrace /start)

Create\_lat\_terminal.com (location: TRACE\_COM): This procedure creates the lat terminal device that is attached to the trace detached process. This device is enabled to receive messages for various operator classes by the command procedure define\_opcom.com. All console messages broadcast to it are then filtered, automatically, by Trace. The default lat device is LTA1000.; sites where this will create a conflict can edit this procedure and change the device number.

Define\_opcom\_class (location: TRACE\_COM): This procedure defines the operator classes for console messages that Trace should trap and filter. By default Trace will monitor all classes. Edit this procedure to configure which classes Trace should monitor.

Trace\_system\_logicals (location: SYSS\$MANAGER): This procedure defines the system logicals for the Trace directories.

Trace\_site\_logicals (location: SYSS\$MANAGER): This procedure defines site specific logicals, if required. This procedure is not replaced during upgrades/installations so it should be used to define site specific Trace logicals.

### General

Install\_cdu.com (location: TRACE\_COM): This procedure loads the OPTRACE command verb into a single processes command verb table. This command procedure can be run from login.com in any accounts that will be used to maintain trace IF the OPTRACE command verb was not added in the DCLTABLES during the installation.

### Action procedures

Sample\_command.com (location: TRACE\_COM): This sample action procedure can be submitted by any trace filter. The procedure mails a copy of the operator console message to the system username using VMS mail. It provides a good starting point for developing your own action procedures because it clearly shows the 4 parameters that get passed to any

action procedures submitted by trace (P1-P4). These parameters are:

- The file containing a copy of the opcom message
- The description field from the filter the message matched
- The match criteria field from the filter the message matched
- The sort ID of the filter the message matched.

Procedures used to rebroadcast messages

`Send_task_message.com` (location: `TRACE_COM`): This command procedure helps centralize operations in a nonclustered Decnet network by forwarding console messages to the master node. The master node runs a network task (`reissue_opcom.com`) to receive and then rebroadcast the console message. A copy of this procedure must be placed in `sys$login` of the Trace account on each node that will have forwarded console messages.

`Reissue_opcom.com` (location: `TRACE_COM`): This command procedure is run on the master node in response to the `send_task_message` procedure. It receives the forwarded console message and then rebroadcasts it on the master node. On the master node filters can be added to Trace to detect these messages and take appropriate action. A copy of this procedure must be placed in `sys$login` for the Trace account on the master node.

Other procedures

Trace does include other command procedures but they are not normally required. Unless you are requested to run one of these procedures by the technical support staff at Saiga Systems they can be safely ignored. The following two procedures should be run after every major OpenVMS upgrade to relink Trace.

`Define_logicals.com` (location: `TRACE_COM`): This procedure prompts for four items, which can normally be defaulted. It defines a set of process level logical names for all the Trace directories.

`Customer_link.com` (location: `TRACE_COM`): This procedure can be run after the `define_logicals.com` procedure to relink Trace. It may also be run after a request by Saiga Systems technical support staff to enable traceback information for problem solving.

## Files

This section contains a brief description of some of the files in the Trace\_dat directory.

Trace\_perm\_data.dat - this is Trace's permanent data file. It contains all the filters.

Trace\_log\_file.node - this is the log file for a specific node. All the history records created by Trace for matches, submissions and Trace vents such as shutdown or reset because of parameter changes get recorded in this file. This file is an ordinary text file and can be typed, printed, edited or searched. Each time you start Trace a new version of this file will be created.

Trace\_error\_file.node - this is the redirected output from the SYSS\$ERROR for the Trace detached process. This file is appended to each time Trace encounters an error. The file is normally locked while Trace is running but it can be typed, printed, edited or searched when Trace is not running.

Trace\_output\_file.node - this is the redirected output from the SYSS\$OUTPUT for the Trace detached process. This file is appended to each time Trace writes to the sys\$output logical unit. The file is normally locked while Trace is running but it can be typed, printed, edited or searched when Trace is not running.

Temp\_mail\_XXXXXXXXXXXXXXXXX.txt - this is a temporary file containing the text from a console message that Trace has received. Normally these files will be deleted as soon as they are acted upon by either being mailed by the sample\_command.com procedure or ignored. If a number of these files are present it is normally safe to delete any that are more than a few hours old.

## Qualifier Reference

This section lists the qualifiers that can be used with OPTRACE command verb. Due to a command verb conflict introduced in OpenVMS 7.x when a TCP/IP trace utility was added to OpenVMS. To prevent a conflict with this Trace utility we have chosen OPTRACE instead of TRACE for our command verb.

Here is a complete list of OPTRACE qualifiers with a brief description of each:

/ACTION=Action_clause	Used to add filters to the trace filter data file
/HISTORY=(history_clause)	Used to control the default logging actions for the detached process
/LIST	Used to list the contents of the trace filter data file
/LOG=value	Used to control the trace log file
/OUTPUT=file	Used to specify an output file for the /list qualifier
/RESEQUENCE=value	Used to resequence filter ID numbers as multiples of #
/START	Used to start the trace detached process
/STATISTICS	Used to generate a statistics report from the detached process
/STOP	Used to stop the trace detached process
/VERSION	Used to display Trace version and license information

Each of these explained in detail in the following pages.

