Introduction

The Intel Management Engine (Intel ME) is an isolated and protected computing resource (Co-processor) residing inside certain Intel chipsets. The Intel ME provides support for computer/IT management and security features. The actual feature set depends on the Intel chipset SKU.

The Intel Management Engine Interface (Intel MEI, previously known as HECI) is the interface between the Host and Intel ME. This interface is exposed to the host as a PCI device, actually multiple PCI devices might be exposed. The Intel MEI Driver is in charge of the communication channel between a host application and the Intel ME features.

Each Intel ME feature, or Intel ME Client is addressed by a unique GUID and each client has its own protocol. The protocol is message-based with a header and payload up to maximal number of bytes advertised by the client, upon connection.

Intel MEI Driver

The driver exposes a character device with device nodes /dev/meiX.

An application maintains communication with an Intel ME feature while /dev/meiX is open. The binding to a specific feature is performed by calling :c:macro:`MEI_CONNECT_CLIENT_IOCTL`, which passes the desired GUID. The number of instances of an Intel ME feature that can be opened at the same time depends on the Intel ME feature, but most of the features allow only a single instance.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 27); *backlink***
>
> Unknown interpreted text role "c:macro".

The driver is transparent to data that are passed between firmware feature and host application.

Because some of the Intel ME features can change the system configuration, the driver by default allows only a privileged user to access it.

The session is terminated calling :c:expr:`close(fd)`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 41); *backlink***
>
> Unknown interpreted text role "c:expr".

A code snippet for an application communicating with Intel AMTHI client:

In order to support virtualization or sandboxing a trusted supervisor can use :c:macro:`MEI_CONNECT_CLIENT_IOCTL_VTAG` to create virtual channels with an Intel ME feature. Not all features support virtual channels such client with answer EOPNOTSUPP.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 45); *backlink***
>
> Unknown interpreted text role "c:macro".

```
struct mei_connect_client_data data;
fd = open(MEI_DEVICE);

data.d.in_client_uuid = AMTHI_GUID;

ioctl(fd, IOCTL_MEI_CONNECT_CLIENT, &data);

printf("Ver=%d, MaxLen=%ld\n",
       data.d.in_client_uuid.protocol_version,
       data.d.in_client_uuid.max_msg_length);

[...]

write(fd, amthi_req_data, amthi_req_data_len);

[...]

read(fd, &amthi_res_data, amthi_res_data_len);
```

```
[...]
close(fd);
```

User space API

IOCTLs:

The Intel MEI Driver supports the following IOCTL commands:

IOCTL_MEI_CONNECT_CLIENT

Connect to firmware Feature/Client.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 86)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

        Usage:

        struct mei_connect_client_data client_data;

        ioctl(fd, IOCTL_MEI_CONNECT_CLIENT, &client_data);

        Inputs:

        struct mei_connect_client_data - contain the following
        Input field:

                in_client_uuid -        GUID of the FW Feature that needs
                                        to connect to.
         Outputs:
                out_client_properties - Client Properties: MTU and Protocol Version.

         Error returns:

                ENOTTY  No such client (i.e. wrong GUID) or connection is not allowed.
                EINVAL  Wrong IOCTL Number
                ENODEV  Device or Connection is not initialized or ready.
                ENOMEM  Unable to allocate memory to client internal data.
                EFAULT  Fatal Error (e.g. Unable to access user input data)
                EBUSY   Connection Already Open
```

Note: max_msg_length (MTU) in client properties describes the maximum data that can be sent or received. (e.g. if MTU=2K, can send requests up to bytes 2k and received responses up to 2k bytes).

IOCTL_MEI_CONNECT_CLIENT_VTAG:

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 121)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

        Usage:

        struct mei_connect_client_data_vtag client_data_vtag;

        ioctl(fd, IOCTL_MEI_CONNECT_CLIENT_VTAG, &client_data_vtag);

        Inputs:

        struct mei_connect_client_data_vtag - contain the following
        Input field:

                in_client_uuid -  GUID of the FW Feature that needs
                                  to connect to.
                vtag - virtual tag [1, 255]

         Outputs:
```

```
                out_client_properties - Client Properties: MTU and Protocol Version.

        Error returns:

                ENOTTY No such client (i.e. wrong GUID) or connection is not allowed.
                EINVAL Wrong IOCTL Number or tag == 0
                ENODEV Device or Connection is not initialized or ready.
                ENOMEM Unable to allocate memory to client internal data.
                EFAULT Fatal Error (e.g. Unable to access user input data)
                EBUSY  Connection Already Open
                EOPNOTSUPP Vtag is not supported
```

IOCTL_MEI_NOTIFY_SET

Enable or disable event notifications.

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 156)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

        Usage:

                uint32_t enable;

                ioctl(fd, IOCTL_MEI_NOTIFY_SET, &enable);


                uint32_t enable = 1;
                or
                uint32_t enable[disable] = 0;

        Error returns:


                EINVAL  Wrong IOCTL Number
                ENODEV  Device  is not initialized or the client not connected
                ENOMEM  Unable to allocate memory to client internal data.
                EFAULT  Fatal Error (e.g. Unable to access user input data)
                EOPNOTSUPP if the device doesn't support the feature
```

Note: The client must be connected in order to enable notification events

IOCTL_MEI_NOTIFY_GET

Retrieve event

System Message: WARNING/2 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\mei\[linux-master][Documentation][driver-api][mei]mei.rst, line 186)

Cannot analyze code. No Pygments lexer found for "none".

```
.. code-block:: none

        Usage:
                uint32_t event;
                ioctl(fd, IOCTL_MEI_NOTIFY_GET, &event);

        Outputs:
                1 - if an event is pending
                0 - if there is no even pending

        Error returns:
                EINVAL  Wrong IOCTL Number
                ENODEV  Device is not initialized or the client not connected
                ENOMEM  Unable to allocate memory to client internal data.
                EFAULT  Fatal Error (e.g. Unable to access user input data)
                EOPNOTSUPP if the device doesn't support the feature
```

Note: The client must be connected and event notification has to be enabled in order to receive an event

Supported Chipsets

82X38/X48 Express and newer

linux-mei@linux.intel.com