CHAPTER 12

Process Sets and Groups

PMIx supports two slightly related, but functionally different concepts known as *process sets* and *process groups*. This chapter describes the two definitions and how they are utilized, along with their corresponding APIs.

12.1 Process Sets

A PMIx *Process Set* is a user-provided label associated with a given set of application processes. Definition of a PMIx process set typically occurs at time of application execution - e.g., on a PRRTE command line:

C

```
$ prun -n 4 --pset ocean myoceanapp : -n 3 --pset ice myiceapp
```

C

In this example, the processes in the first application will be labeled with a **PMIX_PSET_NAME** attribute of *ocean* while those in the second application will be labeled with an *ice* value. During the execution, application processes could lookup the process set attribute for any other process using **PMIx_Get** . Alternatively, other executing applications could utilize the **PMIx_Query_info_nb** API to obtain the number of declared process sets in the system, a list of their names, and other information about them. In other words, the *process set* identifier provides a label by which an application can derive information about a process and its application - it does *not*, however, confer any operational function.

Thus, process *sets* differ from process *groups* in several key ways:

- Process *sets* have no implied relationship between their members - i.e., a process in a process set has no concept of a “pset rank” as it would in a process *group*
- Processes can only have one process *set* identifier, but can simultaneously belong to multiple process *groups*
- Process *set* identifiers are considered job-level information set at launch. No PMIx API is provided by which a user can change the process *set* value of a process on-the-fly. In contrast, PMIx process *groups* can only be defined dynamically by the application.

- Process *groups* can be used in calls to PMIx operations. Members of process *groups* that are involved in an operation are translated by their PMIx server into their *native* identifier prior to the operation being passed to the host environment. For example, an application can define a process group to consist of ranks 0 and 1 from the host-assigned namespace of *210456*, identified by the group id of *foo*. If the application subsequently calls the **PMIx_Fence** API with a process identifier of {foo, PMIX_RANK_WILDCARD}, the PMIx server will replace that identifier with an array consisting of {210456, 0} and {210456, 1} - the host-assigned identifers of the participating processes - prior to passing the request up to the host environment

The two concepts do, however, overlap in one specific area. Process *groups* are included in the process *set* information returned by calls to **PMIx_Query_info_nb** . Thus, a *process group* can effectively be considered an extended version of a *process set* that adds dynamic definition and operational context to the *process set* concept.

Advice to PMIx library implementers

PMIx implementations are required to include all active *group* identifiers in the returned list of process *set* names provided in response to the appropriate **PMIx_Query_info_nb** call.

12.2 Process Groups

PMIx *Groups* are defined as a collection of processes desiring a common, unique identifier for purposes such as passing events or participating in PMIx fence operations. As with processes that assemble via **PMIx_Connect** , each member of the group is provided with both the job-level information of any other namespace represented in the group, and the contact information for all group members. However, *groups* differ from **PMIx_Connect** assemblages in the following key areas:

- Relation to the host environment
  - Calls to **PMIx_Connect** are relayed to the host environment. This means that the host RM should treat the failure of any process in the specified assemblage as a reportable event and take appropriate action. However, the environment is not required to define a new identifier for the connected assemblage or any of its member processes, nor does it define a new rank for each process within that assemblage. In addition, the PMIx server does not provide any tracking support for the assemblage. Thus, the caller is responsible for addressing members of the connected assemblage using their RM-provided identifiers.

- Calls to PMIx Group APIs are first processed within the local PMIx server. When constructed, the server creates a tracker that associates the specified processes with the user-provided group identifier, and assigns a new *group rank* based on their relative position in the array of processes provided in the call to **PMIx_Group_construct** . Members of the group can subsequently utilize the group identifier in PMIx function calls to address the group's members, using either **PMIX_RANK_WILDCARD** to refer to all of them or the group-level rank of specific members. The PMIx server will translate the specified processes into their RM-assigned identifiers prior to passing the request up to its host. Thus, the host environment has no visibility into the group's existence or membership.

Advice to users

User-provided group identifiers must be distinct from anything provided by the RM so as to avoid collisions between group identifiers and RM-assigned namespaces. This can usually be accomplished through the use of an application-specific prefix – e.g., "myapp-foo"

- Construction procedure
  - **PMIx_Connect** calls require that every process call the API before completing – i.e., it is modeled upon the bulk synchronous traditional MPI connect/accept methodology. Thus, a given application thread can only be involved in one connect/accept operation at a time, and is blocked in that operation until all specified processes participate. In addition, there is no provision for replacing processes in the assemblage due to failure to participate, nor a mechanism by which a process might decline participation.
  - PMIx Groups are designed to be more flexible in their construction procedure by relaxing these constraints. While a standard blocking form of constructing groups is provided, the event notification system is utilized to provide a designated *group leader* with the ability to replace participants that fail to participate within a given timeout period. This provides a mechanism by which the application can, if desired, replace members on-the-fly or allow the group to proceed with partial membership. In such cases, the final group membership is returned to all participants upon completion of the operation.