## /ACTION

Use the /action qualifier to add, remove or update filters. Trace can have up to 256 filters defined. When a console message is detected the Trace detached process is woken and it compares the text of the message to each filter, in sequence, until a match is found. If no match is found against any filter the console message will be ignored and a counter for messages not matched incremented. The /statistics qualifier output includes information on how many console messages have been received that could not be matched against an existing filter.

Syntax:        /ACTION=filter

In the following sections all the various action keywords are described; items in “[ ]” are optional and can be negated if they are preceded by [no]. Complete examples of filters, with descriptions and the DCL level commands needed to add those filters, can be found in the reference section for filters that begins on page 8.

To add an action filter:

```
OPTRACE /ACTION=(ID=number, -
    [match="match-string",] -
    [[no]negated,]-
    [[no]case_sensitive,]-
    [node=nodename,]-
    [description="description-string",]-
    [command="command-procedure",]-
    [queue=batch-queue,]-
    [user=username,]-
    [[no]keep_log,]-
    [[no]print_log,]-
    [interval=value,]-
    [range=start:end,]-
    [[no]reset_after=delta-time,]-
    [[no]history=history-clause]
```

To update a filter use the same syntax used when adding a filter and specify the ID number of the existing filter. Only those keywords that you want to change need to be included in the command.

To add an ignore filter:

```
OPTRACE /ACTION=(ID=number, -
    IGNORE,-
    [match="match-string",] -
    [[no]negated,]-
    [[no]case_sensitive,]-
    [node=nodename,]-
```

```
[description="description-string",]-  
[interval=value,]-  
[range=start:end,]-  
[[no]reset_after=delta-time,]-  
[[no]history=history-clause]
```

To delete a filter:

```
/NOACTION=(ID=number)
```

The following section provides a detailed explanation of each keyword with examples of how to use them. Keywords are grouped according to related functions:

1. Filter identification
2. General keywords
3. Matching keywords
4. Control keywords
5. Action keywords

When adding filters the values not specified on the command line will be filled in from the default filter. Trace ships with that filter set to:

```
Sort ID      :      0  
Type         : ACTION  
Match        : *  
Negated      : NO  
Case Sensitive : NO  
Description  :  
Node         :  
Command      : TRACE_COM: SAMPLE_COMMAND.COM  
Queue        : SYS$BATCH  
User         :  
Keep Log     : NO  
Print Log    : NO  
Range        : 1:2000000  
Reset After  : 0 01:00:00.00  
History      :  
    Log Submits : YES  
    Log Matches : NO
```

### Filter identification

The ID keyword is the only required keyword. It is used to specify which filter to add, update or delete. Whenever an opcom message is detected trace compares that messages to the filters in order by the ID number and the action specified in the first filter that matches.

Syntax:

```
ID=number
```

## Examples:

<code>Optrace /action=(id=200, ...)</code>	Add a filter using ID 200
<code>Optrace /noaction=(id=200)</code>	Delete the filter with ID 200
<code>Optrace /action=(id=200, queue=sys\$batch)</code>	Change the batch queue for filter ID 200

## General Keywords

The general keywords are description and history. Description can be omitted but if a description exists for a filter it is included in the filter listing report, the statistics report, the log entry if logging is enabled for this filter, in the new OPCOM message if rebroadcasting is being used and can be used within any command procedures to help make mail messages or pages more meaningful. The history keyword controls whether trace logs all matches against a filter in its log file, all submits of action procedures, both matches and submits or does not log this filter at all.

## Syntax:

<code>Description=text</code>	Provides a text description of this filter. This description is included in the filter listing report, the statistics report, in the new OPCOM message if rebroadcasting is being used and in the default command procedure as part of the mail message created.
<code>History=clause</code>	Tells trace what should be logged for this particular filter; matches, submits, both or nothing.

## Example:

This first example will ignore all hitman idle warning console messages. Since this messages can be safely ignored no log entries will be written for matches.

```
Optrace /action=(id=201, description="Ignore Hitman idle messages", ignore,-
match="*hitman*idle*", history=(nomatch,nosubmit))
```

In this second example the above filter has been modified so that every 10<sup>th</sup> hitman idle warning message is processed to get an idea of how many warnings are generated a day. Both matches and submits will be logged.

```
Optrace /action=(id=201, match="*hitman*idle*", interval=10, -
description="Respond to every 10th Hitman idle message", -
history=(match,submit), queue=node$batch, nokeep_log, noprint_log, -
user=system, command=trace_com:sample_command.com)
```

## Matching Keywords

These keywords control what opcom messages match a particular filter. All opcom messages are checked against the filters, in ID sequence, until the first filter they match; that filter's actions are then taken in response to the opcom message. The `case_sensitive` and `negated` keywords help fine tune a filter when necessary. The `node` keyword can be used in situations where Trace is running on more than one node with a shared data file; specifying a node will limit actions to being performed if the Trace detached process that finds a match is running on the

specified node.

#### Syntax:

<code>match="string"</code>	Match provides the actual filtering string. It can include the normal OpenVMS wildcard characters of "*" for any number of characters and "%" for a single character match. It is important to remember, when adding filters, that the first filter that an opcom message matches will be the only filter applied to that message.
<code>[no]case_sensitive</code>	The <code>case_sensitive</code> keyword can be used to make the match case sensitive; by default filters are not case sensitive ( <code>nocase_sensitive</code> ).
<code>[no]negated</code>	The <code>negated</code> keyword can be used to reverse the function of the match; ie: if the messages does not match this filter then then specified action is taken; by default filters are not negated ( <code>nonegated</code> ).
<code>node=nodename</code>	The <code>node</code> keyword tells trace that it should only take the defined action for this filter if trace is running on the specified node. The default is to act regardless of node.

