The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

MISSION OPERATIONS—MISSION PLANNING AND SCHEDULING SERVICES

PROPOSED DRAFT RECOMMENDED STANDARD

CCSDS 529.1-R-0

PROPOSED RED BOOK

October 2023

AUTHORITY

Issue:	Proposed Red Book, Issue 0
Date:	October 2023
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4), and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the email address below.

This document is published and maintained by:

CCSDS Secretariat
National Aeronautics and Space Administration
Washington, DC, USA
Email: secretariat@mailman.ccsds.org

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - -- The **standard** itself.
  - -- The anticipated date of initial operational capability.
  - -- The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Organization and Processes for the Consultative Committee for Space Data Systems* (CCSDS A02.1-Y-4). Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be sent to the CCSDS Secretariat at the email address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- China National Space Administration (CNSA)/People's Republic of China.
- Deutsches Zentrum für Luft- und Raumfahrt (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (FSA)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- UK Space Agency/United Kingdom.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (BELSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- China Satellite Launch and Tracking Control General, Beijing Institute of Tracking and Telecommunications Technology (CLTC/BITTT)/China.
- Chinese Academy of Sciences (CAS)/China.
- China Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish National Space Center (DNSC)/Denmark.
- Departamento de Ciência e Tecnologia Aeroespacial (DCTA)/Brazil.
- Electronics and Telecommunications Research Institute (ETRI)/Korea.
- Egyptian Space Agency (EgSA)/Egypt.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Geo-Informatics and Space Technology Development Agency (GISTDA)/Thailand.
- Hellenic National Space Committee (HNSC)/Greece.
- Hellenic Space Agency (HSA)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- Mohammed Bin Rashid Space Centre (MBRSC)/United Arab Emirates.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic and Atmospheric Administration (NOAA)/USA.
- National Space Agency of the Republic of Kazakhstan (NSARK)/Kazakhstan.
- National Space Organization (NSPO)/Chinese Taipei.
- Naval Center for Space Technology (NCST)/USA.
- Netherlands Space Office (NSO)/The Netherlands.
- Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Scientific and Technological Research Council of Turkey (TUBITAK)/Turkey.
- South African National Space Agency (SANSA)/Republic of South Africa.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- Swiss Space Office (SSO)/Switzerland.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 529.1-R-0	Mission Operations—Mission Planning and Scheduling Services, Proposed Draft Recommended Standard, Issue 0	October 2023	Current proposed draft

CONTENTS

Figure

Table

1 INTRODUCTION

1.1 GENERAL

Mission planning is an activity that often requires interaction between multiple entities. This may be to support distributed planning, where the responsibility for different aspects of mission operations planning is spread over multiple entities, including the space segment. It may also be to facilitate collaboration between missions, or to allow the planning of payloads by multiple end-users or the planning of multiple payloads from different agencies hosted on the same spacecraft. Other missions, such as observatories, may make payloads available to a wider user community. Some planning responsibility may be delegated to the spacecraft itself and the corresponding capabilities hosted on board. Historically, these interoperable interfaces have typically been defined on a per-mission or per-agency basis.

This Recommended Standard has the objective of specifying generic, interoperable mission planning and scheduling interfaces, for all typical space mission use cases, including the ones identified above. These use cases are elaborated in the associated Informational Report (Green Book) (reference [D2]) *Mission Planning and Scheduling*. This Recommended Standard focuses on the Mission Planning and Scheduling (MPS) Services identified for supporting interoperability and defines an information model that defines the data structures required by the operations of these services.

Mission planning and scheduling are integral parts of Mission Operations (MO) and closely related to the other aspects of the overall monitoring and control of space missions. This close relationship is recognized in the context of the CCSDS Mission Operations and Information Management (MOIMS) Area by the fact that the MPS services have been identified and included from the start among the envisaged MO services described in reference [D1], *Mission Operations Services Concept*. This Recommended Standard defines the MO MPS services in conformance with the CCSDS Mission Operations service framework described therein.

The MPS services are a set of services that support: interaction with a mission planning system and its users at the level of **planning requests**, distribution of the **plans** generated, and control of the execution of those plans. It is expected, but not required, that these are used in conjunction with other mission operations services, such as Monitoring & Control (reference [D4]) and Automation, as identified in reference [D1].

The MPS services are defined in terms of the Message Abstraction Layer (MAL) (see reference [2], *Mission Operations Message Abstraction Layer*), that is the core of the MO service framework.

1.2 PURPOSE AND SCOPE

This Recommended Standard defines, in an abstract manner, the MPS services in terms of:

a) service specifications that define the operations necessary to provide the services, together with their parameter data and required behaviour;

b) an information model that describes the structure of MPS data, including **planning requests**, **plans**, and supporting information objects that are referenced by the MPS services;

c) file-based message formats for the exchange of **planning requests** and **plans**, for use in mission deployments that do not require service-based interfaces.

Not all aspects of this Recommended Standard need to be applied in the context of a specific MPS system in order to support a conformant interface. Each service is optional and may include optional capability sets within it. Some aspects of the MPS information model are optional. A summary of the optional elements of the standard is provided in 2.6.

This Recommended Standard does not specify:

a) individual implementations or products;

b) the implementation of entities or interfaces within real systems;

c) the methods or technologies required for communications;

d) how required MPS service configuration data is made available to deployed MPS functions;

e) the expression language used for representation of conditions and calculations embedded within MPS data.

1.3 APPLICABILITY

This specification is applicable to any mission operations component that provides mission planning functionality or executes mission plans (schedules) and exposes mission planning and scheduling interfaces. This includes interfaces between:

– Mission users and the Mission Planning system;

– Hierarchical or distributed components of a Mission Planning system;

– Mission Planning and Plan Execution (Scheduler) components;

– Plan Execution and Mission Control.

Further detail is given in the associated Informational Report (Green Book) (reference [D2]).

This Recommended Standard is intended to apply to interfaces wherever they may occur in a space system:

- between ground-based components across a terrestrial link;
- between ground-based and on-board components across a space link;
- and potentially between on-board components across an on-board interface.

1.4 RATIONALE

The primary goal is to increase the level of interoperability for mission planning among agencies and space system users at the level of exchanged **planning requests** and **plans**. The service specifications can also be used between systems within an agency and to promote the development of re-usable infrastructure for space systems and interoperability between missions.

Various use case scenarios applicable to the mission planning and scheduling standardization process have been identified by performing a survey of a number of representative space missions of various CCSDS member agencies. The missions subject to the survey have been categorized into mission types, in an attempt to identify commonalities in the mission planning processes, for example, in the areas of planning cycles, execution feedback, navigation services, planning requests, resources, and constraints, and output of the planning phase. The full results of this survey, including the identified mission types, are described in the MPS Green Book (reference [D2]).

The use of the underlying MO framework enables abstract services to be implemented using appropriate encoding and information transfer technologies (file and/or message based) for the deployment context. An extensible set of MAL technology bindings exist to support:

- encoding of the service messages;
- binding of the service operations to a specific messaging technology.

It should be noted that security considerations are partly handled at the MO framework layer or below. This is described in Annex B.

1.5 DOCUMENT STRUCTURE

This Recommended Standard is organized in the following sections:

- **Section 1—Introduction**: provides purpose and scope, applicability, and rationale of this Recommended Standard and lists the definitions, conventions, and references used throughout the document;
- **Section 2—Overview**: describes the mission planning & scheduling concept and how this relates to MO services, as well as giving a high-level overview of both the MPS information model and the set of MPS services specified.

- **Section 3—MPS Service Specifications**: provides the formal specification of the MPS services.
- **Section 4—MPS Data Types**: provides the formal specification of MPS data items, including MO objects, and other data structures that are contained in or referenced by MPS service messages.
- **Section 4—Error Codes**: provides the formal specification of MPS error codes.
- **Section 6—Service Specification XML**: specifies the internet location of the formal service specification eXtensible Markup Language (XML).
- **Section 7—XML File Formats**: specifies the internet location of the formal specification of XML file formats for the exchange of planning requests and plans.

Sections 3-7 contain the normative specification of the Recommended Standard, while sections 1-2 are purely informative.

1.6 DEFINITIONS

For the purposes of this document, the following definitions apply.

NOTES

1 Abbreviations are to be found in annex C.

2 The terms **plan** and **planning** are used throughout this specification, but that this also encompasses schedule and scheduling respectively.

3 The prefix “planning” is used to disambiguate terms used in this specification from other more general uses of a term. This applies to planning activities, constraints, events, requests and resources.

action: A single executable task of an MO M&C service provider. A telecommand is an example of an action.

activity definition: The **definition** element of a **planning activity**. It forms part of the **planning configuration data**.

activity details: The information required to create an **activity instance** from an **activity definition**. It may be contained within a **planning request** to request inclusion of a **planning activity**, or within an **activity definition** (to specify child activities).

activity instance: The **instance** element of a **planning activity**. Activity instances are contained within **plans**.

area: A group of related MO services with an associated identifier and number. This Recommended Standard forms part of the MPS area, with the area number 5.

argument: A run-time parameter provided to various control items on invocation. For example, arguments apply to **planning requests**, **planning activities** and **planning events**.

constraint: (See **planning constraint**.)

custom function: An ancillary MPS data item that allows access to built-in Boolean functions of a planning system, for example in the context of **planning constraints** (specifically a function constraint). The custom function must be pre-defined to be referenceable and the MPS custom function definition holds the declaration of an available function.

definition: The statically declared information associated with an **information object**. This may, for example, include a description, set of defined arguments or any other information that applies to all occurrences of the information object. There may be multiple definitions [**versions**] over the mission lifetime associated with the same **identity** [definitionID].

details: A data structure used to specify the information needed to create an **instance** from a **definition** for an **information object** that has multiple occurrences.

direction: An MPS data type that is used to represent a pointing direction or attitude of a spacecraft, payload instrument, or other object.

domain: A namespace that partitions separately addressable entities (e.g., **planning activities**, **planning events** or **planning resources**) in the mission. The mission is decomposed into a hierarchy of domains within which entity identifiers are unique.

effect: A type of **planning constraint** that is used in the context of modelling **planning resources**. It specifies the impact that executing a **planning activity** will have on a **planning resource**.

event definition: The **definition** element of a **planning event**. It forms part of the **planning configuration data**.

event instance: The **instance** element of a **planning event**. Event instances are contained within **plans**.

expression: A calculation to be performed at run time that supplies a value of a defined data type. Expressions are specified as text strings, together with the identification of the expression language used. No standard expression language is specified in this document.

full plan: A **plan** that contains the full details of a plan. Used to distinguish from a **patch plan**.

identity: A unique identity associated with an **MO object**, which comprises:

- The **domain** of the object;
- The **area** of the object;
- The type of the object;
- A **key** (identifier) for the object, unique within domain, area and type;
- The **version** of the object [optional].

information object: The set of information about a real-world entity that is exchanged across an interface. This may include static definitions, dynamic status and metadata.

NOTE – Mission planning information objects include: **planning requests**, **plans**, **planning activities**, **planning events** and **planning resources**.

instance: A dynamically created object representing each new occurrence of an **information object**. This includes a unique instanceID of the occurrence and any unchanging data associated with it as a set of static attributes. It also includes the current status of the object as a set of dynamic attributes. An instance has a reference to its **definition**.

key: Part of the **identity** of an **MO object**, the key is a unique identifier for the object within the scope of the **domain**, **area** and object type.

MO dynamic item: A pattern of **MO objects** for an **information object** that has separate **definition** and **instance** objects, the latter representing an individual occurrence of the object with an evolving status. **Updates** reference the **instance** object. **Instances** reference the **definition** object.

MO object: An entity defined within the information model of an MO compliant service specification that has a unique **identity** enabling it to be referenced by other MO objects and in the body of MO service messages. **Information objects** may comprise multiple MO objects, adhering to one of the following object patterns: **MO static item**, **MO state** and **MO dynamic item**.

MO state: A pattern of **MO objects** for an **information object** that has a single element comprising both statically declared information and dynamically evolving status. **Updates** reference the **definition** object directly.

MO static item: A pattern of **MO objects** for an **information object** that has a single element comprising statically declared information with no evolving status. Static items comprise only a **definition** object, with no corresponding **updates**.

patch plan: A **plan** that only contains the delta (changes) from a precursor plan. A patch plan must be merged with its **precursor plan** to generate the **target plan**.

plan: The output of the **planning** process. It contains a set of selected **planning activities** associated with time, position, or **planning event**. A plan may contain additional related information.

NOTE – In the context of the mission planning and scheduling standardization activity, there is no distinction between the terms 'plan' and 'schedule' and only the term plan is used.

plan execution: The process of executing **plans** on-board or on the ground.

planning: The process of creating one or more plans (output) from planning requests (input).

NOTE – In the context of the mission planning and scheduling standardization activity, there is no distinction between the terms 'planning' and 'scheduling' and only the term planning is used.

planning activity: A meaningful unit of what can be planned. The granularity of a planning activity depends on the use case, it may be hierarchical. In other words planning activities are the building block for planning.

planning configuration data: The set of configuration data required by MPS service providers and consumers. It includes **activity definitions**, **event definitions**, **resource definitions**, **request definitions**, and MPS system configuration parameters.

planning constraint: Something that limits or restricts the planning of **planning activities**. Different types of constraint exist, including: temporal constraints, sequential constraints between **planning activities** and/or **planning events**, **resource** constraints, and geometric constraints (**position** and **pointing**).

planning event: The meeting of a condition. Planning activities can be associated with a planning event and specify their start or end relative to the event time.

planning request: An input to the planning process, which requests one or more **planning activities**. Each planning request contains all the information that the requester can provide.

planning resource: An abstraction of a real-world resource, physical or virtual, that is represented as a quantity. The level of fidelity in the modelling of a resource only needs to be sufficient to support planning decisions. A planning resource may constrain the execution of planned activities, which may in turn have an **effect** on the value of the resource.

planning user: Any entity that is responsible for submitting **planning requests** to a **planning** function and potentially receiving feedback on the status of planning requests and generated **plans**. For example, this could be an external Principal Investigator (PI) or a mission operations system or role.

position: An MPS data type that is used to represent the physical location of a spacecraft, or other object.

potential event: A type of **planning event** that is not predictable, but may still have a defined response within a **plan**.

precursor plan: A **plan** from which the current plan represents an evolution through replanning or an iterative planning cycle. The current plan contains the specification of the changes from the precursor plan as a set of plan revisions.

predicted event: A type of **planning event** that is expected to occur at a particular time or position that can be predicted as an input to **planning** and contained within a **plan**. Orbital events are an example of predicted events.

procedure: In the context of MPS: an executable process that is invoked to fulfil the execution of a planned activity. Automated operations procedures, on-board control procedures, and procedures supported by an MO Automation service provider (see reference [D1]) are examples of a procedure.

repetition: A data structure used in the context of a **planning request** to request the repeated execution of **planning activities**. Various sub-types of repetition are defined to support the specification of repeat cycles by different criteria, such as time, position, or pointing.

request definition: The optional **definition** element of a **planning request** that contains the specification of a re-usable planning request template. It forms part of the **planning configuration data**.

request instance: The **instance** element of a **planning request**. This may change over time if the request is updated by the user, each comprising a separate version of the request.

resource definition: The **definition** element of a **planning resource**. This may omit dynamic attributes of the resource (its value) and forms part of the **planning configuration data**.

resource profile: Provision for the evolution of the value of a **planning resource** over time.

slider: A relative position with respect to an MPS object, such as a **planning activity**, where 0 represents the start and 1 the end of the activity. The slider is a real number that can represent a specific point between these two extremes.

target plan: A **full plan** that is the result of applying a **patch plan** to its **precursor plan**.

trigger: A construct that allows specification of the specific condition that marks the start or end of something. It is used in the context of both **planning activities** and **plans** to specify when an activity should start or end. Triggers may be defined in terms of time, **position**, **pointing**, or a **planning event**.

update: A data structure used to report the changing value of dynamic attributes and **arguments** of an **MO object** (including its status) at a specific point in time.

version: Part of the **identity** of an **MO object**, typically of a **definition** object, that represents a defined set of values for its static attributes. When a **definition** is updated, the version is incremented, but other elements of the object's identity, including its **key** remain unchanged.

1.7 NOMENCLATURE

1.7.1 NORMATIVE TEXT

The following conventions apply for the normative specifications in this Recommended Standard:

a) the words 'shall' and 'must' imply a binding and verifiable specification;

b) the word 'should' implies an optional, but desirable, specification;

c) the word 'may' implies an optional specification;

d) the words 'is', 'are', and 'will' imply statements of fact.

NOTE – These conventions do not imply constraints on diction in text that is clearly informative in nature.

Specific requirements are identified in the text by a prefix of the following structure:

Req_<section#>.<requirement_type>.<requirement#>

for example, **Req_1.7.1.H.99**

where the section number corresponds to the section of this document that contains the requirement and the requirement number is a serial number assigned to each requirement in this document. The requirement_type may be one of:

– H High-level Requirement;

– F Functional Requirement;

– E Encoding Requirement.

1.7.2 INFORMATIVE TEXT

In the normative sections of this document, informative text is set off from the normative specifications either in notes or under one of the following subsection headings:

– Overview;

– Background;

– Rationale;

– Discussion.

1.8 CONVENTIONS

1.8.1 DIAGRAMS

Unified Modelling Language (UML) notation (reference [3]) is used for diagrams illustrating the service specifications in section 0. Reference Architecture for Space Data Systems (RASDS) notation (reference [7]) is used for the MO protocol stack diagram in section 2.3.

This section does not seek to provide a full description of the standard UML diagram types used, but defines the specific conventions applied in this document.

Sequence Diagrams

In the MPS service specifications, UML sequence diagrams are used to illustrate the sequence of operations and their constituent messages (following MAL interaction patterns).

Timelines show the service consumer and provider. Each service operation is shown as a set of messages, following the appropriate MAL interaction pattern, contained within a fragment box with the name of the operation. The fragment is of type ‘opt’ and greyed out if the operation is an optional element of the service (non-mandatory capability set). Where there is a logical sequence of operations this can be indicated on the diagram.

A broker timeline may also be shown, but is only relevant for publish-subscribe pattern operations. In this case the broker function is itself optional (together with all messages between broker and provider), as a valid implementation is for the provider to act as its own broker.

Messages are labelled with the operation name, followed by the MAL interaction pattern message name in block capitals and the message body fields shown in parentheses by type.

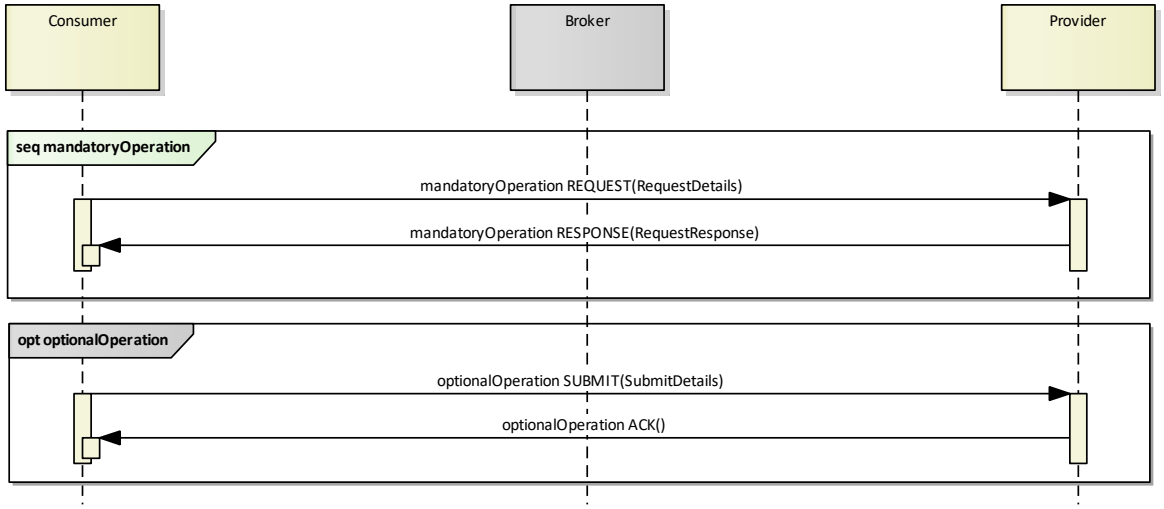

Figure 1-1: Example Sequence Diagram

1.8.2 TABLES

The formal normative definitions of data structures, services and service operations are presented in an abstract tabular format in this document. This is consistent with that specified in the MO MAL Recommended Standard (reference [2]), but has a more compact layout. The table formats used are summarized here to aid in understanding their presentation in this document. A full description can be found in reference [2].

Purple cells (dark grey when printed on a monochrome printer) contain table headings, light grey cells contain fields that are fixed for a pattern, and white cells contain values that are specific to the operation or structure.

Where types are required from other MO specifications, the following notation is used to denote the area in which the referenced definition resides:

<area>::<type name>

It should be noted that all tables described below constitute a normative part of the Recommended Standard, including the data structure tables that are contained within section 4.

Data Structures

Each data structure (or type) definition contained in section 4 contains a table following the standard structure outlined below.

Table 1-1: Example Data Structure Table

Name	**<Data Structure name>**		**Extends**	<Parent name>	**SFP**	<#>
Attribute	**Type**	**Nullable**	**Description**			
<name>	<data type>	Yes¦No	<Description>			
<name>	<data type>	Yes¦No	<Description>			
…	…	…	…			

The first row of the table specifies the name of the MPS data structure (in bold), and that of the structure it extends, which may either be a MAL data type (typically MAL::Composite) or another MPS data type. The Short Form Part (SFP) gives the number used by the MAL to identify this structure within the **area**.

This is followed by a list of attributes or data fields that constitute the data structure. Inherited attributes may optionally be shown with a grey background. In this document this is only used for data structures of type MO Object to highlight the inherited **identity** attribute.

Attribute data types may either be a MAL::Attribute type, or another MPS data structure. In the case of the MAL::ObjectRef attribute type, two forms are supported:

MAL::ObjectRef[1]	Reference is to an explicit object type (encoded within the reference) to an MO object of any type, subject to restrictions defined in the description field.
MAL::ObjectRef<T>	Reference is to an implicit object type (not encoded within the reference) and is restricted to the named concrete type T.

The nullable column indicates whether the attribute is allowed to contain a null value. A nullable field does not need to be provided by the consumer, but must be supported by the provider unless it is an optional element of the standard.

A default value may be specified in the description for a non-nullable attribute. This means that a value must be supplied in any service message ‘on the wire’, to avoid the need for a provider implementation to have knowledge of the default, but that in the context of a user (or Web-based) interface, the default value may be initially populated to avoid the need for the user to specify anything in the general case.

Attributes may also be a list of elements, represented as List<<element>>, where <element> can be of any MAL::Attribute or MPS data type. MAL lists are implicitly unbounded,

[1] From an MO MAL perspective this formally corresponds to MAL::ObjectRef<MAL::Object>, where it is constrained to be any MO object.

ordered and may have zero elements and/or be nullable. A nullable list can be empty [0..*], otherwise it must contain at least one entry [1..*].

By convention data structure names start with an upper case letter. If the data structure is abstract (only used to define an inheritance hierarchy) then its name is italicized and the word 'abstract' is substituted in the SFP. Attribute names start with a lower case letter. In the context of MPS MO objects (definitions and instances), static and dynamic attributes are differentiated by underlining the name of the static attributes.

Enumerations

Enumerations are also contained in section 4 and defined using tables of the following format.

Table 1-2: Example Enumeration Table

Name	**<Enumeration name>**	**SFP**	<#>
Status	**Value**	**Description**	
<STATE NAME>	<#>	<Description>	
…	…	…	

The set of allowed statuses/enumerations is listed together with their corresponding integer values and a description.

By convention the name of an enumeration ends with 'Enum', and the names of the statuses/enumerations are all in upper case.

Services

A summary table is provided for each service specification in section 0. This defines the area, service and version numbers used in MAL message headers, and lists the service operations together with their MAL interaction patterns, operation numbers, and the capability set to which they belong.

Table 1-3: Example Service Table

Area Identifier	**Service Identifier**	**Area Number**	**Service Number**	**Area Version**
MPS	<Service name>	5	<#>	<#>
Interaction Pattern	**Operation Identifier**		**Operation Number**	**Capability Set**
REQUEST	submitRequest		<#>	<#>
…	…		…	…

Service Operations

The definition of each MPS service operation in section 0 contains a table based on the appropriate interaction pattern template for the applicable MAL interaction pattern (see reference [2]).

Table 1-4: Example Operation Template

Operation Identifier	<Operation name>		
Interaction Pattern	PUBLISH-SUBSCRIBE		
Subscription Keys	planID : (MAL::Identifier) precursor : (MAL::Identifier) status : (MAL::UInteger) originator : (MAL::Identifier)		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
OUT	PUBLISH/NOTIFY	Yes¦No	List<Field name : (data type)>
…	…	…	…

The subscription keys row is only included in the case of a PubSub operation and lists the subscription key fields (name and type) specified for the operation.

The message direction denotes the direction of the message relative to the provider of the service and is either IN or OUT. So all messages directed towards the provider are IN messages, and all messages directed away from the provider are OUT messages.

The body signature contains a list of message fields, often a single field referencing an MPS data structure, but in some cases there may be a series of fields each identified by a field name and data type. The nullable column indicates whether each of the listed message fields is nullable or not.

Errors

Errors that may be returned by the operation are listed in a simple table, which references the error by name and the area in which it is defined. MPS errors are defined in section 4, together with the corresponding error number and the type and description of any ExtraInfo field. Standard MAL errors (not listed) may also be returned.

Table 1-5: Example Error References Table

Error	**Area**
<Error name>	<Area>
…	..

1.9 REFERENCES

The following publications contain provisions which, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were valid. All publications are subject to revision, and users of this Recommended Standard are encouraged to investigate the possibility of applying the most recent editions of the publications indicated below. The CCSDS secretariat maintains a register of currently valid CCSDS publications.

[1] *Mission Operations Reference Model*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 520.1-M-1. Washington, D.C.: CCSDS, July 2010.

[2] *Mission Operations Message Abstraction Layer*. Issue 2. Recommendation for Space Data System Standards (Blue Book), CCSDS 521.0-B-2. Washington, D.C.: CCSDS, March 2013.

[3] *Unified Modelling Language.* Issue 2.5.1. Object Management Group (OMG), formal/2017-12-05, December 2017.

[4] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. STD 66. Reston, Virginia: ISOC, January 2005

[5] H. Thompson and C. Lilley. XML Media Types. RFC 7303. Reston, Virginia: ISOC, July 2014.

[6] *Space Assigned Numbers Authority (SANA) - Role, Responsibilities, Policies and Procedures*. Issue 3. CCSDS Record (Yellow Book), CCSDS 313.0-Y-3. Washington, D.C.: CCSDS, October 2020.

[7] *Reference Architecture for Space Data Systems*. Issue 1. Recommendation for Space Data System Practices (Magenta Book), CCSDS 311.0-M-1. Washington, D.C.: CCSDS, Septenber 2008.

NOTE – Informative references are listed in annex D.

2 OVERVIEW

2.1 GENERAL

This section introduces the concepts behind the MPS services. It has the following main sections:

– Mission Planning & Scheduling Concept;

– Relationship to Mission Operations Services;

– MPS Information Model Overview;

– MPS Services Overview;

– Optional Elements of the Recommended Standard.

2.2 MISSION PLANNING & SCHEDULING CONCEPT

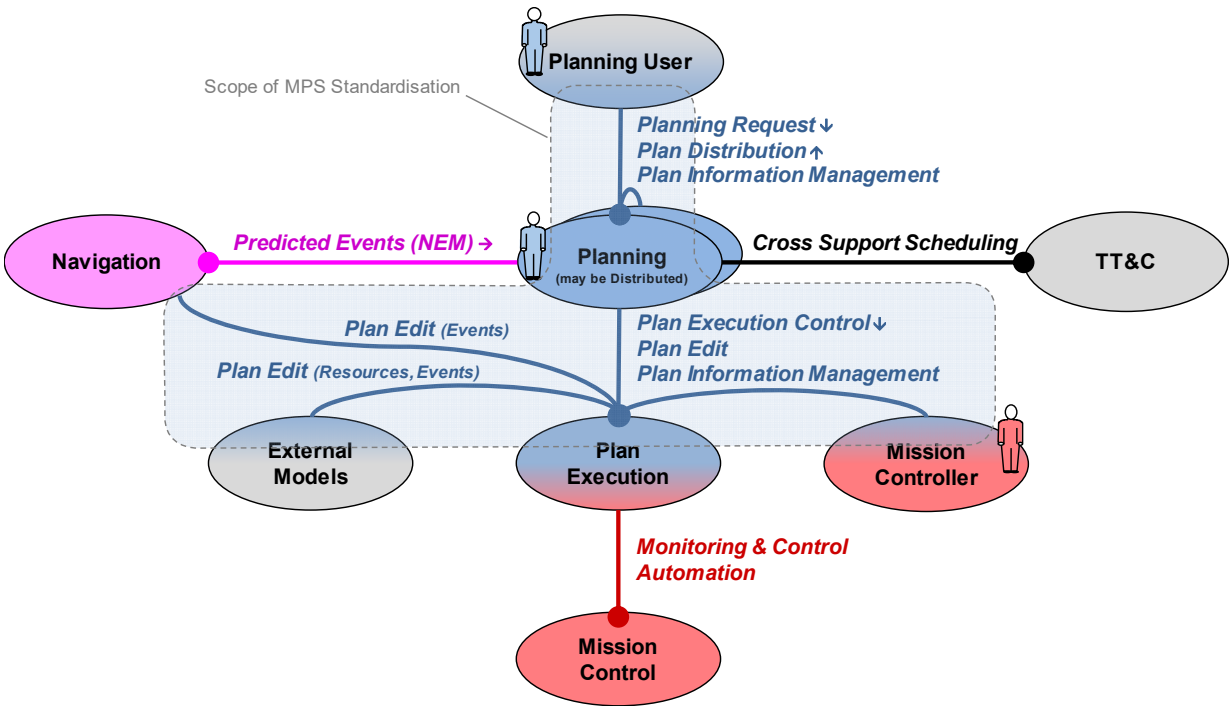

Figure 2-1: Functions involved in Mission Planning

Mission planning & scheduling encompasses application level functions of a space mission system that may be distributed across multiple organizations and physical nodes, both in the space and ground segments. Standardization in this area concerns only the interaction between functions at the application level, and not the mission planning functions themselves.

The scope of standardization includes both the format/model of data exchanged, as well as the semantics of the interactions for their exchange, captured by the associated service level interfaces. A generalized view of the functions involved in mission planning and their interactions with other functions is given in figure 2-1. The entities shown in blue are in the functional area of mission planning. The entities shown in different colours belong to other functional areas of mission operations, such as monitoring and control, navigation, and the ground station and communication network.

The following mission planning functions are identified:

- Planning User: a generic function that is responsible for submitting requests to the planning function. It may also receive feedback on the status of planning requests and the generated plans. It is not a planning function itself, but is a user of planning data and services. A deployment in an actual space mission may contain multiple types of planning user function, some of which correspond to other mission operations functions within the space mission system.

- Planning: the function responsible for performing mission planning. Internally it may be hierarchically organized and/or distributed. Planning requests are received from multiple Planning users (or other mission planning functions) and feedback on their status is provided. The output of the planning function is plans, which may be retrieved by planning users and submitted to plan execution functions. Planning may also control the execution of plans via the plan execution functions. Planning is itself a user of the navigation function and may receive predicted planning events, as a future standard Navigation Event Message (NEM) (reference [D5]) or in a custom format, that are related to orbital information, attitude, or slew times; and negotiates the scheduling of ground station support via Cross Support Services (CSS) services (reference [D7]).

