Rules of the Road

- DO NOT share any confidential information or trade secrets with the group
- DO keep the discussion at a High Level
  - Focus on the specific Agenda topics
  - We are asking for feedback on features for the oneAPI specification (e.g. requirements for functionality and performance)
  - We are NOT asking for feedback on any implementation details
- Please submit any implementation feedback in writing on Github in accordance with the Contribution Guidelines at spec.oneapi.com. This will allow Intel to further upstream your feedback to other standards bodies, including The Khronos Group SYCL* specification.

oneAPI Technical Advisory Board Meeting:

Extension Naming

09-23-2020

Greg Lueck

oneAPI

Why Define an Extension Naming Pattern

- Avoid conflicts between extensions from different vendors
- Avoid conflicts between vendor extension and future SYCL spec
- Make extension's vendor apparent in source code

- All naming patterns share:
  - A namespace that contains extension's types and free functions
  - A prefix to use when adding new member functions to existing SYCL classes (prefix also used when adding new values to existing SYCL enumerations)
  - A prefix to use for extension's macro names

Option 1: All Capitals

- Vendor chooses a unique “vendorstring”
- SYCL org promises never to start an identifier with a capital letter

```
#define SYCL_EXT_ACME_FOO

namespace sycl::ACME {
  void fancy();
}

namespace sycl {
  class device {
    void ACME_fancier();
  };
}
```

Macros start with “SYCL_EXT_ACME”

Free functions and types in namespace “sycl::ACME”

Extended member functions start with “ACME_” prefix

Option 2: Initial Capital

- Vendor chooses a unique “vendorstring”
- SYCL org promises never to start an identifier with a capital letter

```
#define SYCL_EXT_ACME_FOO

namespace sycl::Acme {
  void fancy();
}

namespace sycl {
  class device {
    void Acme_fancier();
  };
}
```

Macros start with “SYCL_EXT_ACME”

Free functions and types in namespace “sycl::Acme”

Extended member functions start with “Acme_” prefix

Option 3: Use “ext” as a namespace

- Vendor chooses a unique “vendorstring”
- SYCL org promises never to start an identifier with “ext_”

```
#define SYCL_EXT_ACME_FOO

namespace sycl::ext::acme {
  void fancy();
}

namespace sycl {
  class device {
    void ext_acme_fancier();
  };
}
```

Macros start with “SYCL_EXT_ACME”

Free functions and types in namespace “sycl::ext::acme”

Extended member functions start with “ext_acme_” prefix

Things to Consider

- Option 1: All caps
  - Less verbose
  - Style similar to other Khronos specifications (e.g. OpenCL uses all caps for extensions)
  - All caps looks like macro name (especially the namespace name)
- Option 2: Initial cap
  - Also less verbose
- Option 3: “ext” as a namespace
  - Most verbose
  - Maintains snake_case style (consistent with core SYCL spec)

Notices and Disclaimers

The content of this oneAPI Specification is licensed under the Creative Commons Attribution 4.0 International License . Unless stated otherwise, the sample code examples in this document are released to you under the MIT license.

This specification is a continuation of Intel's decades-long history of working with standards groups and industry/academia initiatives such as The Khronos Group*, to create and define specifications in an open and fair process to achieve interoperability and interchangeability. oneAPI is intended to be an open specification and we encourage you to help us make it better. Your feedback is optional, but to enable Intel to incorporate any feedback you may provide to this specification, and to further upstream your feedback to other standards bodies, including The Khronos Group SYCL* specification, please submit your feedback under the terms and conditions below. Any contribution of your feedback to the oneAPI Specification does not prohibit you from also contributing your feedback directly to The Khronos Group or other standard bodies under their respective submission policies.

By opening an issue, providing feedback, or otherwise contributing to the specification, *you agree that Intel will be free to use, disclose, reproduce, modify, license, or otherwise distribute your feedback in its sole discretion without any obligations or restrictions of any kind, including without limitation, intellectual property rights or licensing obligations.* For complete contribution policies and guidelines, see Contribution Guidelines on www.spec.oneapi.com.