The OpenVX™ Node Send Command Extension

The Khronos® OpenVX Working Group, Contributor: Jesse Villarreal

Version 1.0, Wed, 15 Oct 2025 22:18:00 +0000: Git branch information not available

Table of Contents

Chapter 1. Extension to support sending asynchronous commands to nodes

1.1. Purpose

This document details an extension to any OpenVX version from 1.1 to 1.3, and references some APIs and symbols that may be found in those APIs: https://www.khronos.org/registry/OpenVX/.

This extension is intended to define support for the application to send asynchronous commands to nodes within OpenVX.

1.2. Example Use Cases

- Automotive camera-based system: When the application detects that the speed of the car has crossed a threshold, it may want to send a command to a specific node in the graph to modify the camera frame rate, or algorithmic tuning parameter accordingly. Since this information comes from outside of the OpenVX graph at a very infrequent rate, there needs to be a means by which the application can efficiently communicate this information to the node in the graph.
- If a node has triggered an error event, then the application may want to query the node which triggered the event to get more information about the type of error so that it can take corrective action.
- An application may want to periodically trigger and report results of some kind of safety diagnostic for a node.

1.3. Background

Prior to this extension, the only standard way that an application has to send commands to an OpenVX node is through input and output parameters.

Node Input Parameters

- For example, if an application wanted to modify the behaviour of a node at run-time, it would need to first ensure that there is an input parameter to the node which can contain the control commands that the application needs to update. Then, at run-time, in between graph process calls, the application can call the specific input object map or copy calls to update the input data object.

Node Output Parameters

- Likewise, if the application needed to get some information from the node, it would need to first ensure that there is an output parameter to the node which the node can fill for every frame to contain any information that the application may need to access.

1.3.1. Limitations in Existing Specification

If the application only needs to send commands, or receive node specific information rarely, the node has to be instrumented with these extra parameters to always consider the input and write the output, even when this is not needed by the application. Depending on the cost of consuming this input and/or providing this output, this could be a very inefficient use of resources.

Furthermore, when pipelining is used, the application is not able to write to or read from a node input or output unless the object is treated as a graph parameter. This further puts additional burden on the application to continuously enqueue and dequeue these parameters when it most of the time may not need to since the input and outputs only need to be rarely asynchronously accessed.

1.4. Summary

In summary, this extension supplies functions to:

- Register a new command callback function pointer to a user kernel.
- Send a command to a node, including any replicas it may have

1.5. Acknowledgements

This specification would not be possible without the contributions from this partial list of the following individuals from the Khronos Working Group and the companies that they represented at the time:

- Kiriti Nagesh Gowda - AMD
- Viktor Gyenes - AI Motive
- Raphael Cano - Robert Bosch GmbH
- Stephen Ramm - ETAS (Robert Bosch GmbH)
- Jesse Villarreal - TI

Chapter 2. Module Documentation

Macros

- VX_TIMEOUT_NO_WAIT
- VX_TIMEOUT_WAIT_FOREVER
- VX_CONTROL_CMD_SEND_TO_ALL_REPLICATED_NODES
- VX_NODE_NUM_WITH_REPLICAS

Typedefs

- vx_kernel_command_f

Functions

- vxAddCommandToKernel
- vxNodeSendCommand

2.1. Macros

2.1.1. VX_TIMEOUT_NO_WAIT

Constant to indicate that a timeout parameter to an api should check and not wait for the call to complete

```
#define VX_TIMEOUT_NO_WAIT (0u)
```

2.1.2. VX_TIMEOUT_WAIT_FOREVER

Constant to indicate that a timeout parameter to an api should wait forever for the call to complete

```
#define VX_TIMEOUT_WAIT_FOREVER ((vx_uint32)0xFFFFFFFFU)
```

2.1.3. VX_CONTROL_CMD_SEND_TO_ALL_REPLICATED_NODES

When sending a control command to a replicated node, this can be used to send control command to all replicated node.

```
#define VX_CONTROL_CMD_SEND_TO_ALL_REPLICATED_NODES ((vx_uint32)0xFFFFFFFFU)
```

2.1.4. VX_NODE_NUM_WITH_REPLICAS

Indicates the number of nodes associated with this node, including first_node and its replicas. A value of '0' indicates that the node has been optimized away; a value of '1' indicates that the node is not replicated; a value greater than 1 indicates that the node is replicated. Read-only. Use a vx_uint32 parameter.

```
#define VX_NODE_NUM_WITH_REPLICAS (VX_ATTRIBUTE_BASE(VX_ID_KHRONOS, VX_TYPE_NODE) + 0xA)
```

2.2. Typedefs

2.2.1. vx_kernel_command_f

[REQ-NC01] The pointer to the control command function for the kernel.

```
typedef vx_status (*vx_kernel_command_f)(
            vx_node node,
            vx_uint32 node_cmd_id,
            const vx_reference *ref,
            vx_uint32 num_refs);
```

[REQ-NC02] It is the responsibility of this callback to verify that the command passed and type and size of the references that are passed are valid prior to processing.