- Plan Execution: the function responsible for executing a plan (or part of it). There may be multiple plan execution functions distributed between space and ground segments. It is not a planning function itself, but it does support a common model of the plan in its interface with planning. It receives or retrieves distributed plans; allows external control of the plan execution process; and provides feedback on execution status of the plan. Plan execution may use underlying mission control services to effect the execution of planned activities. Mission controllers may interact with plan execution functions to control the plan execution process and to edit plans. External functions may also edit plans, for example to update planning events or resources.

It should be noted that in an actual deployment, there may be multiple copies of all the functions identified in figure 2-1.

Figure 2-1 shows MPS services as blue lines. Interactions supported by other CCSDS Recommended Standards are shown in other colours. The circle at one end indicates which function is the service provider: Planning is the service provider for Planning Request and Plan Distribution services. Plan execution is the service provider for Plan Execution Control

and Plan Edit services. Both functions can provide the Plan Information Management service. These services are introduced in 2.5 below and formally defined in section 0.

The identified functions may be distributed over a number of distinct entities (organizations and systems) within a given space mission system. There is not a fixed set of such entities, but typical examples include:

– User Community / PIs;
– Science/Payload Operations Centre;
– Payload Processing Centre;
– Mission Operations Centre;
– Flight Dynamics / Navigation;
– Ground Tracking Network;
– Unmanned Spacecraft;
– Surface Lander / Rover;
– Manned Space Vehicle.

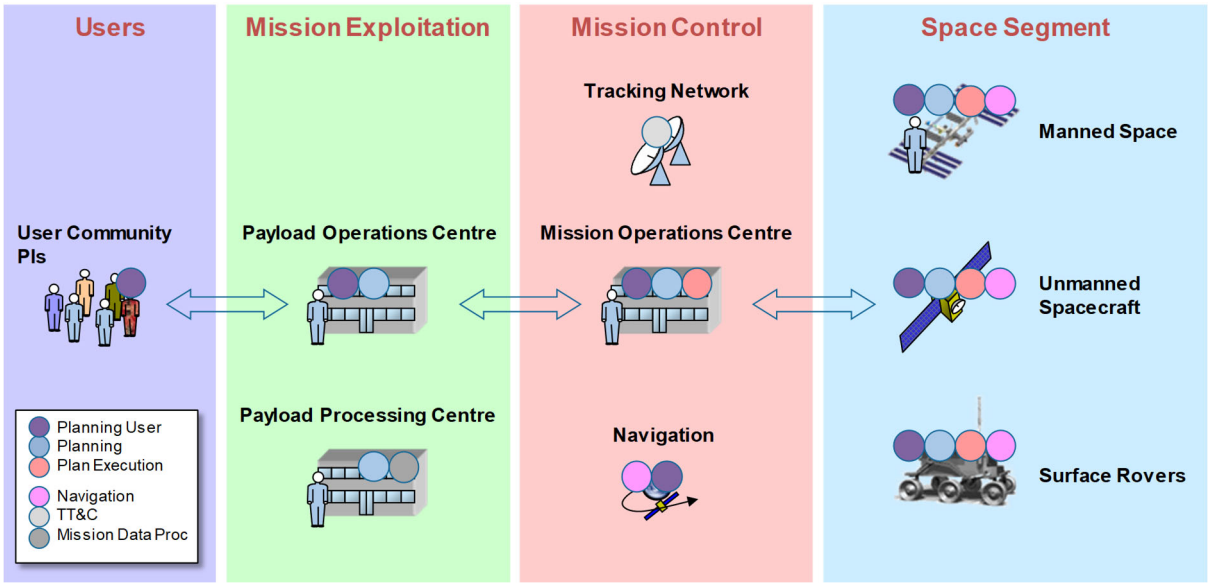

Figure 2-2: Entities and Functions Involved in Mission Planning

As an example, figure 2-2 illustrates potential deployment of each of the functions identified in figure 2-1 to the entities listed above. The circles indicate where each of the functions are typically deployed in existing systems, or where they could potentially be deployed in the future. The arrows indicate the interactions in a typical current deployment, but the potential

distribution of functions indicated by the circles shows that all the functional interfaces shown in figure 2-1 can be exposed to the boundaries between entities. It is where the interactions between the functions are exposed across one or more boundaries between entities that there is a need for standardization within CCSDS as a potentially interoperable interface between agencies.

The interactions within the scope of mission planning and scheduling standardization can be grouped into five services:

- **Planning Request**: submission of planning requests to a planning function, associated responses and their subsequent management and status feedback.
- **Plan Distribution**: distribution and access to plans generated by the planning function.
- **Plan Execution Control**: submission of plans to a plan execution function, management of the execution process, and status feedback.
- **Plan Information Management**: access to planning data definitions.
- **Plan Edit**: direct manipulation of plans outside the planning process, either to update planning events and resources with the latest information or for emergency intervention.

A common MPS information model applies to the planning requests and plans transferred or Referenced by these services and also to the common configuration data required by service providers and consumers to interpret the planning requests and plans. This information model is introduced in 2.4 below and the associated data structures formally defined in section 4.

For those organizations that do not wish to standardize the service level interaction, but only to standardize the data format used for the exchange of planning requests and plans, standard XML-based file formats are defined in section 7.

2.3 RELATIONSHIP TO MISSION OPERATIONS SERVICES

The MO Services Concept provides a standard framework for the specification of end-to-end services between mission operations applications (reference [D1]). MO services are defined in terms of a MAL (reference [2]), which provides a means of specifying data and service interfaces in an implementation, encoding, and communication agnostic manner.

The following figure is taken from the CCSDS Application and Support Layer Architecture (reference [D11]) and shows the generic protocol stack for the MPS services.

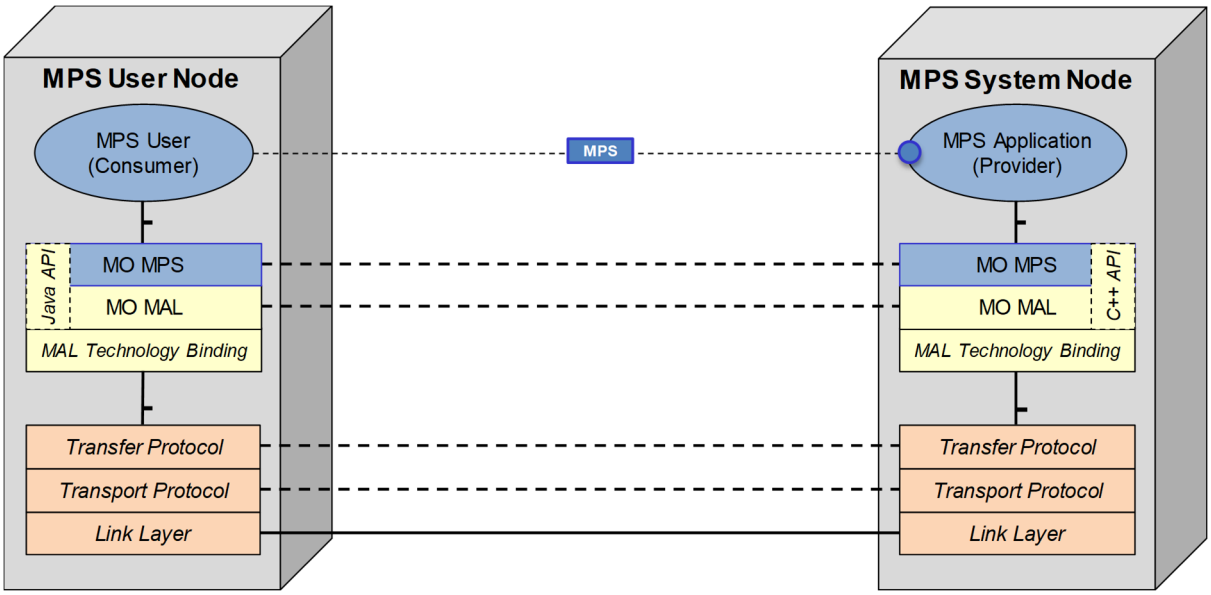

Figure 2-3: MO MPS Services Generic Protocol Stack

The MO MAL defines:

– A set of MAL Attribute data types that can be used to represent the individual data fields of message structures;

– A set of MAL Interaction patterns that correspond to the message exchange behaviour of individual service operations.

The abstract specification of the service interfaces and data can be mapped to a concrete implementation through:

a. a technology binding that defines how the abstract messages (composed of a sequence of MAL attributes) are encoded in a concrete format (e.g., binary, XML, or ASCII);

b. a technology binding that defines how the resulting messages are carried over a concrete message transport protocol by mapping the standard MAL interaction patterns to that protocol;

c. a language binding that transforms the abstract service interface into a concrete API for a given programming language (e.g., Java, C++ or Python).

Figure 2-3 illustrates a generic deployment of MPS services using the MO service framework with service consumer and provider functions hosted on different deployment nodes. MPS specific functions and protocol layers are shown in blue; elements of the 'vanilla' MO framework in yellow; and underlying communications infrastructure layers in tan (light orange). The application level MPS service interaction is shown by the direct interface between service provider and consumer functions, carrying MPS service messages defined in terms of data structures specified in the MPS information model.

Adopting a single MAL technology binding in any specific deployment ensures on-the-wire interoperability. Transfer protocol equates to the messaging or file transfer service used over the underlying Transport and physical Data Link Layers. The diagram illustrates how different language bindings can be used by provider and consumer for the service API, as this does not affect the wire level protocol.

It is noted that while the MAL may be implemented as a specific software layer for reasons of maintainability and reusability, it is not a requirement to do so. The MAL may be used as an abstract specification that enables transformation of the service specification by the applied technology bindings into a concrete implementation of that service with no distinct MAL layer. The MO MPS, MO MAL, and MAL technology binding layers in the diagram are effectively combined into a single software component. This is an important distinction for deployment contexts where the implementation is required to be both compact and efficient, such as on-board a spacecraft.

An MO object is an entity defined within the information model of an MO compliant service specification that has a unique identity enabling it to be referenced by other MO objects and in the body of MO service messages. The **identity** of an MO object is defined by its type and unique **key**, scoped by its **area**, **domain** and optionally by a **version**. The specification of a service-specific MO object class includes a custom set of references to other MO objects that capture the relationships between those objects. In the context of the MPS services, MO objects are defined to represent planning requests, plans and the planning activities, planning events, and resources that they reference.

An MO application-level service specification comprises a set of operations that the service consumer may invoke on the service provider. Each operation is mapped to a standard interaction pattern defined by the MAL and provides the service-specific body of the constituent messages.

In addition to the MAL and technology bindings, the MO service framework includes a set of Common Services (reference [D3]) that can be used in conjunction with any MO application level service. These include a Directory, Login, and Configuration services.

2.4 MPS INFORMATION MODEL OVERVIEW

2.4.1 GENERAL

The information exchanged across the interfaces supported by MPS services is complex. Requests for activities to be planned require the specification of the activities to be performed and any constraints on their execution. Plans express not only the planned activities they contain, but also what triggers their execution relative to time, position or planning events.

The MPS information model describes the information objects that are transferred or operated on by service operations and the relationships between them. It addresses both the content of the messages that constitute service operations and also the configuration data that both consumer and provider must have access to to interpret those messages.

This document contains the specification of data structures derived from the MPS information model that are used in the body of service messages or referenced by them. These data structure definitions are a normative part of the Recommended Standard and are expressed in the tabular format described in 1.8.2.

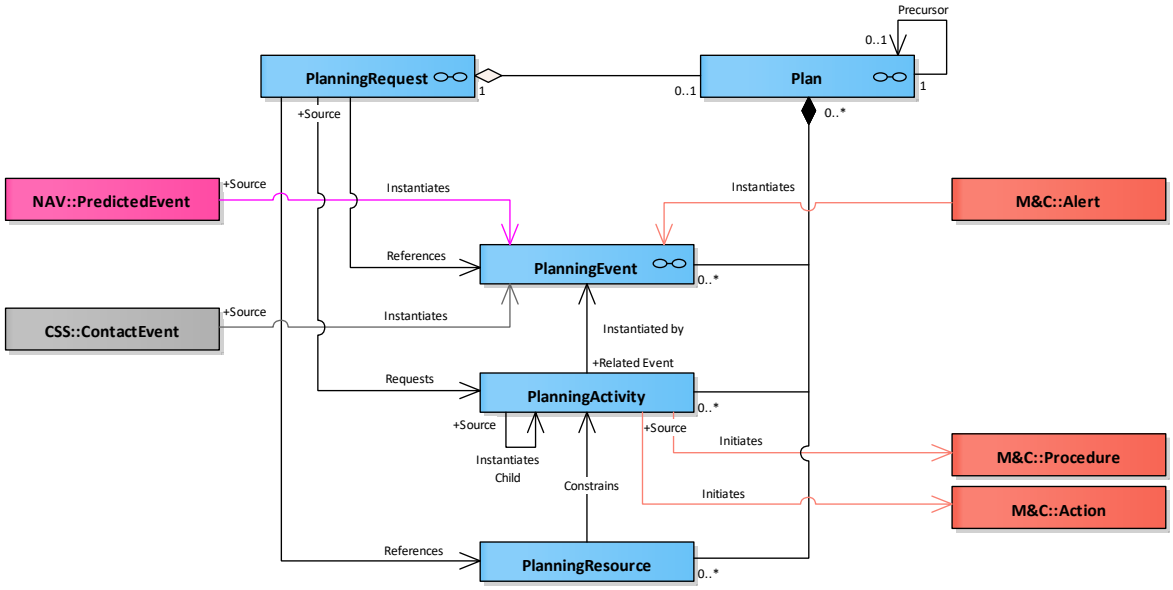

Figure 2-4: MPS Data Items

A high-level overview of the MPS information model is given in figure 2-4. This shows the principal MPS data items and their interrelationships using standard UML notation.

The rectangles in the diagram correspond to standard data items. The lines between them define the relationships between those data items. Data items are color-coded by functional area:

- Mission Planning data items defined in this document are shown in blue.
- Mission Control data items (red) correspond to CCSDS MO Monitoring & Control and the proposed MO Automation standards.
- Navigation data item (magenta) correspond to the CCSDS Navigation Event Message Recommended Standard.
- Cross Support Services data items (grey) correspond to CCSDS Cross Support Service Management Recommended Standards.

Relationships shown in black are within the scope of mission planning standardization, others are color-coded by their respective area.

The following principal MPS data items are shown in the diagram:

– **Planning Requests;**

- **Plans;**
- **Planning Activities;**
- **Planning Events;**
- **Planning Resources.**

These are introduced in the subsections below. Each of these data items comprises a set of **MO objects** with its own object **identity**, following one of the typical MO object patterns defined in 1.6.

The status of planning requests, plans, planning activities, and planning events can evolve over time and is reported through the defined MPS services. State models are associated with each of these data items in the full information model, but only the minimum set of states has been defined to work with the defined services: missions may effectively extend these with additional states using additional information fields.

Planning requests and **plans** are both container objects, whose content relates to a set of planning data items: **planning events**, **planning activities**, and **planning resources**. For each of these three types (or classes) of planning data, there is a defined set of items that can be referenced or instantiated within planning requests and plans. Together these definitions comprise the **planning configuration data**.

Planning constraints are not self-standing data items, but can be attached to planning requests and planning activities. They are defined in 4.3.6. Subsection 4.3 also includes the definition of other MPS supporting data types, including:

- MPS **Position** and **Direction** data types;
- **Expressions;**
- **Arguments;**
- **Triggers;**
- **Repetitions.**

Some aspects of the MPS information model are optional. These aspects are not required to be supported by a compliant MO MPS service provider, although this may limit the set of service capabilities and associated operations that can be supported. Optional aspects of the model include:

- Planning Resources;
- Functions;
- Planning Constraints, other than representation as a text expression using any defined expression syntax supported by the service provider;
- Position and Direction data types;

– Repetitions (the representation of repetitive occurrences of **planning activities**).

2.4.2 MO OBJECTS

Figure 2-5 shows how each of the MPS Data Items comprises multiple MO objects following a 1, 2, or 3 element MO object pattern. MO objects are shown with a bold purple border.

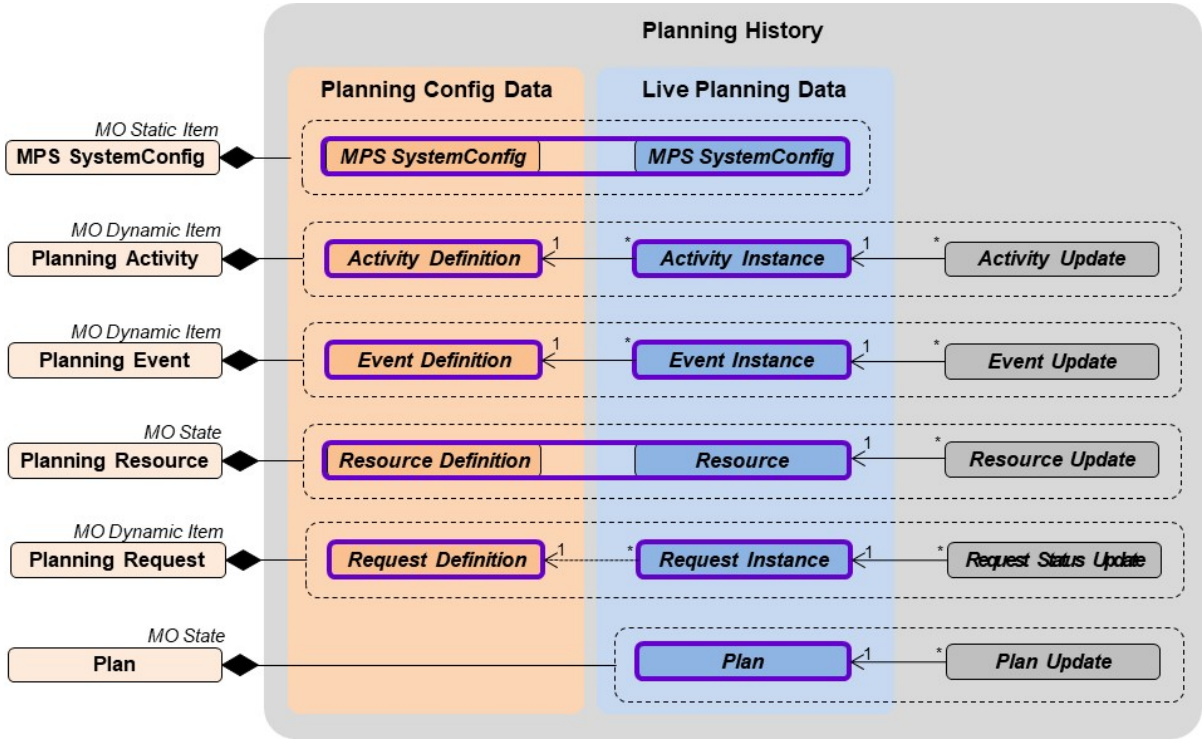

Figure 2-5: MPS Data Items and their Constituent MO Objects

Each MO object has a unique object **identity**, which includes an unchanging key (Identifier) and optionally a **version**. When a **definition** object is updated, it retains the same key, but its **version** is updated to uniquely identify a specific version of the definition. Planning request instances and plans also have an associated version.

The **definition** objects associated with **planning activities**, **planning events**, **planning resources** and **planning requests** all form part of the planning configuration data that must be available to both communicating parties that exchange **planning requests** and **plans**. Planning configuration data also includes a single MPS SystemConfig object that defines overall configuration parameters for the entire MPS system. Definition objects comprise only static attributes. The MPS services defined in this Recommended Standard do not address the bulk transfer of MPS configuration data between communicating parties, but individual definitions can be accessed using the Plan Information Management Service.

The **instance** objects associated with all MPS data items, are the live planning data created by and exchanged between communicating parties during planning and plan execution.

Instance objects may comprise both static and dynamic attributes. For **planning resources** and MPS SystemConfig, the live instance has the same object **identity** as the associated **definition**, the only distinction being that the live instance may include dynamically updating attributes, such as the value of a **planning resource.**

Changes in state of the dynamic attributes of live planning data **instance** objects may be notified through an **update** structure exchanged through MPS service messages and optionally stored in planning history. These updates are not themselves MO objects, but reference the corresponding instance and the timestamp of the update as well as the value of dynamic attributes.

Planning requests may reference **planning activities**, **planning events**, and **planning resources** but do not contain instances of planning activities or planning events. Instead they specify the **activity details** for the requested planning activities.

Plans contain instances of planned activities, and optionally of planning events. They may also optionally contain planning resource profiles that express their planned evolution over time.

2.4.3 PLANNING REQUEST

Planning requests are the main input to the planning function. A planning request is a container for the information needed to be exchanged between the requester and the planner. It supports the specification of a request to plan one or more **planning activities**. Alternatively, it can support a request to use an existing **plan** (already containing a number of **planning activities**) as an input to the planning process. It can constitute a one-off planning request, or request the repetitive planning of activities as a ‘standing order’.

The main characteristic of the planning request is that, being a container, it needs to hold references to, or instances of, the constituent information items that are required by the planner and agreed by the interacting parties for exchange at interface level. It has one or more **planning activities** as the basis of the request. In addition, the request may optionally reference **planning events**. Information about **planning constraints** on when a requested activity can or shall be planned may be exchanged as part of the planning request, by referencing constraints on the timing or position of planning activities, both absolute and relative to planning events or other planning activities, and on the state of **planning resources**.

2.4.4 PLAN

The **plan** is the output of a planning process. The plan is basically a container of one or more selected **planning activities**, optionally associated with **planning events**. In addition, the usage of **planning resources** may be contained in the plan. The plan may contain specific information from the planning process, which applies to the plan as a whole. In the

hierarchical and distributed planning concepts, the output of one planning function could be the input of another one. As such, a **planning request** could refer to an entire plan.

Plans may be iterative, and therefore overlap with the previous plan. This introduces the notion that a plan may have an identified predecessor or **precursor plan**, and also that if a planning data item is contained in multiple iterations of the plan, then it should have the same unique **identity**, except for an updated **version**) in each successive iteration of the plan to avoid ambiguity and duplication.

Plans comprise the following main elements:

- Plan Information: header data relating to the **plan** as a whole;
- Planned Items: the set of contained **planning activities** and **planning events**;
- Plan Revisions [optional]: summaries of the changes between this version of the plan and another specified version of a plan, usually its precursor plan;
- Plan Resources [optional]: value profiles covering the period of the plan for a set of **planning resources**.

2.4.5 PLANNING ACTIVITY

A **planning activity** is the basic building block for the planning: a meaningful unit of what can be planned. As such, it has to be understood by the planning function. It could eventually be translated to something that can be executed by a plan execution function; this includes CCSDS MO M&C **actions** (reference [D4]) (that may represent telecommands) and CCSDS MO Automation **procedures** (reference [D1]) (that may represent any automated telecommand sequence, operational procedure, on-board control procedure, or function).

Planning activities support hierarchy: a planning activity may be composed of one or more subordinate planning activities. A planning activity may define **arguments** (parameters), which could be used to instantiate a specific planning activity in a plan, based on its generic definitions. Arguments of a planning request or planning event can be passed through to the arguments of planning activities resulting from these. Arguments can then similarly cascade down through a hierarchy of planning activities. A plan execution function may then flow down these arguments to any **action** or automated **procedure** initiated.

Planning constraints can also be associated with a planning activity, either generic constraints applicable to all occurrences (or instances) of the planning activity that are contained within its definition, or specific constraints associated with a particular instance that are defined in the context of the **planning request**. These planning constraints can be expressed in terms of the timing or position of a planning activity, both absolute and relative to planning events or other planning activities, and on the state of **planning resources**.

2.4.6 PLANNING EVENT

A **planning event** marks when a condition is being met, expressed in terms of time or position. They may be used to represent predicted or planned events, such as predicted orbital events or planned periods of contact with a spacecraft, that are typically received as an input by the mission planning function, from an external function, such as navigation.

Planning events may be grouped hierarchically to represent a compound event, such as the start and end of a satellite pass over a ground station (AOS/LOS), or a satellite passing through eclipse (penumbra entry, umbra entry, umbra exit, penumbra exit). A planning event may define **arguments** (parameters) to convey additional information relevant to the planning process.

Planning activities may be linked to a related planning event. The start or end of the planning activity can be relative to the planning event, and the **arguments** of the event can be flowed down to the planning activity. **Planning requests** may also reference planning events, associating them with requested planning activities.

Planning events may be classified as **predicted events** or **potential events**:

- Predicted events are those that are expected to occur at a particular time or position that is known at the time of planning and can be contained within a **plan**. Uncertainty in the timing of predicted events may be refined closer to their time of occurrence. This can either be handled by re-planning, or by updating the events within an executing plan.
- Potential events are not predictable, but may still have a defined response within a plan: virtual observatory Target Of Opportunity (TOO) events are an example. Such events can be inserted into an executing plan.

2.4.7 PLANNING RESOURCE

A **planning resource** is an abstract status modelling the state of the system being planned. It may be necessary to model some aspects of system state in order to:

- trigger the execution of a planning activity;
- constrain the execution of a planning activity;
- define the effect that the execution of a planning activity has on the planning resource.

A planning resource is in effect a value of defined type that can evolve over time. A resource profile can be used to capture and communicate that evolution over time in the context of a **plan**.

If an event or constraint on a planning activity needs to be expressed in terms of the state of the system (rather than just time or position) then this corresponds to the state of planning

resources. This is considered not internal to the planning function, if it forms part of the **planning request** or **plan**.

A planning resource could in principle be considered as information that is internal to the planning system. However, some resources may be shared across multiple planning entities. As such, information regarding a resource may need to be communicated between entities, and therefore has to be referenced as part of a planning request or of the plan, in terms of requested or consumed resources respectively. This may include the initialization or synchronization of planning resource values at specific points in the plan.

Planning resources are an optional element of the MPS information model. There is no requirement for a compliant MPS system to support them.

2.5 MPS SERVICES OVERVIEW

2.5.1 GENERAL

This document specifies standards for the following MPS services, which are introduced in the following subsections:

- Planning Request Service (PRS);
- Plan Distribution Service (PDS);
- Plan Execution Control Service (PECS);
- Plan Information Management Service (PIMS);
- Plan Edit Service (PES).

A compliant MPS system may support only a subset of these services. None of the services are mandatory, any subset can be supported.

Each service comprises a set of service operations that the service consumer can invoke on the service provider. Service operations reference the MO objects defined in the MPS information model. Each service operation follows one of the standard MO MAL interaction patterns that may comprise multiple messages flowing in both directions between service consumer and provider. Where not fully defined within the MAL, the body of these messages forms part of the service specification and are defined in terms of MAL Attribute data types and/or data structures defined in the MPS information model.

Service operations are grouped into capability sets. A compliant MPS system may only support a subset of capability sets for each supported service.

2.5.2 PLANNING REQUEST SERVICE

The Planning Request Service is offered by the planning function of an MPS system to enable its users to submit, cancel, and modify planning requests, as well as to receive feedback on their status. The service may be used by another planning function in a hierarchical or distributed MPS system, or by an MPS system user.

Planning requests are defined in 4.2.5 and may include a set of requested planning activities or an existing plan (the output of a planning function in a hierarchical or distributed MPS system).

If a Plan is used as the body of the request, this can either be embedded within the request, or passed by reference. If passed by reference, the Plan Distribution Service can be used to retrieve the Plan. In a hierarchical or distributed planning system, the **domain** can be used to identify where to retrieve it from.

The following capability sets and service operations are defined:

Table 2-1: Planning Request Service Capability Sets and Operations

Capability Set	Mandatory	Description
1	Yes	Submit Planning Request, obtain a list of Planning Request Summaries and retrieve Planning Request Status.
2	No	Cancel Planning Request
3	No	Update (modify) Planning Request
4	No	Subscribe to Planning Request Status Updates
5	No	Retrieve Planning Requests

Operation	Description	Capability Set
submitRequest	Send planning request to service provider, optionally based on a planning request definition. The identity of the planning request instance created is returned.	1
getRequestSummaries	Request a list of available planning requests from the service provider, subject to a specified filter. The returned list is provided as a set of summaries comprising identity, descriptive header fields and status for each planning request instance passing the filter.	
getRequestStatus	Obtain the current status of one or more specified planning requests. The response is a list of planning request status updates, containing the status and other dynamic attributes of each requested planning request instance.	
cancelRequest	Send cancellation of a previously submitted planning request to service provider.	2

updateRequest	Send update of a previously submitted planning request to service provider. A new version of the planning request instance is created and its identity is returned.	3
monitorRequestStatus	Subscribe to receive planning request status updates for a filtered set of planning request instances. The consumer is notified of updates to the status and other dynamic attributes of subscribed planning request instances.	4
getRequest	Retrieve the full content of one or more specified planning request instances.	5

2.5.3 PLAN DISTRIBUTION SERVICE

The Plan Distribution Service is offered by the planning function of an MPS system to enable its users to obtain the plans output by it, as well as to receive feedback on their status. The service may be used by another planning function in a hierarchical or distributed MPS system, or by an MPS system user. Plans are defined in 4.2.6.

The service does not provide the capability to control the planning function itself or to generate plans. This capability can be supported if the planning function exposes a standard set of MO Monitoring & Control services. Submission of plans to a plan execution function is supported by the Plan Execution Control Service (see 2.5.4).

The following capability sets and service operations are defined:

Table 2-2: Plan Distribution Service Capability Sets and Operations

Capability Set	Mandatory	Description
1	Yes	List available Plans and their latest status, obtain specified Plans or their latest status.
2	No	Subscribe to updates in Plan status.
3	No	Subscribe to receive new Plans
4	No	Query to retrieve filtered set of Plans
5	No	Retrieve a partial Plan, based on specified criteria

Operation	Description	Capability Set
getPlanSummaries	Request a list of available plans from the service provider, subject to a specified filter. The returned list is provided as a set of summaries comprising identity, descriptive header fields and status for each plan passing the filter.	1
getPlan	Retrieve the full content of one or more specified plans.	
getPlanStatus	Obtain the current status of one or more specified plans. The response is a list of plan updates, containing the status	

	and other dynamic attributes of each requested plan.	
monitorPlanStatus	Subscribe to receive plan updates for a filtered set of plans. The consumer is notified of updates to the status and other dynamic attributes of subscribed plans.	2
monitorPlan	Subscribe to receive a filtered set of plans, receiving the full content of the plan when published by the provider.	3
queryPlan	Query to receive a filtered set of plans, based on an extended set of filter criteria, including on the planning activities and events contained within a plan.	4
getPartialPlan	Retrieve a subset of a plan, covering a more restricted period or only containing selected planning activities.	5

2.5.4 PLAN EXECUTION CONTROL SERVICE

The Plan Execution Control Service is offered by an MPS system's plan execution function to enable its users to submit (and revoke) plans for execution; to control their execution at plan, sub-plan, and activity levels; and to receive feedback on their execution status.

The Plan Execution Control Service may be used by a planning function, or by an MPS system user responsible for mission operations.

Plans are defined in 4.2.6, including the state model for plans to which the operations of the service correlate.

Sub-plans are not defined as an MO object in the MPS information model, but are specified by an Identifier associated with the constituent ActivityInstances contained in a Plan. This can be used to sub-divide a Plan based on domain (spacecraft or subsystem), operational responsibility, or another criterion. Each ActivityInstance can only be associated with a single sub-plan. Control may be exercised via the service at the level of sub-plans.

The following capability sets and service operations are defined:

Table 2-3: Plan Execution Control Service Capability Sets and Operations

Capability Set	Mandatory	Description
1	Yes	Submit Plan for execution; revoke Plan previously submitted; and retrieve execution status of Plans.
2	No	Activate and deactivate execution of Plans.
3	No	Subscribe to receive execution status of Plans (at the level of the Plan).
4	No	Subscribe to receive detailed execution status of Plans (at the level of contained planning activities, events and resources).
5	No	Activate and deactivate execution of Sub-plans, and retrieve execution status of Sub-plans.

Capability Set	Mandatory	Description
6	No	Subscribe to receive execution status of Sub-plans.
7	No	Suspend and resume execution of ActivityInstances.
8	No	Retrieve a detailed report on the execution status of ActivityInstances.

