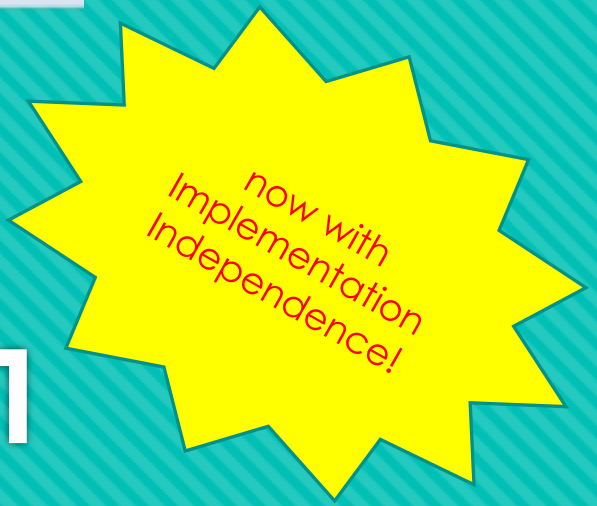

Chapter 1

Brought to you by the Implementation Agnostic Working Group

Goal of the Implementation Agnostic WG

Review the PMIx standard and eliminate text which assumes or requires a specific implementation of the standard

Why?

- The current document was originally written for both client developers and System Software developers implementing the "back end" of OpenPMIx
  - E.g. The **pmix_server_module_t** structure

Example:

Current Text:

The PMIx Standard defines and describes the interface developed by the PMIx Reference Implemenation (PRI). Much of this document is specific to the PRI's design and implementation. Specifically the standard describes the functionality provided by the PRI, and what the PRI requires of the clients and Resource Managers that use it's interface.

Core beliefs

- The PMIx client API's should define PMIx
- PMIx is an API for accessing System Software
- How a PMIx implementation interacts with System Software is not defined by the PMIx standard
- Multiple implementations of PMIx should be encouraged
- Implementation advice should not assume a PMIx implementation must follow a particular architecture or design

The changes

- https://github.com/pmix/pmix-standard/pull/192
- **Overall the text was shortened from about 7 pages of material to slightly over 4**

Overview of changes

- Introduce PMIx based on what it does rather than as an evolution of PMI
- Move PMI historical information to its own section
- Consolidate presentation of the optional nature of API and attribute support was spread throughout the chapter
- Remove text that read as a user's guide rather than as a standard document
- Avoid introducing specific API's except in examples and when their functionality is obvious

TOC

Old chapter sections:

1. **Introduction**
    - 1.1. Charter
    - 1.2. PMIx Standard Overview
        - 1.2.1. Who should use the standard?
        - 1.2.2. What is defined in the standard?
        - 1.2.3 What is not defined in the standard?
        - 1.2.4. General Guidance for PMIx Users and Implementors
    - 1.3. PMIx Architecture
        - 1.3.1. The PMIx Reference Implementation (PRI)
        - 1.3.2. The PMIx Reference RunTime Environment (PRRTE)
    - 1.4 Organization of this document

New chapter sections:

1. Introduction

- 1.0 Background
- 1.1. PMIx Architecture Overview
- 1.2. Portability of Functionality
    - 1.2.1. Optional Nature of Attributes
- 1.3. Organization of this document

Section by Section

- Goal of this presentation is to help you understand the motivation behind the changes
- You need to read it
- Difficult to present 7 pages of changes in an hour presentation.
- Will go through and explain how each section was changed

Old: Introduction (Summarized)

PMI is insufficient for exaflop, existing PMI implementations - would take up to 10 minutes to exchange data across 100,000 nodes

PMIx is different from PMI:

- PMIx adds key-value pair attributers to PMI. (captured in later section)
- Scales to exascale. (In reality it is the PMIx implementation which scales better than PMI)
- PMIx adds new APIs (not really necessary)
- PMIx works in conjunction with SMS to define portable access to system software. (Captured by new introdocution)
- PMIx is a standard (not really necessary to mention)
- PMIx provides a reference implementation (historically interesting, but not for first time readers)

New: Introduction

Process Management Interface - Exascale (PMIx) is an application programming interface standard to provide libraries and programming models with a portable and well-defined access to commonly needed services in distributed and parallel computing systems. A typical example of such a service is the portable and scalable exchange of network addresses to establish communication channels between the processes of a parallel application or service. As such, PMIx gives distributed system software providers a better understanding of how programming models and libraries can interface with and use system-level services. As a standard, PMIx provides Application Programming Interfaces (APIs) that allow for portable access to these varied system software services and the functionalities they offer. Although these services can be defined and implemented directly by the system software components providing them, the community represented by the Administrative Steering Committee (ASC) feels that the development of a shared standard better serves the community. As a result, PMIx enables programming languages and libraries to focus on their core competencies without having to provide their own system-level services.