    Additionally, PMIx supports dynamic definition of group membership based on an invite/join model. A process can asynchronously initiate construction of a group of any processes via the **PMIx_Group_invite** function call. Invitations are delivered via a PMIx event (using the **PMIX_GROUP_INVITED** event) to the invited processes which can then either accept or decline the invitation using the **PMIx_Group_join** API. The initiating process tracks responses by registering for the events generated by the call to **PMIx_Group_join** , timeouts, or process terminations, optionally replacing processes that decline the invitation, fail to respond in time, or terminate without responding. Upon completion of the operation, the final list of participants is communicated to each member of the new group.
- Destruct procedure

- Processes that assemble via **PMIx_Connect** must all depart the assemblage together – i.e., no member can depart the assemblage while leaving the remaining members in it. Even the non-blocking form of **PMIx_Disconnect** retains this requirement in that members remain a part of the assemblage until all members have called **PMIx_Disconnect_nb**
- Members of a PMIx Group may depart the group at any time via the **PMIx_Group_leave** API. Other members are notified of the departure via the **PMIX_GROUP_LEFT** event to distinguish such events from those reporting process termination. This leaves the remaining members free to continue group operations. The **PMIx_Group_destruct** operation offers a collective method akin to **PMIx_Disconnect** for deconstructing the entire group.

  Note that applications supporting dynamic group behaviors such as asynchronous departure take responsibility for ensuring global consistency in the group definition prior to executing group collective operations - i.e., it is the application's responsibility to either ensure that knowledge of the current group membership is globally consistent across the participants, or to register for appropriate events to deal with the lack of consistency during the operation.

In other words, members of PMIx Groups are *loosely coupled* as opposed to *tightly connected* when constructed via **PMIx_Connect** . The relevant APIs are explained below.

Advice to users

The reliance on PMIx events in the PMIx Group concept dictates that processes utilizing these APIs must register for the corresponding events. Failure to do so will likely lead to operational failures. Users are recommended to utilize the **PMIX_TIMEOUT** directive (or retain an internal timer) on calls to PMIx Group APIs (especially the blocking form of those functions) as processes that have not registered for required events will never respond.

