PUBLISH/LOOKUP CHANGES

Client Separation / Implementation Agnostic WG

History

- 9 chapters “done”…. 10 to go!
- IAWG is working through the “next” chapter in our effort: Publish/Lookup
- As we work through a chapter, we look for ways to improve the presentation of the material
- For this chapter we had some significant functionality changes to propose

Major change #1 – Introduction of PMIX_DATA_TO_PUBLISH

- PMIx_Publish(const pmix_info_t info[], size_t ninfo)
  - *”things to publish” and “directives” are in a single input array*
  - *If an implementation does not recognize a directive, it may think it is something to publish.*
  - *If it really is a directive, but it means something to the implementation (some custom directive) it may not get published*
  - *Clients may make assumptions about how “things to publish” and “directives” are ordered. (This didn’t really get addressed by our “fix” though)*

PMIX_DATA_TO_PUBLISH

- Introduced PMIX_DATA_TO_PUBLISH:

PMIX_DATA_TO_PUBLISH "pmix.publishdata" (pmix_data_array_t)

Array of pmix_info_t containing data to be published.

PMIX_DATA_TO_PUBLISH EXPLAINED

- Introduced text to explain PMIX_DATA_TO_PUBLISH:

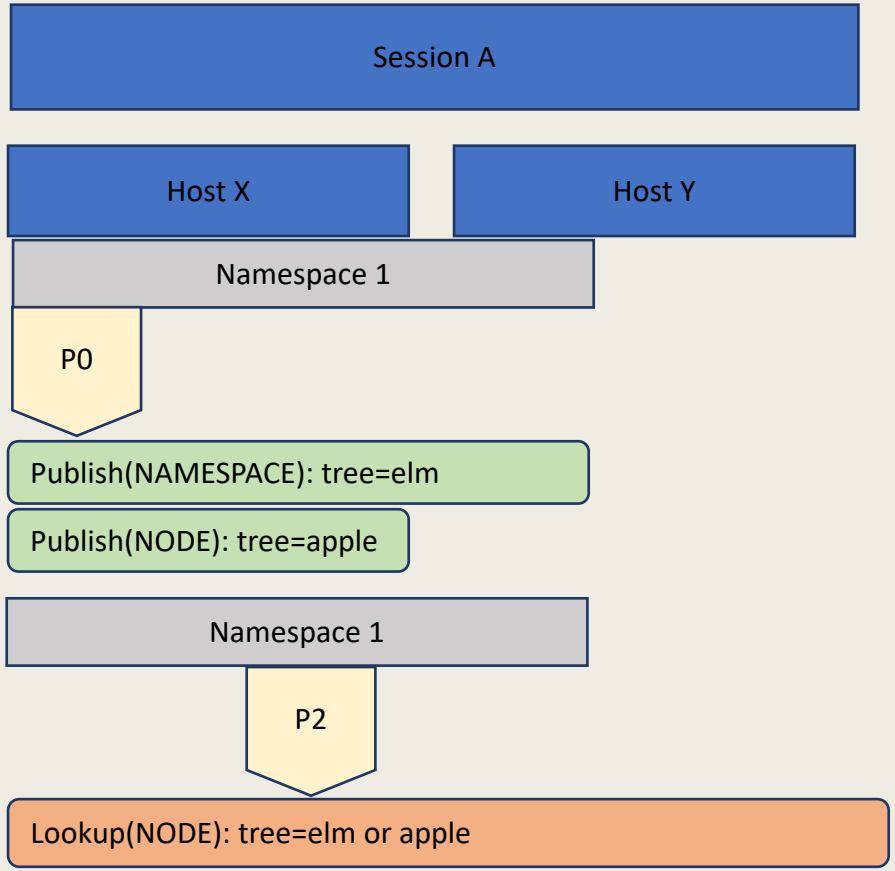
In some cases, implementations may be incapable of distinguishing which info keys in the info array are for publishing and which info keys are directives. To make it clear, it is recommended that the keys to be published are designated by passing them as a pmix_data_array_t using the PMIX_DATA_TO_PUBLISH directive. If the info array contains a PMIX_DATA_TO_PUBLISH info, all other elements of the info array will be treated as directives. If the info array does not include a PMIX_DATA_TO_PUBLISH info, the implementation should distinguish between info array elements that specify keys and directives as follows: All standardized directives to the publish call, including optional attributes the implementation does not support, should be treated as directives. Non-supported directives may be ignored as outlined in Section 1.3.1, but should not be treated as data to publish. The implementation may treat any custom (non-standardized) directives it supports as directives. All other info array elements should it be assumed to be data to be published. Since additional directives may be added to the standard and implementations may add support for additional custom directives, the use of PMIX_DATA_TO_PUBLISH is the only reliable way to ensure that future implementations will not mis-classify elements of an info array.