Operation	Description	Capability Set
submitPlan	Send a plan to a plan execution function, making it available for execution. Note: A plan may be a patch plan.	1
revokePlan	Instructs a plan execution function to revoke a previously submitted Plan, making it unavailable for execution.	
getPlanStatus	Retrieve the current execution status of plans.	
activatePlan	Enables the execution of specified plans.	2
deactivatePlan	Disables the execution of a specified plans, subject to an implementation specific deactivationMode.	
monitorPlanExecution	Subscribe to receive plan updates for a filtered set of plans. The consumer is notified of updates to the status and other dynamic attributes of subscribed plans.	3
monitorPlanExecutionDetail	Subscribe to receive updates that report changes in the detailed execution status for a filtered set of plan contents at the level of planning activities, events and resources.	4
activateSubPlan	Enables the execution of planning activities associated with specified sub-plans.	5
deactivateSubPlan	Disables the execution of planning activities associated with specified sub-plans, subject to an implementation specific deactivationMode.	
getSubPlanStatus	Retrieves the current status of sub-plans.	
monitorSubPlanExecution	Subscribe to receive updates on execution status of sub-plans.	6
SuspendActivity	Request suspension of the execution of selected activities in one or more plans without changing the state of the plan(s), subject to an implementation specific suspensionMode.	7
ResumeActivity	Requests the resumption of previously suspended activities in one or more plans without changing the state of the plan(s).	
getActivityStatus	Requests a detailed report on the status of activities at activity, sub-plan or tag level	8

2.5.5 PLAN INFORMATION MANAGEMENT SERVICE

The Plan Information Management Service is offered by the planning function of an MPS system to enable its users to list and retrieve available definitions for MPS data items, including: planning requests, planning events, planning activities, planning resources, and MPS system configuration data. The service may also be offered by a plan execution function.

The service does not support the transfer of planning configuration data to planning or plan execution functions, which is outside the scope of the current MPS services. Nor does it support the insertion or modification of MPS data item definitions.

The structure of the various MPS data item definitions is given in 4.2.

The following capability sets and service operations are defined, each pertaining to a specific type of MPS data item:

Table 2-4: Plan Information Management Service Capability Sets & Operations

Capability Set	Mandatory	Description
1	No	List and retrieve planning RequestDefinitions
2	No	List and retrieve planning EventDefinitions
3	No	List and retrieve planning ActivityDefinitions
4	No	List and retrieve planning ResourceDefinitions
5	No	Retrieve MPS system configuration data [MPSSystemConfig]

Operation	Description	Capability Set
listRequestDefs	Request a filtered list of available RequestDefinitions.	1
getRequestDefs	Retrieve a set of available RequestDefinitions.	
listEventDefs	Request a filtered list of available EventDefinitions	2
getEventDefs	Retrieve a set of available EventDefinitions.	
listActivityDefs	Request a filtered list of available ActivityDefinitions	3
getActivityDefs	Retrieve a set of available ActivityDefinitions.	
listResourceDefs	Request a filtered list of available Resource definitions	4
getResourceDefs	Retrieve a set of available Resource definitions	
getSystemConfig	Retrieves system configuration data relating to the MPS system.	5

2.5.6 PLAN EDIT SERVICE

The Plan Edit Service is offered by an MPS system's plan execution function to enable its users to modify plans that have already been submitted for execution. It allows an external user or function to update the status of the plan; insert, modify, or delete planning activity and event instances; update the value of resources; and apply a time shift to a plan.

This may be used by expert mission operations users in a non-nominal operational scenario to modify a plan that is executing or about to execute in order to avert or recover from a failure. Where there is sufficient time, it is recommended to re-plan using the nominal planning process rather than to edit the plan directly, as this circumvents any constraint checking performed by the planning function.

Another use of the service is for a third party functions to update elements of the plan to reflect information available in near-real-time. Examples are:

– Update of the input arguments of a planned activity instance to fine tune its behaviour in response to currently observed status.

– Update of a predicted planning event instance that is already contained within a plan, with refined timing or other details.

– The injection of instances of planning events. These may be detected in real-time and correlate to potential (rather than predicted) events for which a planned response is available. An example is a 'Target of Opportunity' event that may be notified to an astronomical observatory mission.

– Update of a resource value by an external modelling function to more accurately reflect currently observed status.

Plans are defined in 4.2.6, while their constituent planned items are defined in 4.2.2 (planning activities), 4.2.3 (planning events) and 4.2.4 (planning resources).

The following capability sets and service operations are defined:

Table 2-5: Plan Edit Service Capability Sets and Operations

Capability Set	Mandatory	Description
1	Yes	Update Plan status.
2	Yes	Insert or Delete ActivityInstances or EventInstances.
3	No	Update ActivityInstances or EventInstances.
4	No	Update Resource value.
5	No	Update Resource profile.
6	No	Apply a time shift to a Plan.

Operation	Description	Capability Set
updatePlanStatus	Update plan status or isAlternate flag.	1
insertActivity	Insert a new ActivityInstance into a plan.	2
insertEvent	Inject a new EventInstance into a plan.	
deleteActivity	Delete a specified ActivityInstance from a plan.	
deleteEvent	Delete a specified EventInstance from a plan.	
updateActivity	Update a specified ActivityInstance in a plan.	3
updateEvent	Update a specified EventInstance in a plan.	
updateResource	Make a discrete update to the value of a specified Resource at a specified time.	4
updateResourceProfile	Update the profile of a specified Resource over time.	5
applyTimeShift	Apply a time shift to a specified plan or its sub-plans.	6

2.6 OPTIONAL ELEMENTS OF THE RECOMMENDED STANDARD

The MPS services specification defines a substantial MPS information model and five services. Compliance of an individual mission planning system deployment to the MPS service specification does not imply that either the complete set of services specified here or the full MPS information model has to be supported.

The level of compliance of a specific deployment to the Recommended Standard can be selected as follows:

a) the set of MPS services supported;

b) the capability sets supported within each MPS service;

c) the optional elements of the MPS information model supported.

The supported capabilities of a service provider can be made available to consumers via the MO Common Directory Service (reference [D3]). An entry in the directory is made for each MPS service supported by each MPS service provided, which lists the supported capability sets for that service.

The elements of the MPS information model are grouped into element sets detailed in table 2-6 below. Of these, only the Core Features are mandatory, subject to the further caveat that only those MPS information model elements exposed by supported capability sets of MPS services need to be implemented. Optional element sets are shown with a grey background.

At the interface level, a deployment must support all data structures that may appear within the messages of supported service operations. However, the deployment is not required to generate any data structure from an element set it does not support. If the deployment

receives a message containing a data structure from an optional element set it does not support, it can either ignore it or reject the service operation with an UNSUPPORTED error.

It should be noted that support for Resource Constraints is dependent on support for Resources, and that support for Geometric Constraints is dependent on support for Position & Direction types.

The optional element sets of the MPS information model are orthogonal to the capability sets of the MPS services. For example, a Planning Request Service provider may support the submission of planning requests, but not the representation of Basic Constraints within those planning constraints.

Table 2-6: Mandatory and Optional Elements of the Information Model

#	Information Model Element Set	MO Objects	MPS Data Types	Constraints
1	Core Features (Mandatory)	Planning Requests Plans Planning Activities Planning Events MPSSystemConfig	Base Data Types (excl. Position & Direction) Expressions Additional MPS Data Types Arguments Time & Event Triggers Time & Event Repetitions	Constraint Expression
2	Basic Constraints			Temporal Constraints Sequential Constraint Exclusion Constraint
3	Plan Revisions	Patch Plans	Plan Revisions	
4	Resources	Resources	Resource Profiles Plan Resources	
5	Resource Constraints (requires Resources)			Resource Constraints Argument Constraint Effects
6	Position & Direction		Position & Direction Types Location, Pointing and Angle Triggers Location, Pointing and Angle Repetitions	
7	Geometric Constraints (requires Pos. & Dir.)			Geometric Constraints
8	Functions	Functions		Function Constraint

The following table summarizes the optional capability sets and information model elements for all MPS services. Optional capabilities and elements are shown with a grey background.

For each service capability set, applicability of information model element sets are shown as follows:

- ✓ Required;
- O Optional;
- - Not Applicable.

If a service capability set is supported, then the information model element sets indicated with a ✓ must also be supported. If the service capability set operations include messages that contain data structures from optional element sets, then this is shown with an O. If the service operations do not contain data structures from optional element sets, then this is shown as not applicable. An indication is also given of which service specific data structures are relevant to the operations of each service capability set. It should be noted this is shown at the level of the MPS data items defined in 4.2: only the specific data structures used in the messages of the associated service operations are required.

Table 2-7: Optional Service Capabilities

Services		Information Model						
Service	Capability Set	Core Features	Basic Constraints	Plan Revisions	Resources	Position & Direction	Functions	MPS Data Items for which MO Objects or Service Data Structures are used in Service Operations
PRS	1: Submit Request / Status	✓	O	-	O	O	O	Planning Request
	2: Cancel Request	✓	-	-	-	-	-	
	3: Update Request	✓	O	-	O	O	O	Planning Request
	4: Subscribe to Status	✓	-	-	-	-	-	Planning Request
	5: Retrieve Request	✓	O	-	O	O	O	Planning Request
PDS	1: List & Obtain Plans	✓	O	O	O	O	O	Plan
	2: Subscribe to Status	✓	-	-	-	-	-	Plan
	3: Subscribe to Plans	✓	O	O	O	O	O	Plan
	4: Query Plans	✓	O	O	O	O	O	Plan
	5: Retrieve Partial Plan	✓	O	O	O	O	O	Plan
PECS	1: Submit Plan / Status	✓	O	O	O	O	O	Plan
	2: Plan Activation	✓	-	O	-	-	-	Plan
	3: Subscribe to Status	✓	-	-	-	-	-	Plan
	4: Subscribe to Detail	✓	-	-	O	O	-	Planning Activity/Event/Resource
	5: SubPlan Activation	✓	-	-	-	-	-	Plan (SubPlan)

Services		Information Model						
Service	Capability Set	Core Features	Basic Constraints	Plan Revisions	Resources	Position & Direction	Functions	MPS Data Items for which MO Objects or Service Data Structures are used in Service Operations
	6: Subscribe to SubPlans	✓	-	-	-	-	-	Plan (SubPlan)
	7: Activity Suspension	✓	-	-	-	-	-	Planning Activity
	8: Activity Status	✓	-	-	-	-	-	Planning Activity
PIMS	1: Request Definitions	✓	O	-	O	O	O	Planning Request
	2: Event Definitions	✓	-	-	-	-	-	Planning Event
	3: Activity Definitions	✓	O	-	O	O	O	Planning Activity
	4: Resource Definitions	✓	-	-	✓	-	-	Planning Resource
	5: MPSSystemConfig	✓	-	-	-	O	-	MPSSystemConfig
PES	1: Update Plan Status	✓	-	-	-	-	-	Plan
	2: Insert/Delete Activity/Event	✓	O	-	O	O	O	Planning Activity/Event
	3: Update Activity/Event	✓	-	-	-	O	-	Planning Activity/Event
	4: Update Resource	✓	-	-	✓	-	-	Planning Resource
	5: Update Resource Profile	✓	-	-	✓	-	-	Planning Resource
	6: Apply Time Shift	✓	-	-	-	-	-	Plan

3 MPS SERVICE SPECIFICATIONS

3.1 OVERVIEW

3.1.1 GENERAL

This section contains the service specifications for the MPS services. An overview of the services has been given previously in 2.5 above. Each of the following services is defined in a separate section below:

- Planning Request Service;
- Plan Distribution Service;
- Plan Execution Control Service;
- Plan Information Management Service;
- Plan Edit Service.

There is no requirement to implement all services. Any subset of the services may be supported by an MPS deployment.

Each service specification comprises the following parts:

1. Summary: contains a table listing the service operations and grouping them into capability sets, and UML sequence diagrams illustrating the service operations.
2. MO Objects: identifies the set of MO objects (defined in section 4) that are operated on by the service or referenced by service operations.
3. High-Level Requirements: such as the set of configuration data required by the service.
4. Functional Requirements: specific requirements on the behaviour of the service provider and consumer.
5. Operations: specification of each service operation.

All services defined in this document are part of the MPS Area, which has the Area number 5.

3.1.2 DIRECTORY SERVICE AND OPTIONAL ELEMENTS

When using the Directory Service of the MO Common Services (reference [D3]), support for capability sets is indicated in the standard way in the ServiceCapability data structure using the supportedCapabilitySets field.

Support for optional element sets (see 2.6) of the MPS information model is represented in the Directory Service by the use of service provider properties as detailed in the following table:

Table 3-1: MPS Service Provider Standard Properties

Property ID	Property Type	Description
ES2_CONSTR	BOOLEAN	Flag indicating support for Basic Constraints
ES3_REVISION	BOOLEAN	Flag indicating support for Plan Revisions
ES4_RESOURCE	BOOLEAN	Flag indicating support for Planning Resources
ES5_RESCONSTR	BOOLEAN	Flag indicating support for Resource Constraints
ES6_POSDIR	BOOLEAN	Flag indicating support for Position and Direction Types
ES7_GEOCONSTR	BOOLEAN	Flag indicating support for Geometric Constraints
ES8_FUNCTION	BOOLEAN	Flag indicating support for Functions and Function Constraints

3.1.3 PUBLISH-SUBSCRIBE OPERATIONS

The defined Publish-Subscribe operations are based on the updated MAL specification which allows for a set of subscription keys (or filters) to be defined for each operation.

Subscription keys are defined in the service standard for each Publish-Subscribe pattern operation with a name and their associated MAL::Attribute type. This may be extended by a given deployment to support additional subscription keys.

When the provider publishes an **update**, the message includes the following entity keys in addition to the update itself:

– domain List<MAL::Identifier>[ordered];

– subscriptionKeys List<MAL::NamedValue>.

A value must be provided for each of the specified subscription keys and any deployment specific keys for the operation. The **domain** may be omitted if the provider only supports a single domain.

When the consumer subscribes to receive **updates**, the subscription filter within the Register message has the following structure:

– domain List<MAL::Identifier> [ordered] [*=wildcard];

– filters List<SubscriptionKey>.

SubscriptionKey:

– Name MAL::Identifier;

– Values List<MAL::Attribute>.

The **domain** field may be omitted where all available domains are required, or where the provider only supports a single domain. A wildcard may be used at any point in the domain hierarchy, for example: if the domain hierarchy is <mission>.<spacecraft>.<subsystem> then specifying MyMission.*.Power would subscribe to updates to the Power subsystem for all spacecraft in MyMission.

The consumer specifies a list of subscription key filters to restrict the set of updates returned. It is not required to supply a filter for every subscription key; all updates are returned for omitted filter criteria. Each filter comprises the name of the subscription key, followed by a list of required values for that key. If any one of the supplied values is met for a given update, then the filter is passed (the values are ORed by the broker).

The broker filters updates by applying all the specified criteria (the **domain** and specified subscription key filters are ANDed by the broker).

For each publish-subscribe operation defined in this Recommended Standard, the set of subscription keys is defined. It should be noted that in addition to defined subscription keys, the **domain** may also be specified as a subscription filter.

Where a defined subscription key corresponds to an MPS enumerated value, the subscription key is defined with the type MAL::UInteger, where the integer corresponds to the enumerated value. The MPS Enum type cannot be used directly as it is unknown to the MAL broker.

3.1.4 SERVICE OPERATIONS USING DOMAIN AS A FILTER

Some service operations have a message field ‘domain’ that is used to filter or restrict the scope of the operation by **domain**. In this context, the domain field is equivalent to that in the subscription filter of a publish-subscribe pattern operation (see 3.1.3 above). The domain field is an ordered list of identifiers representing a domain hierarchy, any node of which can use ‘*’ as a wildcard (meaning any domain identifier at that level of the hierarchy):

- domain List<MAL::Identifier> [ordered] [*=wildcard]

If a filtered set of domains is required that cannot be represented through the use of wildcards, then the operation will need to be repeated using different domain filters.

3.2 SERVICE: PLANNING REQUEST

3.2.1 SUMMARY

The Planning Request Service, introduced in 2.5.2, is provided by a planning function and enables its consumers to manage the submission of planning requests and to receive feedback

on their status. It comprises the following operations, of which only those in capability set 1 are mandatory.

In the context of a hierarchical or federated planning system, the Planning Request Service submitRequest operation can be used to submit a Plan (4.2.6.1) to a planning function, either embedding the Plan in the request itself or passing it by reference. If passed by reference, the Plan can be retrieved using the Plan Distribution Service (3.3). Patch plans are not permitted in the context of a planning request.

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
MPS	PlanningRequest	5	1	1
Interaction Pattern	**Operation Identifier**		**Operation Number**	**Capability Set**
REQUEST	submitRequest		1	1
REQUEST	getRequestSummaries		2	
PROGRESS	getRequestStatus		3	
SUBMIT	cancelRequest		4	2
REQUEST	updateRequest		5	3
PUBSUB	monitorRequestStatus		6	4
PROGRESS	getRequest		7	5

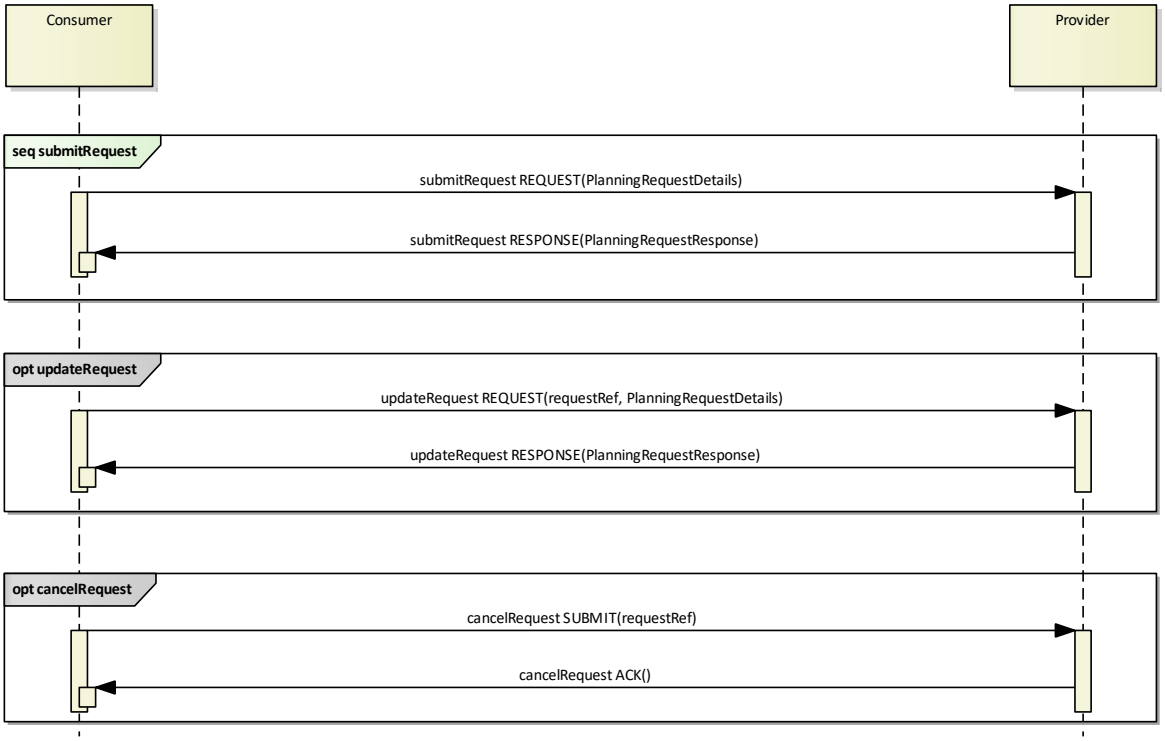

Figure 3-1: Planning Request Submission Operations

Three of the operations are concerned with the submission and subsequent management of planning requests, enabling a service consumer to submit a planning request, and (where supported) to update or cancel that request. The interaction sequence for these operations is illustrated in figure 3-1 above.

The submitRequest operation results in the creation of a new PlanningRequestInstance, returning its identity in the response message. The updateRequest operation results in a new version of an existing PlanningRequestInstance, returning its identity (key and version) in the response message. The cancelRequest operation stops the referenced PlanningRequestInstance being considered in the generation of future Plans, but does not imply that it is immediately deleted by the provider.

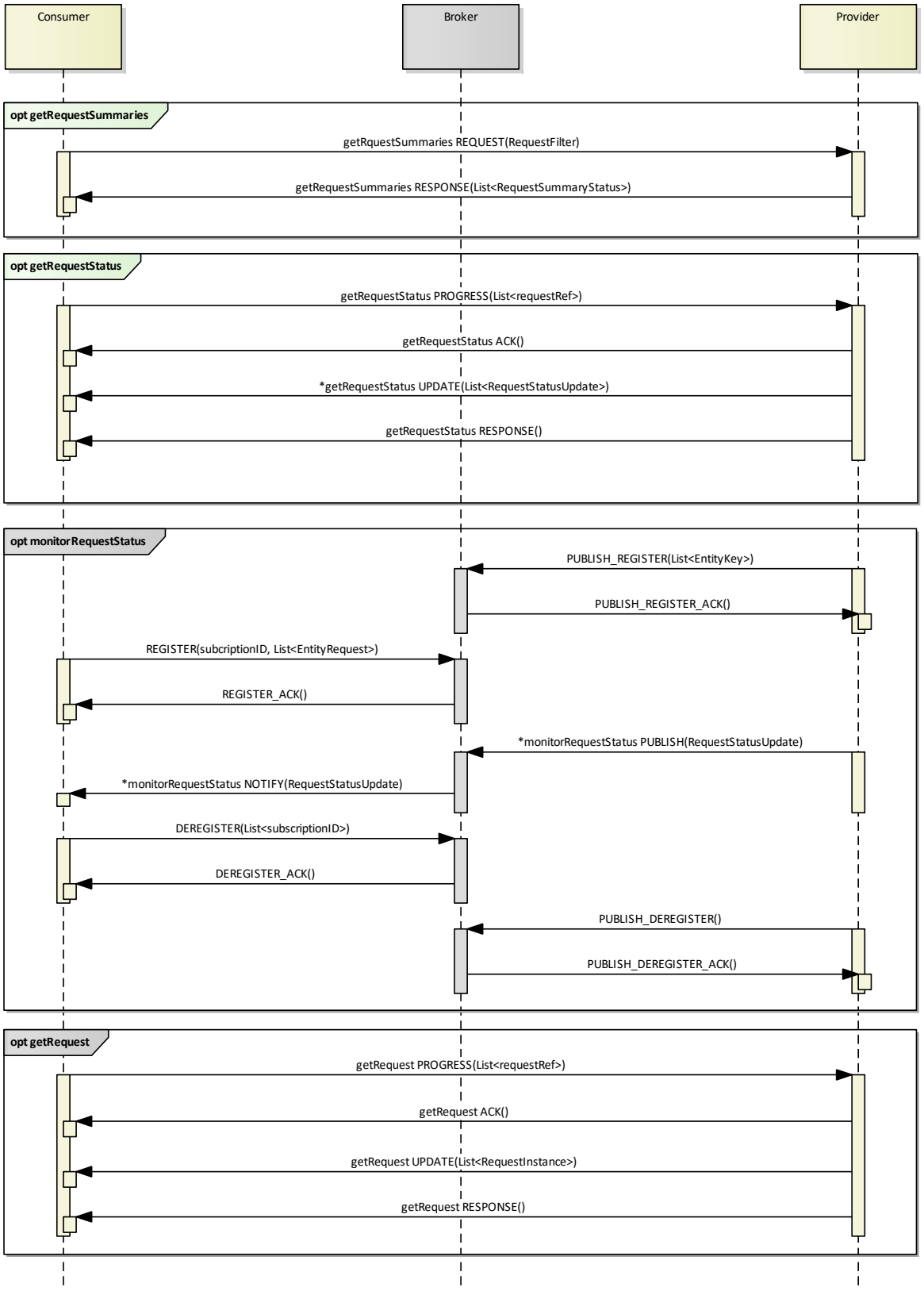

Figure 3-2: Planning Request Feedback Operations

The remaining operations are concerned with obtaining feedback on the status or content of planning requests. The interaction sequence for these operations is illustrated in figure 3-2 above.

The getRequestSummaries operation requests a list of available planning requests based on a supplied filter. It returns a list of RequestSummaryStatuses that contain the identity, header information and status for each available RequestInstance that matches the filter.

A consumer may already holds the identity (and static content) of RequestInstances, either because it has submitted planning requests, or previously performed a getRequestSummaries operation. It can then request an update of their dynamic status using the getRequestStatus operation. The request lists references to RequestInstances and the response provides a corresponding list of RequestStatusUpdates.

Where supported, consumers may also subscribe to receive RequestStatusUpdates as they become available via the monitorRequestStatus. This uses the standard MAL publish-subscribe pattern and notifies the consumer with RequestStatusUpdates corresponding to the specified subscription keys as they are published by the provider.

The getRequest operation, where supported, enables a consumer to retrieve the full content of one or more planning requests. This is similar to the getRequestStatus operation, but instead of returning RequestStatusSummaries it returns RequestInstances. It should be noted that this is not an archive service, only planning requests currently being managed by the provider can be returned.

3.2.2 MPS DATA ITEMS

MPS data items relevant to the planning request service and their relationships are defined within the MPS information model in 4.2.

The following MO objects are directly applicable to the service:

- RequestDefinition;
- RequestInstance.

RequestDefinitions are configuration data that provide re-usable templates for planning requests, but are not required for the submission of an ad-hoc planning request. Their identity comprises both key and version.

RequestInstances are created in response to the submission of a planning request and contain both static data and dynamic status information. As planning requests can be updated, their identity comprises both key and version assigned by the service provider. The planning request submitted by the service consumer contains only the static data needed to specify the planning request in the form of a PlanningRequestDetails structure. If the planning request is derived from a RequestDefinition, this is referenced in the PlanningRequestDetails and the RequestInstance created from it.

As the RequestInstance is unknown at the time of submitting a planning request, the consumer may provide a user reference. This is returned by the provider together with the identity of the RequestInstance in the response to the planning request submission.

Updates to the dynamic status information associated with a RequestInstance are reported by the service provider to the service consumer using the RequestStatusUpdate structure. The RequestSummaryStatus structure is used to provide the identity, descriptive header fields, and current status of available RequestInstances.

The following MO objects may be referenced in the context of the planning request service:

- PlanningUser: the user responsible for submitting the planning request;
- ActivityDefinition: the activities specified to be planned in the form of ActivityDetails structures in the body of the planning request;
- EventDefinition and EventInstance: references to planning events can be contained within ActivityDetails and Constraints contained in the body of the planning request;
- Resource [Optional]: references to planning resources can be contained within the Constraints contained in the body of the planning request;
- Plan: a Plan or reference to a Plan can be used as the body of a planning request. Once the planning request has been planned, a reference to the resulting output Plan may also be included in the RequestInstance.

3.2.3 HIGH-LEVEL REQUIREMENTS

Req_3.2.3.H.1 The following set of mission planning configuration data shall be available to both provider and consumers in any deployment of the planning request service:

a) Planning Request Definitions (as RequestDefinition objects [4.2.5.1]);

b) Planning Activity Definitions (as ActivityDefinition objects [4.2.2.1]);

c) Planning Event Definitions (as EventDefinition objects [4.2.3.1]);

d) Planning Resource Definitions (as Resource objects [4.2.4.2]) [Optional];

e) MPS System Configuration Data (as an MPSSysConfig object [4.2.7]).

3.2.4 FUNCTIONAL REQUIREMENTS

Req_3.2.4.F.2 In response to a submitRequest operation, the service provider shall create a corresponding RequestInstance object, set its creationDate and return its identity (key and version) to the consumer.

Req_3.2.4.F.3 In response to an updateRequest operation, if supported, the service provider shall create a new version of the referenced RequestInstance object, set its creationDate and return its identity (key and version) to the consumer.

Req_3.2.4.F.4 The service provider shall only consider the latest version of a RequestInstance in the generation of future Plans, following a successful updateRequest operation.

Req_3.2.4.F.5 In response to a cancelRequest operation, if supported, the service provider shall set the requestStatus of the referenced RequestInstance object to 'CANCELLED' and stops it being considered in the generation of future Plans.

Req_3.2.4.F.6 In response to internal planning and feedback from external plan execution processes, the service provider shall model the status of RequestInstance objects in accordance with the planning request state model (4.2.5.2).

NOTE – Following a successful updateRequest operation, it is implementation dependent what the status of the previous version of the RequestInstance is set to. Previous versions may already have been incorporated into Plans.

Req_3.2.4.F.7 If a service consumer generates a planning request based on an existing planning request definition, then the PlanningRequestDetails submitted shall:

a) Reference the source RequestDefinition (key and version) in the definition field;

b) Contain a matching set of arguments (name and type) to the RequestDefinition in the arguments field;

c) Have a matching value to the RequestDefinition in the standingOrder field.

NOTE – While typically the PlanningRequestDetails will also have matching content to the RequestDefinition in the activities and constraints fields it is allowed to add and remove individual activities and constraints.

Req_3.2.4.F.8 In the context of a planning request that contains or references an existing Plan, the Plan shall be a full plan, patch plans are not permitted.

Req_3.2.4.F.9 Following successful inclusion of a planning request in a generated Plan, the service provider shall update the outputPlanRef attribute of the RequestInstance with the identity of the Plan (key and version).

NOTE – The consumer may then use the reference to the output Plan to retrieve the Plan or its status using the PlanDistributionService. Where the planning request has been incorporated into multiple alternate plans, these will be listed in the outputPlanRef attribute.

3.2.5 OPERATION: SUBMITREQUEST

3.2.5.1 Overview

The submitRequest operation sends a planning request to the provider, which then creates a corresponding RequestInstance object and returns its identity to the consumer.

Operation Identifier	submitRequest		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	requestDetails : (PlanningRequestDetails)
OUT	RESPONSE	No	requestResponse : (PlanningRequestResponse)

3.2.5.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UNSUPPORTED	MPS

3.2.6 OPERATION: GETREQUESTSUMMARIES

3.2.6.1 Overview

The getRequestSummaries operation allows consumers to obtain a filtered list of currently available RequestInstances. The request uses the RequestFilter structure to select the set of planning requests of interest, using the following keys:

- Domain of the RequestInstance;
- Instance ID (key and version) of the RequestInstance;
- Creation date-time of the RequestInstance version (as a time range);
- Definition ID (key and version) of the RequestInstance (the RequestDefinition from which it was created);
- User ID of the PlanningUser who imitated the RequestInstance;
- User Reference supplied by the User when submitting the RequestInstance.

The response returns a list of RequestSummaryStatus structures containing the identity (key and version), descriptive header fields, and status of the RequestInstances that match the filter.

Operation Identifier	getRequestSummaries		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	requestFilter : (RequestFilter)
OUT	RESPONSE	No	requestSummaries : (List <RequestSummaryStatus>)

3.2.6.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.2.7 OPERATION: GETREQUESTSTATUS

3.2.7.1 Overview

The getRequestStatus operation is used to obtain the current status of one or more known RequestInstances. The operation uses the Progress interaction pattern, to allow the response to be spread across multiple messages.

Operation Identifier	getRequestStatus		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	No	requestRefs : (List <MAL::ObjectRef<RequestInstance>>)
OUT	ACK	No	Empty
OUT	UPDATE	No	requestStatuses : (List <RequestStatusUpdate>)
OUT	RESPONSE	No	Empty

3.2.7.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.2.8 OPERATION: CANCELREQUEST

3.2.8.1 Overview

The cancelRequest operation is used by a consumer to cancel a previously submitted planning request. The service provider acknowledges the cancellation of the RequestInstance or returns an error.

Operation Identifier	cancelRequest		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No	requestRef : (MAL::ObjectRef<RequestInstance>)

3.2.8.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
CANCEL_FAILED	MPS

If it was not possible to cancel the RequestInstance, for example because the resultant activities have already been executed or activated within a plan execution function, then the CANCEL_FAILED error is returned.

