Intel® Media SDK for Linux* Servers Dispatcher

Overview

This document describes the design of the Intel® Media SDK for Linux* Servers dispatcher.

The dispatcher is a layer that lies between applications and Intel Media SDK implementations. Upon initialization, the dispatcher locates the appropriate platform-specific SDK implementation. If there is none, it will select the software SDK implementation. The dispatcher will redirect subsequent function calls to the same functions in the selected SDK implementation.

We recommend that applications always use the Intel Media SDK functions through a dispatcher since platform-specific SDK implementations or software SDK implementations may have different names and reside in separate locations. The dispatching process represents a unique way to locate these SDK implementations.

To maintain forward and backward compatibility, we do not recommend modifying the dispatcher in any way, except in the following cases:

- Debugging in the case that the dispatcher fails to locate an SDK implementation
- Compiling the dispatcher library with a compiler that the Intel Media SDK does not support
- Rebuilding the dispatcher static library to be a dynamically linked library (shared object) for modular design
- Tailoring redirected function entries to those applications actually used for a smaller footprint

SDK Implementation Registration

Both *software* and *platform-specific* SDK implementations installed on target system in the predefined folders.

Software SDK Implementations

Software SDK implementations are installed on the system in the:

Path	Platform type
`/opt/intel/mediasdk/lib64/0000/0000`	for Intel® 64 architecture

Platform-Specific SDK Implementations

Platform-specific SDK implementations are installed on the system in the:

Path	Platform type
`/opt/intel/mediasdk/lib64/<vendor_id>/<device_id>`	for Intel® 64 architecture

where <vendor_id> and <device_id> are IDs of the GPUs supported by Media SDK.

The dispatcher locates SDK implementations by their reserved names:

Library	Platform-Specific SDK Implementation
`libmfxhw64-p.so.<mj>.<mn>`	Reserved hardware library for Intel® 64 architecture

where p stands for **P**roduction build of the library, <mj> - major version of the library's API, <mn> - minor version of the library's API.

Example 1 illustrates an example of registered SDK platform-specific implementations in the file structure:

```
$ls /opt/intel/mediasdk/lib64/8086/0162/
libmfxhw64-p.so.1.8
```

Example 1: SDK Platform-Specific Implementations in the file structure

The Dispatching Procedure

The following describes the dispatching procedure for regular (non-plugin) components:

1. Enumerate graphics adapters on the system and get their device and vendor IDs. This is being done by searching for the adapter devices in the `/sys/bus/pci/devices` folder.
2. Search for the SDK Platform-Specific implementation in the folder `/opt/intel/mediasdk/lib<arch>/<vendor_id>/<device_id>` where vendor/device IDs pair correlates with the **`MFXInit`** parameter as follows:

`MFXInit` Initialization Parameter	Device & Vendor Identifiers
`MFX_IMPL_SOFTWARE`	Use `DeviceId`=0000 and `VendorId`=0000
`MFX_IMPL_HARDWARE` or `MFX_IMPL_AUTO`	First adapter in enumeration thru the adapters in `/sys/bus/pci/devices`.
`MFX_IMPL_HARDWARE2` or `MFX_IMPL_HARDWARE3` or `MFX_IMPL_HARDWARE4`	Second, third or fourth display adapter in enumeration thru the adapters in `/sys/bus/pci/devices`.
`MFX_IMPL_HARDWARE_ANY`	Try to load SDK implementation for adapters one by one. Stop at the first implementation successfully initialized

3. Searching for the SDK *platform-specific* implementation in the folder `/opt/intel/mediasdk/lib<arch>/<vendor_id>/<device_id>` defined on the previous step is done as follows:
    - Load SDK implementation with the requested major API version and maximum minor API version which is higher or equal to requested one
4. Search the default SDK implementations in the default system paths by first searching for *platform-specific* one (`libmfxhw64.so`) and then *software* (`libmfxsw64.so`).

Plug-in Loading

DLL plug-in loading functionality is implemented on dispatcher level. Plug-in is loading in next steps:

1. When application calls **`MFXVideoUSER_Load`** dispatcher firstly looks in global configuration file `/opt/intel/mediasdk/plugins/plugins.cfg` for specified by application plug-in uid.
2. If such uid is found then dispatcher reads plug-in version Vplg and plug-in API version Vapi from registry.

3. Dispatcher compares plug-in version specified by application Vapp with plug-in version. If Vplg<Vapp, dispatcher discards this plug-in and continues search.
4. Dispatcher compares plug-in API version with library version Vlib. Note that dispatcher uses actual version of the loaded library, not the version provided by the application during **`MFXInit`** call.
5. If Vapi is not equal to Vlib, dispatcher discards this plug-in and continues search.
6. Dispatchers creates plug-in by calling **`CreatePlugin`** function. If function fails, dispatcher discards this plug-in and continues search.
7. Dispatcher registers plug-in by calling **`MFXVideoUSER_Register`** function and returns control back to the application.
8. If dispatcher has not been able to load plug-in from registry, it continue search in local application folder.
9. Dispatcher looks for folder with required uid. If required folder does not exist, dispatcher stops search and returns error to the application.
10. If required folder has been found, dispatcher reads plugin.cfg file and extracts plug-in version Vplg, plug-in API version Vapi and file name from it.
11. Dispatcher checks versions and creates plug-in as has been described on steps 3 – 7.
12. If all steps above fail, dispatcher returns error back to the application.

Additional Recommendations

To ensure SDK API compatibility, unless applications *explicitly* specify a version of an SDK implementation, we recommend that a dispatcher always match the SDK API version, during session initialization (**`MFXInit`**), with the latest released SDK API version.

Build Instructions

To build dispatcher go to the `opensource/mfx_dispatch` directory and execute following command sequence:

```
$ mkdir build
$ cd build
$ cmake -D__ARCH:STRING=intel64 ../
$ make
```

At the moment only make files generator for UNIX-like systems is supported. Specifying architecture is mandatory. Possible values are intel64 and ia32. Binaries will appear in the folder `__lib`; for example:

```
$ ls -1 $MFX_HOME/opensource/mfx_dispatch/build/__lib
libdispatch_trace.lib
libmfx.lib
```

Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

Intel, the Intel logo, Intel Core, Intel 64, Intel HD Graphics, Intel Media Software Development Kit (Intel Media SDK) are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

*Other names and brands may be claimed as the property of others.