#### Examples:

1. Ignore all hitman warning messages received on the wrkstn node.  

```
Optrace /action=(id=202, ignore, match="*hitman*warned user*idle*", -  
node=wrkstn)
```
2. Ignore the daily backup confirmation message. Since backups are critical we will only ignore this message if it matches including case. Since there are several different jobs that send variants of this message we will ignore any that have a 3 character flag (ie: EOD, DLY) and any others, such as WKLY, MNTHLY and FULL will not match. The next filter will capture these jobs and submit the trace sample command procedure whic sends an e-mail to system that includes the complete opcom message and description.

```
Optrace /action=(id=203, ignore, match="*PERFORMING %%% BACKUP*", -  
case_sensitive)  
Optrace /action=(id=204, match="*PERFORMING*BACKUP*", keep_log,-  
noprnt_log, command="trace_com:sample_command.com", -  
description="A special backup is being done", user=system, queue=fast)
```

Note: This particular combination of filters will only work if they are defined in this order. If the more general match string were checked first all backup messages would match it.

#### Controlling keywords

These keywords help to fine tune how often particular filters actually match and get acted upon. They facilitate setups where critical actions, such as paging operators, are only performed if the opcom messages is generated at least x times in a specific period of time.

Keyword:

**Range=start:end** The range keyword can be used to control how often trace takes action when a particular match occurs. Used in conjunction with the `reset_after` keyword the number of times, and frequency, that trace acts on a particular message can be fine tuned. The default values for range are 1:20000 which effectively means that all matches are acted on. In situations where a particular opcom message may occur regularly but should be automatically fixed setting a start range higher than 1 will cause trace to ignore the first few times the message is received and only respond if it continues beyond that. The `reset_after` keyword specifies a delta time that is used to control how often the match counter gets reset to 0.

**Interval=value** The interval keyword tells Trace to only act on a particular message every so many times that it occurs. This can limit the number of times trace acts on a particular filter. The default value for interval is 1. If a particular problem situation causes a console message to be generated frequently until the problem is resolved using interval will insure that the filter's action is taken but help keep it from being taken repeatedly.

**Reset\_after=delta** The `reset_after` keyword instructs trace to reset the counter back to zero for the number of times a particular filter has been matched; the historical counter for the statistics purpose is not zeroed. The default is to reset filters after every 1 hour (0 01:00:00.00). Using this keyword in conjunction with range allows very specific fine tuning for how often a particular filter responds to a message.

Examples:

1. When Hitman is running it sends a time stamp to the console every 1 hour. If you want to receive a mail message once a day indicating that Hitman is running you can configure an event to respond only the first time a match occurs and then reset the filter everyday.

```
Optrace /action=(id=150, description="Hitman is running", -
match="*hitman*see you in an hour*", range=1:1, -
reset="1 00:00:00.00", command=trace_com:sample_command.com,-
user=trace, nokeep, noprint, queue=sys$batch)
```

2. The same situation could also be handled with the interval keyword. Since the message is only sent 1 time per hour it is important to provide a `reset_after` value that is many days or the filter may never be triggered.

```
Optrace /action=(id=150, description="Hitman is running", -
match="*hitman*see you in an hour*", interval=24,
```

```
reset="365 00:00:00.00", command=trace_com:sample_command.com,-
user=trace, nokeep, noprint, queue=sys$batch)
```

3. A hitman event has been configured to detect when `sys$sysdevice` has less than 25% free space left. The event submits a job to purge the disk one time; multiple purges will not have any effect. The event is configured to send a console message every data collection interval until the disk has more than 25% free space and the event gets automatically reset. Since purge operations on a large disk can take several minutes a filter can be added that will page the system operator if sufficient disk space has not been cleared up within 10 minutes. If the disk remains too full the filter will be acted on again every 30 minutes.

```
Optrace /action=(id=25, description="System disk is getting full", -
range=10:10, reset_after="0 0:30:00", command=trace_com:page,-
queue=fast$batch, keep, noprint, user=system, -
match="*hitman*sys$sysdevice is getting full*")
```

#### Action keywords

There are two separate structures of action keywords; they are mutually exclusive and only one can be present in a filter. The filter can either contain the information about a batch job that trace should submit in response to a console message OR it can contain the keyword IGNORE which tells trace to ignore any console messages that match this filter.

#### Keyword:

`ignore` Ignore any console messages that match this filter.

#### OR

`command=procedure` Submit this batch job whenever a console message matches this filter and all of the control clauses indicate that action should be taken. The procedure name should be complete.

`queue=queue` The batch queue that the command procedure should be submitted to. It is recommended that the queue execute jobs on the same node as trace is running on.

`user=username` The username that the command procedure should be submitted under. Be sure this username has sufficient privileges to access files in the `trace_dat` directory; the detached process writes temporary files containing opcom messages into this directory.

`[no]keep_log` This flag controls whether the batch job will be submitted with the `/keep` flag to keep the log file from the batch job after the job completes.

`[no]print_log` This flag controls whether the batch job will be submitted with the `/print`

flag to automatically print the log file after the batch job completes.