3.2.9 OPERATION: UPDATEREQUEST

3.2.9.1 Overview

The updateRequest operation may be used to modify the PlanningRequestDetails associated with a previously submitted planning request. This results in the creation of a new version of the RequestInstance (with the same key) by the service provider, which returns the identity (key and version) of the new version to the consumer.

Operation Identifier	updateRequest		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No No	requestRef : (MAL::ObjectRef<RequestInstance>) requestDetails : (PlanningRequestDetails)
OUT	RESPONSE	No	requestResponse : (PlanningRequestResponse)

3.2.9.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UNSUPPORTED	MPS
UPDATE_FAILED	MPS

If it was not possible to update the RequestInstance, for example because the resultant activities have already been executed or activated within a plan execution function, then the UPDATE_FAILED error is returned.

3.2.10 OPERATION: MONITORREQUESTSTATUS

3.2.10.1 Overview

The monitorRequestStatus operation is used to subscribe to status updates for a filtered set of planning RequestInstances. The operation uses the Publish-Subscribe interaction pattern, with the body of the notification message comprising a RequestStatusUpdate for a subscribed RequestInstance.

Operation Identifier	monitorRequestStatus		
Interaction Pattern	PUBLISH-SUBSCRIBE		
Subscription Keys	instanceID : (MAL::Identifier) definitionID : (MAL::Identifier) userID : (MAL::Identifier) userReference : (MAL::Identifier) status : (MAL::UInteger) outputPlanID : (MAL::Identifier)		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
OUT	PUBLISH/NOTIFY	No	requestStatusUpdate : (RequestStatusUpdate)

Other messages of the Publish-Subscribe pattern are fully defined by the MAL.

3.2.10.2 Structures

Req_3.2.10.S.10 The monitorRequestStatus subscription shall be based on the provision of the following keys in addition to the *domain* of the required RequestInstances in the Register message, all of which are nullable:

a) InstanceID: instance **key** as MAL::Identifier;

b) DefinitionID: definition **key** as MAL::Identifier;

c) UserID: user **key** as MAL::Identifier;

d) UserReference: userReference of subscribed RequestInstances as MAL::Identifier;

e) Status: status as MAL::UInteger [RequestStatusEnum];

f) OutputPlanID: outputPlanRef key as MAL::Identifier.

NOTE – For Status, the enumerated value associated with the RequestStatusEnum must be used, and not the associated string.

3.2.10.3 Errors

In addition to standard MAL Errors, the operation may return the following MPS Errors:

Error	Area
INVALID	MPS

3.2.11 OPERATION: GETREQUEST

3.2.11.1 Overview

The getRequest operation is used to obtain the full content of one or more known RequestInstances. The operation uses the Progress interaction pattern, to allow the response to be spread across multiple messages.

Operation Identifier	getRequest		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	No	requestRefs : (List <MAL::ObjectRef<RequestInstance>>)
OUT	ACK	No	Empty
OUT	UPDATE	No	requestInstances : (List <RequestInstance>)
OUT	RESPONSE	No	Empty

3.2.11.2 Errors

In addition to standard MAL Errors, the operation may return the following MPS Errors:

Error	Area
INVALID	MPS

3.3 SERVICE: PLAN DISTRIBUTION SERVICE

3.3.1 SUMMARY

The Plan Distribution Service, introduced in 2.5.3, is provided by a planning function and enables its consumers to access generated plans and to receive updates on their status. It comprises the following operations, of which only those in capability set 1 are mandatory.

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
MPS	PlanDistribution	5	2	1
Interaction Pattern	**Operation Identifier**		**Operation Number**	**Capability Set**
REQUEST	getPlanSummaries		1	1
PROGRESS	getPlan		2	
REQUEST	getPlanStatus		3	
PUBSUB	monitorPlanStatus		4	2
PUBSUB	monitorPlan		5	3
PROGRESS	queryPlan		6	4
REQUEST	getPartialPlan		7	5

NOTE – The getPlanStatus, with the exception of the service number, is identical to the operation of the same name in the Plan Execution Control Service.

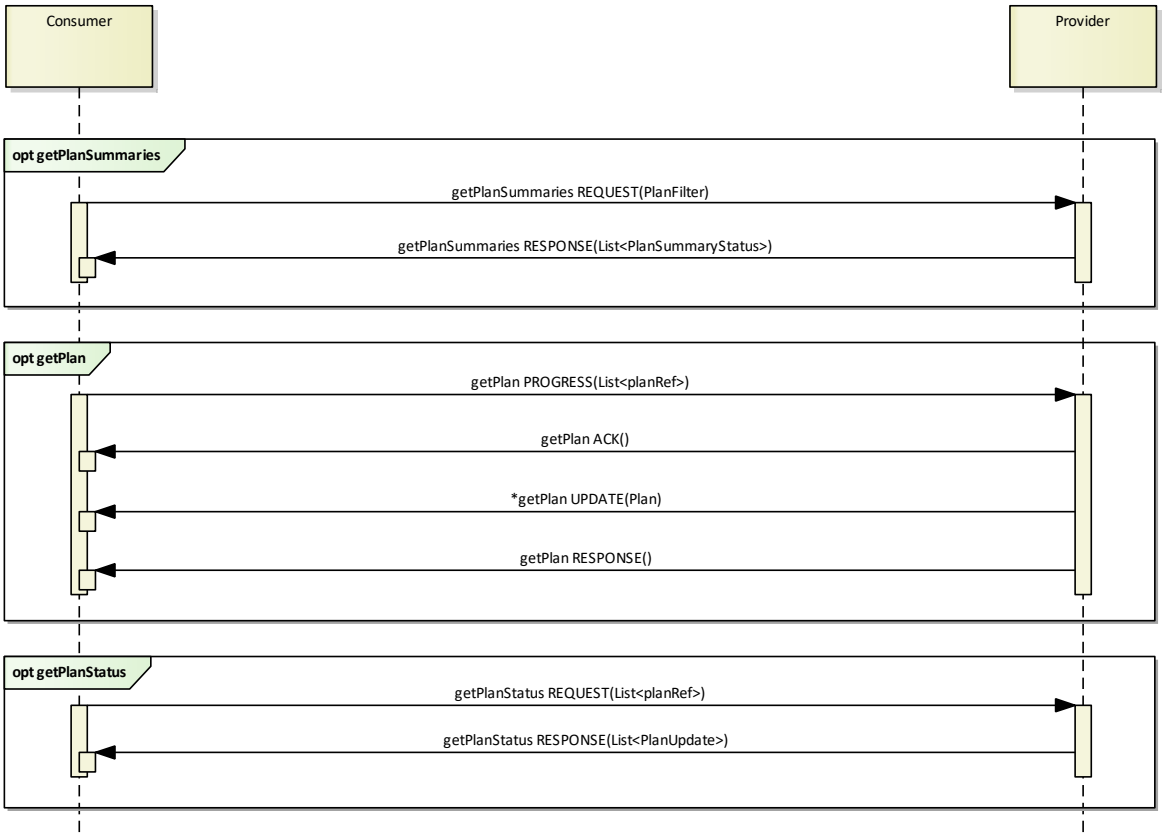

Figure 3-3: Plan Distribution Mandatory Operations

The mandatory operations of capability set 1 allow a service consumer to access plans, or the status of plans, generated by a planning function.

To retrieve a Plan, the consumer needs to have the identity of that Plan (key and optionally version). A reference to the output Plan associated with a planning request may have been received through planning request service feedback. Otherwise, the consumer must use the getPlanSummaries operation to discover available plans.

The getPlanSummaries operation returns PlanSummaries for the set of currently available plans that meet the specified filter criteria. These give the identity of the plans, the associated plan information (context and descriptive fields) and the current status of the Plan, but do not include the full content of the plans.

The getPlan operation returns the full content of the referenced Plan(s).

The getPlanStatus operation returns only the current status of the referenced Plan (s) as PlanUpdates.

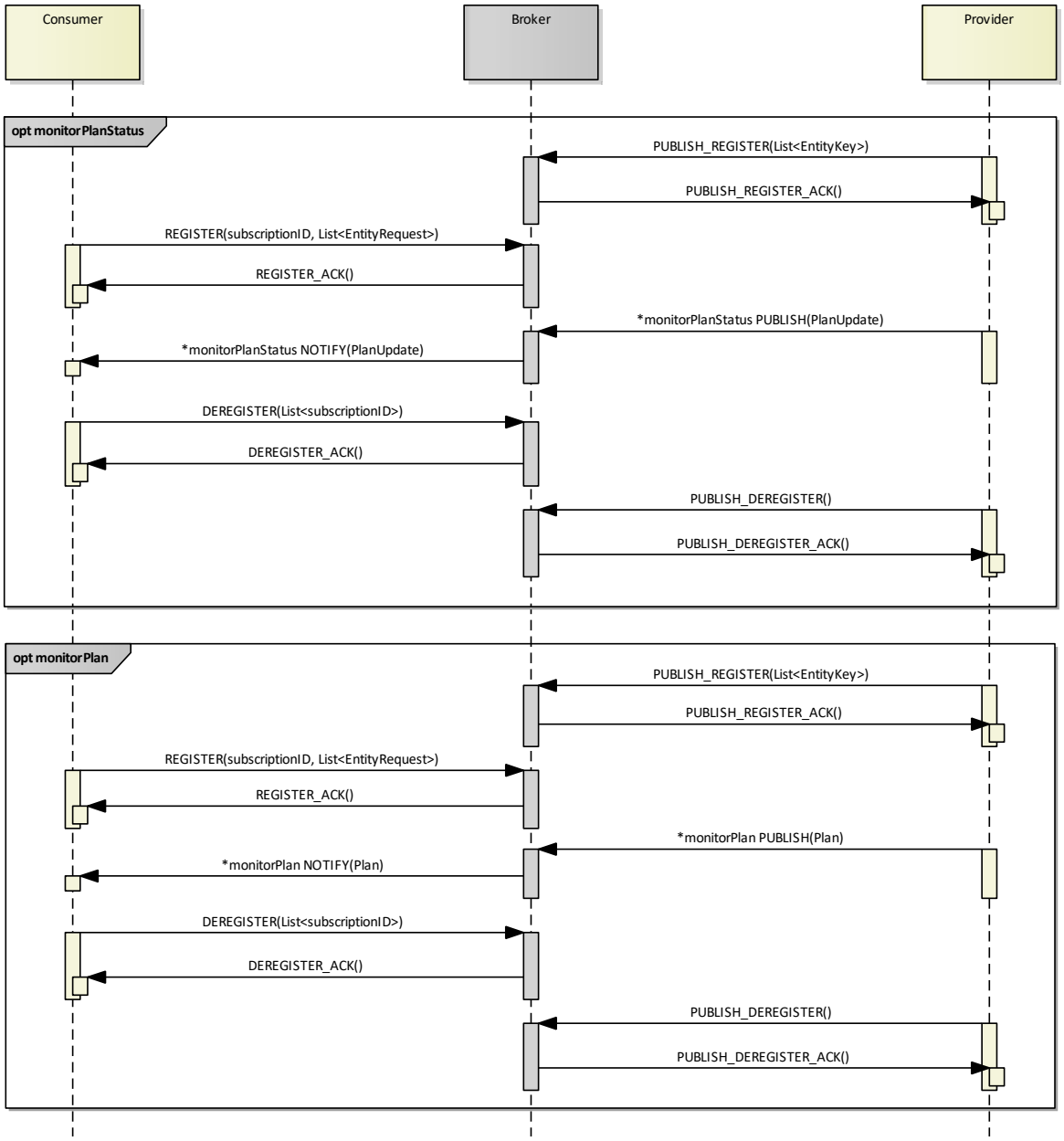

Figure 3-4: Plan Distribution Monitoring Operations

Where supported, the monitorPlanStatus operation enables the consumer to subscribe to automatic notification of changes in status of plans as PlanUpdates. Similarly the monitorPlan operation forwards new Plans matching the subscription to the consumer.

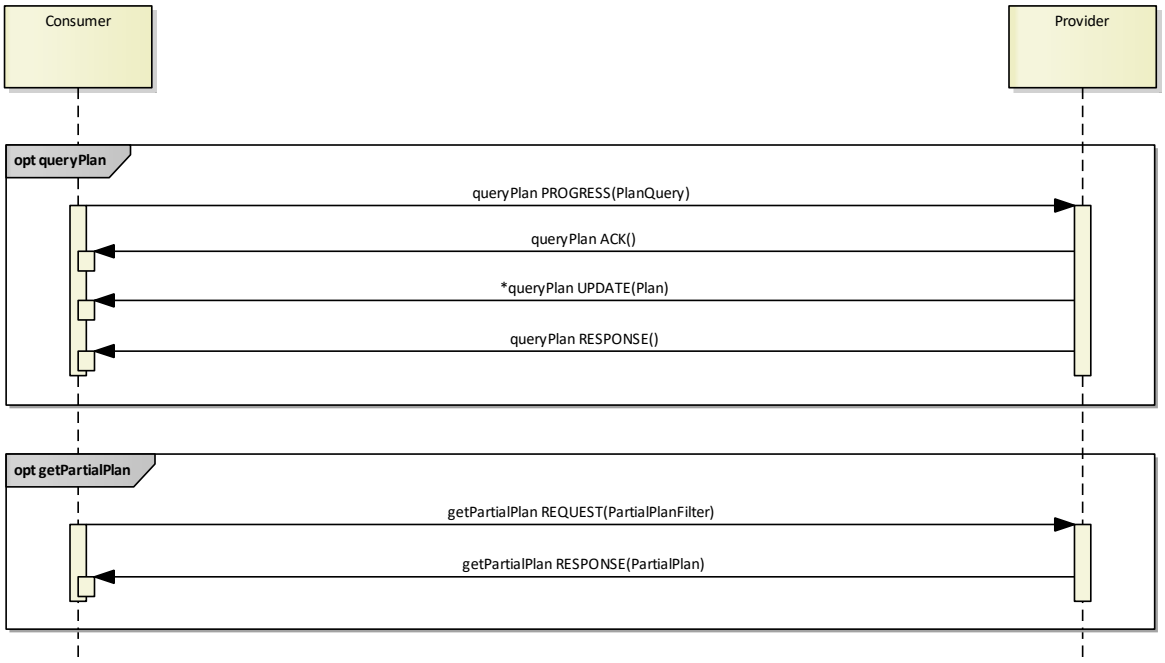

Figure 3-5: Plan Distribution Special Operations

Two further non-mandatory operations allow more complex selection of plans.

The queryPlan operation allows the consumer to request plans based on multiple query criteria, including various header attributes and on whether the Plan includes specific planning activities or events.

The getPartialPlan operation allows the consumer to request a subset of a Plan, based on the domain, subPlan allocation, or tags associated with the contained planning activities, as well as a period of the plan based on time, position, or events.

3.3.2 MPS DATA ITEMS

MPS data items relevant to the plan distribution service and their relationships are defined within the MPS information model in 4.2.

The following MO objects are directly applicable to the service:

– Plan.

The identity of Plans comprises both key and version. Plans are created by a planning function as an output that may be passed to a plan execution function, or to another planning function in a distributed or hierarchical planning system. Where a Plan is updated due to replanning, a new version of the Plan with the same key may be created.

Plans may be stand-alone or reference a defined precursor with which they may overlap. In the latter case, a patch plan may be created which only contains the differences from the precursor plan, and includes a reference to the target Plan which results from applying these changes to the precursor Plan.

A selected subset of a Plan can be extracted as a PartialPlan. This is specified as a shorter period covered by the plan, or by the domain, SubPlan, or tags associated with its contained planning activities.

The following MO objects may be referenced in the context of the planning request service:

- ActivityDefinition and ActivityInstance: the ActivityInstances contained within a plan may be referenced in service operations by their associated ActivityDefinition;
- EventDefinition and EventInstance: the EventInstances contained within a plan may be referenced in service operations by their associated EventDefinition;
- Resources may be contained within a Plan;
- RequestInstance: the ActivityInstances contained within a Plan reference their source planning request.

3.3.3 HIGH-LEVEL REQUIREMENTS

Req_3.3.3.H.11 The following set of mission planning configuration data shall be available to both provider and consumers in a any deployment of the plan distribution service:

a) Planning Activity Definitions (as ActivityDefinition objects [4.2.2.1]);

b) Planning Event Definitions (as EventDefinition objects [4.2.3.1]);

c) Planning Resource Definitions (as Resource objects [4.2.4.2]) [Optional];

d) MPS System Configuration Data (as an MPSSysConfig object [4.2.7]).

3.3.4 FUNCTIONAL REQUIREMENTS

Req_3.3.4.F.12 If no Plan version is specified in the getPlan and getPlanStatus operations, then the service provider shall assume the latest version of the Plan is required.

Req_3.3.4.F.13 In response to internal planning and feedback from external plan execution processes, the service provider shall model the status of Plan objects in accordance with the plan state model (4.2.6.2).

NOTE – A planning function that is a service provider is not required to support real-time provision of Plan status changes. Feedback from plan execution functions may only be processed periodically by the planning function as part of a planning cycle.

3.3.5 OPERATION: GETPLANSUMMARIES

3.3.5.1 Overview

The getPlanSummaries operation allows consumers to obtain a filtered list of currently available Plans. The request uses the PlanFilter structure to select the set of plans of interest, using the following keys:

- Domain of the Plan;
- planID (key and version) of the Plan;
- precursorPlan of the Plan;
- status of the Plan;
- originator of the Plan;
- validity period of the Plan as a time window.

The response returns a list of PlanSummaryStatus structures containing the identity (key and version), descriptive header fields, and status of the Plans that match the filter.

Operation Identifier	getPlanSummaries		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	planFilter : (PlanFilter)
OUT	RESPONSE	No	planSummaries : (List <PlanSummaryStatus>)

3.3.5.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.3.6 OPERATION: GETPLAN

3.3.6.1 Overview

The getPlan operation is used to obtain the full content of one or more known Plans. The operation uses the Progress interaction pattern, to allow the response to be spread across multiple messages.

Operation Identifier	getPlan		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	No	planRefs : (List <MAL::ObjectRef<Plan>>)
OUT	ACK	No	Empty
OUT	UPDATE	No	retrievedPlan : (Plan)
OUT	RESPONSE	No	Empty

3.3.6.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.3.7 OPERATION: GETPLANSTATUS

3.3.7.1 Overview

The getPlanStatus operation is used to obtain the current status of one or more known Plans. The operation uses the Request interaction pattern.

Operation Identifier	getPlanStatus		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	planRefs : (List <MAL::ObjectRef<Plan>>)
OUT	RESPONSE	No	responsePlans : (List <PlanUpdate>)

3.3.7.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.3.8 OPERATION: MONITORPLANSTATUS

3.3.8.1 Overview

The monitorPlanStatus operation is used to subscribe to status updates for a filtered set of Plans. The operation uses the Publish-Subscribe interaction pattern, with the body of the notification message comprising a PlanUpdate for a subscribed Plan.

Operation Identifier	monitorPlanStatus		
Interaction Pattern	PUBLISH-SUBSCRIBE		
Subscription Keys	planID : (MAL::Identifier) precursor : (MAL::Identifier) status : (MAL::UInteger) originator : (MAL::Identifier)		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
OUT	PUBLISH/NOTIFY	No	planUpdate : (PlanUpdate)

Other messages of the Publish-Subscribe pattern are fully defined by the MAL.

3.3.8.2 Structures

Req_3.3.8.S.14 The monitorPlanStatus subscription shall be based on the provision of the following keys in addition to the *domain* of the required Plans in the Register message, all of which are nullable:

a) PlanID: **key** of subscribed Plan as MAL::Identifier;

b) Precursor: **key** of precursorPlan as MAL::Identifier;

c) Status: status as MAL::UInteger [PlanStatusEnum];

d) Originator: originator of subscribed Plan as MAL::Identifier.

NOTE – For Status, the enumerated value associated with the PlanStatusEnum must be used, and not the associated string.

3.3.8.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.3.9 OPERATION: MONITORPLAN

3.3.9.1 Overview

The monitorPlan operation is used by a consumer to subscribe to receive new Plans, or new versions of Plans, as they published. The operation uses the Publish-Subscribe interaction pattern, with the body of the notification message comprising a Plan.

Operation Identifier	monitorPlan		
Interaction Pattern	PUBLISH-SUBSCRIBE		
Subscription Keys	planID : (MAL::Identifier) precursor : (MAL::Identifier) status : (MAL::UInteger) originator : (MAL::Identifier)		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
OUT	PUBLISH/NOTIFY	No	plan : (Plan)

3.3.9.2 Structures

Req_3.3.9.S.15 The monitorPlan subscription shall be based on the provision of the following keys in addition to the **domain** of the required Plans in the Register message, all of which are nullable:

a) PlanID: **key** of subscribed Plan as MAL::Identifier;

b) Precursor: **key** of precursorPlan as MAL::Identifier;

c) Status: status as MAL::UInteger [PlanStatusEnum];

d) Originator: originator of subscribed Plan as MAL::Identifier.

NOTE – For Status, the enumerated value associated with the PlanStatusEnum must be used, and not the associated string.

3.3.9.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.3.10 OPERATION: QUERYPLAN

3.3.10.1 Overview

The queryPlan operation enables a consumer to retrieve a filtered set of plans, based on an extended set of filter criteria, including relevant fields of the plan information sections of the plan, as well as the type of planning activities and planning events contained within the plan.

Operation Identifier	queryPlan		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	No	query : (PlanQuery)
OUT	ACK	No	Empty
OUT	UPDATE	No	queriedPlan : (Plan)
OUT	RESPONSE	No	Empty

3.3.10.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.3.11 OPERATION: GETPARTIALPLAN

3.3.11.1 Overview

The getPartialPlan operation enables a consumer to extract a subset of a Plan that meets the supplied partialPlanFilter. The filter can select the partial plan content based on:

a) a shorter period than that covered by the plan, specified by time, position, or events;

b) a subset of contained ActivityInstances, based on their domain, associated SubPlan or tags.

The PartialPlan returned includes the filter criteria and a version of the plan containing only the ActivityInstances that match those criteria. It is implementation dependent what is returned in terms of events and resources, but it may be assumed that any related events and resources would be included in the returned partial plan.

Operation Identifier	getPartialPlan		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**

Pattern Sequence	Message	Nullable	Body Signature
IN	REQUEST	No	partialPlanFilter : (PartialPlanFilter)
OUT	RESPONSE	No	partialPlan : (PartialPlan)

3.3.11.2 Structures

3.3.11.2.1 Req_3.3.11.S.16 The PartialPlan returned shall include all ActivityInstances contained in the source Plan that pass the specified PartialPlanFilter criteria.

3.3.11.2.2 Req_3.3.11.S.17 The PartialPlan returned should include all EventInstances and ResourceProfiles in the source Plan that lie within the specified period for the partial plan and which are related to the included ActivityInstances.

3.3.11.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4 SERVICE: PLAN EXECUTION CONTROL SERVICE

3.4.1 SUMMARY

3.4.1.1 General

The Plan Execution Control Service, introduced in 2.5.4, is provided by a plan execution function and enables its consumers to submit (and revoke) Plans for execution; to control their execution at plan, sub-plan, and activity levels; and to receive feedback on their execution status.

It comprises the following operations, of which only those in capability set 1 are mandatory.

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
MPS	PlanExecutionControl	5	3	1
Interaction Pattern	**Operation Identifier**		**Operation Number**	**Capability Set**
SUBMIT	submitPlan		1	1
SUBMIT	revokePlan		2	
REQUEST	getPlanStatus		3	
REQUEST	activatePlan		4	2

REQUEST	deactivatePlan	5	
PUBSUB	monitorPlanExecution	6	3
PUBSUB	monitorPlanExecutionDetail	7	4
REQUEST	activateSubPlan	8	5
REQUEST	deactivateSubPlan	9	
REQUEST	getSubPlanStatus	10	
PUBSUB	monitorSubPlanExecution	11	6
REQUEST	suspendActivity	12	7
REQUEST	resumeActivity	13	
REQUEST	getActivityStatus	14	8

NOTE – The getPlanStatus, with the exception of the service number, is identical to the operation of the same name in the Plan Distribution Service.

3.4.1.2 Plan Level Execution Control Operations

Four of the service operations are concerned with submission and subsequent management of plans, submitPlan, and revokePlan being mandatory operations in capability set 1.

The submitPlan operation delivers a Plan to the service provider, making it available for execution. The revokePlan operation makes a previously submitted Plan unavailable for execution. Typically the service provider deletes any local copy of the Plan.

Execution of a Plan is normally a two-stage operation: first the Plan is submitted, then once it has been received by the service provider it can be activated. If the service implementation is restricted to capability set 1, then there is effectively no activation step before a plan is executed: either the plan is activated locally or is immediately activated on submission by the service provider. If an activation step is required, then capability set 2 should also be implemented.

The activatePlan operation enables execution of one or more Plans by the service provider, subject to the scheduling constraints (time, position, or event-based) associated with contained planning activities in each plan.

It is only possible to activate a Plan within its stated validity period and providing the specified start of the Plan is in the future.

The deactivatePlan operation disables the execution of the specified Plans, specifying a deactivation mode for the case where execution has already started.

If a Plan is deactivated before the start of the Plan, then it remains available to the service provider, its status returning to ‘submitted’, and can be re-activated using the activatePlan operation. Any new ActivityInstances and EventInstances are removed from the active plan.

If a Plan is deactivated after the start of the Plan, then its status and that of all contained ActivityInstances and EventInstances that have not completed transition to the 'terminated' state with the status information 'Cancelled'.

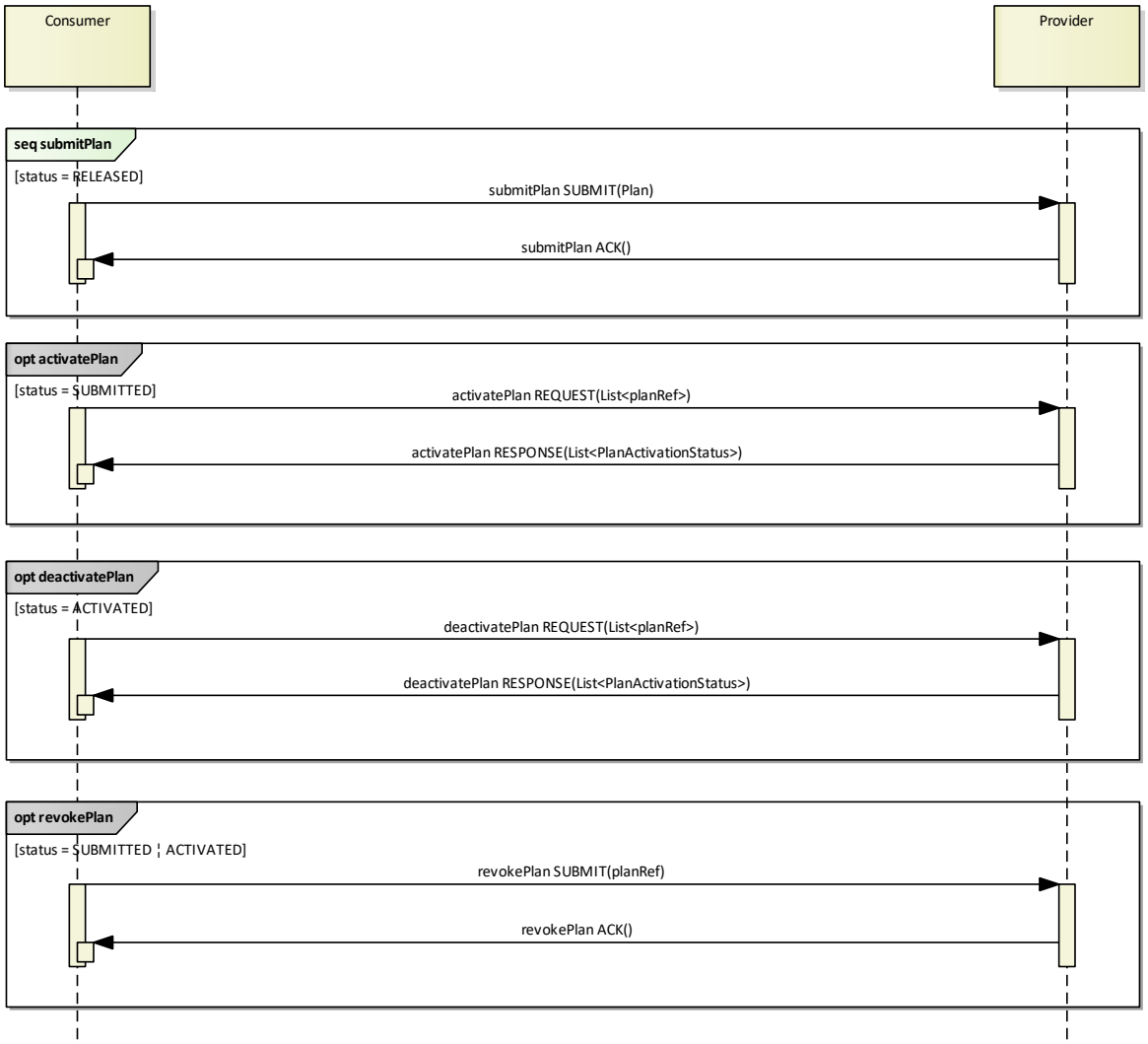

Figure 3-6: Plan Execution Control Operations

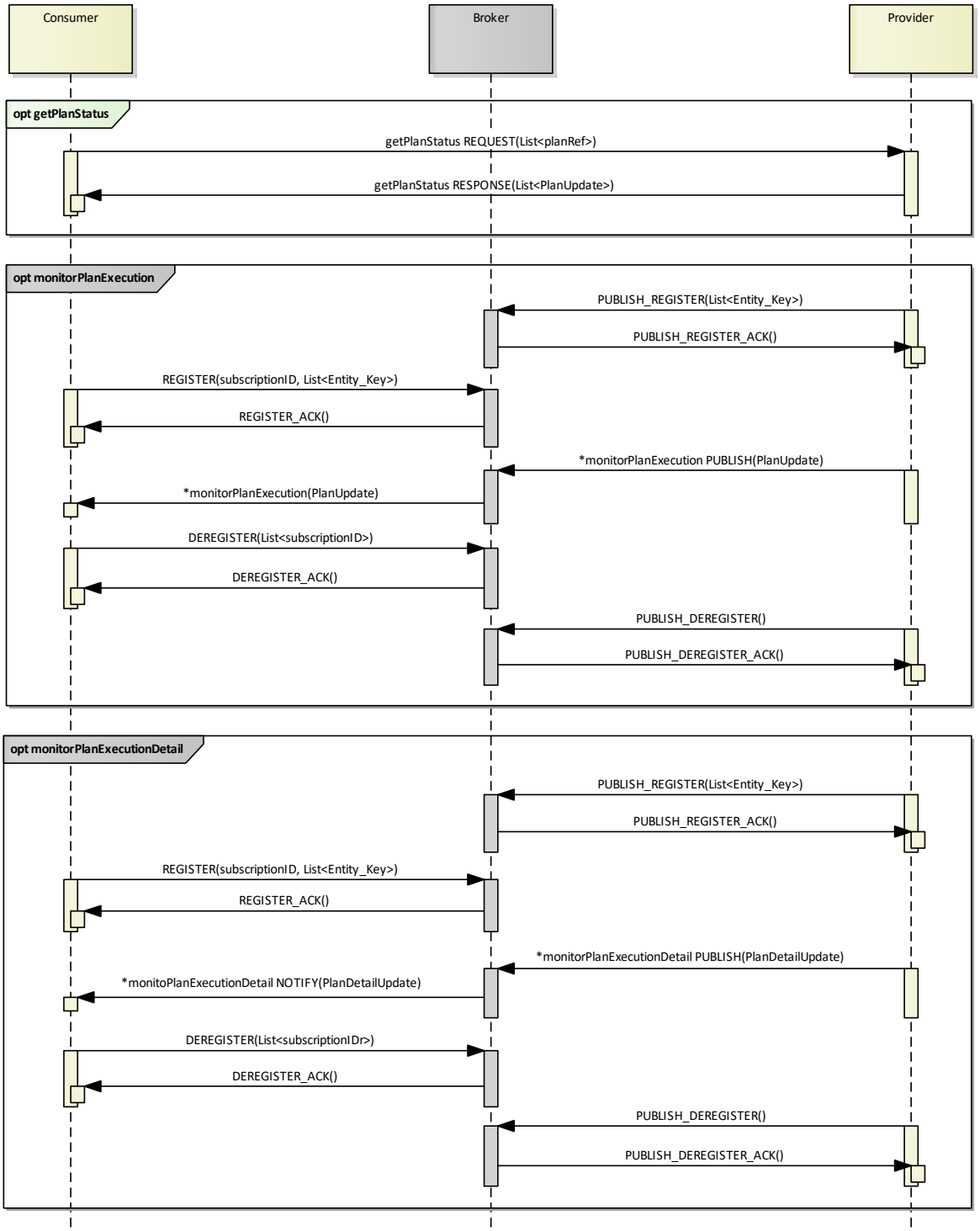

Figure 3-7: Plan Execution Feedback Operations

Three service operations provide feedback on the execution of Plans, of these only the getPlanStatus operation is mandatory in capability set 1. In order to ensure provision of

feedback on the execution of a plan at a sufficiently detailed level to enable feedback for planning requests, it is recommended that capability sets 2 and 3 are implemented.

