# ACPICA Trace Facility

**Copyright:** © 2015, Intel Corporation
**Author:** Lv Zheng <lv.zheng@intel.com>

## Abstract

This document describes the functions and the interfaces of the method tracing facility.

## Functionalities and usage examples

ACPICA provides method tracing capability. And two functions are currently implemented using this capability.

### Log reducer

ACPICA subsystem provides debugging outputs when CONFIG_ACPI_DEBUG is enabled. The debugging messages which are deployed via ACPI_DEBUG_PRINT() macro can be reduced at 2 levels - per-component level (known as debug layer, configured via /sys/module/acpi/parameters/debug_layer) and per-type level (known as debug level, configured via /sys/module/acpi/parameters/debug_level).

But when the particular layer/level is applied to the control method evaluations, the quantity of the debugging outputs may still be too large to be put into the kernel log buffer. The idea thus is worked out to only enable the particular debug layer/level (normally more detailed) logs when the control method evaluation is started, and disable the detailed logging when the control method evaluation is stopped.

The following command examples illustrate the usage of the "log reducer" functionality:

a. Filter out the debug layer/level matched logs when control methods are being evaluated:

```
# cd /sys/module/acpi/parameters
# echo "0xXXXXXXXX" > trace_debug_layer
# echo "0xYYYYYYYY" > trace_debug_level
# echo "enable" > trace_state
```

b. Filter out the debug layer/level matched logs when the specified control method is being evaluated:

```
# cd /sys/module/acpi/parameters
# echo "0xXXXXXXXX" > trace_debug_layer
# echo "0xYYYYYYYY" > trace_debug_level
# echo "\PPPP.AAAA.TTTT.HHHH" > trace_method_name
# echo "method" > /sys/module/acpi/parameters/trace_state
```

c. Filter out the debug layer/level matched logs when the specified control method is being evaluated for the first time:

```
# cd /sys/module/acpi/parameters
# echo "0xXXXXXXXX" > trace_debug_layer
# echo "0xYYYYYYYY" > trace_debug_level
# echo "\PPPP.AAAA.TTTT.HHHH" > trace_method_name
# echo "method-once" > /sys/module/acpi/parameters/trace_state
```

Where:

0xXXXXXXXX/0xYYYYYYYY
: Refer to Documentation/firmware-guide/acpi/debug.rst for possible debug layer/level masking values.

PPPP.AAAA.TTTT.HHHH
: Full path of a control method that can be found in the ACPI namespace. It needn't be an entry of a control method evaluation.

### AML tracer

There are special log entries added by the method tracing facility at the "trace points" the AML interpreter starts/stops to execute a control method, or an AML opcode. Note that the format of the log entries are subject to change:

```
[    0.186427]   exdebug-0398 ex_trace_point        : Method Begin [0xf58394d8:\_SB.PCI0.LPCB.ECOK] execut
[    0.186630]   exdebug-0398 ex_trace_point        : Opcode Begin [0xf5905c88:If] execution.
[    0.186820]   exdebug-0398 ex_trace_point        : Opcode Begin [0xf5905cc0:LEqual] execution.
[    0.187010]   exdebug-0398 ex_trace_point        : Opcode Begin [0xf5905a20:-NamePath-] execution.
[    0.187214]   exdebug-0398 ex_trace_point        : Opcode End [0xf5905a20:-NamePath-] execution.
[    0.187407]   exdebug-0398 ex_trace_point        : Opcode Begin [0xf5905f60:One] execution.
[    0.187594]   exdebug-0398 ex_trace_point        : Opcode End [0xf5905f60:One] execution.
[    0.187789]   exdebug-0398 ex_trace_point        : Opcode End [0xf5905cc0:LEqual] execution.
[    0.187980]   exdebug-0398 ex_trace_point        : Opcode Begin [0xf5905cc0:Return] execution.
[    0.188146]   exdebug-0398 ex_trace_point        : Opcode Begin [0xf5905f60:One] execution.
[    0.188334]   exdebug-0398 ex_trace_point        : Opcode End [0xf5905f60:One] execution.
[    0.188524]   exdebug-0398 ex_trace_point        : Opcode End [0xf5905cc0:Return] execution.
[    0.188712]   exdebug-0398 ex_trace_point        : Opcode End [0xf5905c88:If] execution.
[    0.188903]   exdebug-0398 ex_trace_point        : Method End [0xf58394d8:\_SB.PCI0.LPCB.ECOK] executio
```

Developers can utilize these special log entries to track the AML interpretation, thus can aid issue debugging and performance tuning.

