



Channel Abstraction Layer API V2.0 Errata

The Channel Abstraction Layer (CAL) API V2.0 was released in January 2002. This errata details changes to the V2.0 documentation. The Channel Abstraction Layer API V2.0.1 will include these changes.

Page 15: `cal_devnum_to_name` function syntax

PROBLEM `name` parameter is described to have a type of `char *` in function summary on page 8. It is described to have a type of `uint8_t *` in `cal_g.h`.

SOLUTION `char *` is correct and is how it is defined in `cal_g.h` and implemented. The next release of the CAL specification document will be updated to use `char *` in the function syntax.

PROBLEM Error return `VSTATUS_NODEV` description references constant `CAL_MAX_DEVS`, which is not defined.

SOLUTION `CAL_MAX_DEVS` is a private constant, the example shows correct usage. The reference will be changed to “the maximum number of supported devices – 1.” In the next release of the CAL specification document.

Page 16: `cal_chanhw_open` function description

PROBLEM The documentation references `cal_g.h` for asynchronous event types and associated data parameter descriptions. The documentation should contain the events.

SOLUTION To avoid referencing the `cal_g.h` header file, the next release of the CAL specification document will include a section in the appendix listing the asynchronous events and data description supported.

PROBLEM The description for the `context` parameter is missing.

SOLUTION The following text will be added to the next release of the CAL specification document following the `event_hdl` parameter description: “`context`: A user supplied context, opaque to the CAL that will be associated with this open. The context will be passed to the caller during asynchronous event notifications. This parameter may be NULL.”

Page 19: `cal_chanhw_query` data type definitions

PROBLEM The documentation references `cal_g.h` for data descriptions. The documentation should contain the data structure returned for each query.

SOLUTION To avoid referencing the `cal_g.h` header file, the next release of the CAL specification document will include a section in the appendix to reference the appropriate query specific data structures returned by the CAL.



Page 28: `Cal_event_hdlr_t` definition

PROBLEM The prototype for asynchronous event notification callback uses a 32-bit type for event specification. The `cal_chanhw_set_async_mask` function uses a 64-bit type for setting the asynchronous event notification mask.

SOLUTION The appropriate data type should be `uint64_t`. The next release of the CAL specification document will be updated to use a 64-bit parameter in the `Cal_event_hdlr_t` callback prototype.

Page 30: CAL Management Configuration API Summary

PROBLEM Deprecated argument still listed in summary.

SOLUTION The description on page 73 and `cal_g.h` header file show the correct argument list. The loop back parameter was deprecated prior to the 2.0 API release. The next release of the CAL specification document will be updated to remove this parameter from the function summary, making it consistent with the function description and `cal_g.h` header file.

Page 75: CAL Transport API Summary

PROBLEM `cal_addrvec_destoy` function name in summary is inconsistent with description.

SOLUTION The correct name is `cal_addrvec_destroy`; the next release of the CAL specification document will be updated to use the name `cal_addrvec_destroy` in the summary.

PROBLEM For the function `cal_cq_query`, the parameter `void *context` is inconsistent with function description.

SOLUTION The correct type for the parameter is in the description and `cal_g.h` header file. The function summary will be updated to specify `void **context` in the next release of the CAL specification document.

PROBLEM For the function `cal_mem_region_modify_phys`, the parameter `uint64_t num_phys_bufs` is named inconsistently with the function description and `cal_g.h` header file.

SOLUTION The parameter name will be changed to `uint64_t nbufs` to be consistent with the function description in the next release of the CAL specification document.

PROBLEM For the function `cal_mem_region_reg_shared`, the parameter list is inconsistent with the function description and header file.

SOLUTION The function description and header file are correct. The `phys_buf_list` and `num_phys_bufs` parameters will be deleted from the summary in the next release of the CAL specification document.

PROBLEM For the function `cal_mem_win_post_find`, the parameter `uint64_t length` is named inconsistently with the function description and `cal_g.h` header file.

SOLUTION The parameter name will be changed to `uint64_t len` to be consistent with the function description in the next release of the CAL specification document.



Page 83: `cal_addrvec_query` parameter list

PROBLEM The `cal_addrvec_query` parameter list specified is inconsistent with the function summary and the `cal_g.h` header file. It is missing the `Cal_handle_t *pd_h` parameter.

SOLUTION The documentation for `cal_addrvec_query` description will be updated in the next release of the CAL specification document to be consistent with the function summary and `cal_g.h` header file. The following text will be added to the parameter description as well: “`pd_h`: The location to return the protection domain associated with this address vector.”

Page 88: `cal_cq_query` parameter list

PROBLEM The handle parameter `Cal_handle_t *cq_h` is inconsistent with the function summary and `cal_g.h` header file. It should not be passed by reference.