12.2.1 PMIx_Group_construct

Summary

Construct a PMIx process group

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_construct(const char grp[],
                     const pmix_proc_t procs[], size_t nprocs,
                     const pmix_info_t directives[], size_t ndirs,
                     pmix_info_t **results, size_t *nresults)
```

C

IN **grp**
: **NULL**-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **procs**
: Array of **pmix_proc_t** structures containing the PMIx identifiers of the member processes (array of handles)

IN **nprocs**
: Number of elements in the *procs* array (**size_t**)

IN **directives**
: Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
: Number of elements in the *directives* array (**size_t**)

INOUT **results**
: Pointer to a location where the array of **pmix_info_t** describing the results of the operation is to be returned (pointer to handle)

INOUT **nresults**
: Pointer to a **size_t** location where the number of elements in *results* is to be returned (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request has been successfully completed
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library and/or the host RM does not support this operation
- a PMIx error constant indicating either an error in the input or that the request failed to be completed

Required Attributes

The following attributes are *required* to be supported by all PMIx libraries that support this operation:

PMIX_GROUP_LEADER **"pmix.grp.ldr"** (**bool**)
: This process is the leader of the group

PMIX_GROUP_OPTIONAL **"pmix.grp.opt"** (**bool**)

Participation is optional - do not return an error if any of the specified processes terminate without having joined. The default is false

PMIX_GROUP_LOCAL_ONLY **"pmix.grp.lcl"** (**bool**)
Group operation only involves local processes. PMIx implementations are *required* to automatically scan an array of group members for local vs remote processes - if only local processes are detected, the implementation need not execute a global collective for the operation unless a context ID has been requested from the host environment. This can result in significant time savings. This attribute can be used to optimize the operation by indicating whether or not only local processes are represented, thus allowing the implementation to bypass the scan. The default is false

Host environments that support this operation are *required* to provide the following attributes:

PMIX_GROUP_ASSIGN_CONTEXT_ID **"pmix.grp.actxid"** (**bool**)
Notify remaining members when another member terminates without first leaving the group. The default is false

PMIX_GROUP_NOTIFY_TERMINATION **"pmix.grp.notterm"** (**bool**)
Notify remaining members when another member terminates without first leaving the group. The default is false

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Construct a new group composed of the specified processes and identified with the provided group identifier. The group identifier is a user-defined, **NULL**-terminated character array of length less than or equal to **PMIX_MAX_NSLEN** . Only characters accepted by standard string comparison functions (e.g., *strncmp*) are supported. Processes may engage in multiple simultaneous group construct operations so long as each is provided with a unique group ID. The *directives* array can be used to pass user-level directives regarding timeout constraints and other options available from the PMIx server.

If the **PMIX_GROUP_NOTIFY_TERMINATION** attribute is provided and has a value of **true**, then either the construct leader (if **PMIX_GROUP_LEADER** is provided) or all participants who register for the **PMIX_GROUP_MEMBER_FAILED** event will receive events whenever a process fails or terminates prior to calling **PMIx_Group_construct** – i.e. if a *group leader* is declared, *only* that process will receive the event. In the absence of a declared leader, *all* specified group members will receive the event.

The event will contain the identifier of the process that failed to join plus any other information that the host RM provided. This provides an opportunity for the leader or the collective members to react to the event – e.g., to decide to proceed with a smaller group or to abort the operation. The decision is communicated to the PMIx library in the results array at the end of the event handler. This allows PMIx to properly adjust accounting for procedure completion. When construct is complete, the participating PMIx servers will be alerted to any change in participants and each group member will receive an updated group membership (marked with the **PMIX_GROUP_MEMBERSHIP** attribute) as part of the *results* array returned by this API.

Failure of the declared leader at any time will cause a **PMIX_GROUP_LEADER_FAILED** event to be delivered to all participants so they can optionally declare a new leader. A new leader is identified by providing the **PMIX_GROUP_LEADER** attribute in the results array in the return of the event handler. Only one process is allowed to return that attribute, thereby declaring itself as the new leader. Results of the leader selection will be communicated to all participants via a **PMIX_GROUP_LEADER_SELECTED** event identifying the new leader. If no leader was selected, then the **pmix_info_t** provided to that event handler will include that information so the participants can take appropriate action.

Any participant that returns **PMIX_GROUP_CONSTRUCT_ABORT** from either the **PMIX_GROUP_MEMBER_FAILED** or the **PMIX_GROUP_LEADER_FAILED** event handler will cause the construct process to abort, returning from the call with a **PMIX_GROUP_CONSTRUCT_ABORT** status.

If the **PMIX_GROUP_NOTIFY_TERMINATION** attribute is not provided or has a value of **false**, then the **PMIx_Group_construct** operation will simply return an error whenever a proposed group member fails or terminates prior to calling **PMIx_Group_construct** .

Providing the **PMIX_GROUP_OPTIONAL** attribute with a value of **true** directs the PMIx library to consider participation by any specified group member as non-required - thus, the operation will return **PMIX_SUCCESS** if all members participate, or **PMIX_ERR_PARTIAL_SUCCESS** if

some members fail to participate. The *results* array will contain the final group membership in the latter case. Note that this use-case can cause the operation to hang if the **PMIX_TIMEOUT** attribute is not specified and one or more group members fail to call **PMIx_Group_construct** while continuing to execute. Also, note that no leader or member failed events will be generated during the operation.

Processes in a group under construction are not allowed to leave the group until group construction is complete. Upon completion of the construct procedure, each group member will have access to the job-level information of all namespaces represented in the group plus any information posted via **PMIx_Put** (subject to the usual scoping directives) for every group member.

Advice to PMIx library implementers

At the conclusion of the construct operation, the PMIx library is *required* to ensure that job-related information from each participating namespace plus any information posted by group members via **PMIx_Put** (subject to scoping directives) is available to each member via calls to **PMIx_Get** .

Advice to PMIx server hosts

The collective nature of this API generally results in use of a fence-like operation by the backend host environment. Host environments that utilize the array of process participants as a *signature* for such operations may experience potential conflicts should both a **PMIx_Group_construct** and a **PMIx_Fence** operation involving the same participants be simultaneously executed. As PMIx allows for such use-cases, it is therefore the responsibility of the host environment to resolve any potential conflicts.

12.2.2 PMIx_Group_construct_nb

Summary

Non-blocking form of **PMIx_Group_construct**

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_construct_nb(const char grp[],
                        const pmix_proc_t procs[], size_t nprocs,
                        const pmix_info_t directives[], size_t ndirs,
                        pmix_info_cbfunc_t cbfunc, void *cbdata)
```