## **/HISTORY**

The history qualifier sets the default history actions for the Trace detached process. There are two types of historical entries; matches against a filter and submission of action procedures. When logging is enabled for matches an entry is written to the log file everytime a console messages matches a filter. When logging is enabled for submits an entry is written to the log file everytime trace submits a batch job in response to a filter match. Both types of matches can be further configured by using the keywords action (when a job is submitted), always (whenever a match occurs) or never which effectively turns off that type of logging.

Syntax:

```
/history=(submit=[action][always][never], match=[action][always][never])
```

Example:

To set the default history actions to be logging all matches and all action procedure submits:

```
/history=(submit=action, match=always)
```

## **/LIST**

This qualifier generates a listing of Trace's parameters including default actions and all defined filters. Trace will pause after each filter and wait for the user to hit the <enter> key before it continues. If the /output qualifier is used the listing will be written to a file.

Syntax:

```
optrace /list [/output=trace.lis]
```

## **/LOG**

This qualifier directs Trace to take some action with its log file. A keyword of CLOSE, FLUSH or NEW must be specified with this qualifier; CLOSE instructs trace to close its log file, no further logging will be done until trace is restarted or NEW is used to open a new log file. FLUSH causes trace to update the log file with any information that is currently buffered and has not yet been written out. Stopping trace automatically does a /log=flush.

Syntax:

```
optrace /log=[CLOSE][FLUSH][NEW]
```

Examples:

The trace log file is getting too large. To close the current file and open a new one:

```
optrace /log=new
```

the old file can then be edited or even purged.

## **/OUTPUT**

The output qualifier redirects the output from /list to a file instead of to the users screen. Any valid OpenVMS file name can be specified; the directory will default to the current working directory if no directory is included in the file specification.

Syntax:

```
optrace /output=[filename]
```

Example:

To list and then print the contents of the trace filter file:

```
optrace /list /output=x.x  
print /delete x.x
```

## **/RESEQUENCE**

This qualifier directs Trace to renumber the filters in the parameter file. Since filters are checked in id number sequence it could become necessary to add a filter between two existing filters that are sequential. Using the resequence qualifier will update the first id number to the value specified and then increment each subsequent filters ID number by that value. The numeric value specified must be a positive integer number.

Syntax:

```
optrace /resequence=value
```

Example:

To resequence all filters so that there ID numbers begin with 5 and increment by 5:  
optrace /resequence=5

Caution: Since the ID number is passed to any action procedures submitted by Trace as the fourth parameter (P4) resequencing the parameter file may adversely affect your action procedures if you are using this parameter. If it is necessary to resequence your parameter file the action procedures may need to be updated to reflect the new ID number assigned. We also do not recommend including the ID number in the description since resequencing would leave the descriptions incorrect since they are not updated.

## **/START**

Use the qualifier to start the detached process that monitors all console messages send to the opcom class(es) trace is configured to monitor.

Syntax:

```
Optrace /start
```

On successful completion a message should be displayed showing the UIC that Trace was started under and the process id for the newly created process.

## **/STOP**

Use this qualifier to stop the trace detached opcom monitor process.

Syntax:

```
optrace /stop
```

On successful completion of this command a message will be received at the terminal indicating that Trace has been stopped.

## /STATISTICS

Use this qualifier to generate a status and statistics report for the trace detached process. This report includes some general statistical information about how many opcom messages have been detected, how many of them were ignored, how many were excluded by range or interval, how many were acted on and how many did not match any of the filters. This is followed by a list of all the trace filters with their id number, the match string, the filter's description, a flags indicating whether it is an ignore filter or if it is negated, and counters for how often the filter has been matched since the last reset, how many times it has been reset and how many times it has been matched in total. The statistics are always written to trace\_dat:statistics.node.

This report can be especially helpful when trying to isolate problems with filters. A filter that unexpectedly shows no matches indicates that an earlier filter is capturing the console message, there is an error in the filters match string or the console message is being generated to an OPCOM class that Trace is not monitoring.

Syntax:

Optrace /statistics

Sample report:

Current Time : 7-APR-2000 15:44:50.53  
Start Time : 5-APR-2000 10:00:13.40

Total Messages : 127  
Total Messages Ignored : 5  
Total Messages Acted On : 122  
Total Messages Not Matching : 0  
Total Messages Excluded (Range or Interval) : 0

ID	Total Count	Current Count	Times Reset	Match Field	Ignore	Negated	Description
20	0	0	0	*Please mount device*	T	F	Ignore device mount requests
26	0	0	0	*Operator*has been enabled, username SYS	T	F	Ignore system enabled as operator
27	1	1	0	*operator status for operator*	T	F	Ignore operator status
298	2	2	0	*Accounting running at *	T	F	Accounting running normally: IGNORE
299	2	2	0	*Accounting rescheduled for tomorrow*	T	F	Accounting rescheduled normally: IGNORE
300	122	1	51	*	F	F	FORWARD UNKNOWN MESSAGES TO SWIFT
301	0	0	0	*	F	F	Forward all unmatched messages

Note:

The statistics command must interrupt the detached process with an AST so it can output the report. As a result of this the counters will be reset each time the report is run.

## **/VERSION**

This qualifier outputs version and license information for Trace.