The getPlanStatus operation returns the current status of the referenced plan(s) as PlanUpdates. It is equivalent to the service operation of the same name in the Plan Distribution Service, but only reports those plan statuses relevant to the plan execution function.

Where supported, the monitorPlanExecution operation enables the consumer to subscribe to automatic notification of changes in status of plans as PlanUpdates. It is equivalent to the monitorPlanStatus operation of the Plan Distribution Service.

The monitorPlanExecutionDetail operation enables the consumer to subscribe to automatic notification of changes in status of the ActivityInstances, EventInstances, and Resources contained within Plans, provided as ActivityUpdates, EventUpdates, and ResourceUpdates respectively. It should be noted that as resource profiles are an optional element of a Plan, ResourceUpdates are only provided where they are supported.

3.4.1.3 SubPlan Level Execution Control Operations

Support for SubPlans is optional, but where supported an additional 4 service operations enable control at the level of SubPlans.

Planning ActivityInstances contained within a Plan may be associated with a single SubPlan ID. This can be used as a mechanism to identify multiple sub-plans within plans, for example to support individual spacecraft within a constellation, or separate payloads. SubPlan IDs are not unique to an individual Plan, but apply globally to a mission planning system. SubPlan operations apply to all activated Plans managed by the service provider. Individual ActivityInstances are only executed if both their container Plan and associated SubPlan are in the ACTIVATED state.

The activateSubPlan operation enables execution of one or more SubPlans by the service provider. It is implementation dependent whether it is initially assumed that all SubPlans are active, in which case the operation would only be needed to re-enable execution following a deactivateSubPlan operation.

The deactivateSubPlan operation disables execution of one or more SubPlans by the service provider.

The getSubPlanStatus operation returns the current status of the referenced SubPlans as SubPlanUpdates.

Where supported, the monitorSubPlanExecution operation enables the consumer to subscribe to automatic notification of changes in status of SubPlans as SubPlanUpdates.

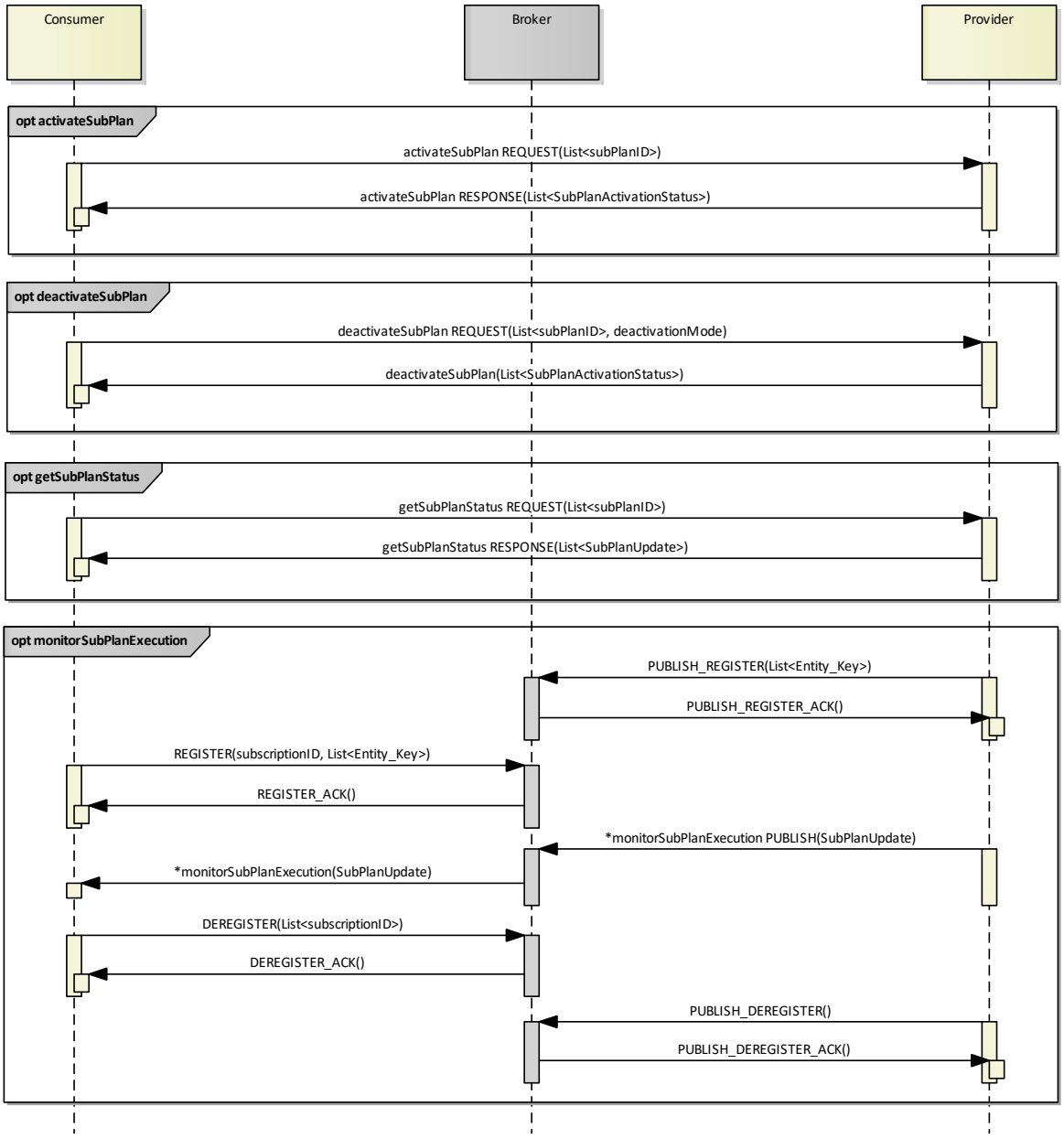

Figure 3-8: SubPlan Execution Control Operations

3.4.1.4 Activity Level Execution Control Operations

Figure 3-9: Activity Execution Control Operations

Optional support for execution control at the level of ActivityInstances is provided by an additional 3 service operations.

Selected ActivityInstances are identified by Plan, instance ID, and/or tag. The use of tags[2] allows arbitrary (and overlapping) groups of ActivityInstances to be managed together.

The suspendActivity operation requests suspension of the execution of selected activities, in one or more plans, without changing the state of those plans. The response gives the resulting ActivitySuspensionStatus of each referenced ActivityInstance.

The resumeActivity operation requests the resumption of previously suspended activities in one or more plans without changing the state of those plans. The response gives the resulting ActivitySuspensionStatus of each referenced ActivityInstance.

The getActivityStatus operation returns the status of ActivityInstances, as ActivityUpdates, at activity, sub-plan, or tag level.

[2] Tags are used in the MPS information model to capture the equivalent concept to subschedules of PUS service 11 in the ECSS Packet Utilization Standard (reference [D8]).

3.4.2 MPS DATA ITEMS

MPS data items relevant to the plan execution control service and their relationships are defined within the MPS information model in 4.2.

The following MO objects are directly applicable to the service:

- Plan;
- ActivityInstance;
- EventInstance;
- Resource.

The identity of Plans comprises both key and version. Plans are submitted to a plan execution function for execution.

Plans contain ActivityInstances and EventInstances, the same instance of which may occur in multiple overlapping Plans or versions of Plans. Plans may also optionally contain ResourceProfiles referencing Resources.

The plan execution control service does not result in the creation of new MO objects.

The following MO objects may be referenced by the ActivityInstances and EventInstances contained within a Plan:

- ActivityDefinition;
- EventDefinition;
- RequestInstance.

3.4.3 HIGH-LEVEL REQUIREMENTS

Req_3.4.3.H.18 The following set of mission planning configuration data shall be available to both provider and consumers in a any deployment of the plan execution control service:

a) Planning Activity Definitions (as ActivityDefinition objects [4.2.2.1]);

b) Planning Event Definitions (as EventDefinition objects [4.2.3.1]);

c) Planning Resource Definitions (as Resource objects [4.2.4.2]) [Optional];

d) MPS System Configuration Data (as an MPSSysConfig object [4.2.7]).

3.4.4 FUNCTIONAL REQUIREMENTS

Req_3.4.4.F.19 In response to a submitPlan operation, the service provider shall make the submitted Plan available for execution.

Req_3.4.4.F.20 If the service does not support plan activation (capability set 2), the service provider shall provide an internal mechanism to activate submitted Plans.

NOTE – This may be as simple as to automatically activate Plans on submission.

Req_3.4.4.F.21 In response to a revokePlan operation, the service provider shall make the submitted Plan unavailable for execution. If this is not possible (for example, if the plan has already executed), then a REVOKE_FAILED error shall be returned.

Req_3.4.4.F.22 In response to internal plan execution processes, the service provider shall model the status of Plans in accordance with the plan state model (4.2.6.2).

Req_3.4.4.F.23 In response to an activatePlan operation, or internal plan activation where not supported, the service provider shall enable the execution of the referenced Plans and the ActivityInstances contained within them, subject to the scheduling constraints specified within the Plans.

Req_3.4.4.F.24 In response to an activatePlan operation, if a referenced Plan overlaps with its already activated precursor Plan, then the service provider should merge plan revisions into the currently active Plan.

NOTE – It is implementation dependent whether overlapping Plans are supported and how this is managed internally by the plan execution function. However, it is common for successive iterations of a Plan to overlap, for example with a Plan being generated daily for the coming week. In this case, the same ActivityInstance may appear in the overlap period of both Plans, and may be unchanged, modified or deleted in the successor Plan. New ActivityInstances may also be inserted in the overlap period or in the extended period covered by the successor Plan. It is anticipated that when the successor Plan is activated, then its predecessor is superseded from the start point of the successor Plan with any revisions applicable to the overlap period being applied. Only the latest revision of the ActivityInstance is executed.

Req_3.4.4.F.25 In response to an activatePlan operation, if a referenced Plan is a target or Patch Plan, then the service provider shall also activate the associated precursor Plan.

NOTES

1 It is implementation dependent whether the target Plan is reconstituted prior to activation, or the precursor is activated and the Patch Plan merged into this.

2 If the precursor Plan is already activated, then the Patch Plan can be merged into the currently active Plan. If the precursor Plan is not available, then the operation cannot be performed.

Req_3.4.4.F.26 It shall not be possible to activate a Plan if it is outside the validity period of the Plan, or if the start of the Plan period has already passed. In this case, the activatePlan operation shall fail.

Req_3.4.4.F.27 In response to a deactivatePlan operation, if supported, the service provider shall disable the execution of the referenced Plans and the ActivityInstances contained within them, subject to the specified deactivationMode.

NOTE – Supported deactivationModes are specific to the implementation of the service provider.

Req_3.4.4.F.28 If no Plan version is specified in the getPlanStatus, monitorPlanExecution, and monitorPlanExecutionDetail operations, then the service provider shall assume the latest version of the Plan is required.

Req_3.4.4.F.29 If the monitorPlanExecutionDetail operation (or activity level operations) are supported, then in response to internal plan execution processes, the service provider shall model the status of ActivityInstances and EventInstances in accordance with their respective state models (4.2.2.2 and 4.2.3.2).

Req_3.4.4.F.30 If the monitorPlanExecutionDetail operation is supported, the service provider may model the evolving value of planning Resources in accordance with the ResourceProfiles (4.2.4.3) defined within Plans and any Effects (4.3.6.3) associated with ActivityInstances.

Req_3.4.4.F.31 If SubPlans are supported, the service provider shall maintain the current status of SubPlans as ACTIVATED or DEACTIVATED.

NOTE – The status of SubPlans is independent to that of Plans and a change in their activation status has no impact on the status of the Plan. It may impact the status of individual ActivityInstances, which can be observed using the monitorPlanExecutionDetail operation.

Req_3.4.4.F.32 In response to an activateSubPlan operation, if supported, the service provider shall activate the referenced SubPlans and enable the execution of ActivityInstances that are contained in activated Plans and allocated to activated SubPlans.

NOTE – It is implementation dependent whether SubPlans are initially ACTIVATED and therefore do not require activation unless previously deactivated.

Req_3.4.4.F.33 In response to a deactivateSubPlan operation, if supported, the service provider shall deactivate the referenced SubPlans and disable the execution of ActivityInstances that are contained in activated Plans and allocated to deactivated SubPlans, subject to the specified deactivationMode.

NOTE – Supported deactivationModes are specific to the implementation of the service provider.

Req_3.4.4.F.34 In response to a suspendActivity operation, if supported, the service provider shall suspend the execution of referenced ActivityInstances, subject to the specified suspensionMode.

NOTE – Supported suspensionModes are specific to the implementation of the service provider.

Req_3.4.4.F.35 In response to a resumeActivity operation, if supported, the service provider shall resume the execution of the referenced ActivityInstances where it is safe to do so.

3.4.5 OPERATION: SUBMITPLAN

3.4.5.1 Overview

The submitPlan operation is used to send a plan to a plan execution function (the service provider), making it available for execution. The service provider acknowledges the reception of the plan or returns an error.

NOTE – The submitted plan may be a full plan or a patch plan.

Operation Identifier	submitPlan		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No	plan : (Plan)

3.4.5.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UNSUPPORTED	MPS

3.4.6 OPERATION: REVOKEPLAN

3.4.6.1 Overview

The revokePlan operation is used to request a plan execution function to revoke a previously submitted Plan, making it unavailable for execution. The service provider acknowledges the revocation of the Plan or returns an error.

Operation Identifier	revokePlan
Interaction Pattern	SUBMIT

Pattern Sequence	Message	Nullable	Body Signature
IN	SUBMIT	No	planRef : (MAL::ObjectRef<Plan>)

3.4.6.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
REVOKE_FAILED	MPS

3.4.7 OPERATION: GETPLANSTATUS

3.4.7.1 Overview

The getPlanStatus operation is used to obtain the current status of one or more known Plans that have been previously submitted to a plan execution function.

Operation Identifier	getPlanStatus		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	planRefs : (List <MAL::ObjectRef<Plan>>)
OUT	RESPONSE	No	planStatus : (List <PlanUpdate>)

3.4.7.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.8 OPERATION: ACTIVATEPLAN

3.4.8.1 Overview

The activatePlan operation is used to request the execution of specified Plans that have previously been submitted to a plan execution function. The service provider enables the execution of the referenced Plans and the ActivityInstances contained within them, subject to the scheduling constraints specified within the Plans. It is not possible to activate a Plan outside its validity period, or after the start of the Plan period.

NOTES

1 Multiple plans with a common precursor may have been submitted to a plan execution function. Usually only one of these is considered the nominal plan, the other alternative or contingency plans having the isAlternate flag set. It is implementation dependent whether the service provider will allow activation of Plans that have the isAlternate flag set, but this may be blocked for operational safety. Where this is the case, the plan edit service can be used to change the state of the isAlternate flag prior to activation (see 3.6.4).

2 In order to activate a patch Plan, the precursor Plan on which it is based must also be activated. It is recommended that the activatePlan operation references the target Plan (the result of merging the patch Plan with its precursor), rather than the patch Plan itself (although this is allowed). It is implementation dependent how it is achieved (merge patch with precursor prior to activation, or activate precursor and then merge patch), but if the precursor Plan is not already activated, then activating a target or patch Plan implies that the precursor is also activated. If the precursor plan has not previously been submitted to the service provider (or has been revoked), then it is not possible to activate the target or patch Plan.

Operation Identifier	activatePlan		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	planRefs : (List <MAL::ObjectRef<Plan>>)
OUT	RESPONSE	No	activationStatus : (List <PlanActivationStatus>)

3.4.8.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
ACTIVATE_FAILED	MPS

3.4.9 OPERATION: DEACTIVATEPLAN

3.4.9.1 Overview

The deactivatePlan operation is used to request deactivation of specified Plans that have previously been activated. The service provider disables the execution of the referenced Plans and the ActivityInstances contained within them, where it is possible to do so.

The deactivationMode argument allows selection of the deactivation behaviour. For example:

- Orderly (ceases execution of any new activities, but allows those already initiated to complete);
- Rapid (ceases execution of the Plan, but allows activities already initiated to continue until their next defined breakpoint);
- Immediate (ceases execution of the Plan and all activities currently in progress).

It should be noted that it is dependent on the service provider implementation which deactivationModes are supported, and that the above list is not exhaustive.

The service provider returns a list of PlanActivationStatus data structures comprising Plan status and activationInfo as a String for each Plan in the deactivation list. The activationInfo allows the return of deployment specific details on the deactivation, such as the deactivation mode applied or reasons for a failure to deactivate.

If a Plan is deactivated prior to any of its constituent ActivityInstances being executed (or before the specified planPeriodStart), then all new ActivityInstances and EventInstances contained in the Plan are unloaded or removed, and the status of the Plan reverts to SUBMITTED.

If a Plan is deactivated after any of its constituent ActivityInstances have been executed (or after the specified planPeriodStart), then the status of the Plan and the status of all contained ActivityInstances and EventInstances that will not be executed are set to TERMINATED with the additional statusInfo ‘CANCELLED’.

Operation Identifier	deactivatePlan		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No No	planRefs : (List <MAL::ObjectRef<Plan>>) deactivationMode : (MAL::String)
OUT	RESPONSE	No	activationStatus : (List <PlanActivationStatus>)

3.4.9.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.10 OPERATION: MONITORPLANEXECUTION

3.4.10.1 Overview

The monitorPlanExecution operation is used to subscribe to status updates for a filtered set of Plans that have been submitted to a plan execution function. The operation uses the Publish-Subscribe interaction pattern, with the body of the notification message comprising a PlanUpdate for a subscribed Plan.

The operation is equivalent to the monitorPlanStatus operation of the Plan Distribution Service, but only reports the status of plans currently being managed by a plan execution function.

Operation Identifier	monitorPlanExecution
Interaction Pattern	PUBLISH-SUBSCRIBE
Subscription Keys	planID : (MAL::Identifier) precursor : (MAL::Identifier) status : (MAL::UInteger) originator : (MAL::Identifier)

Pattern Sequence	Message	Nullable	Body Signature
OUT	PUBLISH/NOTIFY	No	planUpdate : (PlanUpdate)

3.4.10.2 Structures

Req_3.4.10.S.36 The monitorPlanExecution subscription shall be based on the provision of the following keys in addition to the *domain* of the required Plans in the Register message, all of which are nullable:

a) PlanID: **key** of subscribed Plan as MAL::Identifier;

b) Precursor: **key** of precursorPlan as MAL::Identifier;

c) Status: status as MAL::UInteger [PlanStatusEnum];

d) Originator: originator of subscribed Plan as MAL::Identifier.

NOTE – For Status, the enumerated value associated with the PlanStatusEnum must be used, and not the associated string.

3.4.10.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.11 OPERATION: MONITORPLANEXECUTIONDETAIL

3.4.11.1 Overview

The monitorPlanExecutionDetail operation is used to subscribe to updates that report changes in the detailed execution status for a filtered set of Plan contents at the level of planning activities, events and resources. A planning function requires feedback at the level of planning activities and events to be able to reconstitute the status of planning requests, as well to support re-planning. The operation uses the Publish-Subscribe interaction pattern.

It is implementation dependent which details are reported on, but this may be any combination of planning activities, events, and resources. The notification message body comprises a single structure of the abstract class *PlanDetailUpdate*, which corresponds to one of the concrete classes ActivityUpdate, EventUpdate, or ResourceUpdate.

Operation Identifier	monitorPlanExecutionDetail		
Interaction Pattern	PUBLISH-SUBSCRIBE		
Subscription Keys	planID : (MAL::Identifier) subPlan : (MAL::Identifier) tags : (MAL::String) type : (MAL::Identifier)		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
OUT	PUBLISH/NOTIFY	No	detailUpdate : (*PlanDetailUpdate*)

3.4.11.2 Structures

3.4.11.2.1 Req_3.4.11.S.37 The monitorPlanExecutionDetail subscription shall be based on the provision of the following keys in addition to the *domain* of the required Plans in the Register message, all of which are nullable:

a) PlanID: **key** of subscribed Plan as MAL::Identifier;

b) SubPlan: subPlan of subscribed ActivityInstances as MAL::Identifier;

c) Tags: tags of subscribed ActivityInstances as MAL::String;

d) Type: type of the MO object to which the *PlanDetailUpdate* relates as MAL::Identifier. This must be one of ActivityInstance, EventInstance, or Resource.

3.4.11.2.2 Req_3.4.11.S.38 The service provider shall publish a combination of ActivityUpdates, EventUpdates, and ResourceUpdates.

NOTE – It is implementation dependent which types of update are supported by the service provider.

3.4.11.2.3 Req_3.4.11.S.39 If the subscription includes subPlan or tags keys, then the consumer shall only be notified of updates to ActivityInstances associated with the specified

subPlan or keys. If the service provider supports EventUpdates and/or ResourceUpdates then these should be notified irrespective of subscription by subPlan or tag.

3.4.11.2.4 Req_3.4.11.S.40 If the subscription includes Type keys, then the consumer shall only be notified with updates to the specified MO object types (ActivityInstance, EventInstance, or Resource).

3.4.11.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.12 OPERATION: ACTIVATESUBPLAN

3.4.12.1 Overview

The activateSubPlan operation is used to request that the service provider activates the referenced SubPlans and enables the execution of ActivityInstances that are contained in activated Plans and allocated to activated SubPlans.

NOTE

1 It is implementation dependent whether SubPlans are initially ACTIVATED and therefore do not require activation unless previously deactivated.

2 Where the operation is directly supported by the service provider there is little reason for the activation to fail, but if the operation is delegated, for example to an on-board scheduler, there is the potential for the operation to fail.

Operation Identifier	activateSubPlan		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	subPlanIDs : (List <MAL::Identifier>)
OUT	RESPONSE	No	activationStatus : (List <SubPlanActivationStatus>)

3.4.12.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

ACTIVATE_SUBPLAN_FAILED	MPS

3.4.13 OPERATION: DEACTIVATESUBPLAN

3.4.13.1 Overview

The deactivateSubPlan operation is used to request that the service provider deactivates the referenced SubPlans and disables the execution of ActivityInstances that are contained in activated Plans and allocated to the deactivated SubPlans, where it is possible to do so.

The deactivationMode argument allows selection of the deactivation behaviour. For example:

– Orderly (ceases execution of any new activities, but allows those already initiated to complete);

– Rapid (ceases execution of the Sub-plan, but allows activities already initiated to continue until their next defined breakpoint);

– Immediate (ceases execution of the Sub-plan and all activities currently in progress).

It should be noted that it is dependent on the service provider implementation which deactivationModes are supported, and that the above list is not exhaustive.

The service provider returns a list of SubPlanActivationStatus data structures comprising sub-plan status and activationInfo as a String for each sub-plan in the deactivation list. The activationInfo allows the return of deployment specific details on the deactivation, such as the deactivation mode applied or reasons for a failure to deactivate.

NOTE – Where the operation is directly supported by the service provider there is little reason for the deactivation to fail, but if the operation is delegated, for example to an on-board scheduler, there is the potential for the operation to fail.

Operation Identifier	deactivateSubPlan		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No No	subPlanIDs : (List <MAL::Identifier>) deactivationMode : (MAL::String)
OUT	RESPONSE	No	activationStatus : (List <SubPlanActivationStatus>)

3.4.13.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	**Area**

INVALID	MPS
DEACTIVATE_SUBPLAN_FAILED	MPS

3.4.14 OPERATION: GETSUBPLANSTATUS

3.4.14.1 Overview

The getSubPlanStatus operation is used to obtain the current status of one or more SubPlans.

Operation Identifier	getSubPlanStatus		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	subPlanIDs : (List <MAL::Identifier>)
OUT	RESPONSE	No	subPlanStatus : (List <SubPlanUpdate>)

3.4.14.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	**Area**
INVALID	MPS

3.4.15 OPERATION: MONITORSUBPLANEXECUTION

3.4.15.1 Overview

The monitorSubPlanExecution operation is used to subscribe to status updates for a filtered set of SubPlans. The operation uses the Publish-Subscribe interaction pattern, with the body of the notification message comprising a SubPlanUpdate for a subscribed sub-plan.

Operation Identifier	monitorSubPlanExecution		
Interaction Pattern	PUBLISH-SUBSCRIBE		
Subscription Keys	subPlan : (MAL::Identifier)		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
OUT	PUBLISH/NOTIFY	No	subPlanUpdate : (SubPlanUpdate)

3.4.15.2 Structures

Req_3.4.15.S.41 The monitorSubPlanExecution subscription shall be based on the provision of the following keys in addition to the *domain* of the required SubPlans in the Register message, all of which are nullable:

SubPlan: subPlan ID as MAL::Identifier

3.4.15.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.16 OPERATION: SUSPENDACTIVITY

3.4.16.1 Overview

The suspendActivity operation is used to request suspension of the execution of selected activities in one or more plans, without changing the state of the plan(s).

The suspentionMode argument allows selection of the suspension behaviour. For example:

- Orderly (suspends execution of any new activities, but allows those already initiated to complete);
- Rapid (suspends execution of any new activities, but allows any activities and their sub-activities already initiated to continue until their next defined breakpoint);
- Immediate (suspends execution of all activities, including those currently in progress).

It should be noted that it is dependent on the service provider implementation which deactivationModes are supported, and that the above list is not exhaustive.

The service provider responds with a list of ActivitySuspensionStatus data structures comprising activity status and suspensionInfo (as a String) for each activity subject to the suspension request.

The suspensionInfo allows the return of deployment specific details on the suspension, such as the suspension mode applied or reasons for a failure to suspend.

Operation Identifier	suspendActivity		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No Yes Yes No	planRefs : (List <MAL::ObjectRef<Plan>>) activityRefs : (List <MAL::ObjectRef<ActivityInstance>>) tags : (List <MAL::String>) suspensionMode : (MAL::String)
OUT	RESPONSE	No	suspensionStatus : (List <ActivitySuspensionStatus>)

3.4.16.2 Structures

Req_3.4.16.S.42 The ActivityInstances to be suspended shall be specified in the request message of the suspendActivity operation, by the following filter fields, each of which is nullable:

a) planRefs;

b) activityRefs;

c) tags (associated with ActivityInstances to be suspended).

NOTE – As for other filters, the planRefs, activityRefs, and tags filters are ANDed together, but multiple items within each filter are ORed. This also applies to the tag filter, where if an ActivityInstance has any one of the listed tag values, it passes the filter. activityRefs and tags filters would not normally be used in conjunction.

3.4.16.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.17 OPERATION: RESUMEACTIVITY

3.4.17.1 Overview

The resumeActivity operation is used to request resumption of the execution of selected activities in one or more plans, without changing the state of the plan(s).

The service provider responds with a list of ActivitySuspensionStatus data structures comprising activity status and suspensionInfo (as a String) for each activity subject to the resumption request.

The suspensionInfo allows the return of deployment specific details on the resumption, such as the reasons for a failure to resume.

Operation Identifier	resumeActivity		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No Yes Yes	planRefs : (List <MAL::ObjectRef<Plan>>) activityRefs : (List <MAL::ObjectRef<ActivityInstance>>)

Pattern Sequence	Message	Nullable	Body Signature
			tags : (List <MAL::String>)
OUT	RESPONSE	No	suspensionStatus : (List <ActivitySuspensionStatus>)

3.4.17.2 Structures

Req_3.4.17.S.43 The ActivityInstances to be resumed shall be specified in the request message of the resumeActivity operation, by the following filter fields, each of which is nullable:

a) planRefs;

b) activityRefs;

c) tags (associated with ActivityInstances to be resumed).

NOTE – As for other filters, the planRefs, activityRefs, and tags filters are ANDed together, but multiple items within each filter are ORed. This also applies to the tag filter, where if an ActivityInstance has any one of the listed tag values, it passes the filter. activityRefs and tags filters would not normally be used in conjunction.

3.4.17.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.4.18 OPERATION: GETACTIVITYSTATUS

3.4.18.1 Overview

The getActivityStatus operation is used to request a detailed report from the service provider on the current status of ActivityInstances, selected at activity, sub-plan, or tag levels.

Operation Identifier	getActivityStatus		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	planRefs : (List <MAL::ObjectRef<Plan>>)
		Yes	activityRefs : (List <MAL::ObjectRef<ActivityInstance>>)
		Yes	subPlans : (List <MAL::Identifier>)

Pattern Sequence	Message	Nullable	Body Signature
		Yes	tags : (List <MAL::String>)
OUT	RESPONSE	No	activityStatus : (List <ActivityUpdate>)

3.4.18.2 Structures

Req_3.4.18.S.44 The ActivityInstances whose status is to be reported shall be specified in the request message of the getActivityStatus operation, by the following filter fields, each of which is nullable:

a) planRefs;

b) activityRefs;

c) subPlans (associated with ActivityInstances to be reported);

d) tags (associated with ActivityInstances to be reported).

NOTES

1 It is implementation dependent what is reported if none of the above are provided (all ActivityInstances or none).

2 As for other filters, the planRefs, activityRefs, subPlans, and tags filters are ANDed together, but multiple items within each filter are ORed. This also applies to the tag filter, where if an ActivityInstance has any one of the listed tag values, it passes the filter. activityRefs would not normally be used in conjunction with subPlans or tags filters.

3.4.18.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5 SERVICE: PLAN INFORMATION MANAGEMENT SERVICE

3.5.1 OVERVIEW

The Plan Information Management Service, introduced in 2.5.5, is provided by a planning function and enables its consumers to list and retrieve available MPS configuration data. This includes definitions for planning requests, planning activities, planning events and planning resources, and also MPS system configuration data. The service may also be

provided by a plan execution function (excluding planning request definitions). It comprises the following operations, none of which are mandatory.