C

IN **grp**
NULL-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **procs**
Array of **pmix_proc_t** structures containing the PMIx identifiers of the member processes (array of handles)

IN **nprocs**
Number of elements in the *procs* array (**size_t**)

IN **directives**
Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
Number of elements in the *directives* array (**size_t**)

IN **cbfunc**
Callback function **pmix_info_cbfunc_t** (function reference)

IN **cbdata**
Data to be passed to the callback function (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** indicating that the request has been accepted for processing and the provided callback function will be executed upon completion of the operation. Note that the library *must not* invoke the callback function prior to returning from the API.
- **PMIX_OPERATION_SUCCEEDED** , indicating that the request was immediately processed and returned *success* - the *cbfunc* will *not* be called
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library does not support this operation - the *cbfunc* will *not* be called
- a non-zero PMIx error constant indicating a reason for the request to have been rejected - the *cbfunc* will *not* be called

If executed, the status returned in the provided callback function will be one of the following constants:

- **PMIX_SUCCESS** The operation succeeded and all specified members participated.

- **PMIX_ERR_PARTIAL_SUCCESS** The operation succeeded but not all specified members participated - the final group membership is included in the callback function
- **PMIX_ERR_NOT_SUPPORTED** While the PMIx server supports this operation, the host RM does not.
- a non-zero PMIx error constant indicating a reason for the request's failure

Required Attributes

PMIx libraries that choose not to support this operation *must* return **PMIX_ERR_NOT_SUPPORTED** when the function is called.

The following attributes are *required* to be supported by all PMIx libraries that support this operation:

PMIX_GROUP_LEADER **"pmix.grp.ldr"** (**bool**)
This process is the leader of the group

PMIX_GROUP_OPTIONAL **"pmix.grp.opt"** (**bool**)
Participation is optional - do not return an error if any of the specified processes terminate without having joined. The default is false

PMIX_GROUP_LOCAL_ONLY **"pmix.grp.lcl"** (**bool**)
Group operation only involves local processes. PMIx implementations are *required* to automatically scan an array of group members for local vs remote processes - if only local processes are detected, the implementation need not execute a global collective for the operation unless a context ID has been requested from the host environment. This can result in significant time savings. This attribute can be used to optimize the operation by indicating whether or not only local processes are represented, thus allowing the implementation to bypass the scan. The default is false

Host environments that support this operation are *required* to provide the following attributes:

PMIX_GROUP_ASSIGN_CONTEXT_ID **"pmix.grp.actxid"** (**bool**)
Notify remaining members when another member terminates without first leaving the group. The default is false

PMIX_GROUP_NOTIFY_TERMINATION **"pmix.grp.notterm"** (**bool**)
Notify remaining members when another member terminates without first leaving the group. The default is false

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Non-blocking version of the **PMIx_Group_construct** operation. The callback function will be called once all group members have called either **PMIx_Group_construct** or **PMIx_Group_construct_nb** .

12.2.3 PMIx_Group_destruct

Summary

Destruct a PMIx process group

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_destruct(const char grp[],
                    const pmix_info_t directives[], size_t ndirs)
```

C

IN **grp**
: **NULL**-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the identifier of the group to be destructed (string)

IN **directives**
: Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
: Number of elements in the *directives* array (**size_t**)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request has been successfully completed
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library and/or the host RM does not support this operation
- a PMIx error constant indicating either an error in the input or that the request failed to be completed

Required Attributes

For implementations and host environments that support the operation, there are no identified required attributes for this API.

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
: Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Destruct a group identified by the provided group identifier. Processes may engage in multiple simultaneous group destruct operations so long as each involves a unique group ID. The *directives* array can be used to pass user-level directives regarding timeout constraints and other options available from the PMIx server.

The destruct API will return an error if any group process fails or terminates prior to calling **PMIx_Group_destruct** or its non-blocking version unless the **PMIX_GROUP_NOTIFY_TERMINATION** attribute was provided (with a value of **false**) at time of group construction. If notification was requested, then the **PMIX_GROUP_MEMBER_FAILED** event will be delivered for each process that fails to call destruct and the destruct tracker updated to account for the lack of participation. The **PMIx_Group_destruct** operation will subsequently return **PMIX_SUCCESS** when the remaining processes have all called destruct – i.e., the event will serve in place of return of an error.

Advice to PMIx server hosts

The collective nature of this API generally results in use of a fence-like operation by the backend host environment. Host environments that utilize the array of process participants as a *signature* for such operations may experience potential conflicts should both a **PMIx_Group_destruct** and a **PMIx_Fence** operation involving the same participants be simultaneously executed. As PMIx allows for such use-cases, it is therefore the responsibility of the host environment to resolve any potential conflicts.

12.2.4 PMIx_Group_destruct_nb

Summary

Non-blocking form of **PMIx_Group_destruct**

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_destruct_nb(const char grp[],
                       const pmix_info_t directives[], size_t ndirs,
                       pmix_op_cbfunc_t cbfunc, void *cbdata)
```

