# Overview of MPS PubSub Operations

MPS PubSub Operations are typically subscribing to updates associated with MPS MO Objects, whose identity is defined by the ObjectIdentity structure

* domain (ordered list of MAL::Identifiers)
* area (MAL::Identifier) = MPS
* type (MAL::Identifier) specific to the MO Object Type
* key (MAL::Identifier) unique to the MO Object, within the scope of the above
* version (MAL::UInteger) of the MO Object, or =1 if the object is not versioned

In all cases of MPS PubSub Operations, the area and type are implicit in the service operation itself. However, in one case (MonitorPlanExecutionDetail), the operation effectively subscribes to updates to multiple object types (depending on what the provider supports), so the update messages themselves must identify the type of update they contain.

There are 5 defined MPS services. Some of these are anticipated to be supported by Planning functions:

* Planning Request Service PS
* ­Plan Distribution Service PS
* Plan Information Management Service

And others by Plan Execution (automated schedulers, etc.):

* Plan Execution Control Service PS
* Plan Edit Service
* Plan Information Management Service

Only those marked PS have PubSub operations.

In each case the Planning or Plan Execution function acts as the service provider. A given deployment of these functions may support a single domain or multiple domains. For the case of a provider that only supports a single domain, there is no requirement to specify the domain as it is defined by context. Similarly for a multi-domain provider, domains can be specified relative to the root domain supported by the provider.

# Detail of MPS PubSub Operations

MonitorRequestStatus [Planning Request Service]

This operation is used to subscribe to updates to the status of Planning Requests, to observe their evolving status. This is provided as RequestStatusUpdates relating to a RequestInstance. The following subscription keys are identified:

1. Domain: domain as MAL::Identifier
2. InstanceID: instance key as MAL::Identifier
3. DefinitionID: definition key as MAL::Identifier
4. UserID: user key as MAL::Identifier
5. UserReference: userReference of subscribed RequestInstances as MAL::Identifier
6. Status: status as MAL::UInteger [RequestStatusEnum]
7. OutputPlanID: outputPlanRef key as MAL::Identifier

The Domain and InstanceID keys relate to the ObjectIdentity of the subscribed Planning Requests. The area and type of the ObjectIdentity are defined by the operation (MPS, RequestInstance).

The DefinitionID is the key of the associated RequestDefinition – multiple RequestInstances may be created from the same RequestDefinition.

The UserID is the key of the MPS User who generated the Planning Request – Users have an MO object identity.

UserReference is associated with a planning request by the User when submitting it – it is distinct from the RequestInstance key which is assigned by the Planning system.

Status is an enum that indicates the current status of the Planning Request.

OutputPlanID is the key associated with a Plan (MO Object) that satisfies the Planning Request.

Example use cases:

1. Consumer subscribes to all Planning Requests pertaining to a given Domain – only the Domain is specified. It would be advantageous to be able to wildcard any node in the domain hierarchy.
2. Consumer subscribes to a specific Planning Request (or set of Planning Requests) – this may be specified by the InstanceID (if known) or the UserReference (if known). The Domain may need to be specified if the Provider manages multiple domains. It would be advantageous to be able to subscribe to multiple Planning Requests with a single subscription.
3. Consumer subscribes to all Planning Requests generated by a specified User. This allows a User to obtain updates to all their Planning Requests (including those not yet submitted). May be refined by specifying the Domain.
4. Consumer subscribes to all Planning Requests generated from the same Definition. The Definition is effectively template for a particular class of mission operation. May be refined by also specifying the Domain and/or User. It would be advantageous to be able to subscribe to multiple DefinitionIDs with a single subscription.
5. Consumer subscribes to Planning Requests that have a given Status. May be refined by specifying Domain, User and/or DefinitionID.
6. Consumer subscribes to Planning Requests that are associated with a given OutputPlanID. May be refined by specifying Domain, User and/or DefinitionID.

