



The OpenVX™ Installable Client Driver Loader Extension

The Khronos OpenVX Working Group, Editors: Radhakrishna Giduthuri, Xin
Wang

Version 1.0.1 (provisional), Wed, 15 Aug 2018 06:03:14 +0000

Table of Contents

1. The OpenVX Installable Client Driver Loader Extension	2
1.1. Overview	2
1.2. Dependencies	2
1.3. External Interface	2
1.4. Inferring Vendor ICD Calls from Arguments	2
1.5. Vendor Enumerations on Linux	3
1.6. Vendor Enumerations on Android	3
1.7. Vendor Enumerations on Windows	4
1.8. ICD Compatible Khronos Sample Implementation	4
1.9. Sample Implementation of ICD Loader	5
1.10. Updates to ICD Loader source code	5
1.11. Contributors	6
2. Module Documentation	7
2.1. OpenVX ICD Loader API	7
2.1.1. Typedefs	7
2.1.2. Functions	7



Copyright 2013-2018 The Khronos Group Inc.

This specification is protected by copyright laws and contains material proprietary to Khronos. Except as described by these terms, it or any components may not be reproduced, republished, distributed, transmitted, displayed, broadcast or otherwise exploited in any manner without the express prior written permission of Khronos.

This specification has been created under the Khronos Intellectual Property Rights Policy, which is Attachment A of the Khronos Group Membership Agreement available at www.khronos.org/files/member_agreement.pdf. Khronos Group grants a conditional copyright license to use and reproduce the unmodified specification for any purpose, without fee or royalty, EXCEPT no licenses to any patent, trademark or other intellectual property rights are granted under these terms. Parties desiring to implement the specification and make use of Khronos trademarks in relation to that implementation, and receive reciprocal patent license protection under the Khronos IP Policy must become Adopters and confirm the implementation as conformant under the process defined by Khronos for this specification; see <https://www.khronos.org/adopters>.

Khronos makes no, and expressly disclaims any, representations or warranties, express or implied, regarding this specification, including, without limitation: merchantability, fitness for a particular purpose, non-infringement of any intellectual property, correctness, accuracy, completeness, timeliness, and reliability. Under no circumstances will Khronos, or any of its Promoters, Contributors or Members, or their respective partners, officers, directors, employees, agents or representatives be liable for any damages, whether direct, indirect, special or consequential damages for lost revenues, lost profits, or otherwise, arising from or in connection with these materials.

Khronos and OpenVX are trademarks of The Khronos Group Inc. OpenCL is a trademark of Apple Inc., used under license by Khronos. All other product names, trademarks, and/or company names are used solely for identification and belong to their respective owners.

Chapter 1. The OpenVX Installable Client Driver Loader Extension

1.1. Overview

The `vx_khr_icd` extension defines a simple mechanism through which the Khronos installable client driver loader (ICD Loader) may expose multiple separate vendor installable client drivers (Vendor ICDs) for OpenVX. An application written against the ICD Loader will be able to access all `vx_platform` exposed by all vendor implementations with the ICD Loader acting as a demultiplexer.

1.2. Dependencies

OpenVX 1.0.1 or later

1.3. External Interface

The ICD Loader interface can be accessed by application using:

```
#include <VX/vx_khr_icd.h>
```

The `VX/vx_khr_icd.h` includes definition of a new object `vx_platform` and new functions `vxIcdGetPlatforms`, `vxQueryPlatform`, `vxCreateContextFromPlatform`. An ICD compatible vendor implementation is required to implement the function `vxCreateContextFromPlatform`. And the implementation of `vxIcdGetPlatforms`, `vxQueryPlatform`, and `struct _vx_platform` will be part of ICD Loader source. See [OpenVX ICD Loader API](#) for further details.

1.4. Inferring Vendor ICD Calls from Arguments

At every OpenVX function call, the ICD Loader infers the Vendor ICD function to call from the ICD compatible object that is passed as the first argument. All OpenVX objects are said to be ICD compatible if the struct `_vx_reference` contains a placeholder for `vx_platform` as its first field, as shown below:

```
struct _vx_reference {
    struct _vx_platform * platform;
    // ... remainder of internal data
};
```

The structure `_vx_platform` has a function pointer dispatch table which is used to make direct calls to a particular vendor implementation. All objects created from ICD compatible objects must be ICD compatible.