C

IN **grp**
NULL-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the identifier of the group to be destructed (string)

IN **directives**
Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
Number of elements in the *directives* array (**size_t**)

IN **cbfunc**
Callback function **pmix_op_cbfunc_t** (function reference)

IN **cbdata**
Data to be passed to the callback function (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request is being processed - result will be returned in the provided *cbfunc*. Note that the library *must not* invoke the callback function prior to returning from the API.
- **PMIX_OPERATION_SUCCEEDED** , indicating that the request was immediately processed and returned *success* - the *cbfunc* will *not* be called
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library does not support this operation - the *cbfunc* will *not* be called
- a PMIx error constant indicating either an error in the input or that the request was immediately processed and failed - the *cbfunc* will *not* be called

If executed, the status returned in the provided callback function will be one of the following constants:

- **PMIX_SUCCESS** The operation was successfully completed
- **PMIX_ERR_NOT_SUPPORTED** While the PMIx server supports this operation, the host RM does not.
- a non-zero PMIx error constant indicating a reason for the request's failure

Required Attributes

PMIx libraries that choose not to support this operation *must* return **PMIX_ERR_NOT_SUPPORTED** when the function is called. For implementations and host environments that support the operation, there are no identified required attributes for this API.

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Non-blocking version of the **PMIx_Group_destruct** operation. The callback function will be called once all members of the group have executed either **PMIx_Group_destruct** or **PMIx_Group_destruct_nb** .