Major change #2: Require publish/lookup ranges to match

- PMIx_Publish takes a PMIX_RANGE that describes the range across which data is published
  - *E.g. Publish foo on range=session makes foo available to any process in the caller's same session.*
  - *Fairly straightforward*
- PMIx_Lookup takes a PMIX_RANGE
  - *According to the text: Specifies the range the publisher must be in relative to the caller. i.e. if range=session, publisher of the data must be in the same session as the caller*
- Standard says: "Publishing duplicate keys is permitted provided they are published to different ranges"

Review of why is it a problem

- Non-deterministic behavior:
  - *P2 can see 2 possible values*
- Non-intuitive behavior
  - *P2 asks for something on RANGE node, but can get something that was published to range NAMESPACE*



Quick review of alternative solutions:

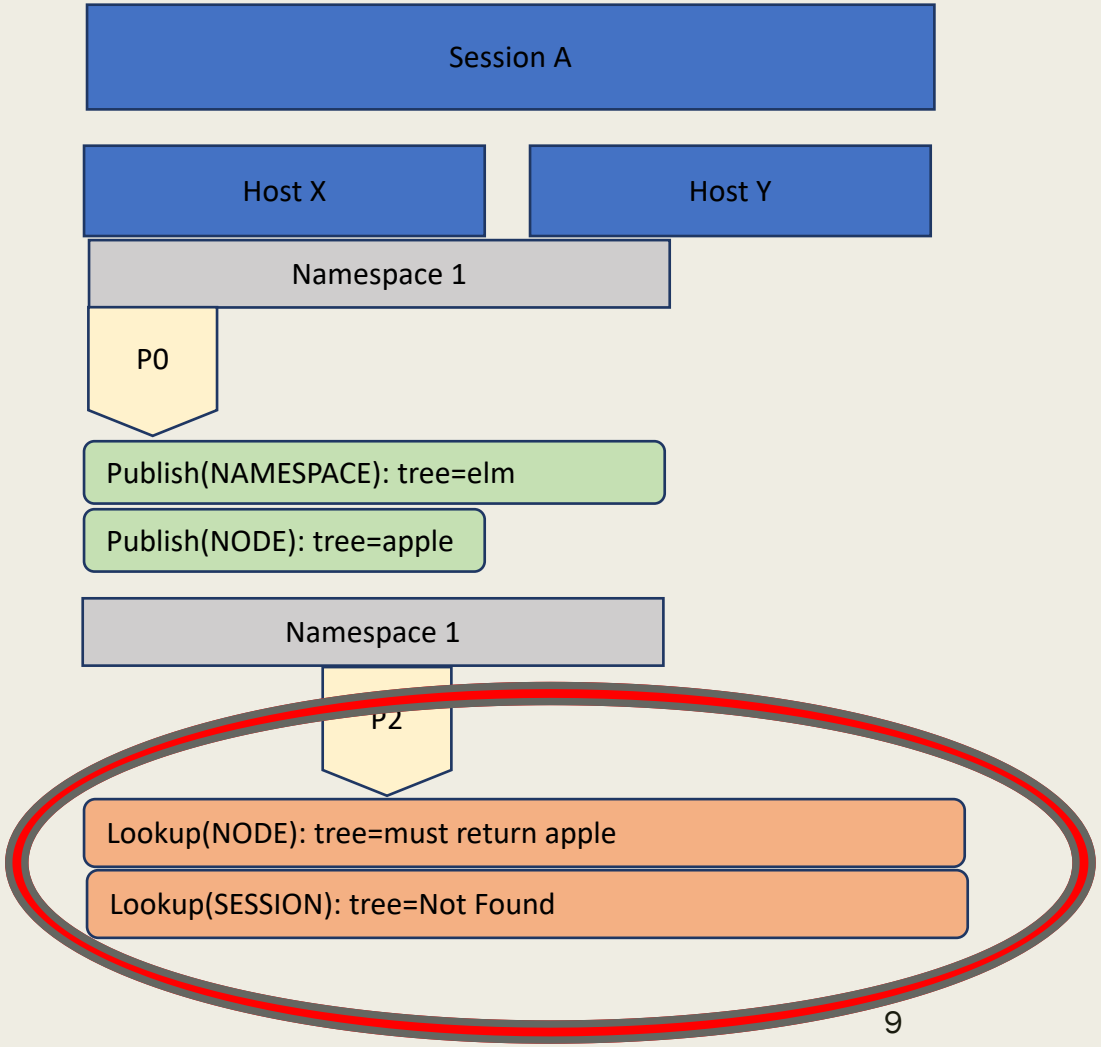
- Only allow publishing of a key to a single range
- Force implementations to prefer to choose the same publishing range as the lookup range
- Add more documentation around the issue (there already was a fair amount, but add more pictures and examples, etc)