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
MPS	PlanInformationManagement	5	4	1

Interaction Pattern	Operation Identifier	Operation Number	Capability Set
PROGRESS	listRequestDefs	1	1
REQUEST	getRequestDefs	2	
PROGRESS	listEventDefs	3	2
REQUEST	getEventDefs	4	
PROGRESS	listActivityDefs	5	3
REQUEST	getActivityDefs	6	
PROGRESS	listResourceDefs	7	4
REQUEST	getResourceDefs	8	
REQUEST	getSystemConfig	9	5

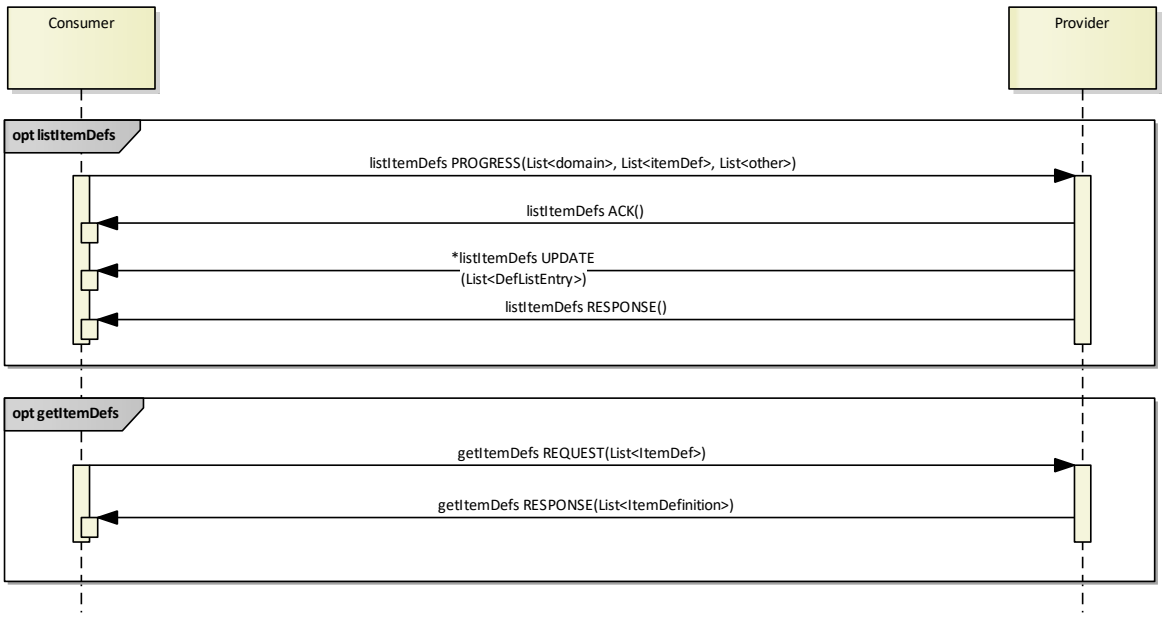

Figure 3-10: Plan Information Management Operations for Generic Data Item

Each of the first four capability sets corresponds to a particular type of MPS data item and comprises two operations:

a) listDefs: request a list of available definitions, filtered by domain and/or key;

b) getDef: obtain the full definitions for a specified set of definition IDs.

The first enables a consumer to discover which definitions are available, and the second enables their retrieval.

The fifth capability set comprises a single operation to retrieve the MPS system configuration data: MPSSystemConfig.

3.5.2 DISCUSSION—MPS DATA ITEMS

MPS data items relevant to the plan information management service and their relationships are defined within the MPS information model in section 4, and specifically identified as planning configuration data in 4.2.7.

The following MO objects are applicable to the service:

- RequestDefinition;
- ActivityDefinition;
- EventDefinition;
- Resource [Definition];
- MPSSystemConfig.

Each definition has its own identity (key and version). It should be noted that for MPSSystemConfig, as a singleton object, no key is required.

In the case of planning Resources, the definition can omit the value attribute, although it can also be used to provide a default or initial value.

3.5.3 HIGH-LEVEL REQUIREMENTS

Req_3.5.3.H.45 The following set of mission planning configuration data shall be available to the provider in a any deployment of the plan information management service:

a) Planning Request Definitions (as RequestDefinition objects [4.2.5.1]);

b) Planning Activity Definitions (as ActivityDefinition objects [4.2.2.1]);

c) Planning Event Definitions (as EventDefinition objects [4.2.3.1]);

d) Planning Resource Definitions (as Resource objects [4.2.4.2]) [Optional];

e) MPS System Configuration Data (as an MPSSysConfig object [4.2.7]).

3.5.4 FUNCTIONAL REQUIREMENTS

Req_3.5.4.F.46 If no version is specified for the definition to be retrieved in a listItem]Defs or operation then the service provider shall list all available versions of the definition.

Req_3.5.4.F.47 If no version is specified for the definition to be retrieved in a get[Item]Defs or getSystemConfig operation then the service provider shall return the latest available version of the definition.

3.5.5 OPERATION: LISTREQUESTDEFS

3.5.5.1 Overview

The listRequestDefs operation is used to obtain a list of available RequestDefinitions (key and version) together with their descriptions. The list can be filtered by domain or restricted to specified definition IDs. All available versions are listed.

The domain field is an ordered list of identifiers representing a domain hierarchy, any node of which can use '*' as a wildcard (meaning any domain identifier at that level of the hierarchy). If a set of domains is required that cannot be represented through the use of wildcards, then the operation will need to be repeated using different domain filters.

Operation Identifier	listRequestDefs		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	Yes Yes	domain : (List <MAL::Identifier>) requestDefs : (List <MAL::ObjectRef<RequestDefinition>>)
OUT	ACK	No	Empty
OUT	UPDATE	No	requestDefs : (List <DefListEntry>)
OUT	RESPONSE	No	Empty

3.5.5.2 Structures

Req_3.5.5.S.48 The RequestDefinitions to be listed shall be specified in the request message of the listRequestDefs operation, by the following fields, each of which is nullable:

a) domain;

b) requestDefs.

3.5.5.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5.6 OPERATION: GETREQUESTDEFS

3.5.6.1 Overview

The getRequestDefs operation is used to retrieve one or more available RequestDefinitions, whose identity is known to the consumer.

Operation Identifier	getRequestDefs		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	requestDefs : (List <MAL::ObjectRef<RequestDefinition>>)
OUT	RESPONSE	No	definitions : (List <RequestDefinition>)

3.5.6.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5.7 OPERATION: LISTEVENTDEFS

3.5.7.1 Overview

The listEventDefs operation is used to obtain a list of available EventDefinitions (key and version) together with their descriptions. The list can be filtered by domain or restricted to specified definition IDs. All available versions are listed.

The domain field is an ordered list of identifiers representing a domain hierarchy, any node of which can use '*' as a wildcard (meaning any domain identifier at that level of the hierarchy). If a set of domains is required that cannot be represented through the use of wildcards, then the operation will need to be repeated using different domain filters.

Operation Identifier	listEventDefs		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	Yes	domain : (List <MAL::Identifier>)

Pattern Sequence	Message	Nullable	Body Signature
		Yes	eventDefs : (List <MAL::ObjectRef<EventDefinition>>)
OUT	ACK	No	Empty
OUT	UPDATE	No	eventDefs : (List <DefListEntry>)
OUT	RESPONSE	No	Empty

3.5.7.2 Structures

Req_3.5.7.S.49 The EventDefinitions to be listed shall be specified in the request message of the listEventDefs operation, by the following fields, each of which is nullable:

a) domain;

b) eventDefs.

3.5.7.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5.8 OPERATION: GETEVENTDEFS

3.5.8.1 Overview

The getEventDefs operation is used to retrieve one or more available EventDefinitions, whose identity is known to the consumer.

Operation Identifier	getEventDefs		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	eventDefs : (List <MAL::ObjectRef<EventDefinition>>)
OUT	RESPONSE	No	definitions : (List <EventDefinition>)

3.5.8.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area

INVALID	MPS

3.5.9 OPERATION: LISTACTIVITYDEFS

3.5.9.1 Overview

The listActivityDefs operation is used to obtain a list of available ActivityDefinitions (key and version) together with their descriptions. The list can be filtered by domain or restricted to specified definition IDs. All available versions are listed.

The domain field is an ordered list of identifiers representing a domain hierarchy, any node of which can use '*' as a wildcard (meaning any domain identifier at that level of the hierarchy). If a set of domains is required that cannot be represented through the use of wildcards, then the operation will need to be repeated using different domain filters.

Operation Identifier	listActivityDefs		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	Yes Yes Yes	domain : (List <MAL::Identifier>) activityDefs : (List <MAL::ObjectRef<ActivityDefinition>>) defaultTags : (List<MAL::String>)
OUT	ACK	No	Empty
OUT	UPDATE	No	activitytDefs : (List <DefListEntry>)
OUT	RESPONSE	No	Empty

3.5.9.2 Structures

Req_3.5.9.S.50 The ActivityDefinitions to be listed shall be specified in the request message of the listActivityDefs operation, by the following fields, each of which is nullable:

a) domain;

b) activityDefs;

c) defaultTags (associated with ActivityDefinitions to be included).

3.5.9.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	**Area**
INVALID	MPS

3.5.10 OPERATION: GETACTIVITYDEFS

3.5.10.1 Overview

The getActivityDefs operation is used to retrieve one or more available ActivityDefinitions, whose identity is known to the consumer.

Operation Identifier	getActivityDefs		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	activityDefs : (List <MAL::ObjectRef<ActivityDefinition>>)
OUT	RESPONSE	No	definitions : (List <ActivityDefinition>)

3.5.10.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5.11 OPERATION: LISTRESOURCEDEFS

3.5.11.1 Overview

The listResourceDefs operation is used to obtain a list of available Resources (key and version) together with their descriptions. The list can be filtered by domain or restricted to data types. All available versions are listed.

The domain field is an ordered list of identifiers representing a domain hierarchy, any node of which can use '*' as a wildcard (meaning any domain identifier at that level of the hierarchy). If a set of domains is required that cannot be represented through the use of wildcards, then the operation will need to be repeated using different domain filters.

Operation Identifier	listResourceDefs		
Interaction Pattern	PROGRESS		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	PROGRESS	Yes Yes	domain : (List <MAL::Identifier>) dataType : (List <MAL::AttributeType>)
OUT	ACK	No	Empty
OUT	UPDATE	No	resourceDefs : (List <DefListEntry>)
OUT	RESPONSE	No	Empty

3.5.11.2 Structures

Req_3.5.11.S.51 The Resources to be listed shall be specified in the request message of the listResourceDefs operation, by the following fields, each of which is nullable:

a) domain;

b) dataType: list of Resource data types to be included.

3.5.11.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5.12 OPERATION: GETRESOURCEDEFS

3.5.12.1 Overview

The getResourceDefs operation is used to retrieve the definition of one or more available Resources, whose identity is known to the consumer.

It should be noted that this operation is designed to retrieve the resource definition and not the current value of the resource (the value field may contain a default value for the resource).

Operation Identifier	getResourceDefs		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	resources : (List <MAL::ObjectRef<Resource>>)
OUT	RESPONSE	No	definitions : (List <Resource>)

3.5.12.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.5.13 OPERATION: GETSYSTEMCONFIG

3.5.13.1 Overview

The getSystemConfig operation is used to retrieve system configuration data relating to the MPS system.

Operation Identifier	getSystemConfig		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	version : (MAL::UInteger)
OUT	RESPONSE	No	systemConfig : (MPSSystemConfigDetails)

3.5.13.2 Structures

Req_3.5.13.S.52 The version field of the getSystemConfig operation request message body shall reference the required version of the MPSSystemConfig object.

3.5.13.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS

3.6 SERVICE: PLAN EDIT SERVICE

3.6.1 OVERVIEW

The Plan Edit Service, introduced in 2.5.6, is provided by a plan execution function and enables its consumers to modify Plans that have already been submitted for execution. It allows an external user or function to update the status of the Plan; insert, modify, or delete its constituent ActivityInstances and EventInstances; update the value of Resources; and apply a time shift to a Plan. It comprises the following operations, of which those in capability sets 1 and 2 are mandatory.

In some deployments, the Plan Edit Service could also be provided by a planning function to enable users to make adjustments to their planned activities prior to submission of the plan for execution.

Area Identifier	Service Identifier	Area Number	Service Number	Area Version
MPS	PlanEdit	5	5	1

Interaction Pattern	Operation Identifier	Operation Number	Capability Set
SUBMIT	updatePlanStatus	1	1
REQUEST	insertActivity	2	2
REQUEST	insertEvent	3	
SUBMIT	deleteActivity	4	
SUBMIT	deleteEvent	5	
SUBMIT	updateActivity	6	3
SUBMIT	updateEvent	7	
SUBMIT	updateResource	8	4
SUBMIT	updateResourceProfile	9	5
SUBMIT	applyTimeShift	10	6

The updatePlanStatus operation allows the service consumer to modify the status of a previously submitted Plan, including the isAlternate flag.

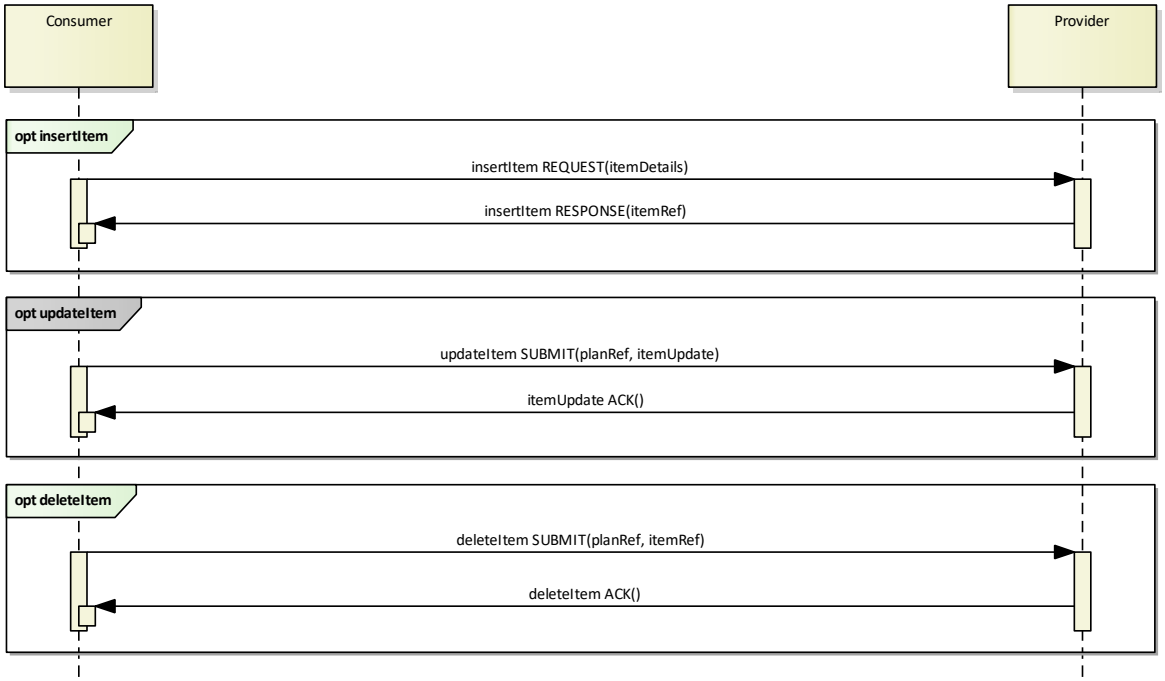

Figure 3-11: Plan Edit Operations for Generic Data Item (Activity or Event)

For planning activities and events, three operations are defined for each item type to insert, optionally to update, and to delete an ActivityInstance or EventInstance in a referenced Plan that has previously been submitted to a plan execution function (the service provider). The insert operation results in the creation by the service provider of a new ActivityInstance or EventInstance in the specified Plan. A reference to the new instance (key and version) is returned in the response message.

For planning resources, if supported, it is only possible to update the value of the Resource. This is supported in two ways, as two distinct operations:

- updateResource: a discrete update to a specified value at a specified instant of time;
- updateResourceProfile: a revised profile covering a period of time.

The applyTimeShift operation is a special operation that allows the timings contained in an entire Plan, or selected SubPlans to be shifted by a specified offset.

3.6.2 DISCUSSION—MPS DATA ITEMS

MPS data items relevant to the plan edit service and their relationships are defined within the MPS information model in 4.2.

The following MO objects are directly applicable to the service:

- Plan;
- ActivityInstance;
- EventInstance;
- Resource.

The identity of Plans comprises both key and version. Plans must have been previously submitted to a plan execution function for execution (using the plan execution control service).

Plans contain ActivityInstances and EventInstances, the same instance of which may occur in multiple overlapping Plans or versions of Plans. Plans can also optionally contain ResourceProfiles referencing Resources.

The insertActivity and insertEvent operations of the plan edit service result in creation of new ActivityInstances and EventInstances respectively. Deletion of ActivityInstances and EventInstances does not necessarily result in their removal from the plan execution function, but rather their transition to a terminated state.

The following MO objects can be referenced by the ActivityInstances and EventInstances contained within a Plan and plan edit service operations:

- ActivityDefinition;
- EventDefinition.

3.6.3 HIGH-LEVEL REQUIREMENTS

Req_3.6.3.H.53 The following set of mission planning configuration data shall be available to both provider and consumers in a any deployment of the plan edit service:

a) Planning Activity Definitions (as ActivityDefinition objects [4.2.2.1]);

b) Planning Event Definitions (as EventDefinition objects [4.2.3.1]);

c) Planning Resource Definitions (as Resource objects [4.2.4.2]) [Optional];

d) MPS System Configuration Data (as an MPSSysConfig object [4.2.7]);

e) Functional Requirements.

Req_3.6.3.F.54 In response to an updatePlanStatus operation, the service provider shall update the isAlternate flag of the Plan accordingly and apply any change to Plan status consistently with the plan state model (4.2.6.2).

NOTE – It is implementation dependent what the service provider does in the event of a Plan status change when the Plan has executing ActivityInstances. It is recommended that the plan execution control service is used to manage Plan deactivation in an orderly manner.

Req_3.6.3.F.55 In response to an insertActivity operation, the service provider shall create a new ActivityInstance in the referenced Plan and return its identity to the consumer.

Req_3.6.3.F.56 In response to an insertEvent operation, the service provider shall create a new EventInstance in the referenced Plan and return its identity to the consumer.

Req_3.6.3.F.57 In response to a deleteActivity operation, the service provider shall transition the referenced ActivityInstance to the TERMINATED state.

NOTE – If the ActivityInstance is in the EXECUTING state, then it is implementation dependent what action is taken by the service provider.

Req_3.6.3.F.58 In response to a deleteEvent operation, the service provider shall transition the referenced EventInstance to the TERMINATED state.

NOTE – This may also affect any ActivityInstances whose start or end trigger is linked to the deleted EventInstance as a terminated event will not result in the activity being triggered.

Req_3.6.3.F.59 In response to an updateActivity operation, the service provider shall apply the changes contained in the ActivityUpdate to the referenced ActivityInstance.

Req_3.6.3.F.60 In response to an updateEvent operation, the service provider shall apply the changes contained in the EventUpdate to the referenced EventInstance.

Req_3.6.3.F.61 In response to an updateResource operation, the service provider shall apply the changes contained in the ResourceUpdate to the referenced Resource.

Req_3.6.3.F.62 In response to an updateResourceProfile operation, the service provider shall apply the supplied ResourceProfile to the referenced Resource.

Req_3.6.3.F.63 In response to an applyTimeShift operation, the service provider shall apply the specified time shift to the unexpired (future) content, or the specified time period of the referenced Plan or its SubPlan(s), including temporal start and end triggers on ActivityInstances.

NOTE – It is implementation dependent whether any other elements of the Plan are time shifted, particularly with regard to the eventTime of EventInstances and, where supported, the start and end times on resource ProfileSegments and the times on Profile Entries.

3.6.4 OPERATION: UPDATEPLANSTATUS

3.6.4.1 Overview

The updatePlanStatus operation may be used to modify the status of a previously submitted Plan. Directly modifying the status field of a Plan may be used by a third party function to autonomously terminate (or activate) a Plan, but the operation also allows the isAlternate flag to be set or cleared.

It is implementation dependent what action the service provider takes in response to a change of Plan status. The service provider may not permit certain state changes (for example to modify the status of a TERMINATED plan, which is inconsistent with the plan status model), in which case an UPDATE_FAILED error is returned.

A set of Plans with a common precursor may be submitted to a plan execution function to cater for alternative or contingency scenarios. All but one of these Plans should have the isAlternate flag set, to inform the plan execution function (and the mission operations team) which is the nominal Plan. It is implementation dependent whether a plan execution control service provider will allow a Plan to be activated with the isAlternate flag set, but for operational safety reasons this may be blocked. In a contingency scenario, the updatePlanStatus operation can be used to set the flag on the nominal Plan, and reset the flag on the required contingency Plan, making it operational.

Operation Identifier	updatePlanStatus		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No No No	planRef : (MAL::ObjectRef<Plan>) status : (PlanStatusEnum) isAlternate : (MAL::Boolean)

3.6.4.2 Structures

Req_3.6.4.S.64 The status and isAlternate fields of the updatePlanStatus submit message shall both be nullable, but one must be present.

3.6.4.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UPDATE_FAILED	MPS

3.6.5 OPERATION: INSERTACTIVITY

3.6.5.1 Overview

The insertActivity operation sends an InsertedActivityDetails structure (an ActivityDetails structure with Plan reference and start/end triggers) to the provider, which then creates a corresponding ActivityInstance object in the referenced Plan and returns its identity to the consumer. It is up to the planning system, how to manage concurrent access to the plan.

Insertion may fail if the Plan is already in the TERMINATED state, in which case an INSERT_FAILED error is returned.

Operation Identifier	insertActivity		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	activityDetails : (InsertedActivityDetails)
OUT	RESPONSE	No	activityRef : (MAL::ObjectRef<ActivityInstance>)

3.6.5.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UNSUPPORTED	MPS
INSERT_FAILED	MPS

3.6.6 OPERATION: INSERTEVENT

3.6.6.1 Overview

The insertEvent operation sends an InsertedEventDetails structure, which includes a Plan reference, to the provider, which then creates a corresponding EventInstance object in the referenced Plan and returns its identity to the consumer. It is up to the planning system, how to manage concurrent access to the plan.

Insertion may fail if the Plan is already in the TERMINATED state, in which case an INSERT_FAILED error is returned.

Operation Identifier	insertEvent		
Interaction Pattern	REQUEST		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	REQUEST	No	eventDetails : (InsertedEventDetails)
OUT	RESPONSE	No	eventRef : (MAL::ObjectRef<EventInstance>)

3.6.6.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
INSERT_FAILED	MPS

3.6.7 OPERATION: DELETEACTIVITY

3.6.7.1 Overview

The deleteActivity operation requests that a specified ActivityInstance within a Plan is deleted by the service provider. In practice, the activity is not removed, but transitioned to the TERMINATED state with deletion indicated in the statusInfo field. The ActivityInstance is not subsequently executed by the service provider, but it is implementation dependent what action is taken by the service provider if the ActivityInstance is in the EXECUTING state. It is up to the planning system, how to manage concurrent access to the plan.

Deletion may fail if the referenced Plan or ActivityInstance is already in the TERMINATED state, in which case the DELETE_FAILED error is returned.

Operation Identifier	deleteActivity		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**

Pattern Sequence	Message	Nullable	Body Signature
IN	SUBMIT	No No	planRef : (MAL::ObjectRef<Plan>) activityRef : (MAL::ObjectRef<ActivityInstance>)

3.6.7.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
DELETE_FAILED	MPS

3.6.8 OPERATION: DELETEEVENT

3.6.8.1 Overview

The deleteEvent operation requests that a specified EventInstance within a Plan is deleted by the service provider. In practice, the event is not removed, but transitioned to the TERMINATED state with deletion indicated in the statusInfo field. The EventInstance is not subsequently triggered by the service provider. It is up to the planning system, how to manage concurrent access to the plan.

Deletion may fail if the referenced Plan or EventInstance is already in the TERMINATED state, in which case the DELETE_FAILED error is returned.

Operation Identifier	deleteEvent		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No No	planRef : (MAL::ObjectRef<Plan>) eventRef : (MAL::ObjectRef<EventInstance>)

3.6.8.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
DELETE_FAILED	MPS

3.6.9 OPERATION: UPDATEACTIVITY

3.6.9.1 Overview

The updateActivity operation may be used to modify an ActivityInstance in a Plan that has already been submitted to the service provider. The consumer submits an ActivityUpdate structure which is applied by the service provider to the referenced ActivityInstance. It is up to the planning system, how to manage concurrent access to the plan.

Update may fail if the referenced Plan or ActivityInstance is already in the TERMINATED state, in which case the UPDATE_FAILED error is returned.

Operation Identifier	updateActivity		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No No	planRef : (MAL::ObjectRef<Plan>) activityUpdate : (ActivityUpdate)

3.6.9.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UPDATE_FAILED	MPS

3.6.10 OPERATION: UPDATEEVENT

3.6.10.1 Overview

The updateEvent operation may be used to modify an EventInstance in a Plan that has already been submitted to the service provider. The consumer submits an EventUpdate structure which is applied by the service provider to the referenced EventInstance. It is up to the planning system, how to manage concurrent access to the plan.

Update may fail if the referenced Plan or EventInstance is already in the TERMINATED state, in which case the UPDATE_FAILED error is returned.

Operation Identifier	updateEvent		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No No	planRef : (MAL::ObjectRef<Plan>) eventUpdate : (EventUpdate)

3.6.10.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UPDATE_FAILED	MPS

3.6.11 OPERATION: UPDATERESOURCE

3.6.11.1 Overview

The updateResource operation may be used to modify the value of a Resource at the specified point in time, in a Plan that has already been submitted to the service provider. The consumer submits an ResourceUpdate structure which is applied by the service provider to the referenced Resource. It is up to the planning system, how to manage concurrent access to the plan.

Update may fail if the referenced Plan is already in the TERMINATED state, in which case the UPDATE_FAILED error is returned.

Operation Identifier	updateResource		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No No	planRef : (MAL::ObjectRef<Plan>) resourceUpdate : (ResourceUpdate)

3.6.11.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UNSUPPORTED	MPS
UPDATE_FAILED	MPS

3.6.12 OPERATION: UPDATERESOURCEPROFILE

3.6.12.1 Overview

The updateResourceProfile operation may be used to modify the value of a Resource over a period of time, in a Plan that has already been submitted to the service provider. The

consumer submits an ResourceProfile structure which is applied by the service provider to the referenced Resource. It is up to the planning system, how to manage concurrent access to the plan.

Update may fail if the referenced Plan is already in the TERMINATED state, in which case the UPDATE_FAILED error is returned.

Operation Identifier	updateResourceProfile		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No No	planRef : (MAL::ObjectRef<Plan>) resourceProfile : (ResourceProfile)

3.6.12.2 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UNSUPPORTED	MPS
UPDATE_FAILED	MPS

3.6.13 OPERATION: APPLYTIMESHIFT

3.6.13.1 Overview

The applyTimeShift operation may be used to request a shift in the timing by a fixed offset of the ActivityInstances, EventInstances, and ResourceProfiles contained within a Plan that has previously been submitted to a plan execution function. The operation may also be restricted to one or more SubPlans within the referenced Plan and/or to a specified time period within the Plan. The service provider applies the time shift to the timing of ActivityInstances, EventInstances, and ResourceProfiles contained within the Plan or SubPlan(s).

The time shift may fail if the referenced Plan is already in the TERMINATED state, in which case the UPDATE_FAILED error is returned.

The operation is designed to support backward compatibility[3] with simple time-based on-board schedules, and may not be appropriate for use with plans that include event or position-based triggers and resource profiles. What is shifted within the Plan is implementation dependent, but shall include time-based start and end triggers on ActivityInstances.

[3] For example, the operation is equivalent to the ECSS PUS Service 11 Time Shift operation (15). Service 11 is for the management of on-board time-based schedules (also known as on-board queues or mission timelines).

EventInstances may also be shifted, but it is noted that some EventInstances correspond to predicted orbital events that cannot meaningfully be shifted. Similarly, where supported, resource profiles may reflect the ActivityInstances contained within the Plan and if those are shifted, the corresponding changes in Resource value should also be shifted.

NOTE – ActivityInstances have duration which means they may overlap the start or end of the specified TimeWindow for the applicability of the time shift. It is implementation dependent how this is managed, but a reasonable assumption is that the start time of the ActivityInstances must be within the specified TimeWindow. Given the potential to introduce inconsistencies into a Plan, it must be assumed that users of this service operation understand both its operational implications and its specific implementation.

Operation Identifier	applyTimeShift		
Interaction Pattern	SUBMIT		
Pattern Sequence	**Message**	**Nullable**	**Body Signature**
IN	SUBMIT	No Yes No No	planRef : (MAL::ObjectRef<Plan>) subPlans : (List <MAL::Identifier>) timePeriod : (TimeWindow) offset : (MAL::Duration)

3.6.13.2 Structures

Req_3.6.13.S.65 The subPlans and timePeriod fields of the applyTimeShift submit message shall both be nullable.

3.6.13.3 Errors

In addition to standard MAL errors, the operation may return the following MPS errors:

Error	Area
INVALID	MPS
UPDATE_FAILED	MPS

4 MPS DATA TYPES

4.1 OVERVIEW

4.1.1 GENERAL

This section defines the data types (MO objects and other data classes) applicable to the Mission Planning and Scheduling Recommended Standard. An overview of the MPS information model has been given in 2.4 above.

The MPS information model has been defined in terms of the CCSDS Mission Operations (MO) framework, specifically the MO Message Abstraction Layer (MAL) and the associated set of MAL Attribute types. This is to enable the specification of MO compliant data formats and services that reference elements of the information model.

In the case of the MPS File Formats defined in section 7, an explicit XML encoding of MAL attribute types is used, which means the implementation dependency on the MAL is removed, although data structures defined in the MAL are used.

It describes both the data actively exchanged by MPS Services and that either referenced by or required as common configuration data by service providers and users.

As previously introduced in 2.4, some elements of the MPS information model are optional. This is the case for individual data elements where:

- they may not be required by the supported capability sets of supported services;
- they form an optional element of a required data structure.

Where this is the case, this is indicated in the description of each element in the body of this section.

The section is organized into the following main sections:

- Overview (this subsection);
- MPS Data Items;
- MPS Data Types;

4.1.2 MO OBJECT NUMBERS

The **identity** of MO objects requires the specification of the **area** and object type. In an efficient encoding this may be represented by a number, which is the same as the Short Form Part for the corresponding data structure. The following table specifies the number codes assigned for efficient encoding of the defined MO object types:

Table 4-1: MO Object Numbers

Area	Area #	MO Object Type	Type #
MPS	5	ActivityDefinition	101
		ActivityInstance	102
		EventDefinition	201
		EventInstance	202
		Resource	301
		RequestDefinition	401
		RequestInstance	402
		Plan	501
		PlanningUser	601
		MPSSystemConfig	701

4.1.3 TIME SYSTEMS

Time system references allow specification the time system used for time attributes within an MPS system. This may be specified in the context of a **planning request** or a **plan** or as a system-wide default within the MPSSystemConfig object.

The set of allowed time system values is not defined by this Recommended Standard, but specified in the SANA registry for time systems:

https://sanaregistry.org/r/time_systems/

To allow for evolution, both of the set of time systems defined within this registry and through mission specific extension, time systems are not defined as an enumeration but represented as a MAL::String.

4.1.4 LISTS AND NULLABILITY

Whether a field (attribute) of a data structure or service operation message can be null or not, is indicated in the 'nullable' column of the corresponding definition table.

Where a nullable field (attribute) is a List, then this gives rise to two possible representations where there are no entries in the List:

- A Null List—there is no List in the encoded message;
- An Empty List—there is a List with no entries in the encoded message.

In most cases these are equivalent. However, where the List represents an optional filter for a subscription or query operation, there may be a distinction in their interpretation. A null list

shall always represent the absence of a filter and implies that any value in the filtered field passes the filter. The interpretation of an empty list is implementation dependent, but could be one of the following:

– no filter (equivalent to a null list);

– a filter for which there is no value that passes the filter, which would be of little use in practice;

– a filter for which only a null or empty value passes the filter. This may be of used for an attribute that is itself a list of values, for example, the 'tags' attribute of an ActivityInstance, where only those items that have no 'tags' would pass the filter.

4.2 MPS DATA ITEMS

4.2.1 GENERAL

4.2.1.1 Introduction

The principle MPS Data Items defined are those introduced in 2.4 that correspond to a set of MO objects that can be directly referenced in the context of the MPS services:

- **Planning Requests**;
- **Plans**;
- **Planning Activities**;
- **Planning Events**;
- **Planning Resources** [Optional];
- **Planning Configuration Data**;
- Functions [Optional].