Functions which do not take ICD compatible object or a pointer to ICD compatible object as its first

argument needs to be implemented by ICD Loader. The OpenVX functions that are required for an implementation in ICD Loader source are:

```
void* vxFormatImagePatchAddress1d(
    void* ptr,
    vx_uint32 index,
    const vx_imagepatch_addressing_t* addr);
```

```
void* vxFormatImagePatchAddress2d(
    void* ptr,
    vx_uint32 x,
    vx_uint32 y,
    const vx_imagepatch_addressing_t* addr);
```

```
vx_context vxCreateContext(void);
```

The ICD Loader's `vxCreateContext` implementation is required to pick the default platform and to call the vendor specific implementation of `vxCreateContextFromPlatform`.

The ICD Loader's `vxHint` implementation is required to check the OpenVX version of the vendor implementation and handle function signature changes between OpenVX 1.0.1 and OpenVX 1.1.

1.5. Vendor Enumerations on Linux

To enumerate vendor ICDs on Linux, the ICD Loader scans the files under `/etc/OpenVX/vendors`. For each file in this path, the ICD Loader opens the file as a text file. The expected format for the file is a single line of text which specifies the Vendor ICD's library. If the Vendor ICD comes with a separate library for immediate mode functions (VXU), the expected format for the file is a single line of text with OpenVX and VXU libraries separated by semi-colon(;) in that order.

The ICD Loader will attempt to open that file as a shared object using `dlopen()`. Note that the library specified may be an absolute path or just a file name.

EXAMPLE

```
If the following file exists
    /etc/OpenVX/vendors/VendorA.icd
and contains the text
    libopenvx.so;libvxu.so
then the ICD Loader will load the libraries "libopenvx.so" and "libvxu.so"
```

1.6. Vendor Enumerations on Android

To enumerate vendor ICDs on Android, the ICD Loader scans the files under

`/system/vendor/Khronos/OpenVX/vendors/`. For each file in this path, the ICD Loader opens the file as a text file. The expected format for the file is a single line of text which specifies the Vendor ICD's library. If the Vendor ICD comes with a separate library for immediate mode functions (VXU), the expected format for the file is a single line of text with OpenVX and VXU libraries separated by semi-colon(;) in that order.

The ICD Loader will attempt to open that file as a shared object using `dlopen()`. Note that the library specified may be an absolute path or just a file name.

EXAMPLE

```
If the following file exists
    /system/vendor/Khronos/OpenVX/vendors/VendorA.icd
and contains the text
    libopenvx.so
then the ICD Loader will load the library "libopenvx.so"
```

1.7. Vendor Enumerations on Windows

To enumerate Vendor ICDs on Windows, the ICD Loader scans the values in the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Khronos\OpenVX\Vendors`. For each value in this key which has `DWORD` data set to 0, the ICD Loader opens the dynamic link library specified by the name of the value using `LoadLibraryA`. If the Vendor ICD comes with a separate library for immediate mode functions (VXU), the expected format for the name of the value is a single line of text with OpenVX and VXU libraries separated by semi-colon(;) in that order.

EXAMPLE

```
If the registry contains the following value
    [HKEY_LOCAL_MACHINE\SOFTWARE\Khronos\OpenVX\Vendors]
    "c:\\vendor_a\\openvx.dll;c:\\vendor_a\\vxu.dll"=dword:00000000
then the ICD will open the libraries "c:\\vendor_a\\openvx.dll" and "c:\\vendor_a\\vxu.dll"
```

1.8. ICD Compatible Khronos Sample Implementation

To make the sample implementation compatible with ICD implementation, the following two changes are **required**

1. Add "struct _vx_platform * platform;" as first field to "struct _vx_reference"
2. Every derived reference should copy "platform" from it's parent: add "ref->platform = context ? context->base.platform : NULL;" statement to vxInitReference()
3. Create a new vxCreateContextFromPlatform() which initializes context->base.platform with the function argument and performs same functionality as vxCreateContext().

1.9. Sample Implementation of ICD Loader

An implementation of ICD Loader is available in `vx_khr_icd` folder of sample implementation tree. Use cmake to build ICD Loader library to a static library with the name "openvx". Applications that use ICD Loader library can use any ICD compatible vendor implementation picked during run-time.