MonitorPlanStatus [Plan Distribution Service]

This operation is used to subscribe to updates to the status of Plans, in the context of Planning, to observe their evolving status. This is provided as PlanUpdates relating to a Plan. The following subscription keys are identified:

1. Domain: domain as MAL::Identifier
2. PlanID: key of subscribed Plan as MAL::Identifier
3. Precursor: key of precursorPlan as MAL::Identifier
4. Status: status as MAL::Uinteger [PlanStatusEnum]
5. Originator: originator of subscribed Plan as MAL::Identifier

The Domain and PlanID keys relate to the ObjectIdentity of the subscribed Plans. The area and type of the ObjectIdentity are defined by the operation (MPS, Plan).

Plans may be standalone or relative to the Plan preceding it. The Precursor is the key of the precursor Plan (also of type MPS, Plan).

Status is an enum that indicates the current status of the Plan.

Originator is a MAL::Identifier that identifies the system or individual that created the Plan.

Example use cases:

1. Consumer subscribes to all Plans pertaining to a given Domain – only the Domain is specified. It would be advantageous to be able to wildcard any node in the domain hierarchy.
2. Consumer subscribes to a specific Plans (or set of Plans) by specifying the PlanID. The Domain may need to be specified if the Provider manages multiple domains. It would be advantageous to be able to subscribe to multiple Plans with a single subscription.
3. Consumer subscribes to all Plans that share a common Precursor. This is useful to identify all possible follow-on Plans, which may include alternative or contingency options. The Domain may need to be specified if the Provider manages multiple domains.
4. Consumer subscribes to Plans that have a given Status. May be refined by specifying Domain, Precursor and/or Originator.
5. Consumer subscribes to Plans that have a given Originator. May be refined by specifying Domain, Status and/or Precursor.

MonitorPlan [Plan Distribution Service]

This operation is used to subscribe to Plans (the whole Plan, not just its status), the update is therefore a Plan. The subscription keys are identical to MonitorPlanStatus. A consumer would use this operation to receive new Plans as they are released.

MonitorPlanExecution [Plan Execution Control Service]

This operation is used to subscribe to the status of Plans, in the context of their execution, to observe their evolving status. It is equivalent to the MonitorPlanStatus operation and has the same set of subscription keys.

MonitorPlanExecutionDetail [Plan Execution Control Service]

This operation is used to subscribe to the status of the Planning Activities, Planning Events and [optionally] Planning Resources contained within Plans, in the context of their execution. Multiple types of Update message, depending on what is supported by the deployment (ActivityUpdate, EventUpdate and ResourceUpdate). It is expected that, as a minimum, updates for ActivityInstances will be supported, but additionally updates for EventInstances and Resources may be supported. It is anticipated that the Provider will only provide updates for Plans currently loaded into the execution environment. The following subscription keys are identified:

1. Domain: domain as MAL::Identifier
2. PlanID: key of subscribed Plan as MAL::Identifier
3. SubPlan: subplan of subscribed ActivityInstances as MAL::Identifier
4. Tags: tags of subscribed ActivityInstances as MAL::String

The Domain and PlanID keys relate to the ObjectIdentity of the subscribed Plans. The area and type of the ObjectIdentity are defined by the operation (MPS, Plan).

The SubPlan key is a MAL::Identifier that may be associated with ActivityInstances contained within the Plan(s). SubPlans are a means to decompose a Plan and allow control over activities at a lower level than the Plan itself. Only one SubPlan can be associated with each ActivityInstance. The SubPlan IDs are fixed for a given deployment and can be used across multiple Plans. SubPlans are not identified as MO Objects: they are fully defined by the SubPlanID.

Tags may also be associated with ActivityInstances. Tags are arbitrary strings associated with an ActivityInstance by the user, multiple Tags can be associated with an ActivityInstance. They provide a means to specify an operationally meaningful selection criteria for filtering purposes. For example it could be used to identify the responsible operations team, operations campaign that the activity relates to, or any other topic not already covered by domain and SubPlan.

Note that only ActivityInstances have associated SubPlan and Tags. It is implementation dependent which if any EventUpdates and ResourceUpdates are returned if the subscription is filtered by SubPlan or Tag.

