User-Space EC Interface (cdev)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 3)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 3)

Substitution definition contains illegal element <problematic>:

```
<problematic ids="id2" refid="id1">
    :c:type:`struct ssam_cdev_request <ssam_cdev_request>`

.. |ssam_cdev_request| replace:: :c:type:`struct ssam_cdev_request <ssam_cdev_request>`
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 4)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 4)

Substitution definition contains illegal element <problematic>:

```
<problematic ids="id4" refid="id3">
    :c:type:`enum ssam_cdev_request_flags <ssam_cdev_request_flags>`

.. |ssam_cdev_request_flags| replace:: :c:type:`enum ssam_cdev_request_flags <ssam_cdev_request_flag
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 5)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 5)

Substitution definition contains illegal element <problematic>:

```
<problematic ids="id6" refid="id5">
    :c:type:`struct ssam_cdev_event <ssam_cdev_event>`

.. |ssam_cdev_event| replace:: :c:type:`struct ssam_cdev_event <ssam_cdev_event>`
```

The `surface_aggregator_cdev` module provides a misc-device for the SSAM controller to allow for a (more or less) direct connection from user-space to the SAM EC. It is intended to be used for development and debugging, and therefore should not be used or relied upon in any other way. Note that this module is not loaded automatically, but instead must be loaded manually.

The provided interface is accessible through the `/dev/surface/aggregator` device-file. All functionality of this interface is provided via IOCTLs. These IOCTLs and their respective input/output parameter structs are defined in `include/uapi/linux/surface_aggregator/cdev.h`.

A small python library and scripts for accessing this interface can be found at https://github.com/linux-surface/surface-aggregator-module/tree/master/scripts/ssam.

Contents

- Receiving Events
- Controller IOCTLs
  - SSAM_CDEV_REQUEST
  - SSAM_CDEV_NOTIF_REGISTER

- SSAM_CDEV_NOTIF_UNREGISTER
- SSAM_CDEV_EVENT_ENABLE
- SSAM_CDEV_EVENT_DISABLE

- Structures and Enums

Receiving Events

Events can be received by reading from the device-file. The are represented by the |ssam_cdev_event| datatype.

Before events are available to be read, however, the desired notifiers must be registered via the SSAM_CDEV_NOTIF_REGISTER IOCTL. Notifiers are, in essence, callbacks, called when the EC sends an event. They are, in this interface, associated with a specific target category and device-file-instance. They forward any event of this category to the buffer of the corresponding instance, from which it can then be read.

Notifiers themselves do not enable events on the EC. Thus, it may additionally be necessary to enable events via the SSAM_CDEV_EVENT_ENABLE IOCTL. While notifiers work per-client (i.e. per-device-file-instance), events are enabled globally, for the EC and all of its clients (regardless of userspace or non-userspace). The SSAM_CDEV_EVENT_ENABLE and SSAM_CDEV_EVENT_DISABLE IOCTLs take care of reference counting the events, such that an event is enabled as long as there is a client that has requested it.

Note that enabled events are not automatically disabled once the client instance is closed. Therefore any client process (or group of processes) should balance their event enable calls with the corresponding event disable calls. It is, however, perfectly valid to enable and disable events on different client instances. For example, it is valid to set up notifiers and read events on client instance A, enable those events on instance B (note that these will also be received by A since events are enabled/disabled globally), and after no more events are desired, disable the previously enabled events via instance C.

Controller IOCTLs

The following IOCTLs are provided:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 65)

Unknown directive type "flat-table".

```
.. flat-table:: Controller IOCTLs
   :widths: 1 1 1 1 4
   :header-rows: 1

   * - Type
     - Number
     - Direction
     - Name
     - Description

   * - ``0xA5``
     - ``1``
     - ``WR``
     - ``REQUEST``
     - Perform synchronous SAM request.

   * - ``0xA5``
     - ``2``
     - ``W``
     - ``NOTIF_REGISTER``
     - Register event notifier.

   * - ``0xA5``
     - ``3``
     - ``W``
     - ``NOTIF_UNREGISTER``
     - Unregister event notifier.

   * - ``0xA5``
     - ``4``
     - ``W``
     - ``EVENT_ENABLE``
     - Enable event source.

   * - ``0xA5``
     - ``5``
     - ``W``
     - ``EVENT_DISABLE``
     - Disable event source.
```

SSAM_CDEV_REQUEST

Defined as `_IOWR(0xA5, 1, struct ssam_cdev_request)`.

