Request creation and submission

IN THIS DOCUMENT

Request creation and submission

construct_default_request_settings

Create capture settings for standard camera use cases. The device must return a settings buffer that is configured to meet the requested use case, which must be one of the CAMERA3_TEMPLATE_* enums. All request control fields must be included.
The HAL retains ownership of this structure, but the pointer to the structure must be valid until the device is closed. The framework and the HAL may not modify the buffer once it is returned by this call. The same buffer may be returned for subsequent calls for the same template, or for other templates.

Return values

- Valid metadata: On successful creation of a default settings buffer.
- NULL: In case of a fatal error. After this is returned, only the close() method can be called successfully by the framework.

process_capture_request

Send a new capture request to the HAL. The HAL should not return from this call until it is ready to accept the next request to process. Only one call to process_capture_request() will be made at a time by the framework, and the calls will all be from the same thread. The next call to process_capture_request() will be made as soon as a new request and its associated buffers are available. In a normal preview scenario, this means the function will be called again by the framework almost instantly.
The actual request processing is asynchronous, with the results of capture being returned by the HAL through the process_capture_result() call. This call requires the result metadata to be available, but output buffers may simply provide sync fences to wait on. Multiple requests are expected to be in flight at once, to maintain full output frame rate.
The framework retains ownership of the request structure. It is only guaranteed to be valid during this call. The HAL device must make copies of the information it needs to retain for the capture processing. The HAL is responsible for waiting on and closing the buffers' fences and returning the buffer handles to the framework.
The HAL must write the file descriptor for the input buffer's release sync fence into input_buffer->release_fence, if input_buffer is not NULL. If the HAL returns -1 for the input buffer release sync fence, the framework is free to immediately reuse the input buffer. Otherwise, the framework will wait on the sync fence before refilling and reusing the input buffer.

Return values

- 0: On a successful start to processing the capture request
- -EINVAL: If the input is malformed (the settings are NULL when not allowed, there are 0 output buffers, etc) and capture processing cannot start. Failures during request processing should be handled by calling camera3_callback_ops_t.notify(). In case of this error, the framework will retain responsibility for the stream buffers' fences and the buffer handles; the HAL should not close the fences or return these buffers with process_capture_result.
- -ENODEV: If the camera device has encountered a serious error. After this error is returned, only the close() method can be successfully called by the framework.

Miscellaneous methods

get_metadata_vendor_tag_ops

Get methods to query for vendor extension metadata tag information. The HAL should fill in all the vendor tag operation methods, or leave ops unchanged if no vendor tags are defined. The definition of vendor_tag_query_ops_t can be found in system/media/camera/include/system/camera_metadata.h.

dump

Print out debugging state for the camera device. This will be called by the framework when the camera service is asked for a debug dump, which happens when using the dumpsys tool, or when capturing a bugreport. The passed-in file descriptor can be used to write debugging text using dprintf() or write(). The text should be in ASCII encoding only.

flush

Flush all currently in-process captures and all buffers in the pipeline on the given device. The framework will use this to dump all state as quickly as possible in order to prepare for a configure_streams() call.
No buffers are required to be successfully returned, so every buffer held at the time of flush() (whether successfully filled or not) may be returned with CAMERA3_BUFFER_STATUS_ERROR. Note the HAL is still allowed to return valid (STATUS_OK) buffers during this call, provided they are successfully filled.
All requests currently in the HAL are expected to be returned as soon as possible. Not-in-process requests should return errors immediately. Any interruptible hardware blocks should be stopped, and any uninterruptible blocks should be waited on.
flush() should only return when there are no more outstanding buffers or requests left in the HAL. The framework may call configure_streams (as the HAL state is now quiesced) or may issue new requests.
A flush() call should only take 100ms or less. The maximum time it can take is 1 second.

Version information

This is available only if device version >= CAMERA_DEVICE_API_VERSION_3_1.

Return values

- 0: On a successful flush of the camera HAL.
- -EINVAL: If the input is malformed (the device is not valid).
- -ENODEV: If the camera device has encountered a serious error. After this error is returned, only the close() method can be successfully called by the framework.