12.2.5 PMIx_Group_invite

Summary

Asynchronously construct a PMIx process group

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_invite(const char grp[],
                  const pmix_proc_t procs[], size_t nprocs,
                  const pmix_info_t directives[], size_t ndirs,
                  pmix_info_t **results, size_t *nresult)
```

C

IN **grp**
: **NULL**-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **procs**
: Array of **pmix_proc_t** structures containing the PMIx identifiers of the processes to be invited (array of handles)

IN **nprocs**
: Number of elements in the *procs* array (**size_t**)

IN **directives**
: Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
: Number of elements in the *directives* array (**size_t**)

INOUT **results**
: Pointer to a location where the array of **pmix_info_t** describing the results of the operation is to be returned (pointer to handle)

INOUT **nresults**
: Pointer to a **size_t** location where the number of elements in *results* is to be returned (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request has been successfully completed
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library and/or the host RM does not support this operation
- a PMIx error constant indicating either an error in the input or that the request failed to be completed

Required Attributes

The following attributes are *required* to be supported by all PMIx libraries that support this operation:

PMIX_GROUP_OPTIONAL **"pmix.grp.opt"** (**bool**)
: Participation is optional - do not return an error if any of the specified processes terminate without having joined. The default is false

Host environments that support this operation are *required* to provide the following attributes:

PMIX_GROUP_ASSIGN_CONTEXT_ID **"pmix.grp.actxid"** (**bool**)
: Notify remaining members when another member terminates without first leaving the group. The default is false

PMIX_GROUP_NOTIFY_TERMINATION **"pmix.grp.notterm"** (**bool**)
: Notify remaining members when another member terminates without first leaving the group. The default is false

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
: Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Explicitly invite the specified processes to join a group. The process making the **PMIx_Group_invite** call is automatically declared to be the *group leader*. Each invited process will be notified of the invitation via the **PMIX_GROUP_INVITED** event - the processes being invited must therefore register for the **PMIX_GROUP_INVITED** event in order to be notified of the invitation. Note that the PMIx event notification system caches events - thus, no ordering of invite versus event registration is required.

The invitation event will include the identity of the inviting process plus the name of the group. When ready to respond, each invited process provides a response using either the blocking or non-blocking form of **PMIx_Group_join** . This will notify the inviting process that the invitation was either accepted (via the **PMIX_GROUP_INVITE_ACCEPTED** event) or declined (via the **PMIX_GROUP_INVITE_DECLINED** event). The **PMIX_GROUP_INVITE_ACCEPTED** event is captured by the PMIx client library of the inviting process – i.e., the application itself does not need to register for this event. The library will track the number of accepting processes and alert the inviting process (by returning from the blocking form of **PMIx_Group_invite** or calling the callback function of the non-blocking form) when group construction completes.

The inviting process should, however, register for the **PMIX_GROUP_INVITE_DECLINED** if the application allows invited processes to decline the invitation. This provides an opportunity for the application to either invite a replacement, declare "abort", or choose to remove the declining process from the final group. The inviting process should also register to receive **PMIX_GROUP_INVITE_FAILED** events whenever a process fails or terminates prior to responding to the invitation. Actions taken by the inviting process in response to these events must be communicated at the end of the event handler by returning the corresponding result so that the PMIx library can adjust accordingly.

Upon completion of the operation, all members of the new group will receive access to the job-level information of each other's namespaces plus any information posted via **PMIx_Put** by the other members.

The inviting process is automatically considered the leader of the asynchronous group construction procedure and will receive all failure or termination events for invited members prior to completion. The inviting process is required to provide a **PMIX_GROUP_CONSTRUCT_COMPLETE** event once the group has been fully assembled – this event is used by the PMIx library as a trigger to release participants from their call to **PMIx_Group_join** and provides information (e.g., the final group membership) to be returned in the *results* array.

Advice to users

Applications are not allowed to use the group in any operations until group construction is complete. This is required in order to ensure consistent knowledge of group membership across all participants.

Failure of the inviting process at any time will cause a **PMIX_GROUP_LEADER_FAILED** event to be delivered to all participants so they can optionally declare a new leader. A new leader is identified by providing the **PMIX_GROUP_LEADER** attribute in the results array in the return of the event handler. Only one process is allowed to return that attribute, declaring itself as the new leader. Results of the leader selection will be communicated to all participants via a **PMIX_GROUP_LEADER_SELECTED** event identifying the new leader. If no leader was selected, then the status code provided in the event handler will provide an error value so the participants can take appropriate action.

12.2.6 PMIx_Group_invite_nb

Summary

Non-blocking form of **PMIx_Group_invite**

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_invite_nb(const char grp[],
                     const pmix_proc_t procs[], size_t nprocs,
                     const pmix_info_t directives[], size_t ndirs,
                     pmix_info_cbfunc_t cbfunc, void *cbdata)
```

C

IN **grp**
NULL-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **procs**
Array of **pmix_proc_t** structures containing the PMIx identifiers of the processes to be invited (array of handles)

IN **nprocs**
Number of elements in the *procs* array (**size_t**)

IN **directives**
Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
Number of elements in the *directives* array (**size_t**)

IN **cbfunc**
Callback function **pmix_info_cbfunc_t** (function reference)

IN **cbdata**
Data to be passed to the callback function (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request is being processed - result will be returned in the provided *cbfunc*. Note that the library *must not* invoke the callback function prior to returning from the API.
- **PMIX_OPERATION_SUCCEEDED** , indicating that the request was immediately processed and returned *success* - the *cbfunc* will *not* be called
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library does not support this operation - the *cbfunc* will *not* be called
- a PMIx error constant indicating either an error in the input or that the request was immediately processed and failed - the *cbfunc* will *not* be called

If executed, the status returned in the provided callback function will be one of the following constants:

- **PMIX_SUCCESS** The operation succeeded and all specified members participated.
- **PMIX_ERR_PARTIAL_SUCCESS** The operation succeeded but not all specified members participated - the final group membership is included in the callback function
- **PMIX_ERR_NOT_SUPPORTED** While the PMIx server supports this operation, the host RM does not.
- a non-zero PMIx error constant indicating a reason for the request's failure

Required Attributes

The following attributes are *required* to be supported by all PMIx libraries that support this operation:

PMIX_GROUP_OPTIONAL **"pmix.grp.opt"** (**bool**)
: Participation is optional - do not return an error if any of the specified processes terminate without having joined. The default is false

Host environments that support this operation are *required* to provide the following attributes:

PMIX_GROUP_ASSIGN_CONTEXT_ID **"pmix.grp.actxid"** (**bool**)
: Notify remaining members when another member terminates without first leaving the group. The default is false

PMIX_GROUP_NOTIFY_TERMINATION **"pmix.grp.notterm"** (**bool**)
: Notify remaining members when another member terminates without first leaving the group. The default is false

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Non-blocking version of the **PMIx_Group_invite** operation. The callback function will be called once all invited members of the group (or their substitutes) have executed either **PMIx_Group_join** or **PMIx_Group_join_nb** .