Syntax:

```
optrace /version
```

Sample:

```
$ optrace/version
TRACE 2.0
Copyright Saiga Systems Inc. 1989-1999, All Rights Reserved
Phone 1 (800) 561 8876
      1 (403) 263 1151
Fax   1 (403) 263 0744

WWW   www.saiga.com
FTP   ftp.saiga.com
E-mail sales@saiga.com
      support@saiga.com

Licensed to:      SAIGA SYSTEMS
License name:     COHORT3AXP
License node:     -SITE-
License expiry:   12/31/2099

Current date:     10-APR-2000 09:45:23.05
Node name:        SWIFT
VMS version:      V7.2
HW type:          AlphaServer 800 5/333
Cluster member:   N
Number of nodes:  0
```

# Appendix

## A. Trace Account; recommended settings

Sites that choose to forward console messages from one node to another node using Trace require a Trace account. A sample of a valid trace account follows:

```
Username: TRACE                               Owner: SAIGA
Account:                                       UIC: [205,300] ([TRACE])
CLI: DCL                                       Tables: DCLTABLES
Default: DKA0:[TRACE]
LGICMD:
Flags:
Primary days: Mon Tue Wed Thu Fri
Secondary days:                               Sat Sun
Primary 000000000001111111112222 Secondary 000000000001111111112222
Day Hours 012345678901234567890123 Day Hours 012345678901234567890123
Network: ##### Full access #####           ##### Full access #####
Batch:   ##### Full access #####           ##### Full access #####
Local:   ----- No access -----          ----- No access -----
Dialup:  ----- No access -----          ----- No access -----
Remote:  ----- No access -----          ----- No access -----
Expiration: (none) Pwdminimum: 6 Login Fails: 0
Pwdlifetime: (none) Pwdchange: 30-MAR-2000 09:14
Last Login: (none) (interactive), 10-APR-2000 09:20
(non-interactive)
Maxjobs: 0 Fillm: 100 Bytlim: 64000
Maxacctjobs: 0 Shrfillm: 0 Pbytlim: 0
Maxdetach: 0 BIOlm: 150 JTquota: 4096
Prclm: 8 DIOlm: 150 WSdef: 2000
Prio: 4 ASTlm: 250 WSquo: 4000
Queprio: 4 TQElm: 10 WSextent: 16384
CPU: (none) Enqlm: 2000 Pgflquo: 50000
Authorized Privileges:
NETMBX OPER SYSPRV TMPMBX
Default Privileges:
NETMBX OPER SYSPRV TMPMBX
```

Since this account requires proxies to be set up on each node that will be receiving forwarded console messages we automatically set the account so that INTERACTIVE logins (local, dialup and remote) are disabled. Only network and batch logins are enabled. The Owner and Account fields can be set to any desired values.

## B. Error Messages

Trace uses VMS-style error messages. This appendix lists these messages in alphabetical order. Since Trace runs as a primarily detached process many of these messages will only be found in one or both of the Trace\_output\_file.node and Trace\_error\_file.node files since these contain the redirected output from sys\$output and sys\$error.

**Deldefrec - Cannot delete default action record** You cannot delete the default action record. Check to make sure you are using the correct ID number and reenter the command.

**Detrunaway - Trace Stopping, possible runaway condition** Trace is stopping because it believe it maybe in a runaway condition. The logical name Trace\$loop\_limit determines how many messages Trace can receive in one minute before it considers itself runaway. The default value is 15.

**Errclosefile - Error closing file *filename*** Trace encountered an error trying to close the file *filename*. Try the operation again and if it continues to fail contact technical support at Saiga Systems Software Inc.

**Errinqual - Error in processes the qualifier *qualifier*** A qualifier has been entered that is either invalid or has an invalid value specified. Please check the qualifier against the reference section of this manual and try again.

**Erropenfile - Error opening the file *filename*** Trace was unable to open the specified file. Check that all the required Trace logical names are defined and valid and that any disks referenced are mounted and online. If the problem persists please contact technical support at Saiga Systems Software Inc.

**Errreadfile - Error reading file *filename*** Trace was unable to read the specified file. Check that all the required Trace logical names are defined and valid and that any disks references are mounted and online. Also check the file protections on the specified file as well as its parent directory(s). If the problem persists please contact technical support at Saiga Systems Software Inc.

**Errwritefile - Error writing file *filename*** Trace was unable to write to the specified file. Check that all the required Trace logical names are defined and valid and that any disks referenced are mounted and online. If the problem persists please contact technical support at Saiga Systems Software Inc.

**Errstrtdet - Error starting detached process** Trace was unable to get started. Check that all the required Trace logical names are defined and valid and that any disks referenced are mounted and online. Also check your account privileges and quotas. The trace\_error\_file.node may contain a more detailed explanation of the error. Checking the OpenVMS Accounting record for the detached process with /full may provide a text explanation for the error that prevented the detached process from starting. If the problem persists please contact technical support at Saiga Systems Software Inc.

**Invdate - date is not a valid date for the qualifier *qualifier*** You have entered a date incorrectly. Correct the problem and reenter the command.

**Invdelta - Invalid delta time entered, please re-enter** You have entered an invalid delta time. Correct the problem and reenter the command. When entering delta times at the DCL level they are normally required to be enclosed in double quotes since a delta time may contain a space.

**Invrange - Qualifier *qualifier* has an invalid range** You have entered an invalid range for the specified qualifier. Be sure to enter the lower number then the higher number separated by a colon.

**Itemreq - You must enter a value for the qualifier *qualifier*** You must enter this qualifier with a valid value. Reenter the command with a valid value.