Executes a synchronous SAM request. The request specification is passed in as argument of type |ssam_cdev_request|, which is then written to/modified by the IOCTL to return status and result of the request.

Request payload data must be allocated separately and is passed in via the `payload.data` and `payload.length` members. If a response is required, the response buffer must be allocated by the caller and passed in via the `response.data` member. The `response.length` member must be set to the capacity of this buffer, or if no response is required, zero. Upon completion of the request, the call will write the response to the response buffer (if its capacity allows it) and overwrite the length field with the actual size of the response, in bytes.

Additionally, if the request has a response, this must be indicated via the request flags, as is done with in-kernel requests. Request flags can be set via the `flags` member and the values correspond to the values found in |ssam_cdev_request_flags|.

Finally, the status of the request itself is returned in the `status` member (a negative errno value indicating failure). Note that failure indication of the IOCTL is separated from failure indication of the request: The IOCTL returns a negative status code if anything failed during setup of the request (`-EFAULT`) or if the provided argument or any of its fields are invalid (`-EINVAL`). In this case, the status value of the request argument may be set, providing more detail on what went wrong (e.g. `-ENOMEM` for out-of-memory), but this value may also be zero. The IOCTL will return with a zero status code in case the request has been set up, submitted, and completed (i.e. handed back to user-space) successfully from inside the IOCTL, but the request `status` member may still be negative in case the actual execution of the request failed after it has been submitted.

A full definition of the argument struct is provided below.

SSAM_CDEV_NOTIF_REGISTER

Defined as `_IOW(0xA5, 2, struct ssam_cdev_notifier_desc)`.

Register a notifier for the event target category specified in the given notifier description with the specified priority. Notifiers registration is required to receive events, but does not enable events themselves. After a notifier for a specific target category has been registered, all events of that category will be forwarded to the userspace client and can then be read from the device file instance. Note that events may have to be enabled, e.g. via the `SSAM_CDEV_EVENT_ENABLE` IOCTL, before the EC will send them.

Only one notifier can be registered per target category and client instance. If a notifier has already been registered, this IOCTL will fail with `-EEXIST`.

Notifiers will automatically be removed when the device file instance is closed.

SSAM_CDEV_NOTIF_UNREGISTER

Defined as `_IOW(0xA5, 3, struct ssam_cdev_notifier_desc)`.

Unregisters the notifier associated with the specified target category. The priority field will be ignored by this IOCTL. If no notifier has been registered for this client instance and the given category, this IOCTL will fail with `-ENOENT`.

SSAM_CDEV_EVENT_ENABLE

Defined as `_IOW(0xA5, 4, struct ssam_cdev_event_desc)`.

Enable the event associated with the given event descriptor.

Note that this call will not register a notifier itself, it will only enable events on the controller. If you want to receive events by reading from the device file, you will need to register the corresponding notifier(s) on that instance.

Events are not automatically disabled when the device file is closed. This must be done manually, via a call to the `SSAM_CDEV_EVENT_DISABLE` IOCTL.

SSAM_CDEV_EVENT_DISABLE

Defined as `_IOW(0xA5, 5, struct ssam_cdev_event_desc)`.

Disable the event associated with the given event descriptor.

Note that this will not unregister any notifiers. Events may still be received and forwarded to user-space after this call. The only safe way of stopping events from being received is unregistering all previously registered notifiers.

Structures and Enums

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst, line 204)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/uapi/linux/surface_aggregator/cdev.h
```

Docutils System Messages

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst`, **line 31**); *backlink*

Undefined substitution referenced: "ssam_cdev_event".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst`, **line 111**); *backlink*

Undefined substitution referenced: "ssam_cdev_request".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\surface_aggregator\clients\[linux-master][Documentation][driver-api][surface_aggregator][clients]cdev.rst`, **line 124**); *backlink*

Undefined substitution referenced: "ssam_cdev_request_flags".