Example use cases:

1. Consumer subscribes to plan execution detail (ActivityInstances, EventInstances and Resources – as applicable to the deployment) in all Plans pertaining to a given Domain – only the Domain is specified. It would be advantageous to be able to wildcard any node in the domain hierarchy.
2. Consumer subscribes to a specific Plan (or set of Plans) by specifying the PlanID. The Domain may need to be specified if the Provider manages multiple domains. Only updates to the content of those Plans will be reported. It would be advantageous to be able to subscribe to multiple Plans with a single subscription.
3. Consumer subscribes to a specific SubPlan (or set of SubPlans) by specifying the SubPlan ID. If no PlanID is specified, then this will apply to ActivityInstances assigned to the subscribed SubPlan, irrespective of the Plan they are contained in. The Domain may need to be specified if the Provider manages multiple domains. If a PlanID (or PlanIDs) is specified, then updates will only relate to the content of those Plans. It would be advantageous to be able to subscribe to multiple SubPlans with a single subscription.
4. Consumer subscribes to ***one or more*** Tags. Only updates for ActivityInstances with one or more of those Tags are returned. The subscription may also be scoped by specifying Domain, PlanID or SubPlanID.

MonitorSubPlanExecution [Plan Execution Control Service]

The operation is used to subscribe to the execution status of SubPlans, which is provided as SubPlanUpdates relating to a SubPlan. The following subscription keys are identified:

1. Domain: domain as MAL::Identifier
2. SubPlan: subplan ID as MAL::Identifier

The Domain and SubPlan keys relate to the identity of the subscribed SubPlans. SubPlans are not defined as MO Objects, and are fully defined by the combination of domain and SubPlan ID.

Use cases:

1. Consumer subscribes to all SubPlan updates for a given domain.
2. Consumer subscribes to updates for a specific SubPlan (or set of SubPlans). It would be advantageous to be able to subscribe to multiple SubPlans with a single subscription.

# Conclusions

The following is the complete set of subscription keys currently required by MPS:

| Subscription Key | Type | Comments |
| --- | --- | --- |
| domain | List<MAL::Identifier>[ordered] | The domain may be redundant if the provider only supports a single domain.May be relative to the root domain supported by the provider.It should be possible to wildcard any node in the domain hierarchy, e.g. Galileo.\*.Power, where \* implies any spacecraft in the constellation. |
| [Request] InstanceIDPlanID | MAL::Identifier | Corresponds to the identity key of the subscribed objects to which the updates relate.It should be possible to specify multiple keys within the same request (to be ORed). |
| [Request] DefinitionIDUserIDOutputPlanIDPrecursor [Plan] | MAL::Identifier | Corresponds to the identity key of another MO object with which the subscribed object has a relationship.It should be possible to specify multiple values of a given subscription key within the same request (to be ORed). |
| UserReferenceSubPlanID | MAL::Identifier | Corresponds to an identity associated with the subscribed object that is not itself an MO object.It should be possible to specify multiple values of a given subscription key within the same request (to be ORed). |
| Status | Enum (Integer or String) | Corresponds to a defined status of the subscribed object.It should be possible to specify multiple values of a given subscription key within the same request (to be ORed) |
| Tags | MAL::String | Corresponds to a Tag associated with the subscribed object.It should be possible to specify multiple tags within the same request (to be ORed) |

In all cases, the requirement is that the filter required is the AND of all subscription keys, but where multiple values of a subscription key are supplied these should be ORed.

The domain is a special case, as it is an ordered list with the possibility of wildcards at any point in the list. It may also be empty. It is acceptable if a subscription request only allows a single domain specification – where more than one domain is required and this cannot be defined using a wildcard, then multiple subscriptions can be made.

For the remaining subscription keys, the required functionality could be achieved by using a list of subscription key entries that each comprise the key name, followed by a list of key values, rather than the Name-Value pair with a single value. This would seem a more concise structure where long lists of IDs need to be supplied (name once, value many times). It also avoids the need for nested structures.

The rule is then that the values for an individual key are ORed, but that the keys themselves are ANDed.

Structure of the subscription would then look like:

* Domain List<MAL::Identifier> [ordered] [\*=wildcard]
* Filters List<SubscriptionKeys>

SubscriptionKey

* Name MAL::Identifier
* Values List<MAL::Attributes>

Name and Type of SubscriptionKeys are defined in Service Specification, but this may be extended by a given deployment.

I note the use of “key” is overloaded between the object identity key and subscription keys – maybe subscription keys should be termed “filters” or criteria”.