The following subsections contain the definition of each MPS Data Item and are structured as follows:

- MO Objects defined for the Data Item;
- Status enumeration for the Data Item (if relevant)
- Subordinate Data Types defined for the Data Item (if relevant);
- Service-specific Data Types defined for the Data Item (these are used in the context of MPS service messages).

4.2.1.2 Updates to Multiple Object Types

PlanDetailUpdate

Specifically in the case of reporting the detailed execution status of a plan, **updates** may be reported for multiple object types: **planning activities**, **planning events**, and **planning resources**. To support this an abstract type of PlanDetailUpdate is defined as follows.

Name	***PlanDetailUpdate***	**Extends**	MAL::Composite	**SFP**	Abstract

4.2.2 PLANNING ACTIVITIES

4.2.2.1 Activity Objects

ActivityDefinition

An ActivityDefinition is an MO object that contains static configuration data relating to multiple occurrences of a planning activity. Its **identity** is defined by a definitionID, which includes a constant **key** and an evolving **version** that is updated each time the definition is revised. ActivityDefinitions form part of the planning configuration data.

Name	**ActivityDefinition**		**Extends**	MAL::Object	**SFP**	101
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the ActivityDefinition, including version.			
description	MAL::String	No	Description of the Activity.			
argDefs	List <ArgDef>	Yes	List of Argument Definitions.			
constraints	ConstraintNode	Yes	Set of Constraints applicable to all instances of the Activity.			
executionDefinition	MAL::Identifier	Yes	Reference to the definition of an executable body for the Activity (procedure, action sequence, etc.) [external to the MPS Information Model].			
durationSpec	Expression <MAL::Duration>	Yes	Supports calculation of an estimated duration of an Activity Instance.			
children	ActivityNode	Yes	Set of **Activity Details** specifying child activities, optionally with repetition.			
activityType	MAL::String	Yes	Enables a planning system to customise behavior for activities, such as their presentation in displays, based on the specified value.			
defaultTags	List <MAL::String>	Yes	Default set of Tags that may be used to associate the Activity with others, grouping activities by operational responsibility (controller/group/system) or other criteria.			

ActivityInstance

An ActivityInstance is an MO object that contains the identity of a specific occurrence of a planning activity, together with both static and dynamic information associated with that occurrence. It supports relationships to its definition, source, a related planning event and any child activities.

ActivityInstances may be contained within a Plan.

NOTE – The start and end attributes specify the trigger conditions (including time) that specify when the ActivityInstance starts and/or ends in the context of a Plan. The duration is an estimate of the time taken to execute the ActivityInstance rather than an offset, which may for example be used in the visualization of a Plan. Duration may be used in conjunction with a specified end trigger to determine the planned start time of an ActivityInstance.

Name	**ActivityInstance**		**Extends**	MAL::Object	**SFP**	102
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the ActivityInstance			
definition	MAL::ObjectRef <ActivityDefinition>	No	Reference to the ActivityDefinition			
source	MAL::ObjectRef	No	Object Type: RequestInstance | ActivityInstance | PlanningUser Reference to the source of the ActivityInstance, which is either its parent ActivityInstance, a RequestInstance if it is a root Activity, or a PlanningUser if directly inserted.			
relatedEvent	MAL::ObjectRef <EventInstance>	Yes	Optional reference to an EventInstance that is specifically associated with this instance of the Activity. Typically the Activity is placed in response to the Event.			
children	List <MAL::ObjectRef <ActivityInstance>>	Yes	References to any child ActivityInstances.			
comments	MAL::String	Yes	Any notes associated with this instance of the Activity.			
constraints	ConstraintNode	Yes	Set of Constraints applicable to this instance of the Activity.			
arguments	List <Argument>	Yes	Argument values for each Argument defined in the Activity Definition.			
start	Trigger	Yes	Optionally specifies the trigger that initiates the Activity: may be Time, Position or Event based.			
end	Trigger	Yes	Optionally specifies the trigger that ends the Activity			
duration	MAL::Duration	Yes	Optional duration of the Activity (estimated until execution, actual post execution).			
subPlan	MAL::Identifier	Yes	Optional association of the Activity with a defined sub-plan.			
tags	List <MAL::String>	Yes	Set of Tags that may be used to associate the Activity with others, grouping activities by operational responsibility (controller/group/system) or other criteria.			
status	ActivityStatusEnum	No	Current Status of the Activity Instance (see Activity State Model in §4.2.2.2)			

Attribute	Type	Nullable	Description
executionInstance	MAL::Identifier	Yes	Reference to the instance of an executable body for the Activity (procedure, action sequence, etc.) [external to the MPS Information Model].
returnData	List<MAL::NamedValue>	Yes	Optional return data from the planning process, provided as a list of ID-Value pairs. This can be used to provide additional information required by the User to interpret the planned activity.
statusInfo	MAL::String	Yes	StatusInfo provides the reason for entering the Terminated State and is customisable, but if the following conditions exist then the specified text shall be used: - Completed (nominal) - Expired (prior to Activation or during plan Supension) - Deleted - Failed (see ErrorCode/ErrorInfo)
errorCode	MAL::Integer	Yes	Error Code optional in the case of a failure status for the planning activity (for example Terminated state with statusInfo Failed). The codes are implementation specific.
errorInfo	MAL::String	Yes	Supplementary Error Information

4.2.2.2 Activity Status

ActivityStatusEnum

The following states are defined for **ActivityStatus**:

Name	**ActivityStatusEnum**	SFP	103
Status	**Value**	**Description**	
PLANNED	1	The Activity Instance has been included in the Plan	
ACTIVATED	2	The Plan including the Activity Instance has been Activated within the plan execution function.	
EXECUTING	3	Execution of the Activity Instance has been initiated	
SUSPENDED	4	Execution of the Activity Instance has been suspended	
TERMINATED	5	Execution of the Activity Instance has been terminated (further information is provided in statusInfo)	

4.2.2.3 Activity Details and Nodes

ActivityDetails

Contains the information required to create one or more ActivityInstances, including the specification of argument values and constraints.

It should be noted that the activityRef and activityOffset attributes are only relevant in the case that a Repetition has been specified in a parent ActivityNode. Temporal and sequential constraints associated with the ActivityInstance can be specified as constraints attached to a concrete SimpleActivityDetails structure.

Name	***ActivityDetails***	**Extends**	MAL::Composite	**SFP**	Abstract
Attribute	**Type**	**Nullable**	**Description**		
activityRef	Slider	Yes	Specifies how the ActivityInstance is placed with respect to any defined Repetition (0=Start; 1=End). Default is Start.		
activityOffset	Expression <MAL::Duration>	Yes	Specifies an offset in time for the ActivityInstance from any defined Repetition. Default is no offset.		
relatedEvent	Expression <MAL::ObjectRef>	Yes	Object Type: EventInstance. Specifies a related Event (or Event Group) for the ActivityInstance. Argument specifications and constraints may reference arguments and attributes of the RelatedEvent.		
comments	MAL::String	Yes	Any notes associated with the ActivityDetails.		

ActivityNode

A concrete sub-type of ActivityDetails, an ActivityNode is a container node for a set of ActivityDetails together with an optional Repetition specification. An ActivityNode is used as the root node of any ActivityDetails specification within a **planning request** or parent **activity definition**.

Name	**ActivityNode**	**Extends**	*ActivityDetails*	**SFP**	104
Attribute	**Type**	**Nullable**	**Description**		
repetition	Repetition	Yes	Optional Repetition specification.		
activities	List <ActivityDetails>	Yes	Set of ActivityDetails.		

SimpleActivityDetails

A concrete sub-type of ActivityDetails, a SimpleActivityDetails provides the information required to instantiate a single ActivityInstance.

Name	SimpleActivityDetails		Extends	*ActivityDetails*	SFP	105
Attribute	**Type**	**Nullable**	**Description**			
activityDefinition	MAL::ObjectRef <ActivityDefinition>	No	Reference to the ActivityDefinition.			
argSpecs	List <ArgSpec>	Yes	Set of argument specifications for each argument definition contained in the referenced activity definition. These supply a value for each argument, or an expression to enable the value to be derived.			
constraints	ConstraintNode	Yes	Set of Constraints specific to the ActivityInstance to be created.			
subPlan	MAL::Identifier	Yes	Optional association of the ActivityInstance with a defined sub-plan.			
tags	List <MAL::String>	Yes	Set of tags that may be used to associate the Activity with an identified subset of the Plan, grouping activities by operational responsibility (controller/group/system) or other criteria.			

4.2.2.4 Activity Service Structures

ActivityUpdate

ActivityUpdate is a data structure that is used to report the dynamic status of an ActivityInstance in the context of the MPS Plan Execution Control service monitorPlanExecutionDetail and getActivityStatus operations.

ActivityUpdates may be distributed to subscribing applications, including status displays, to inform them of the latest status of the activity. This may be particularly relevant in conjunction with a plan execution function. ActivityUpdates may be stored in activity history to provide a complete record of evolving status over time.

Name	ActivityUpdate		Extends	*PlanDetailUpdate*	SFP	106
Attribute	**Type**	**Nullable**	**Description**			
activityInstance	MAL::ObjectRef <ActivityInstance>	No	Reference to the ActivityInstance to which the status update relates.			
timestamp	MAL::Time	Yes	Time of status update. Only nullable in the context of an updateActivity operation: the timestamp must be provided when reporting ActivityInstance status.			
plan	MAL::ObjectRef <Plan>	Yes	Optional reference to the Plan containing the ActivityInstance to which this update pertains.			
arguments	List <Argument>	Yes	Argument values.			
start	Trigger	Yes	Optionally specifies the trigger that initiates the ActivityInstance: may be time, position, or event			

			based.
end	Trigger	Yes	Optionally specifies the trigger that ends the ActivityInstance
duration	MAL::Duration	Yes	Optional duration of the ActivityInstance (estimated until execution, actual post execution).
subPlan	MAL::Identifier	Yes	Optional association of the ActivityInstance with a defined sub-plan.
tags	List <MAL::String>	Yes	Set of tags that may be used to associate the ActivityInstance with an identified subset of the Plan, grouping activities by operational responsibility (controller/group/system) or other criteria.
status	ActivityStatusEnum	No	Current status of the ActivityInstance
executionInstance	MAL::Identifier	Yes	Reference to the instance of an executable body for the ActivityInstance (procedure, action sequence, etc.) [external to the MPS information model].
returnData	List<MAL::NamedValue>	Yes	Optional return data from the planning process, provided as a list of ID-Value pairs. This can be used to provide additional information required by the User to interpret the planned activity.
statusInfo	MAL::String	Yes	StatusInfo provides the reason for entering the Terminated State and is customizable, but if the following conditions exist then the specified text shall be used: - Completed (nominal) - Expired (prior to Activation or during plan Suspension) - Deleted - Failed (see ErrorCode/ErrorInfo)
errorCode	MAL::Integer	Yes	Error Code optional in the case of a failure status for the planning activity (for example Terminated state with statusInfo Failed). The codes are implementation specific.
errorInfo	MAL::String	Yes	Supplementary error information

InsertedActivityDetails

A concrete sub-type of ActivityDetails (see 4.2.2.3) that is a variation of SimpleActivityDetails providing additional details for a single ActivityInstance to be inserted into a Plan using the MPS Plan Edit service.

Name	**InsertedActivityDetails**		**Extends**	ActivityDetails	**SFP**	107
Attribute	**Type**	**Nullable**	**Description**			
plan	MAL::ObjectRef <Plan>	No	Reference to the Plan into which the ActivityInstance is to be inserted.			
start	Trigger	Yes	Optionally specifies the trigger that initiates the			

			ActivityInstance: may be time, position, or event based.
end	Trigger	Yes	Optionally specifies the trigger that ends the ActivityInstance
activityDefinition	MAL::ObjectRef <ActivityDefinition>	No	Reference to the ActivityDefinition.
argSpecs	List <ArgSpec>	Yes	Set of argument specifications for each argument definition contained in the referenced activity definition. These supply a value for each argument, or an expression to enable the value to be derived.
constraints	ConstraintNode	Yes	Set of Constraints specific to the ActivityInstance to be created.
subPlan	MAL::Identifier	Yes	Optional association of the ActivityInstance with a defined sub-plan.
tags	List <MAL::String>	Yes	Set of tags that may be used to associate the Activity with a subset of the Plan, grouping activities by operational responsibility (controller/group/system) or other criteria.

ActivitySuspensionStatus

A data structure that returns the status and supplementary suspension information for an ActivityInstance affected by an MPS Plan Execution Control service SuspendActivity or ResumeActivity operation.

Name	**ActivitySuspensionStatus**		**Extends**	MAL::Composite	**SFP**	108
Attribute	**Type**	**Nullable**	**Description**			
activityInstance	MAL::ObjectRef <ActivityInstance>	No	Reference to an ActivityInstance.			
plan	MAL::ObjectRef <Plan>	Yes	Optional reference to the Plan containing the ActivityInstance.			
status	ActivityStatusEnum	No	Current Status of the ActivityInstance			
suspensionInfo	MAL::String	Yes	Supplementary information on the suspension/resumption status of the ActivityInstance. This may detail the point of suspension, which may be specific to the suspension mode; or a reason why resumption was not possible.			

4.2.3 PLANNING EVENTS

4.2.3.1 Event Objects

EventDefinition

An EventDefinition is an MO object that contains static configuration data relating to multiple occurrences of a planning event. Its **identity** is defined by a definitionID, which includes a constant **key** and an evolving **version**, which is updated each time the definition is revised. Event definitions form part of the planning configuration data.

Events may be either Predicted or Potential. Events that are predictable either by time or position can have specific instances included in a Plan. Potential events are those that may occur during the execution of a Plan, but the specific time or position is not predicted.

Name	**EventDefinition**		**Extends**	MAL::Object	**SFP**	201
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the EventDefinition, including version.			
description	MAL::String	No	Description of the event.			
predictability	PredictabilityEnum	No	Enumeration: one of {Predicted, Potential} indicating whether the event occurrence is known in advance or can occur at any time.			
eventType	MAL::String	Yes	Enables a planning system to customize behaviour for events, such as their presentation in displays, based on the specified value.			
argDefs	List <ArgDef>	Yes	List of argument definitions.			
eventDefinitions	List <MAL::ObjectRef <EventDefinition>>	Yes	List of child event definitions. For a single event, this list shall be empty; for a group event, the list shall be populated.			

EventInstance

An EventInstance is an MO object that contains the identity of a specific occurrence of a planning event, together with both static and dynamic information associated with that occurrence. It supports relationships to its definition and source.

The source of an EventInstance may be an external event, corresponding to a NAV Predicted Event or a CSS Contact Event.

EventInstances may be contained within a Plan.

EventInstances may be referenced as a related event by an ActivityInstance, so that the ActivityInstance can reference the timing and arguments of the related EventInstance.

Name	EventInstance		Extends	MAL::Object	SFP	202
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the EventInstance			
definition	MAL::ObjectRef <EventDefinition>	No	Reference to the EventDefinition			
sourceEvent	MAL::Identifier	Yes	Reference to an external source event (e.g., NAV predicted event, or CSS contact event)			
events	List <MAL::ObjectRef <EventInstance>>	Yes	List of references to child EventInstances. For a single event, this list is empty; for a group event, the list will be populated.			
eventTime	MAL::FineTime	Yes	Predicted or actual time of the event. EventTime is nullable: it can be predicted without an eventTime (e.g., if position based)			
arguments	List <Argument>	Yes	Argument values for each argument defined in the EventDefinition.			
eventStatus	EventStatusEnum	No	Current status of the event instance (see event state model in 4.2.3.2)			
statusInfo	MAL::String	Yes	StatusInfo provides the reason for entering the terminated state and is customizable, but if the following conditions exist then the specified text shall be used: - Occurred (Event has been triggered) - Did Not Occur (Event expired or did not occur within validity period) - Deleted (Event was deleted)			

PredictabilityEnum

Name	PredictabilityEnum		SFP	203
Enumeration	**Value**	**Description**		
PREDICTED	1	Events that are predictable either by time or position can have specific instances included in a Plan.		
POTENTIAL	2	Potential events are those that may occur during the execution of a Plan, but the specific time or position is not predicted.		

4.2.3.2 Event Status

EventStatusEnum

The following states are defined for **EventStatus**:

Name	EventStatusEnum	SFP	204

Status	Value	Description
GROUP	1	The EventInstance is a group event
PLANNED	2	The EventInstance has been included in the Plan
ACTIVATED	3	The Plan including the EventInstance has been Activated within the plan execution function.
TERMINATED	4	The EventInstance has reached a terminal status (further information is provided in statusInfo)

4.2.3.3 Event Service Structures

EventUpdate

EventUpdate is a data structure that is used to report the dynamic status of an EventInstance in the context of the MPS Plan Execution Control service monitorPlanExecutionDetail operation.

EventUpdates may be distributed to subscribing applications, including status displays, to inform them of the latest status of the event. This may be particularly relevant in conjunction with a plan execution function. EventUpdates may be stored in event history to provide a complete record of evolving status over time.

Name	EventUpdate	Extends	*PlanDetailUpdate*	SFP	205

Attribute	Type	Nullable	Description
eventInstance	MAL::ObjectRef <EventInstance>	No	Reference to the EventInstance to which the status update relates.
timestamp	MAL::Time	No	Time of status update.
eventTime	MAL::FineTime	No	Predicted or actual time of the event. EventTime is nullable: it can be predicted without an EventTime (e.g., if position based)
arguments	List <Argument>	Yes	Argument values.
eventStatus	EventStatusEnum	No	Current status of the EventInstance.
statusInfo	MAL::String	Yes	StatusInfo provides the reason for entering the Terminated state and is customizable, but if the following conditions exist then the specified text shall be used:: - Occurred (Event has been triggered) - Did Not Occur (Event expired or did not occur within validity period) - Deleted (Event was deleted)

InsertedEventDetails

A data structure that provides the information required to create the EventInstance to be inserted into a Plan using the MPS Plan Edit service.

Name	InsertedEventDetails		Extends	MAL::Composite	SFP	206
Attribute	**Type**	**Nullable**	**Description**			
plan	MAL::ObjectRef <Plan>	No	Reference to the Plan into which the Event is to be inserted.			
eventDefinition	MAL::ObjectRef <EventDefinition>	No	Reference to the EventDefinition			
eventTime	MAL::FineTime	No	Specifies the predicted or actual time of the event. For an inserted event this must be present.			
arguments	List <Argument>	Yes	Argument values.			

4.2.4 PLANNING RESOURCES [OPTIONAL]

4.2.4.1 General

Planning resources are an optional element of the MPS information model. Support for planning resources is not a requirement for compliance of an MPS system with the MPS service interfaces.

4.2.4.2 Resource Objects

Resource

A resource is an MO object that contains both the static attributes that define a **planning resource** and a dynamic attribute that holds its current value. Its **identity** is defined by a constant **key** and evolving **version**, which is updated each time the definition is revised. Resource definitions form part of the planning configuration data and in practice the value attribute may be omitted in this context, although it may also be used to provide an initial or default value.

Depending on the resource data type, the resource definition may require additional type specific attributes to support data validation. Sub-types are defined for Numeric, String, and enumerated Status type resources. The base Resource MO object type can be used where no data validation is applicable. The following attributes are applicable to the base type and all sub-types.

Name	Resource		Extends	MAL::Object	SFP	301
Attribute	**Type**	**Nullable**	**Description**			

identity	MAL::ObjectIdentity	No	Identity of the Resource, including version of the Resource definition.
description	MAL::String	No	Description of the Resource.
dataType	MAL::AttributeType	No	Specifies the data type of the Resource, which must be a supported MAL Attribute type.
units	MAL::String	Yes	Optional. Specifies the units the value of the Resource is expressed in, as defined in reference [D6] annex D
validationData	ValidationDetails	Yes	Optional. Specifies the allowed range of values for the Resource, with concrete subtypes specific to the data type of the Resource.
value	MAL::Attribute	Yes	Value of the resource. MAL Attribute type must match the dataType of the Resource definition. The value is only nullable in the context of a Resource definition (planning configuration data).

NumericProfile

An additional concrete sub-type of ValidationDetails applicable only to Resources of any numeric type, including Duration, that provides additional attributes for the specification of numeric data validation.

Name	**NumericResource**		**Extends**	*ValidationDetails*	**SFP**	302
Attribute	**Type**	**Nullable**	**Description**			
minimum	ResourceProfile	No	Defines the permitted minimum value over time.			
maximum	ResourceProfile	No	Defines the permitted maximum value over time.			

4.2.4.3 Resource Profiles

ResourceProfile

A ResourceProfile provides the evolution of a value for a single **planning resource** over time as a set of ProfileSegments.

Name	**ResourceProfile**		**Extends**	MAL::Composite	**SFP**	303
Attribute	**Type**	**Nullable**	**Description**			
resource	MAL::ObjectRef <Resource>	No	Object Type: *Resource* Reference to a Resource			
profileSegments	List <ProfileSegment>	No	Set of Profile Segments			

ProfileSegment

A ProfileSegment defines the time range and interpolation method for a set of ProfileEntries.

Name	ProfileSegment		Extends	MAL::Composite	SFP	304
Attribute	**Type**	**Nullable**	**Description**			
interpolation	InterpolationTypeEnum	No	Interpolation method to be applied for values lying between points defined in the profile segment. Default = Step.			
start	Expression <MAL::Time>	No	Start of time range covered by the profile segment.			
end	Expression <MAL::Time>	No	End of time range covered by the profile segment			
startIncluded	MAL::Boolean	No	Indicates whether the start time is included in the profile segment. Default = True.			
endIncluded	MAL::Boolean	No	Indicates whether the end time is included in the profile segment. This allows the same time to be used as the end of one segment and the start of another. Default = False.			
profileEntries	List <ProfileEntry>	No	Set of profile entries (resource value points).			

InterpolationTypeEnum

The following InterpolationTypes are defined:

Name	InterpolationTypeEnum		SFP	305
Enumeration	**Value**	**Description**		
STEP	1	No interpolation: resource values remain unchanged between defined points.		
LINEAR	2	Linear interpolation: resource values follow a straight line between defined points.		
POLYNOMIAL	3	Polynomial interpolation: resource values follow a curve fitting the defined points.		

ProfileEntry

Defines the value (or minimum/maximum value) of a resource at a particular point in time.

The data type of the value can be any valid MAL Attribute type, but must match the defined dataType in the corresponding Resource definition.

Name	ProfileEntry		Extends	MAL::Composite	SFP	306
Attribute	**Type**	**Nullable**	**Description**			
time	Expression <MAL::Time>	No	Time of resource data point			
value	MAL::Attribute	No	Value of resource data point			

RelativeResourceProfile

A variation on ResourceProfile, the RelativeResourceProfile uses relative timestamps of type Duration (indicating an offset from a reference time, such as the start time of an Activity). RelativeResourceProfiles are used in the context of a complex resource constraint.

Name	RelativeResourceProfile		Extends	MAL::Composite	SFP	307
Attribute	**Type**	**Nullable**	**Description**			
resource	MAL::ObjectRef <Resource>	No	Object Type: *Resource* Reference to a Resource			
profileSegments	List <RelativeProfileSegment>	No	Set of RelativeProfileSegments			

RelativeProfileSegment

A RelativeResourceSegment defines the time range and interpolation method for a set of RelativeProfileEntries.

Name	RelativeProfileSegment		Extends	MAL::Composite	SFP	308
Attribute	**Type**	**Nullable**	**Description**			
interpolation	InterpolationTypeEnum	No	Interpolation method to be applied for values lying between points defined in the relative profile segment. Default = Step.			
start	Expression <MAL::Duration>	No	Relative start of time range covered by the relative profile segment.			
end	Expression <MAL::Duration>	No	Relative end of time range covered by the relative profile segment			
startIncluded	MAL::Boolean	No	Indicates whether the start time is included in the relative profile segment. Default = True.			
endIncluded	MAL::Boolean	No	Indicates whether the end time is included in the relative profile segment. This allows the same time to be used as the end of one segment and the start of another.			

			Default = False.
profileEntries	List <RelativeProfileEntry>	No	Set of relative profile entries (resource value points).

RelativeProfileEntry

Defines the value (or minimum/maximum value) of a resource at a relative point in time.

Specific sub-types exist for each allowed data type for a Resource. These replace the variant type value attribute with one of concrete data type.

Name	**RelativeProfileEntry**		**Extends**	MAL::Composite	**SFP**	309
Attribute	**Type**	**Nullable**	**Description**			
time	Expression <MAL::Duration>	No	Relative time of resource data point			
value	MAL::Attribute	No	Value of resource data point			

4.2.4.4 Resource Service Structures

ResourceUpdate

ResourceUpdate is a data structure that is used to report the value of a Resource at a given point in time in the context of the MPS Plan Execution Control service monitorPlanExecutionDetail operation, or to supply an updated value for a Resource in the context of the MPS Plan Edit service.

Resource updates may be distributed to subscribing applications, including status displays, to inform them of the latest value of the Resource. This may be particularly relevant in conjunction with a plan execution function. Resource updates may be stored in resource history to provide a complete record of evolving value over time.

Resource updates are also effectively contained within a Plan to describe the predicted evolution of Resources over the duration of that Plan. However, in this context the ResourceProfile construct is used (see 4.2.4.3 above).

Name	**ResourceUpdate**		**Extends**	*PlanDetailUpdate*	**SFP**	310
Attribute	**Type**	**Nullable**	**Description**			
resource	MAL::ObjectRef <Resource>	No	Object Type: *Resource* Reference to the Resource to which the value update relates.			
timestamp	MAL::Time	No	Time of Resource value update.			
value	MAL::Attribute	No	Value of the resource. MAL Attribute type must			

			match the dataType of the resource definition.

4.2.5 PLANNING REQUESTS

4.2.5.1 Planning Request Objects

RequestDefinition

A RequestDefinition is an MO object that contains the specification of a re-usable planning request template.

Name	**RequestDefinition**		**Extends**	MAL::Object	**SFP**	401
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the RequestDefinition, including version.			
description	MAL::String	No	Description of the re-usable RequestDefinition.			
argDefs	List <ArgDef>	Yes	List of argument definitions. Arguments may be referenced in ActivityDetails and constraints.			
standingOrder	MAL::Boolean	No	A flag that indicates whether the planning request is for a repetitive standing order (unbounded other than by the validity period), or is a one-off request. If it is a standing order, then the ActivityNode must include specification of the repetition criteria. It should be noted that a one-off request can still include repetition.			
constraints	ConstraintNode	Yes	Set of Constraints applicable to the whole planning request. It should be noted that constraints specific to a **planning activity** can be specified within the ActivityDetails for that activity within the ActivityNode.			
activities	ActivityNode	No	Set of **activity details** specifying requested activities, optionally with repetition.			

RequestInstance

A RequestInstance is an MO object that contains the specification of a planning request. This may change over time if the request is updated by the user, each comprising a separate **version** of the request with the same object **key**.

In the context of a hierarchical or federated planning system, a RequestInstance can be used to submit a Plan (4.2.6.1) to a planning function, either embedding the Plan in the RequestInstance or passing it by reference. If passed by reference, the Plan can be retrieved using the Plan Distribution Service (3.3). Patch plans are not permitted in this context.

NOTE – RequestInstances may be created from a RequestDefinition that has defined arguments (as ArgDefs) and will in this case have the associated Arguments. An ad-hoc RequestInstance is not anticipated to hold any Arguments. The values that can be parameterized through the arguments of a re-usable RequestDefinition can be directly entered in a RequestInstance, and there would be no corresponding ArgDef associated with any Arguments supplied.

Name	**RequestInstance**		**Extends**	MAL::Object	**SFP**	402
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the RequestInstance, including version.			
userReference	MAL::Identifier	Yes	Optional user supplied reference for the planning request. This is distinct from the instancetID of the RequestInstance that is assigned by the planning function.			
creationDate	MAL::Time	No	Creation date-time of the RequestInstance version.			
definition	MAL::ObjectRef <RequestDefinition>	Yes	Reference to the RequestDefinition from which the RequestInstance was created, if a planning request template was used.			
planningPeriod	MAL::Identifier	No	Specifies which planning period the planning request applies to. Planning period IDs are mission specific, but can be used to indicate mission phase; planning cycle; or 'semester' in observatory missions.			
validityTimes	List <TimeWindow>	Yes	Validity period for the planning request, expressed as one or more time windows. The planning request must be satisfied within this period. Only one of validityTime or validityEvent should be present in a planning request.			
validityEvents	List <EventWindow>	Yes	Validity period for the planning request, expressed as one or more event windows. The planning request must be satisfied within this period. Only one of validityTime or validityEvent should be present in a planning request.			
timeSystem	MAL::String	Yes	Specifies the time system used for all time attributes within the planning request (see 4.1.3). If null, the default time system is used (see 4.2.8).			
user	MAL::ObjectRef <PlanningUser>	No	The User ID for the person or organization raising the planning request.			
description	MAL::String	No	Description of the request.			
arguments	List <Argument>	Yes	List of named argument values. If created from a template planning request, this will include the arguments defined in the RequestDefinition.			
standingOrder	MAL::Boolean	No	A flag that indicates whether the planning request is for a repetitive standing order (unbounded other than by the validity period), or is a one-off request. If it is a standing order, then the ActivityNode must include specification of the repetition criteria. It should be			

			noted that a one-off request can still include repetition.
constraints	ConstraintNode	Yes	Set of Constraints applicable to the whole planning request. It should be noted that constraints specific to a **planning activity** can be specified within the ActivityDetails for that activity within the ActivityNode.
activities	ActivityNode	No	Set of **activity details** specifying requested activities, optionally with repetition.
inputPlanRef	MAL::ObjectRef <Plan>	Yes	Reference to an existing Plan (output of one planning function) submitted as a planning request to another planning function in the context of a distributed or hierarchical planning system. Only one of inputPlanRef and inputPlan should be present within the planning request.
inputPlan	Plan	Yes	An existing Plan (output of one planning function) submitted as a planning request to another planning function in the context of a distributed or hierarchical planning system. The Plan is embedded within the planning request. Only one of inputPlanRef and inputPlan should be present within the planning request.
comments	MAL::String	Yes	Free text for any additional user comments about the request.
status	RequestStatusEnum	No	Current status of the ActivityInstance (see planning request state model in 4.2.5.2)
outputPlanRef	List<MAL::ObjectRef<Plan>>	Yes	Reference to the output Plan(s) that contains the activities resulting from the planning request. Where multiple alternate plans have been generated, these may be listed here.
returnData	List<MAL::NamedValue>	Yes	Optional return data from the planning process, provided as a list of ID-Value pairs. This can be used to provide additional information required by the User to interpret the planned operations.
statusInfo	MAL::String	Yes	StatusInfo provides the reason for termination and is customizable, but if the following conditions exist then the specified text shall be used: - Completed (all constituent activities completed successfully) - Expired (constituent activities expired prior to execution) - Failed (constituent activities failed during execution) - Deleted (constituent activities were deleted) - Partially Completed It may also be used to provide the reason for rejection.
errorCode	MAL::Integer	Yes	Error Code optional in the case of a failure status for

			the planning request (for example Terminated state with statusInfo Failed). The codes are implementation specific.
errorInfo	MAL::String	Yes	Supplementary error information

4.2.5.2 Planning Request Status

RequestStatusEnum

The following states are defined for **RequestStatus**:

Name	**RequestStatusEnum**		**SFP**	403
Status	**Value**	**Description**		
REQUESTED	1	The planning request has been submitted to the planning function		
ACCEPTED	2	The planning request has been accepted by the planning function		
REJECTED	3	The planning request has been rejected by the planning function		
CANCELLED	4	The planning request has been cancelled by the user		
PLANNED	5	The planning request has been incorporated into a Plan (see outputPlanRef)		
PROCESSING	6	The corresponding Plan has been activated within plan execution		
PROCESSED	7	Execution of the all constituent activities of the planning request have terminated. Awaiting confirmation of the status of the planning request.		
TERMINATED	8	The planning request has completed, either successfully or with a failure condition (further information is provided in statusInfo)		

4.2.5.3 Planning Request Service Structures

RequestStatusUpdate

RequestStatusUpdate is a data structure that is used to report changes in status of the RequestInstance as it proceeds through both planning and plan execution functions. Reporting is the responsibility of the planning function.