SOLUTION The documentation for `cal_cq_query` will be updated in the next release of the CAL specification document to be consistent with the summary and `cal_g.h` header file.

Page 92: `cal_cq_poll` error return

PROBLEM The error return `VSTATUS_QEMPTY` is not documented, and is returned when the CQ is empty.

SOLUTION The `VSTATUS_QEMPTY` error return will be added to the next release of the CAL API specification document.

Page 94: `cal_qp_create` parameter list

PROBLEM The syntax for the function lists the parameter type for QP attributes as `Cal_attrib_t` which is inconsistent with the summary and `cal_g.h` header file.

SOLUTION The correct data type is `Cal_qp_attrib_t` as defined in the summary, description, and header file. The syntax description will be updated in the next release of the CAL API specification document.



Page 107: `cal_qp_attach_mcast` parameter list

PROBLEM The parameter `Gid_t *addr` is incorrect and creates an unnecessary data abstraction.

SOLUTION The parameter should be `uint8_t addr[16]`. The summary, description, and `cal_g.h` header file will be updated to reflect this in the next release of the CAL API specification document.

Page 108: `cal_qp_detatch_mcast` parameter list

PROBLEM The parameter `Gid_t *addr` is incorrect and creates an unnecessary data abstraction.

SOLUTION The parameter should be `uint8_t addr[16]`. The summary, description, and `cal_g.h` header file will be updated to reflect this in the next release of the CAL API specification document.

Page 110: `cal_qp_create_special` description

PROBLEM The name for the channel hardware handle is inconsistent with the function summary and `cal_g.h` header file.

SOLUTION The syntax will be changed from `Cal_handle_t ch_h` to `Cal_handle_t chw_h` in the next release of the CAL specification document.

PROBLEM The name for the requested QP attribute parameter is inconsistent with the function summary and `cal_g.h` header file.

SOLUTION The syntax will be changed from `Cal_qp_attrib_t *request_attrib` to `Cal_qp_attrib_t *attrib` in the next release of the CAL specification document.

PROBLEM The name for the actual QP attribute parameter is inconsistent with the function summary and `cal_g.h` header file.

SOLUTION The syntax will be changed from `Cal_qp_attrib_t *actt_attrib` to `Cal_qp_attrib_t *actual_attrib` in the next release of the CAL specification document.

Page 119: `cal_mem_region_query` parameter list

PROBLEM The syntax for the parameter `uint64_t io_virt_addr` is inconsistent with the function summary and `cal_g.h` header file. In addition, actual protection bounds may be returned for both the local and remote protection bounds and only a single start/length return is allowed.

SOLUTION The parameter should be passed by reference as documented in the function summary and `cal_g.h` header file. The prototype should be changed to:

```
Status_t cal_mem_region (Cal_handle_t mr_h,
                        uint64_t *loc_io_virt_addr,
                        uint64_t *loc_length,
                        uint64_t *rem_io_virt_addr,
                        uint64_t *rem_length,
                        Cal_handle_t *pd_h,
                        Cal_access_t *access,
                        Cal_mkey_t *lkey,
                        Cal_mkey_t *rkey);
```

The summary, description, and `cal_g.h` header file will be updated in the next release of the CAL specification document.



Page 129, 131, 139 Cal_wr_t definition

PROBLEM The definition of `Cal_wr_t` is inconsistent with the `cal_g.h` header file.

SOLUTION The `cal_g.h` header file is correct. The next release of the CAL specification document will be updated to be consistent with the `cal_g.h` header file.

Cal_global_attrib_t scatter/gather list count

PROBLEM The maximum scatter/gather entries should provide the flexibility to be different for sends and receives.

SOLUTION Added additional fields to `Cal_global_attrib_t`:

```
uint32_t max_sndsg_wr; /* Max scatter/gather per */
                        /* send work request, excl. RD */
uint32_t max_rcvsg_wr; /* Max scatter/gather per */
                        /* rcv work request, excl. RD */
```

The next release of the CAL API specification document will be updated to reflect these new fields.

Insufficient error return status

PROBLEM Function error returns do not always provide sufficient resolution to map to IB error returns associated with IB Verb implementations.

SOLUTION The return status for each transport API function will be examined, and where required, will be updated to allow the desired mapping. Additional return codes will be added in the next official release of the CAL API specification document.

Overview of event and completion handler usage required

PROBLEM High-level overview of CAL event and completion handler not included in the document.

SOLUTION The next release of the CAL specification document will be updated to describe asynchronous and completion handlers usage, and expected return values.

PROBLEM It is unclear what should be returned by asynchronous and completion handlers.

SOLUTION The need for the handler return value has been deprecated by the CAL event selection and notification paradigm. The prototypes will be modified to be `void` functions in the next release of the CAL specification document.



[This page intentionally left blank.]