Example: Build and Run Conformance Tests using ICD Loader

```
# Build ICD Loader sample implementation from vx_khr_icd folder
% pushd <path-to-sample-implementation-trunk>
% export OPENVX_DIR=$PWD
% popd
% mkdir -p build/vx_khr_icd
% cd build/vx_khr_icd
% cmake $OPENVX_DIR/vx_khr_icd
% make
% export VX_KHR_ICD_LIB=$PWD
% cd ../../

# Build OpenVX Conformance Tests using ICD Loader
% mkdir -p build/conformance_tests
% cd build/conformance_tests
% cmake -DOPENVX_INCLUDES=$OPENVX_DIR/include \
        -DOPENVX_LIBRARIES=$VX_KHR_ICD_LIB/libopenvx.a\;pthread\;dl\;m \
        $OPENVX_DIR/conformance_tests
% make

# Run Conformance Tests
# Note: use of sample implementation requires LD_LIBRARY_PATH to be set properly
% export VX_TEST_DATA_PATH=$OPENVX_DIR/conformance_tests/test_data
% <build binary path>/vx_test_conformance
```

1.10. Updates to ICD Loader source code

The sample implementation tree has a python script `vx_khr_icd/vx_khr_icd.py` to update ICD Loader source code from OpenVX header files in `include/VX` folder.

```
# To update vx_dispatch.h, vx_dispatch.c, and ../include/VX/vx_khr_icd.h files
# with the API in VX/vx_api.h, VX/vx_nodes.h, VX/vxu.h, and VX/vx_compatibility.h,
# run below command from vx_khr_icd folder
% python vx_khr_icd.py \
    ../include/VX/vx_api.h ../include/VX/vx_nodes.h \
    ../include/VX/vxu.h \
    ../include/VX/vx_compatibility.h

# To create ICD Loader source code for OpenVX 1.0.1:
# - unzip extract openvx_sample_1.0.1.tar.bz2 in openvx_sample folder
# - copy vx_khr_icd folder into openvx_sample folder
# - run below command from openvx_sample/vx_khr_icd folder
% python vx_khr_icd.py
    ../include/VX/vx_api.h ../include/VX/vx_nodes.h \
    ../include/VX/vxu.h
```

1.11. Contributors

- Radhakrishna Giduthuri (radha.giduthuri@amd.com)

Chapter 2. Module Documentation

2.1. OpenVX ICD Loader API

The OpenVX Installable Client Driver (ICD) Loader API.

The `vx_khr_icd` extension provides a mechanism for vendors to implement Installable Client Driver (ICD) for OpenVX. The OpenVX ICD Loader API provides a mechanism for applications to access these vendor implementations.

Typedefs

- `vx_platform`

Functions

- `vxCreateContextFromPlatform`
- `vxIcdGetPlatforms`
- `vxQueryPlatform`

2.1.1. Typedefs

`vx_platform`

Platform handle of an implementation.

```
typedef struct _vx_platform *vx_platform;
```

2.1.2. Functions

`vxCreateContextFromPlatform`

Creates a `vx_context` from a `vx_platform`.

```
NOAPI vx_context vxCreateContextFromPlatform(  
    vx_platform platform);
```

This creates a top-level object context for OpenVX from a platform handle.

Returns: The reference to the implementation context `vx_context`. Any possible errors preventing a successful creation should be checked using `vxGetStatus`.

`vxIcdGetPlatforms`

Queries list of available platforms.

```
NOAPI vx_status vxIcdGetPlatforms(
    vx_size          capacity,
    vx_platform[]    platform[],
    vx_size*         pNumItems);
```

Parameters

- **[in]** *capacity* - Maximum number of items that *platform[]* can hold.
- **[out]** *platform[]* - List of platform handles.
- **[out]** *pNumItems* - Number of platform handles returned.

Returns: A `vx_status_e` enumeration.

Return Values

- `VX_SUCCESS` - No errors.
- `VX_FAILURE` - If no platforms are found.

vxQueryPlatform

Queries the platform for some specific information.

```
NOAPI vx_status vxQueryPlatform(
    vx_platform    platform,
    vx_enum        attribute,
    void*          ptr,
    vx_size        size);
```

Parameters

- **[in]** *platform* - The platform handle.
- **[in]** *attribute* - The attribute to query. Use one of the following: `VX_CONTEXT_VENDOR_ID`, `VX_CONTEXT_VERSION`, `VX_CONTEXT_EXTENSIONS_SIZE`, `VX_CONTEXT_EXTENSIONS`.
- **[out]** *ptr* - The location at which to store the resulting value.
- **[in]** *size* - The size in bytes of the container to which *ptr* points.

Returns: A `vx_status_e` enumeration.

Return Values

- `VX_SUCCESS` - No errors.
- `VX_ERROR_INVALID_REFERENCE` - If the platform is not a `vx_platform`.
- `VX_ERROR_INVALID_PARAMETERS` - If any of the other parameters are incorrect.
- `VX_ERROR_NOT_SUPPORTED` - If the attribute is not supported on this implementation.