**Logreq - logical name must be defined** The specified logical name is not defined on this system. Check to ensure that the Trace\_startup procedure has been run since it defines the logical names. Redefine the logical name or run Trace\_startup manually if required.

**Nonexpr - Failure starting TRACE's detached process, check console for error** Trace was unable to start. Check the Trace\_error\_file.node for any messages related to the failure. Also check your account privileges and quotas, the file protections on the Trace directories, that all the required Trace logical names have been defined and that their definitions are valid and point to disks that are mounted and online.

**Normal - Normal successful completion** The command has been completed successfully. No further action is required.

**Nosuchrec - No such action record** You have attempted to modify or delete an action record that doesn't exist. Reenter the command with a valid ID number.

**Notrun - Trace is not running** Trace is not currently running and may need to be restarted.

**Prsstp - Stopping Trace, please wait ...** This message is displayed when the command to stop Trace has been issued. You should be returned to the DCL level \$ prompt shortly.

**Pracsstrt - Starting Trace, please wait ...** This message is displayed when the command to start Trace has been issued. You should be returned to the \$ prompt shortly and the UIC and process identification of the created process displayed.

**Procid - Identification of created process is *process-id*** Trace has just been started. The process ID for the new process is displayed.

**Reqpriv - Operation required *privilege*** Trace requires the listed privilege for the operation you are attempting. Grant yourself the required privilege and retry the command. If this message is seen in the Trace\_error\_file.node file please make sure that the Trace account has been created with the default and authorized privileges shown in Appendix B, The Trace Account.

**Running - Trace is already running** Trace is already running, it is not necessary to start it at this time.

**Toomanychar - the item *item* specified with the qualifier *qualifier* is too long, maximum length is *length*** The item you specified is too long. Reenter the command with a shorter value for the item.

**Toomanyitem - Too many items specified for *qualifier*, maximum allows is *value*** You have entered too many items for this qualifier. Reenter your command with the correct number of items.

**Toomanyrec - Maximum action record limit already reached, limit = *limit*** You have tried to add an additional action record when the action file is full. Delete an existing action and retry your command or modify an existing action.

**Valuetoohigh - The item *item* specified with the qualifier *qualifier* is too high, maximum is *value*** The value specified for the item was too high. Reenter the command with a value less than or equal to the displayed maximum value.

**Valuetoowlow - The item *item* specified with the qualifier *qualifier* is too low, minimum is *value*** The value specified for the item was too low. Reenter the command with a value greater than or equal to the displayed minimum value.

**Version - Saiga Systems - Trace 2.0.0** this message is displayed as output from the Trace/version command. It shows the version, subversion and patch level.

## C. Installing Trace

This section provides a complete sample installation on a master node with notes describing each step necessary in the installation. The procedure is similar on a remote node. If Trace is being installed on a stand-alone node or in a cluster where message rebroadcasting will not be enabled or required it is not necessary to create a Trace account or specify a master node.

### 1. Install Trace on an individual node OR on the master node

Prior to beginning this step you should have a tape distribution mounted and on-line, run the saveset extraction program on the distribution CD or have downloaded and unzipped Trace from the download Cohort 3 web page at [www.saiga.com](http://www.saiga.com).

```
Welcome to OpenVMS (TM) Alpha OS, Version V7.2

Username: system
Password:
  Welcome to OpenVMS (TM) Alpha Operating System, Version V7.2 on node SWIFT
  Last interactive login on Thursday, 30-MAR-2000 09:08:12.66
  Last non-interactive login on Thursday, 30-MAR-2000 09:00:44.68
$ set def sys$update
$ @vmsinstal
  OpenVMS AXP Software Product Installation Procedure V7.2
It is 30-MAR-2000 at 09:13.
Enter a question mark (?) at any time for help.
%VMSINSTAL-W-ACTIVE, The following processes are still active:
  DECW$SERVER_0
  DECW$TE_OC3A
  _FTA26:
* Do you want to continue anyway [NO]? yes
* Are you satisfied with the backup of your system disk [YES]?
* Where will the distribution volumes be mounted: dka0:[software]

Enter the products to be processed from the first distribution volume set.
* Products: trace020
* Enter installation options you wish to use (none):
The following products will be processed:
  TRACE V2.0
  Beginning installation of TRACE V2.0 at 09:13

%VMSINSTAL-I-RESTORE, Restoring product save set A ...

  You can have this installation automatically performed if the following
  defaults are acceptable:

  1. Help will be installed in a user help library rather than in
     the system help library.
  2. TRACE will be installed on the system disk SYS$SYSDEVICE:[TRACE].
  3. The installation will not purge existing files.
  4. The OPTRACE command will be installed in the DCLTABLES.
```

5. You will not be using the operator message re-broadcasting.

\* Are the above defaults acceptable [YES]? NO

%TRACE-I-NOAUTOINSTALL, This installation will ask questions relating to site specific options.

Note: If you answer YES to the following question, the DCLTABLES will be purged if you have the installation add the OPTRACE command to the DCLTABLES. To avoid purging this file, enter NO. See the Installation Manual for more information.

\* Do you want to purge files replaced by this installation [NO]? yes

To install this product in the root directory, enter the device name of the disk where you want the product installed (e.g. DUA0:). Please include the colon.