12.2.7 PMIx_Group_join

Summary

Accept an invitation to join a PMIx process group

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_join(const char grp[],
                const pmix_proc_t *leader,
                pmix_group_opt_t opt,
                const pmix_info_t directives[], size_t ndirs,
                pmix_info_t **results, size_t *nresult)
```

C

IN **grp**
: **NULL**-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **leader**
: Process that generated the invitation (handle)

IN **opt**
: Accept or decline flag (**pmix_group_opt_t**)

IN **directives**
: Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
: Number of elements in the *directives* array (**size_t**)

INOUT **results**
: Pointer to a location where the array of **pmix_info_t** describing the results of the operation is to be returned (pointer to handle)

INOUT **nresults**
: Pointer to a **size_t** location where the number of elements in *results* is to be returned (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request has been successfully completed
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library and/or the host RM does not support this operation
- a PMIx error constant indicating either an error in the input or that the request failed to be completed

Required Attributes

There are no identified required attributes for implementers.

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
: Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Respond to an invitation to join a group that is being asynchronously constructed. The process must have registered for the **PMIX_GROUP_INVITED** event in order to be notified of the invitation. When called, the event information will include the **pmix_proc_t** identifier of the process that generated the invitation along with the identifier of the group being constructed. When ready to respond, the process provides a response using either form of **PMIx_Group_join** .

Advice to users

Since the process is alerted to the invitation in a PMIx event handler, the process *must not* use the blocking form of this call unless it first "thread shifts" out of the handler and into its own thread context. Likewise, while it is safe to call the non-blocking form of the API from the event handler, the process *must not* block in the handler while waiting for the callback function to be called.

Calling this function causes the inviting process (aka the *group leader*) to be notified that the process has either accepted or declined the request. The blocking form of the API will return once the group has been completely constructed or the group's construction has failed (as described below) – likewise, the callback function of the non-blocking form will be executed upon the same conditions.

Failure of the leader during the call to **PMIx_Group_join** will cause a **PMIX_GROUP_LEADER_FAILED** event to be delivered to all invited participants so they can optionally declare a new leader. A new leader is identified by providing the **PMIX_GROUP_LEADER** attribute in the results array in the return of the event handler. Only one process is allowed to return that attribute, declaring itself as the new leader. Results of the leader selection will be communicated to all participants via a **PMIX_GROUP_LEADER_SELECTED** event identifying the new leader. If no leader was selected, then the status code provided in the event handler will provide an error value so the participants can take appropriate action.

Any participant that returns **PMIX_GROUP_CONSTRUCT_ABORT** from the leader failed event handler will cause all participants to receive an event notifying them of that status. Similarly, the leader may elect to abort the procedure by either returning **PMIX_GROUP_CONSTRUCT_ABORT** from the handler assigned to the **PMIX_GROUP_INVITE_ACCEPTED** or **PMIX_GROUP_INVITE_DECLINED** codes, or by generating an event for the abort code. Abort events will be sent to all invited participants.

12.2.8 PMIx_Group_join_nb

Summary

Non-blocking form of **PMIx_Group_join**

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_join_nb(const char grp[],
                   const pmix_proc_t *leader,
                   pmix_group_opt_t opt,
                   const pmix_info_t directives[], size_t ndirs,
                   pmix_info_cbfunc_t cbfunc, void *cbdata)
```

C

IN **grp**
NULL-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **leader**
Process that generated the invitation (handle)

IN **opt**
Accept or decline flag (**pmix_group_opt_t**)

IN **directives**
Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
Number of elements in the *directives* array (**size_t**)

IN **cbfunc**
Callback function **pmix_info_cbfunc_t** (function reference)

IN **cbdata**
Data to be passed to the callback function (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request is being processed - result will be returned in the provided *cbfunc*. Note that the library *must not* invoke the callback function prior to returning from the API.
- **PMIX_OPERATION_SUCCEEDED** , indicating that the request was immediately processed and returned *success* - the *cbfunc* will *not* be called
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library does not support this operation - the *cbfunc* will *not* be called
- a PMIx error constant indicating either an error in the input or that the request was immediately processed and failed - the *cbfunc* will *not* be called

If executed, the status returned in the provided callback function will be one of the following constants:

- **PMIX_SUCCESS** The operation succeeded and group membership is in the callback function parameters
- **PMIX_ERR_NOT_SUPPORTED** While the PMIx server supports this operation, the host RM does not.
- a non-zero PMIx error constant indicating a reason for the request's failure

Required Attributes

There are no identified required attributes for implementers.