Old: Charter (Summarized)

The charter of the PMIx community

- Define APIs to support interactions between applications and the SMS (already covered)
- Develop an open source "reference" implementation (not a goal of the standard)
- Retain transparent backward compatibility with the PMI-1 and PMI-2 (not a goal)
- Support "Instant On" for rapid startup (not really central to PMIx goals)
- Work with the HPC community to define and implement new APIs that support evolving programming model requirements for application interactions with the SMS. (covered)

Participation in the PMIx community is open to anyone, and not restricted to only code contributors to the reference implementation (separate operations of group from standard)

Background

The Process Management Interface (PMI) has been used for quite some time as a means of exchanging wireup information needed for inter-process communication. Two versions (PMI-1 and PMI-2) have been released as part of the MPICH effort, with PMI-2 demonstrating better scaling properties than its PMI-1 predecessor.

PMI-1 and PMI-2 can be implemented using PMIx though PMIx is not a strict superset of either. Since its introduction, PMIx has expanded on earlier PMI efforts by providing an extended version of the PMI APIs which provide necessary functionality for launching and managing parallel applications and tools at scale.

The increase in adoption has motivated the creation of this document to formally specify the intended behavior of the PMIx APIs.

More information about the PMIx standard and affiliated projects can be found at the PMIx web site: https://pmix.org

Old: PMIx Standard Overview

- PMIx Standard Overview
  - Standard defined by reference implementation. (this is exactly what we want to change)
  - Who should use the standard

    - client developers vs RM vs system software providers. (covered in new introduction)
  - What is defined in the standard

    - Again restates that standard = reference implementation. (exactly what we want to change)
  - What is not defined in the standard

    - Really gives detail about optional nature of attributes and returning “not supported”

    (moved to new section covering portability of functionality, much of the text was retained)

Old: General Guidance for PMIx Users and Implementors. (Summarized)

- The Standard defines the behavior of the Reference Implementation. (irrelevant)
- PMIx is an API to enable client access to PMIx-enabled RM (use SMS instead of RM)
- The standard defines this interaction (already covered by introduction)
- Not all the API's might be needed by a client (leaving this topic to slices WG)
- Clients should define what they require & desire so RM's know what to enable (slices WG)
- PMIx-enabled RMs should document what they support (unnecessary to tell implementations to document their product)

Old/New: Architecture

- Mostly kept intact
- Some language rephrased to be clear that the architecture presented is an example
- Interactions between PMIx implementation and backend are presented as implementation details and not port of the standard

Old: PRI, PRRTE

- Although the Reference Implementation and PRRTE are vital to the success of PMIx, both the OpenPMIx/PRRTE and PMIx Standard communities were in favor of dropping this presentation

New: Portability of Functionality

It is difficult to define a portable API that will provice access to the many and varied features underlying the operations for which PMIx provides access. For example, the options and features provided to request the creation of new processes varied dramatically between different systems existing at the time PMIx was introduced. Many workload managers (WLMs) provide rich interfaces to specify the resources assigned to processes. As a result, PMIx is faced with the challenge of attempting to meet the seamingly conflicting goals of creating an API which allows access to these diverse features while being portable across a wide range of existing software environments. In addition, the functionalities required by different clients vary greatly. Producing a PMIx implementation which can provide the needs of all possible clients on all of its target systems could be so burdensome as to discourage PMIx implementations.

To help address this issue, the PMIx APIs are designed to allow resource managers and other system management stack components to decide on support of a particular function and allow client applications to query and adjust to the level of support available. The PMIx community continues to look at ways to assist SMS implementers in their decisions on what functionality to support by highlighting functions and attributes that are critical to basic application execution (e.g., PMIx_Get) for certain classes of applications.

New: Optional nature of attributes

An area where differences between support on different systems can be challenging is regarding the attributes that provide information to the client process and/or control the behavior of a PMIx API. Most PMIx API calls can accept additional information or attributes specified in the form of key/value pairs. These attributes provide information to the PMIx implementation which influence the behavior of the API call. In addition to API calls being optional, support for the individual attributes of an API call can vary between systems or implementations.

An application can adapt to the attribute support on a particular system in one of two ways. PMIx provides an API to enable an application to query the attributes supported by a particular API (See 7.1.3.2). Through this API, the PMIx implementation can provide detailed information about the attributes supported on a system for each API call queried. Alternatively, the application can mark attributes as required using a flag within the pmix_info_t (See 3.2.16). If the required attribute is not available on the system or the desired value for the attribute is not available, the call will return the error code for not supported.

Final Thoughts

- Not claiming that our version of Chapter 1 is ideal.
- Much unedited text could be improved, but was outside the scope of our effort
- Our goal was only to detach the standard from a particular implementation
- Re-organization of the chapter was necessary to meet that goal
- We could define a back-end standard but at present we think it is best to only define client-side