To install this product in a subdirectory, enter the full device and directory specification (e.g. DUA0:[LARRY.]). The "." after LARRY is required.

\* Name of disk [SYS\$SYSDEVICE:]: dka0:

You may choose one of the following options when installing TRACE's help files:

- 1) Add the help as an additional user help library.
- 2) Add the help in the system help library
- 3) Do not install help files.

Note: Exclusive access is required to install the help text into the system help library. Otherwise, if someone accesses help during the installation, it will fail.

\* Enter help option number [1]: 3

To permanently add the OPTRACE command verb to the DCLTABLES, answer YES to the following question. If you answer NO, you must add the command to your process command table. See the Installation Manual for more information.

Note: If you add the OPTRACE command to the DCLTABLES a check will be to ensure you have sufficient global pages. The installation will fail if the system does not have the required global pages. Refer to the Installation Manual for more information.

\* Add OPTRACE command to DCLTABLES [YES]?

Trace has the ability to take operator messages from one node on a DECNET network and rebroadcast the message on another node. This can be useful to sites that have remote systems.

\* Enable the rebroadcasting of operator messages [NO]? yes

To use Trace's facility to rebroadcast operator messages, an account must be created for Trace's use. This account will use proxies to send the information to the master node.

\* Name of Trace Account [TRACE]:

Please specify a UIC for the Trace account in the standard format [group,member]. If no UIC is specified Trace will default to [1,5].

\* UIC for Trace Account [[1,5]]: [205,300]

Trace requires the name of the node to receive the operator messages.

Note: This node will require proxies for all of the other nodes.

\* Name of master node: SWIFT

%TRACE-I-ASKDONE, This installation will not ask any more questions.

%VMSINSTAL-I-RESTORE, Restoring product save set C ...

%TRACE-I-COMMAND, Adding TRACE command to dcltables

%TRACE-I-RELINK, Relinking images

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.CDU].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.COM].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.DAT].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.DEF].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.DOC].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.EXE].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.EXE.AXP].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.EXE.VAX].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.LIS].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.MAP].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.MSG].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.OBJ].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.OBJ.AXP].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.OBJ.VAX].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.SRC].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.SRC.AXP].

%VMSINSTAL-I-SYSDIR, This product creates system disk directory DKA0:[TRACE.SRC.VAX].

%VMSINSTAL-I-ACCOUNT, This installation creates an ACCOUNT named TRACE.

%UAF-I-ADDMSG, user record successfully added

%UAF-I-RDBADDMSGU, identifier TRACE value [000205,000300] added to rights database

%TRACE-I-CREATEFILE, Creating TRACE systartup command file

A command file containing the logical names needed to use TRACE has been created. It is called TRACE\_STARTUP.COM. A copy of this file has been placed in SYS\$MANAGER: and a backup copy has been placed in the [TRACE.COM] directory.

To ensure the required logicals are defined after a system re-boot, the file SYS\$SYSMANAGER:SYSTARTUP\_VMS.COM should be modified to contain the following text:

```
$ @SYS$MANAGER:TRACE_STARTUP.COM
```

Please refer to the "After Running VMSINSTAL" section of the installation

guide for detailed instructions on how to finish upgrading / installing TRACE. To install a new license use:

```
$ @SYS$MANAGER:SAIGA_LICENSE.COM
```

If you have any questions or problems please contact us at:

Saiga Systems Software  
#215 801 - 6th Street S.W.  
Calgary, Alberta, Canada  
T2P 3V8

Phone - 1 800 561 8876 or  
1 403 263 1151  
FAX - 1 403 263 0744  
Email - support@saiga.com or  
sales@saiga.com

WWW <http://www.saiga.com/>

\* Press return to continue:

To be automatically notified of patches and new releases and to receive helpful tips on how to better use product please join the product list-server.

Send an e-mail message to product@saiga.com with the body of the message:  
join product your-e-mail-address

the list server will respond with an introductory e-mail advising you how to post questions to the list and how to make changes to, or delete, your subscription.

%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...

Installation of TRACE V2.0 completed at 09:14

Adding history entry in VMI\$ROOT:[SYSUPD]VMSINSTAL.HISTORY

Creating installation data file: VMI\$ROOT:[SYSUPD]TRACE020.VMI\_DATA

Enter the products to be processed from the next distribution volume set.

\* Products: EXIT

VMSINSTAL procedure done at 09:16

## 2. Define which operator classes trace should monitor

```
$ edit/edt trace_com:define_opcom_class.com
1      $!++
*'class_to_monitor
42     $! class_to_monitor = 'OPC$M_NM_SECURITY + 'OPC$M_NM_CLUSTER + 'OPC$M_NM_CENTRL
*s/!/42:43
42     $ class_to_monitor = 'OPC$M_NM_SECURITY + 'OPC$M_NM_CLUSTER + 'OPC$M_NM_CENTRAL
43     $ define/system TRACE_OPERATOR_CLASS 'class_to_monitor
2 substitutions
*exit
DKA0:[TRACE.COM]DEFINE_OPCOM_CLASS.COM;2 44 lines
```

```
$ @trace_com:define_opcom_class
```

### 3. Define proxies for the trace account on remote nodes

```
$ mcr authorize
UAF> add /proxy roc::trace trace/default
%UAF-I-NAFADDMSG, proxy from LOCAL:.ROC::TRACE to TRACE added
UAF> add/proxy eagle::trace trace/default
%UAF-I-NAFADDMSG, proxy from LOCAL:.EAGLE::TRACE to TRACE added
UAF> exit
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-NAFDONEMSG, network proxy database modified
%UAF-I-RDBNOMODS, no modifications made to rights database
```

### 4. Modify the trace\_com:create\_lat\_terminal.com procedure

In this example a LAT terminal number of 1000 is not in conflict with an existing LAT terminal so no changes are necessary.

