Error and stream handling

IN THIS DOCUMENT

Error management

Camera HAL device ops functions that have a return value will all return -ENODEV / NULL in case of a serious error. This means the device cannot continue operation, and must be closed by the framework. Once this error is returned by some method, or if notify() is called with ERROR_DEVICE, only the close() method can be called successfully. All other methods will return -ENODEV / NULL.
If a device op is called in the wrong sequence, for example if the framework calls configure_streams() is called before initialize(), the device must return -ENOSYS from the call, and do nothing.
Transient errors in image capture must be reported through notify() as follows:

- The failure of an entire capture to occur must be reported by the HAL by calling notify() with ERROR_REQUEST. Individual errors for the result metadata or the output buffers must not be reported in this case.
- If the metadata for a capture cannot be produced, but some image buffers were filled, the HAL must call notify() with ERROR_RESULT.
- If an output image buffer could not be filled, but either the metadata was produced or some other buffers were filled, the HAL must call notify() with ERROR_BUFFER for each failed buffer.

In each of these transient failure cases, the HAL must still call process_capture_result, with valid output buffer_handle_t. If the result metadata could not be produced, it should be NULL. If some buffers could not be filled, their sync fences must be set to the error state.
Invalid input arguments result in -EINVAL from the appropriate methods. In that case, the framework must act as if that call had never been made.

Stream management

configure_streams

Reset the HAL camera device processing pipeline and set up new input and output streams. This call replaces any existing stream configuration with the streams defined in the stream_list. This method will be called at least once after initialize() before a request is submitted with process_capture_request().
The stream_list must contain at least one output-capable stream, and may not contain more than one input-capable stream.
The stream_list may contain streams that are also in the currently-active set of streams (from the previous call to configure_stream()). These streams will already have valid values for usage, maxbuffers, and the private pointer. If such a stream has already had its buffers registered, register_stream_buffers() will not be called again for the stream, and buffers from the stream can be immediately included in input requests.
If the HAL needs to change the stream configuration for an existing stream due to the new configuration, it may rewrite the values of usage and/or maxbuffers during the configure call. The framework will detect such a change, and will then reallocate the stream buffers, and call register_stream_buffers() again before using buffers from that stream in a request.
If a currently-active stream is not included in stream_list, the HAL may safely remove any references to that stream. It will not be reused in a later configure() call by the framework, and all the gralloc buffers for it will be freed after the configure_streams() call returns.
The stream_list structure is owned by the framework, and may not be accessed once this call completes. The address of an individual camera3streamt structure will remain valid for access by the HAL until the end of the first configure_stream() call which no longer includes that camera3streamt in the stream_list argument. The HAL may not change values in the stream structure outside of the private pointer, except for the usage and maxbuffers members during the configure_streams() call itself.
If the stream is new, the usage, maxbuffer, and private pointer fields of the stream structure will all be set to 0. The HAL device must set these fields before the configure_streams() call returns. These fields are then used by

the framework and the platform gralloc module to allocate the gralloc buffers for each stream.
Before such a new stream can have its buffers included in a capture request, the framework will call register_stream_buffers() with that stream. However, the framework is not required to register buffers for _all streams before submitting a request. This allows for quick startup of (for example) a preview stream, with allocation for other streams happening later or concurrently.

Preconditions

The framework will only call this method when no captures are being processed. That is, all results have been returned to the framework, and all in-flight input and output buffers have been returned and their release sync fences have been signaled by the HAL. The framework will not submit new requests for capture while the configure_streams() call is underway.

Postconditions

The HAL device must configure itself to provide maximum possible output frame rate given the sizes and formats of the output streams, as documented in the camera device's static metadata.

Performance expectations

This call is expected to be heavyweight and possibly take several hundred milliseconds to complete, since it may require resetting and reconfiguring the image sensor and the camera processing pipeline. Nevertheless, the HAL device should attempt to minimize the reconfiguration delay to minimize the user-visible pauses during application operational mode changes (such as switching from still capture to video recording).

Return values

- 0: On successful stream configuration
- undefined
- -EINVAL: If the requested stream configuration is invalid. Some examples of invalid stream configurations include:
  - Including more than 1 input-capable stream (INPUT or BIDIRECTIONAL)
  - Not including any output-capable streams (OUTPUT or BIDIRECTIONAL)
  - Including streams with unsupported formats, or an unsupported size for that format.
  - Including too many output streams of a certain format.
  - Note that the framework submitting an invalid stream configuration is not normal operation, since stream configurations are checked before configure. An invalid configuration means that a bug exists in the framework code, or there is a mismatch between the HAL's static metadata and the requirements on streams.
- -ENODEV: If there has been a fatal error and the device is no longer operational. Only close() can be called successfully by the framework after this error is returned.

register_stream_buffers

Register buffers for a given stream with the HAL device. This method is called by the framework after a new stream is defined by configure_streams, and before buffers from that stream are included in a capture request. If the same stream is listed in a subsequent configure_streams() call, register_stream_buffers will not be called again for that stream.
The framework does not need to register buffers for all configured streams before it submits the first capture request. This allows quick startup for preview (or similar use cases) while other streams are still being allocated.
This method is intended to allow the HAL device to map or otherwise prepare the buffers for later use. The buffers passed in will already be locked for use. At the end of the call, all the buffers must be ready to be returned to the stream. The bufferset argument is only valid for the duration of this call.
If the stream format was set to HAL_PIXEL_FORMAT_IMPLEMENTATION_DEFINED, the camera HAL should inspect the passed-in buffers here to determine any platform-private pixel format information.

Return values

- 0: On successful registration of the new stream buffers

- -EINVAL: If the streambufferset does not refer to a valid active stream, or if the buffers array is invalid.
- -ENOMEM: If there was a failure in registering the buffers. The framework must consider all the stream buffers to be unregistered, and can try to register again later.
- -ENODEV: If there is a fatal error, and the device is no longer operational. Only close() can be called successfully by the framework after this error is returned.