NOTE: Since it was determined that no known implementation (PMIx+host environment) supports the current model anyway, we decided to do a more complete fix.

Proposed solution

- Lookup range must match publishing range
  - *Pros: Mostly addresses issue 1 & 2*
  - *Cons: Lookup code must know the range of the publisher. You can't just say, "I don't care… use RANGE=global and give me any publisher"**
  - *Big Cons: functionality change*

*Note: publish/lookup already requires a certain level of prior agreement between publishers and lookup'ers



Old Retrieval Rules

- If the requester specified the range, then the search shall be constrained to data where the publishing process falls within the specified range
- If the key of the stored information does not match the specified key, then the search will continue.
- If the requester's identifier does not fall within the range specified by the publisher, then the search will continue.
- If the publisher specified access permissions, the effective UID and GID of the requester shall be checked against those permissions, with the datastore rejecting the match if the requester fails to meet the requirements.

New Retrieval Rules

- The lookup key matches the published key.
- The type of range specified by the publisher is the same as the type of range specified by the requester.
- If a custom range is specified by the publisher and the requester, the members described in both cases must be identical. The publisher is not required to be a member of a custom range.
- The requestor must be a member of the range specified by the publisher.
- If the publisher specified access permissions, the effective UID and GID of the requester must meet those requirements.

Minor change #1 – make range definitions clearer for both events and publish

- Ranges (e.g. PMIX_RANGE_NAMESPACE) are used for both events and publish/lookup.
- Make the descriptions clear what it means for both cases

NOTE: I don't have these changes listed in this slide deck, you must look at the PR to see them

Minor change #2 – move lookup callback function to struct chapter

- It's not clear where callback functions are to be defined.
- Chapter 2 has a section "General callback functions"/. Probably the original thought was that specific ones would be presented with their API's and ones that are used by multiple API's would be presented in this Chapter 2 section.
- However, it also contains pmix_value_cbfunc_t.
  - *Maybe our WG moved this when we did that chapter?*
- Probably we need to revisit this section and change it from "General" to simple "Callback functions".
- Or should we move the callback functions back to the API's when they are only used by one API? Thoughts?

Minor change #3 – clarify custom range matching

- Not clear if custom ranges match based on description or set membership (e.g. PMIX_RANK_WILDCARD vs listing each process explicitly)

  Added: Custom ranges are considered different if they have different members.

  NOTE: Continuing to look at how other API's handle descriptions of sets/groups or processes.

Minor grammatical changes

- Unpublish description:
  - *OLD: Unpublish data posted by this process using the given keys.*
  - *NEW: Unpublish a list of keys published by the calling process.*
- Description of ranges and access requirements:
  - *OLD: Mechanisms for constraining availability of the information are also provided as a means for better targeting of the eventual recipient(s).*
  - *NEW: Mechanisms for constraining the scope of availability of the information are also provided as a means for better targeting of the eventual recipient(s).*

Minor grammatical changes (cont).

- Description of when publish/lookup may be used instead of put/get:
  - *OLD: However, another requirement exists for an asynchronous exchange of data where neither the posting nor the retrieving process is known in advance. For example, two separate namespaces may need to rendezvous with each other without knowing in advance the identity of the other namespace or when that namespace might become active.*
  - *NEW: However, another requirement exists for an asynchronous exchange of data where neither the posting nor the retrieving process is known in advance, for example, two namespaces that do not share a child-parent relationship.*

Misc changes

- Added definition for PMIX_ERR_NO_PERMISSIONS:

  All of the requested data was found and range restrictions were met for each specified key, but none of the matching data could be returned due to lack of access permissions.

- Definition of PMIX_PERSIST_INDEF:
  - *OLD: Retain data until specifically deleted.*
  - *NEW: Retain data until unpublished.*

- Added missing reference to PMIX_ACCESS_USERIDS and PMIX_ACCESS_GRPIDS