Note that, as the "AML tracer" logs are implemented via ACPI_DEBUG_PRINT() macro, CONFIG_ACPI_DEBUG is also required to be enabled for enabling "AML tracer" logs.

The following command examples illustrate the usage of the "AML tracer" functionality:

a. Filter out the method start/stop "AML tracer" logs when control methods are being evaluated:

```
# cd /sys/module/acpi/parameters
# echo "0x80" > trace_debug_layer
# echo "0x10" > trace_debug_level
# echo "enable" > trace_state
```

b. Filter out the method start/stop "AML tracer" when the specified control method is being evaluated:

```
# cd /sys/module/acpi/parameters
# echo "0x80" > trace_debug_layer
# echo "0x10" > trace_debug_level
# echo "\PPPP.AAAA.TTTT.HHHH" > trace_method_name
# echo "method" > trace_state
```

c. Filter out the method start/stop "AML tracer" logs when the specified control method is being evaluated for the first time:

```
# cd /sys/module/acpi/parameters
# echo "0x80" > trace_debug_layer
# echo "0x10" > trace_debug_level
# echo "\PPPP.AAAA.TTTT.HHHH" > trace_method_name
# echo "method-once" > trace_state
```

d. Filter out the method/opcode start/stop "AML tracer" when the specified control method is being evaluated:

```
# cd /sys/module/acpi/parameters
# echo "0x80" > trace_debug_layer
# echo "0x10" > trace_debug_level
# echo "\PPPP.AAAA.TTTT.HHHH" > trace_method_name
# echo "opcode" > trace_state
```

e. Filter out the method/opcode start/stop "AML tracer" when the specified control method is being evaluated for the first time:

```
# cd /sys/module/acpi/parameters
# echo "0x80" > trace_debug_layer
# echo "0x10" > trace_debug_level
# echo "\PPPP.AAAA.TTTT.HHHH" > trace_method_name
# echo "opcode-opcode" > trace_state
```

Note that all above method tracing facility related module parameters can be used as the boot parameters, for example:

```
acpi.trace_debug_layer=0x80 acpi.trace_debug_level=0x10 \
acpi.trace_method_name=\_SB.LID0._LID acpi.trace_state=opcode-once
```

## Interface descriptions

All method tracing functions can be configured via ACPI module parameters that are accessible at /sys/module/acpi/parameters/:

trace_method_name

The full path of the AML method that the user wants to trace.

Note that the full path shouldn't contain the trailing "_"s in its name segments but may contain "" to form an absolute path.

trace_debug_layer

The temporary debug_layer used when the tracing feature is enabled.

Using ACPI_EXECUTER (0x80) by default, which is the debug_layer used to match all "AML tracer" logs.

trace_debug_level

The temporary debug_level used when the tracing feature is enabled.

Using ACPI_LV_TRACE_POINT (0x10) by default, which is the debug_level used to match all "AML tracer" logs.

trace_state

The status of the tracing feature.

Users can enable/disable this debug tracing feature by executing the following command:

```
# echo string > /sys/module/acpi/parameters/trace_state
```

Where "string" should be one of the following:

"disable"

Disable the method tracing feature.

"enable"

Enable the method tracing feature.

ACPICA debugging messages matching "trace_debug_layer/trace_debug_level" during any method execution will be

logged.

"method"

Enable the method tracing feature.

ACPICA debugging messages matching "trace_debug_layer/trace_debug_level" during method execution of "trace_method_name" will be logged.

"method-once"

Enable the method tracing feature.

ACPICA debugging messages matching "trace_debug_layer/trace_debug_level" during method execution of "trace_method_name" will be logged only once.

"opcode"

Enable the method tracing feature.

ACPICA debugging messages matching "trace_debug_layer/trace_debug_level" during method/opcode execution of "trace_method_name" will be logged.

"opcode-once"

Enable the method tracing feature.

ACPICA debugging messages matching "trace_debug_layer/trace_debug_level" during method/opcode execution of "trace_method_name" will be logged only once.

Note that, the difference between the "enable" and other feature enabling options are:

1. When "enable" is specified, since "trace_debug_layer/trace_debug_level" shall apply to all control method evaluations, after configuring "trace_state" to "enable", "trace_method_name" will be reset to NULL.
2. When "method/opcode" is specified, if "trace_method_name" is NULL when "trace_state" is configured to these options, the "trace_debug_layer/trace_debug_level" will apply to all control method evaluations.