Optional Attributes

The following attributes are optional for host environments that support this operation:

PMIX_TIMEOUT **"pmix.timeout"** (**int**)
: Time in seconds before the specified operation should time out (*0* indicating infinite) in error. The timeout parameter can help avoid "hangs" due to programming errors that prevent the target process from ever exposing its data.

Advice to PMIx library implementers

We recommend that implementation of the **PMIX_TIMEOUT** attribute be left to the host environment due to race condition considerations between completion of the operation versus internal timeout in the PMIx server library. Implementers that choose to support **PMIX_TIMEOUT** directly in the PMIx server library must take care to resolve the race condition and should avoid passing **PMIX_TIMEOUT** to the host environment so that multiple competing timeouts are not created.

Description

Non-blocking version of the **PMIx_Group_join** operation. The callback function will be called once all invited members of the group (or their substitutes) have executed either **PMIx_Group_join** or **PMIx_Group_join_nb** .

12.2.9 PMIx_Group_leave

Summary

Leave a PMIx process group

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_leave(const char grp[],
                 const pmix_info_t directives[], size_t ndirs)
```

C

IN **grp**
: **NULL**-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **directives**
: Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
: Number of elements in the *directives* array (**size_t**)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request has been communicated to the local PMIx server
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library and/or the host RM does not support this operation
- a PMIx error constant indicating either an error in the input or that the request is unsupported

Required Attributes

There are no identified required attributes for implementers.

Description

Leave a PMIx Group. Calls to **PMIx_Group_leave** (or its non-blocking form) will cause a **PMIX_GROUP_LEFT** event to be generated notifying all members of the group of the caller's departure. The function will return (or the non-blocking function will execute the specified callback function) once the event has been locally generated and is not indicative of remote receipt. All PMIx-based collectives such as **PMIx_Fence** in action across the group will automatically be adjusted if the collective was called with the **PMIX_GROUP_FT_COLLECTIVE** attribute (default is false) – otherwise, the standard error return behavior for that collective will be executed.

Advice to users

The PMIx_Group_leave API is intended solely for asynchronous departures of individual processes from a group as it is not a scalable operation – i.e., when a process determines it should no longer be a part of a defined group, but the remainder of the group retains a valid reason to continue in existence. Developers are advised to use PMIx_Group_destruct (or its non-blocking form) for all other scenarios as it represents a more scalable operation.

12.2.10 PMIx_Group_leave_nb

Summary

Non-blocking form of **PMIx_Group_leave**

Format

PMIx v4.0

C

```
pmix_status_t
PMIx_Group_leave_nb(const char grp[],
                    const pmix_info_t directives[], size_t ndirs,
                    pmix_op_cbfunc_t cbfunc, void *cbdata)
```

C

IN **grp**
NULL-terminated character array of maximum size **PMIX_MAX_NSLEN** containing the group identifier (string)

IN **directives**
Array of **pmix_info_t** structures (array of handles)

IN **ndirs**
Number of elements in the *directives* array (**size_t**)

IN **cbfunc**
Callback function **pmix_op_cbfunc_t** (function reference)

IN **cbdata**
Data to be passed to the callback function (memory reference)

Returns one of the following:

- **PMIX_SUCCESS** , indicating that the request is being processed - result will be returned in the provided *cbfunc*. Note that the library *must not* invoke the callback function prior to returning from the API.
- **PMIX_OPERATION_SUCCEEDED** , indicating that the request was immediately processed and returned *success* - the *cbfunc* will *not* be called
- **PMIX_ERR_NOT_SUPPORTED** The PMIx library does not support this operation - the *cbfunc* will *not* be called
- a PMIx error constant indicating either an error in the input or that the request was immediately processed and failed - the *cbfunc* will *not* be called

If executed, the status returned in the provided callback function will be one of the following constants:

- **PMIX_SUCCESS** The operation succeeded - i.e., the **PMIX_GROUP_LEFT** event was generated

- **PMIX_ERR_NOT_SUPPORTED** While the PMIx library supports this operation, the host RM does not.
- a non-zero PMIx error constant indicating a reason for the request's failure

Required Attributes

There are no identified required attributes for implementers.

Description

Non-blocking version of the **PMIx_Group_leave** operation. The callback function will be called once the event has been locally generated and is not indicative of remote receipt.