Parameters

- [in] *node* - The handle to the node that contains this kernel.
- [in] *node_cmd_id* - The kernel-specific command identifier for the node.
- [in] *ref* - The array of references.
- [in] *num_refs* - The number of references in the refs array.

Returns: An error code describing the status of the command .

2.3. Functions

2.3.1. vxAddCommandToKernel

[REQ-NC03] Allows users to add a control command callback to the kernel.

```
vx_status vxAddCommandToKernel(
    vx_kernel                                   kernel,
    vx_kernel_command_f                         command_func_ptr);
```

This function can be optionally called when creating a user kernel to register a command callback function with the kernel

Parameters

- [in] *kernel* - The reference to the kernel added with vxAddUserKernel.
- [in] *command_func_ptr* - The process-local function pointer to be invoked when application calls vxNodeSendCommand.

Returns: A vx_status_e enumerated value.

Return Values

- VX_SUCCESS [REQ-NC04] Command callback is successfully set on kernel; any other value indicates failure .
- VX_ERROR_INVALID_REFERENCE - kernel is not a valid vx_kernel reference.

- VX_ERROR_INVALID_PARAMETERS - If the parameter is not valid for any reason.

Precondition: vxAddUserKernel

2.3.2. vxNodeSendCommand

[REQ-NC05] Send node-specific control command

```
vx_status vxNodeSendCommand(
    vx_node                                     node,
    vx_uint32                                   replicate_nodex_idx,
    vx_uint32                                   node_cmd_id,
    vx_reference                                ref[],
    vx_uint32                                   num_refs,
    vx_uint32                                   timeout);
```

This is used to send a specific control command to the specified node asynchronously. Not all nodes support commands, so refer to the node documentation for the specific control command.

- [REQ-NC06] Note that this blocks for at most 'timeout' milli-seconds; i.e. this returns either when the command execution finishes or when the timeout occurs, whichever occurs first.
- [REQ-NC07] Multiple commands can be sent to same or different nodes from different threads.

Parameters

- [in] *node* - Reference of the node to which this command is to be sent.
- [in] *replicate_nodex_idx*
  - [REQ-NC08] In case of a non-replicated node this should be 0. For a replicated node this is the index of the node replica to which the command is targeted.
  - [REQ-NC09] To send same command to all replicated nodes use VX_CONTROL_CMD_SEND_TO_ALL_REPLICATED_NODES
- [in] *node_cmd_id* - The Node-specific control command id
  - [REQ-NC10]Node-specific control command id, refer to node-specific documentation
- [inout] *ref* - List of references, they can be any OpenVX object, created using the create API
  - [REQ-NC11] This is a list of references, required as parameters for this control command. They can be any OpenVX object, created using the create API. They are bidirectional parameters, can be used for INPUT, OUTPUT or both. Refer to node documentation to get details about the parameters required for given control command.
- [in] *num_refs* - Number of valid entries/references in ref[] array
  - [REQ-NC12] Number of valid entries/references in ref[] array shall valid depending on the given control command
- [in] *timeout* - Timeout in units of msecs,
  - [REQ-NC13] Timeout in units of msecs, use VX_TIMEOUT_WAIT_FOREVER to wait forever 5 **Returns:** A vx_status_e enumerated value.

Return Values

- `VX_SUCCESS` **`[REQ-NC14]`** No errors; any other value indicates failure.
- `VX_ERROR_INVALID_NODE` - if the node parameter is not a valid node.
- `VX_ERROR_GRAPH_NOT_VERIFIED` - if the graph that the node is a part of is not yet verified
- `VX_ERROR_INVALID_REFERENCE` - if one or more of the ref parameter references is not a valid reference.
- `VX_ERROR_TIMEOUT` - if the operation exceeds the timeout value.
- `VX_ERROR_INVALID_PARAMETERS` - if any of the other parameters are not valid or out of bounds

Precondition: `vxVerifyGraph`