### 5. Start the trace detached process

```
$ @sys$manager:trace_startup
%DCL-I-SUPERSEDE, previous value of TRACE_EXE has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE_DAT has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE_DOC has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE_COM has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE_CDU has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE$UIC has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE$MASTER_NODE has been superseded
%DCL-I-SUPERSEDE, previous value of TRACE_OPERATOR_CLASS has been superseded
Starting Trace detached process under UIC [205,300]
%TRACE-S-PRCSSTRT, Starting TRACE, please wait ...
%TRACE-S-PROCID, Identification of created process is 00001941
```

### 6. Add a generic filter to send any opcom messages that have not yet been filtered to the system account using VMS mail.

```
$ optrace /action=(id=300, match="*", description="Unfiltered opcom message",-
command=trace_com:sample_command.com, queue=swift$batch, keep, noprint, -
user=trace, node=swift)
```

### 7. View the contents of the trace filter file

```
$ optrace/list/output=x.x
$ type x.x
TRACE Version: 1.0.0 Node: SWIFT
/HISTORY
```

```

MATCH = ACTION
SUBMIT = ACTION
Default Record      1
Sort ID             :          0
Type                : ACTION
Match               : *
Negated             : NO
Case Sensitive      : NO
Description         :
Node                :
Command             : TRACE_COM:SAMPLE_COMMAND.COM
Queue               : SYS$BATCH
User                :
Keep Log            : NO
Print Log           : NO
Range               : 1:2000000
Reset After         : 0 01:00:00.00
History             :
    Log Submits     : YES
    Log Matches     : NO
Action Record       2
Sort ID             :          300
Type                : ACTION
Match               : *
Negated             : NO
Case Sensitive      : NO
Description         : Unfiltered opcom message
Node                : SWIFT
Command             : TRACE_COM:SAMPLE_COMMAND.COM
Queue               : SWIFT$BATCH
User                : TRACE
Keep Log            : YES
Print Log           : NO
Range               : 1:2000000
Reset After         : 0 01:00:00.00
History             :
    Log Submits     : YES
    Log Matches     : NO

```

**8. Periodically check for mail in the receiving account. Add filters as necessary.**

\$ mail

You have 1 new message.

```

MAIL>
    #1          30-MAR-2000 09:33:23.00
NEWMAIL
From: SWIFT::TRACE
To:  SYSTEM
CC:

```

Subj: TRACE Action (300) Unfiltered opcom message

```
$%%%%%%%%%% OPCOM 30-MAR-2000 09:33:22.34 %%%%%%%%%%%  
Message from user SYSTEM on SWIFT  
CHECK FREE ON DKA0  
Check Free Event # 3 notification, Device: DKA0:
```

```
MAIL> exit  
$ optrace /action=(id=10, queue=swift$batch, command=sample_command.com, -  
node=swift, user=trace, match="*Check*3 notification*dka0*", range=1:1,-  
reset="0 02:30", keep, noprint, -  
description="If DKA0 is low, notify every 2-1/2 hours")
```

```
$ mail  
You have 1 new messages.
```

```
MAIL>  
#1 30-MAR-2000 09:36:25.15  
NEWMAIL  
From: SWIFT::TRACE  
To: SYSTEM  
CC:  
Subj: TRACE Action (300) Unfiltered opcom message
```

```
$%%%%%%%%%% OPCOM 30-MAR-2000 09:36:24.77 %%%%%%%%%%%  
Message from user SYSTEM on SWIFT  
This is HITMAN 9.0 (hit mode), see you in an hour
```

```
MAIL> exit  
$ optrace /action=(id=11, match="*see you in an hour*", -  
_ $ description="Hitman time stamp: ignore", ignore)
```

## 9. Install Trace on the remote node(s)

The installation procedure is the same as on the Master Node. Do not take the default installation since rebroadcasting must be enabled. Be sure to specify the master node when prompted. To minimize the need to manage Trace on the individual nodes we recommend giving the same name and UIC when prompted for the account.

Make sure that the UAF record for the account has a valid default login directory and that the Trace command procedure `send_task_message.com` has been copied into this default directory. You should also verify that the command procedure `reissue_opcom.com` has been copied to the default login directory for Trace on the master node.

## 10. Set up filters, as needed, to limit network overhead

We recommend that filters be added locally for as many messages as possible; this limits the overhead involved in forwarded messages to the master node and rebroadcasting them. Since Trace V2 is limited to 256 filters it makes sense to take advantage of the filtering options locally.

## **11. Add a final filter to rebroadcast all unfiltered messages**

By making the final filter on the local node match "\*" and using the `send_task_message.com` procedure Trace will be configured to send all opcom messages that don't get filtered locally to the master node where they will be rebroadcast and potentially filtered. The action procedure might look like this:

```
$ Optrace /action=(id=300, match="*", nokeep, noprint, user=Trace, -  
    description="Forward unfiltered OPCOM messages", -  
    command=send_task_message.com, queue=node$batch)
```