Planning request status updates may be distributed to subscribing applications, including both Users and status displays, to inform them of the latest status of the planning request. This may be particularly relevant in conjunction with a plan execution function. Status updates may be stored in planning request history to provide a complete record of evolving status over time.

Name	RequestStatusUpdate		Extends	MAL::Composite	SFP	404
Attribute	**Type**	**Nullable**	**Description**			
requestInstance	MAL::ObjectRef <RequestInstance>	No	Reference to the planning request instance to which the status update relates.			
timestamp	MAL::Time	No	Time of status update.			
status	RequestStatusEnum	No	Current status of the planning request.			
outputPlanRef	MAL::ObjectRef <Plan>	Yes	Reference to the Plan that contains the planned activities resulting from the planning request. It should be noted that this is only available once the planning request has been processed and successfully planned. The outputPlanRef may be updated following iterative planning cycles or re-planning.			
returnData	List<MAL::NamedValue>	Yes	Optional return data from the planning process, provided as a list of ID-Value pairs. This can be used to provide additional information required by the User to interpret the planned operations.			
statusInfo	MAL::String	Yes	StatusInfo provides the reason for termination and is customizable, but if the following conditions exist then the specified text shall be used: - Completed (all constituent activities completed successfully) - Expired (constituent activities expired prior to execution) - Failed (constituent activities failed during execution) - Deleted (constituent activities were deleted) - PartiallyCompleted It may also be used to provide the reason for rejection.			
errorCode	MAL::Integer	Yes	Error Code optional in the case of a failure status for the planning request (for example Terminated state with statusInfo Failed). The codes are implementation specific.			
errorInfo	MAL::String	Yes	Supplementary error information.			

PlanningRequestDetails

PlanningRequestDetails is a data structure used in the context of the MPS Planning Request service SubmitRequest and UpdateRequest operations, where the RequestInstance MO object cannot be used as the full identity of the resulting RequestInstance (key and version) is not yet known.

Its structure is equivalent to that of RequestInstance, but omitting the identity attributes and dynamic attributes used to report its status.

Name	PlanningRequestDetails		Extends	MAL::Composite	SFP	405
Attribute	**Type**	**Nullable**	**Description**			
userReference	MAL::Identifier	No	User supplied reference for the planning request. This is distinct from the instancetID of the RequestInstance that is assigned by the planning function.			
definition	MAL::ObjectRef <RequestDefinition>	Yes	Reference to the RequestDefinition from which the RequestInstance was created, if a planning request template was used.			
planningPeriod	MAL::Identifier	No	Specifies which planning period the planning request applies to. Planning period IDs are mission specific, but can be used to indicate mission phase; planning cycle; or 'semester' in observatory missions.			
validityTimes	List <TimeWindow>	Yes	Validity period for the planning request, expressed as one or more time windows. The planning request must be satisfied within this period. Only one of validityTime or validityEvent should be present in a planning request.			
validityEvents	List <EventWindow>	Yes	Validity period for the planning request, expressed as one or more event windows. The planning request must be satisfied within this period. Only one of validityTime or validityEvent should be present in a planning request.			
timeSystem	MAL::String	Yes	Specifies the time system used for all time attributes within the planning request (see 4.1.3). If null, the default time system for is used (see 4.2.8).			
user	MAL::ObjectRef <PlanningUser>	No	The User ID for the person or organization raising the planning request.			
description	MAL::String	No	Description of the request.			
arguments	List <Argument>	Yes	List of named argument values. If created from a template planning request, this will include the arguments defined in the RequestDefinition.			
standingOrder	MAL::Boolean	No	A flag that indicates whether the planning request is for a repetitive standing order (unbounded other than by the validity period), or is a one-off request. If it is a standing order, then the ActivityNode must include specification of the repetition criteria. It should be noted that a one-off request can still include repetition.			
constraints	ConstraintNode	Yes	Set of Constraints applicable to the whole planning request. It should be noted that constraints specific to a **planning activity** can be specified within the ActivityDetails for that activity within the ActivityNode.			
activities	ActivityNode	No	Set of **activity details** specifying requested activities, optionally with repetition.			

inputPlanRef	MAL::ObjectRef <Plan>	Yes	Reference to an existing Plan (output of one planning function) submitted as a planning request to another planning function in the context of a distributed or hierarchical planning system. Only one of inputPlanRef and inputPlan should be present within the planning request.
inputPlan	Plan	Yes	An existing Plan (output of one planning function) submitted as a planning request to another planning function in the context of a distributed or hierarchical planning system. The Plan is embedded within the planning request. Only one of inputPlanRef and inputPlan should be present within the planning request.
comments	MAL::String	Yes	Free text for any additional user comments about the request.

PlanningRequestResponse

PlanningRequestResponse is a data structure used in the context of the MPS Planning Request service SubmitRequest and UpdateRequest operations, in response to the submitted PlanningRequestDetails defined above. It contains a reference to the created RequestInstance and the supplied userReference to allow the user to correlate the two.

Name	**PlanningRequestResponse**		**Extends**	MAL::Composite	**SFP**	406
Attribute	**Type**	**Nullable**	**Description**			
instance	MAL::ObjectRef <RequestInstance>	No	Reference to the RequestInstance created in response to a SubmitRequest operation, or the updated version of the RequestInstance following an UpdateRequest operation.			
userReference	MAL::Identifier	No	User supplied reference for the planning request. This is distinct from the instanceID of the RequestInstance that is assigned by the planning function.			

RequestSummaryStatus

RequestSummaryStatus is a data structure used in the context of the MPS Planning Request service getRequestSummaries operation, where a list of these structures is returned. It contains header fields of the planning request and its status, but not the request content (arguments, activities and constraints).

Name	**RequestSummaryStatus**		**Extends**	MAL::Composite	**SFP**	407
Attribute	**Type**	**Nullable**	**Description**			
requestInstance	MAL::ObjectRef <RequestInstance>	No	Reference to the RequestInstance (key and version).			

userReference	MAL::Identifier	Yes	Optional user supplied reference for the planning request. This is distinct from the instancetID of the RequestInstance that is assigned by the planning function.
creationDate	MAL::Time	No	Creation date-time of the RequestInstance version.
definition	MAL::ObjectRef <RequestDefinition>	Yes	Reference to the RequestDefinition from which the RequestInstance was created, if a planning request template was used.
planningPeriod	MAL::Identifier	No	Specifies which planning period the planning request applies to. Planning period IDs are mission specific, but can be used to indicate mission phase; planning cycle; or 'semester' in observatory missions.
validityTime	List <TimeWindow>	Yes	Validity period for the planning request, expressed as one or more time windows. The planning request must be satisfied within this period. Only one of validityTime or validityEvent should be present in a planning request.
validityEvent	List <EventWindow>	Yes	Validity period for the planning request, expressed as one or more event windows. The planning request must be satisfied within this period. Only one of validityTime or validityEvent should be present in a planning request.
user	MAL::ObjectRef <PlanningUser>	No	The User ID for the person or organization raising the planning request.
description	MAL::String	No	Description of the request.
standingOrder	MAL::Boolean	No	A flag that indicates whether the planning request is for a repetitive standing order (unbounded other than by the validity period), or is a one-off request. If it is a standing order, then the ActivityNode must include specification of the repetition criteria. It should be noted that a one-off request can still include repetition.
comments	MAL::String	Yes	Free text for any additional user comments about the request.
status	RequestStatusEnum	No	Current status of the ActivityInstance (see planning request state model in 4.2.5.2)
outputPlanRef	List <MAL::ObjectRef <Plan>>	Yes	References to output Plans that contains the activities resulting from the planning request.
statusInfo	MAL::String	Yes	StatusInfo provides the reason for termination and is customizable, but includes: - Completed (all constituent activities completed successfully) - Expired (constituent activities expired prior to execution) - Failed (constituent activities failed during execution)

			- Deleted (constituent activities were deleted) - PartiallyCompleted It may also be used to provide the reason for rejection.

RequestFilter

RequestFilter is a data structure used in the context of MPS Planning Request Service operations to specify a filtered set of planning requests. All filter criteria specified are applied (logical AND, not OR).

NOTE – All attributes are nullable and it is valid to specify a RequestFilter with no filter criteria; this corresponds to an open filter in which all available planning requests are returned.

Name	**RequestFilter**		**Extends**	MAL::Composite	**SFP**	408
Attribute	**Type**	**Nullable**	**Description**			
domain	List<MAL::Identifier>	Yes	Domain of the RequestInstance. An ordered list representing a domain hierarchy, '*' can be used to represent a wildcard at that level.			
instanceID	MAL::ObjectRef <RequestInstance>	Yes	Identity (key and version) of the RequestInstance			
creationDate	MAL::Time	Yes	Creation date-time of the RequestInstance version.			
definitionID	MAL::ObjectRef <RequestDefinition>	Yes	Identity (key and version) of the RequestDefinition from which the RequestInstance was created.			
userID	MAL::ObjectRef <PlanningUser>	Yes	userID of the User who initiated the RequestInstance			
userReference	MAL::Identifier	Yes	Reference supplied by User when submitting the RequestInstance.			
status	RequestStatusEnum	Yes	Current status (enum) of the RequestInstance			
outputPlanRef	MAL::ObjectRef <Plan>	Yes	Reference to the output Plan generated in response to the RequestInstance			

4.2.6 PLANS

4.2.6.1 Plan Object

4.2.6.1.1 General

Plan

A Plan is an MO object that contains both the static attributes that define a version of a **plan** and dynamic attributes that hold its current state. Its **identity** is defined by a constant **key** and an evolving **version**, which is updated each time the Plan is revised.

Name	**Plan**		**Extends**	MAL::Object	**SFP**	501
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the Plan, including version.			
isPatchPlan	MAL::Boolean	No	Flag indicating if the Plan is a patch plan that only contains details of the changes from the precursor Plan. A patch plan must have a precursor. It must also include a single PlanRevision relative to the precursor Plan.			
precursorPlan	MAL::ObjectRef <Plan>	Yes	Reference to a precursor (or predecessor) Plan from which the changes are detailed in the Plan. This may be used if there is an iterative re-planning cycle in which successive plans overlap, or where a previous Plan has been updated through re-planning. If there is no precursor, then the Plan must be a self-standing full plan. If the Plan is a Patch Plan, then a precursor plan must be specified.			
targetPlan	MAL::ObjectRef <Plan>	Yes	Applicable only for patch plans, this is a reference to the target Plan. This target Plan is the result of applying the patch plan to the precursor Plan and is distinct from the identity of the patch plan itself. Patch plans are not permitted in the context of a planning request.			
information	PlanInformation	No	Contains header information relating to the Plan, including its originator and validity period.			
items	PlannedItems	No	Contains the planned activities and events that constitute the Plan.			
revisions	PlanRevisions	Yes	Details the changes between this Plan and other Plans (or other versions of the same Plan), usually the precursor Plan. Optional, but must be present in a patch plan.			
resources	PlanResources	Yes	Optional. Contains resource profiles for planning resources covering the period of the Plan.			
isAlternate	MAL::Boolean	No	Flag indicating if the Plan has currently been released as an Operational or Alternate plan.			

status	PlanStatusEnum	No	Current status of the Plan.
statusInfo	MAL::String	Yes	Supplementary information for a Plan in the Terminated state. This is customizable, but if the following conditions exist then the specified text shall be used: - Completed (nominal) - Superseded (by a successor Plan) - Revoked - Cancelled (deactivated after start of execution) - Expired

4.2.6.1.2 Plan Information

PlanInformation

The PlanInformation section of a plan contains administrative and validity details associated with the plan as a whole.

Name	**PlanInformation**	**Extends**	MAL::Composite	**SFP**	502
Attribute	**Type**	**Nullable**	**Description**		
originator	MAL::Identifier	No	Identity of the entity or system responsible for the production of the plan.		
productionDate	MAL::Time	No	Date and time of production of the plan.		
description	MAL::String	No	Description of the plan.		
comments	MAL::String	Yes	Field for additional comments or notes to the operations team regarding the plan.		
validityStart	MAL::Time	No	Start of validity period for the plan. The validity period defines when the plan is available for operational use. It cannot be used outside its validity period.		
validityEnd	MAL::Time	No	End of validity period for the plan.		
planPeriodStart	Trigger	No	Start of the plan period. The plan period defines the start and end points of the plan. Planned items (planning activities and events) contained within the plan must at least partially overlap the plan period. The use of the trigger structure allows this to be specified in terms of time, position, pointing, or planning events. Examples are: - a specified period of time - an orbital repeat cycle - a period between two events		
planPeriodEnd	Trigger	No	End of the plan period.		
timeSystem	MAL::String	Yes	Specifies the time system used for all time attributes		

			within the Plan (see 4.1.3). If Null, the default time system is used (see 4.2.8).

4.2.6.1.3 Planned Items

The PlannedItems section of the Plan specifies the set of planning activities and planning events contained within the Plan. It comprises two lists of contained MO objects: one of EventInstances and one of ActivityInstances. Both lists can be empty.

If the Plan is a **full plan**, then there must be an entry in the list for all planned items contained within the plan, whether changed or not. If the plan is a **patch plan**, then there is only an entry in the list for those planned items that have changed (new or modified).

PlannedItems

Name	**PlannedItems**		**Extends**	MAL::Composite	**SFP**	503
Attribute	**Type**	**Nullable**	**Description**			
plannedEvents	List <EventInstance>	Yes	List of planned events contained within the Plan.			
plannedActivities	List <ActivityInstance>	Yes	List of planned activities contained within the Plan.			

4.2.6.1.4 Plan Revisions [Optional]

PlanRevisions

The optional PlanRevisions section details the changes between this Plan and another Plan. PlanRevisions comprise a collection of PlanRevision structures, each of which details the changes with respect to one version of a Plan (the revised Plan).

In the typical case, there is only one PlanRevision corresponding to the changes between the Plan and its predecessor (the **precursor plan**). However, it is possible to include multiple PlanRevision structures documenting the differences with any other version of a Plan. This can be used to provide a change history for successive versions of the same Plan, or to document the differences between alternate Plans.

In the case of a patch plan, the Plan must include a single PlanRevision relative to the precursor Plan.

Each PlanRevision comprises an ordered set of ItemRevisions that document the change to individual planned items (planning events and activities). The order should be from earliest to latest modification within the plan period, to allow for the earliest and most critical changes to be applied first to a currently executing plan. Each ItemRevision references an individual EventInstance or ActivityInstance and indicates whether the planned item is new,

modified or deleted in the current Plan. New or modified items must also exist within the PlannedItems section of the Plan, but deleted items are not contained within the current Plan. It should be noted that unmodified items do not appear in the PlanRevision.

Name	PlanRevisions	Extends	MAL::Composite	SFP	504

Attribute	Type	Nullable	Description
planRevisions	List <PlanRevision>	No	Set of PlanRevision structures, each detailing the change with respect to an identified Plan.

PlanRevision

Name	PlanRevision	Extends	MAL::Composite	SFP	505

Attribute	Type	Nullable	Description
revisedPlan	MAL::ObjectRef <Plan>	No	Reference to the Plan (key and version) with respect to which the plan revisions are detailed. Typically this is the precursor Plan, but any other Plan can be used.
revisionStart	MAL::Time	No	Start time of the earliest revision.
revisionEnd	MAL::Time	No	End time of the latest revision.
itemRevisions	List <ItemRevision>	Yes	Ordered list (earliest to latest) of revisions to planned items (activity and event instances).

ItemRevision

Name	ItemRevision	Extends	MAL::Composite	SFP	506

Attribute	Type	Nullable	Description
itemRef	MAL::ObjectRef	No	Object Type: ActivityInstance | EventInstance Reference to a planned ActivityInstance or EventInstance that is new or modified in the current Plan, or has been deleted with respect to the referenced revisedPlan.
revisionStatus	RevisionStatusEnum	No	Revision status of the referenced item. May be one of New, Modified, or Deleted. Default = Undefined.

RevisionStatusEnum

Name	RevisionStatusEnum	SFP	507

Status	Value	Description
NEW	1	The item is new in this revision of the Plan.
MODIFIED	2	The item has been modified in this revision of the Plan.
DELETED	3	The item has been deleted in this revision of the Plan.

UNDEFINED	4	The item is unchanged in this revision of the Plan, or its revision status is undefined.

4.2.6.1.5 Plan Resources [Optional]

PlanResources

The optional PlanResources section allows the projected values of **planning resources** to be communicated between distributed planning and plan execution functions as part of a **plan**.

This is provided as a set of ResourceProfiles, one per planning resource included. Resource profiles can provide the projected evolution of the value of a planning resource over the period of the Plan. Alternatively it can provide a single value at the start of the Plan to enable synchronization between planning systems. Which approach is used is a deployment choice.

Name	**PlanResources**		**Extends**	MAL::Composite	**SFP**	508
Attribute	**Type**	**Nullable**	**Description**			
resourceProfiles	List <ResourceProfile>	No	Set of resource profiles, one per planning resource, containing the initial value of the resource and optionally the projected evolution of the resource value over the period of the Plan.			

4.2.6.2 Plan Status

PlanStatusEnum

The following states are defined for **PlanStatus**:

Name	**PlanStatusEnum**		**SFP**	509
Status	**Value**	**Description**		
DRAFT	1	The Plan has been saved by the planning function		
RELEASED	2	The Plan has been released for operational use by the planning function		
SUBMITTED	3	The Plan has been submitted to the plan execution function and is available for use, but will not execute until activated.		
ACTIVATED	4	The Plan has been activated by the plan execution function		
TERMINATED	5	The Plan has reached a terminal state, as detailed in the statusInfo. This includes the following cases: - Completed (nominal) - Superseded by a successor Plan - Revoked by a User - Cancelled (deactivated after start of execution) - Expired (reached the end of its validity period without being activated)		

4.2.6.3 Plan Service Structures

PlanUpdate

PlanUpdate is a data structure that is used to report changes in status of the Plan as it proceeds through both planning and plan execution functions. It is returned in the context of the MPS Plan Distribution service GetPlanStatus and MonitorPlanStatus operations, and also the MPS Plan Execution Control service MonitorPlanExecution and GetPlanStatus operations.

PlanUpdates may be distributed to subscribing applications, including status displays, to inform them of the latest status of a Plan. PlanUpdates may be stored in plan history to provide a complete record of evolving status over time.

Name	**PlanUpdate**		**Extends**	MAL::Composite	**SFP**	510
Attribute	**Type**	**Nullable**	**Description**			
plan	MAL::ObjectRef <Plan>	No	Reference to the Plan (key and version) to which the status update relates.			
timestamp	MAL::Time	No	Time of status update.			
isAlternate	MAL::Boolean	No	Flag indicating if the Plan has currently been released as an Operational or Alternate plan.			
status	PlanStatusEnum	No	Current status of the Plan.			
statusInfo	MAL::String	Yes	Supplementary information for a Plan in the Terminated state. This is customizable, but if the following conditions exist then the specified text shall be used: - Completed (nominal) - Superseded by a successor Plan - Revoked by a User - Cancelled (deactivated after start of execution) - Expired (reached the end of its validity period without being activated)			

PlanSummaryStatus

PlanSummaryStatus is a data structure that provides an summary view of a Plan that includes the PlanInformation section and current status, but not the full details of the Plan. It is returned in the context of the MPS Plan Distribution service GetPlanSummaries operation.

Name	**PlanSummaryStatus**		**Extends**	MAL::Composite	**SFP**	511
Attribute	**Type**	**Nullable**	**Description**			
plan	MAL::ObjectRef <Plan>	No	Reference to the Plan (key and version) to which the summary status relates.			

isPatchPlan	MAL::Boolean	No	Flag indicating if the Plan is a patch plan that only contains details of the changes from the precursor Plan. A patch plan must have a precursor. It must also include a single PlanRevision relative to the precursor Plan.
precursorPlan	MAL::ObjectRef <Plan>	Yes	Reference to a precursor (or predecessor) Plan from which the changes are detailed in the Plan. This may be used if there is an iterative re-planning cycle in which successive plans overlap, or where a previous Plan has been updated through re-planning. If there is no precursor, then the Plan must be a self-standing full plan. If the Plan is a Patch Plan, then a precursor plan must be specified.
targetPlan	MAL::ObjectRef <Plan>	Yes	Applicable only for patch plans, this is a reference to the target Plan. This target Plan is the result of applying the patch plan to the precursor Plan and is distinct from the identity of the patch plan itself. Patch plans are not permitted in the context of a planning request.
information	PlanInformation	No	Contains header information relating to the Plan, including its originator and validity period.
isAlternate	MAL::Boolean	No	Flag indicating if the Plan has currently been released as an Operational or Alternate plan.
status	PlanStatusEnum	No	Current status of the Plan.
statusInfo	MAL::String	Yes	Supplementary information for a Plan in the Terminated state.

PlanActivationStatus

PlanActivationStatus is a data structure that returns the activation status of a Plan in the context of the MPS Plan Execution Control service ActivatePlan and DeactivatePlan operations.

Name	**PlanActivationStatus**		**Extends**	MAL::Composite	**SFP**	512
Attribute	**Type**	**Nullable**	**Description**			
plan	MAL::ObjectRef <Plan>	No	Reference to the Plan (key and version) to which the status relates.			
status	PlanStatusEnum	No	Current status of the Plan.			
activationInfo	MAL::String	No	ActivationInfo provides customizable detailed information on the result of the activation/deactivation request for the referenced Plan.			

SubPlanUpdate

SubPlanUpdate is a data structure that is used to report changes in status of a sub-plan during plan execution. It is returned in the context of the MPS Plan Execution Control service MonitorSubPlanExecution and GetSubPlanStatus operations.

Sub-plans are not defined as objects within the MPS model. Individual activities within a Plan may be associated with a single sub-plan via its Identifier. The plan execution function is responsible for managing and reporting sub-plan status associated with relevant Plan Execution Control service operations, if supported.

Name	SubPlanUpdate		Extends	MAL::Composite	SFP	513
Attribute	**Type**	**Nullable**	**Description**			
subPlan	MAL::Identifier	No	Identifier of the sub-plan to which the update relates.			
timestamp	MAL::Time	No	Time of status update.			
status	SubPlanStatusEnum	No	Current status of the sub-plan, which may be Activated or Deactivated.			

SubPlanStatusEnum

Name	SubPlanStatusEnum	SFP	514
Status	**Value**	**Description**	
ACTIVATED	1	The sub-plan is active.	
DEACTIVATED	2	The sub-plan is not active.	

SubPlanActivationStatus

SubPlanActivationStatus is a data structure that returns the activation status of a sub-plan in the context of the MPS Plan Execution Control service ActivateSubPlan and DeactivateSubPlan operations.

Name	SubPlanActivationStatus		Extends	MAL::Composite	SFP	515
Attribute	**Type**	**Nullable**	**Description**			
plan	MAL::Identifier	No	Identifier of the sub-plan to which the status relates.			
status	SubPlanStatusEnum	No	Current status of the sub-plan, which may be Activated or Deactivated.			
activationInfo	MAL::String	No	ActivationInfo provides customizable detailed information on the result of the activation/deactivation request for the referenced sub-plan.			

PlanQuery

PlanQuery is a data structure used in the context of queryPlan operation of the MPS Plan Distribution Service. It is used to specify search criteria for querying the available set of Plans. All fields are nullable, in which case they do not apply as a search criteria.

Name	PlanQuery		Extends	MAL::Composite	SFP	516
Attribute	**Type**	**Nullable**	**Description**			
planID	MAL::ObjectIdentity	Yes	Query for Plans with the specified PlanID.			
hasPrecursor	MAL::Boolean	Yes	Query for Plans with or without a precursor.			
isPatchPlan	MAL::Boolean	Yes	Query for Plans that are or are not patch plans			
precursorPlan	MAL::ObjectRef <Plan>	Yes	Query for Plans with the specified precursor Plan.			
targetPlan	MAL::ObjectRef <Plan>	Yes	Applicable only for patch plans. Query for patch plans that have the specified target Plan.			
originator	MAL::Identifier	Yes	Query for Plans with the specified originator.			
productionDate	TimeWindow	Yes	Query for Plans with a production date in the specified range.			
validityPeriod	TimeWindow	Yes	Query for Plans with a validity period within (overlapping with) the specified range.			
isAlternate	MAL::Boolean	Yes	Query for Plans that are or are not Alternate plans.			
status	List <PlanStatusEnum>	Yes	Query for Plans that have a current status matching one of the specified list of Plan statuses.			
containedEvents	List <MAL::ObjectRef <EventDefinition>>	Yes	Query for Plans that contain EventInstances whose definition matches one of the specified list of EventDefinitions.			
containedActivities	List <MAL::ObjectRef <ActivityDefinition>>	Yes	Query for Plans that contain ActivityInstances whose definition matches one of the specified list of ActivityDefinitions.			

PartialPlan

A PartialPlan is a data structure returned from the getPartialPlan operation of the Plan Distribution Service that contains a reference to the source Plan, the criteria used to select the partial plan, and the partial plan itself. The partial plan uses the same structure as a normal Plan, with header fields matching those of the source Plan, but only containing the subset of ActivityInstances that matches the selection criteria. Whether EventInstances and Resources are included is implementation specific, but it might be assumed that any events and resources related to the selected ActivityInstances would be included in the returned partial plan.

Name	PartialPlan	Extends	MAL::Composite	SFP	517

Attribute	Type	Nullable	Description
sourcePlan	MAL::ObjectRef <Plan>	No	Reference to the Plan of which the partial plan is a selected subset.
domain	List<MAL::Identifier >	Yes	Selection criterion based on the domain of contained ActivityInstances. An ordered list representing a domain hierarchy, '*' can be used to represent a wildcard at that level.
subPlan	MAL::Identifier	Yes	Selection criterion based on the subPlan of contained ActivityInstances.
tags	List <MAL::String>	Yes	Selection criterion based on tags associated with contained ActivityInstances
partialPlanStart	Trigger	Yes	Selection criterion indicating the start of a range of time, position, or events associated with contained ActivityInstances.
partialPlanEnd	Trigger	Yes	Selection criterion indicating the end of a range of time, position, or events associated with contained ActivityInstances.
partialPlan	Plan	No	The returned partial plan.

PlanFilter

PlanFilter is a data structure used in the context of MPS Plan Distribution Service operations to specify a filtered set of Plans. All filter criteria specified are applied (logical AND, not OR).

Name	PlanFilter	Extends	MAL::Composite	SFP	518

Attribute	Type	Nullable	Description
domain	List<MAL::Identifier>	Yes	Domain of the Plan An ordered list representing a domain hierarchy, '*' can be used to represent a wildcard at that level.
planID	MAL::ObjectRef <Plan>	Yes	Identity (key and version) of the Plan
precursorPlan	MAL::ObjectRef <Plan>	Yes	Identity (key and version) of the precursor Plan
status	PlanStatusEnum	Yes	Current status (enum) of the Plan
originator	MAL::Identifier	Yes	Originator of the Plan
validityPeriod	TimeWindow	Yes	Period of time with which the validity period of the Plan overlaps.

PartialPlanFilter

PartialPlanFilter is a data structure input to the getPartialPlan operation of the Plan Distribution Service that contains a reference to the source Plan, and specifies the criteria used to select the partial plan.

Name	PartialPlanFilter		Extends	MAL::Composite	SFP	519
Attribute	**Type**	**Nullable**	**Description**			
sourcePlan	MAL::ObjectRef <Plan>	No	Reference to the Plan of which the partial plan is a selected subset.			
domain	List<MAL::Identifier >	Yes	Selection criterion based on the domain of contained ActivityInstances. An ordered list representing a domain hierarchy, '*' can be used to represent a wildcard at that level.			
subPlan	MAL::Identifier	Yes	Selection criterion based on the subPlan of contained ActivityInstances.			
tags	List <MAL::String>	Yes	Selection criterion based on tags associated with contained ActivityInstances			
partialPlanStart	Trigger	Yes	Selection criterion indicating the start of a range of time, position, or events associated with contained ActivityInstances.			
partialPlanEnd	Trigger	Yes	Selection criterion indicating the end of a range of time, position, or events associated with contained ActivityInstances.			

4.2.7 PLANNING USERS

PlanningUser

The source of a planning request is the user that raises it, and this is identified in the user field of a RequestInstance as a reference to a PlanningUser object.

The information held on planning users is outside the scope of this Recommended Standard. The only requirement on the PlanningUser object is that it is formulated as an MO object, with an associated object **identity**. Any additional content [attributes] of the PlanningUser object are system specific. As the PlanningUser object is not transferred in any message of the MPS services, there is no requirement to fully define the data structure within this Recommended Standard, but it is referenced in other MPS objects and data structures using an attribute of type MAL::ObjectRef.

Name	PlanningUser		Extends	MAL::Object	SFP	601
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of the PlanningUser, including version.			

4.2.8 PLANNING CONFIGURATION DATA

MPSSystemConfigDetails

The referenced timeSystem allows specification the time system used for time attributes within an MPS system. This may be specified in the context of a **planning request**, a **plan,** or as a system-wide default here within the MPSSystemConfig object.

Other configuration parameters specific to the MPS system can be defined using the customConfig attribute as a list of name-value pairs.

Name	MPSSystemConfigDetails		Extends	MAL::Object	SFP	701
Attribute	**Type**	**Nullable**	**Description**			
identity	MAL::ObjectIdentity	No	Identity of MPS system config, including version.			
timeSystem	MAL::String	No	Specifies the default time system used by the MPS system (see 4.1.3).			
customConfig	List <MAL::NamedValue >	Yes	Optional set of custom configuration parameters defined as a set of named values.			

4.2.9 CUSTOM FUNCTIONS [OPTIONAL]

FunctionDefinitionDetails

FunctionDefinition is a data structure that contains static configuration data relating to custom functions: built-in Boolean functions of an MPS system, each of which has a specified Identifier and optional set of argument definitions. This may change over time, each comprising a separate version of the definition. FunctionDefinitions form part of the planning configuration data.

Name	FunctionDefinitionDetails		Extends	MAL::Composite	SFP	801
Attribute	**Type**	**Nullable**	**Description**			
functionID	MAL::Identifier	No	ID of the custom function			
version	MAL::UInteger	No	Version of the FunctionDefinition.			
description	MAL::String	No	Description of the custom function.			
argDefs	List <ArgDef>	Yes	List of argument definitions.			

FunctionDetails

Contains the information required to invoke a defined function, including the specification of argument values.

Name	FunctionDetails		Extends	MAL::Composite	SFP	802
Attribute	**Type**	**Nullable**	**Description**			
function	MAL::Identifier	No	ID of a specific FunctionDefinition.			
argSpecs	List <ArgSpec>	Yes	Set of argument specifications for each argument definition contained in the referenced function definition. These supply a value for each argument, or an expression to enable the value to be derived.			

4.3 MPS DATA TYPES

MPS Data Types are supporting data structures used in the context of MPS Data Items and MPS service messages:

- MPS Base Data Types;
- MPS Position and Direction Types [Optional];
- Expressions;
- Additional MPS Data Types;
- Arguments;
- Constraints;
- Triggers;
- Repetitions.

4.3.1 MPS BASE DATA TYPES

The MPS information model optionally extends the set of MAL attribute types with the following additional MAL composite data types:

- Position coordinates defining a physical location;
- Direction coordinates defining a target pointing angle.

ArgTypeEnum

ArgTypeEnum is an MPS extension of the MAL::AttributeType enum that also allows specification of the data type as Position or Direction.

Name	ArgTypeEnum		SFP	1
Status	**Value**	**Description**		

BLOB	1	Binary object
BOOLEAN	2	Boolean value (True or False)
DURATION	3	Length of time in seconds
FLOAT	4	Floating point number (32 bits)
DOUBLE	5	Double precision floating point number (64 bits)
IDENTIFIER	6	The Identifier structure is used to store an identifier and can be used for indexing. It is a variable-length, unbounded, Unicode string.
OCTET	7	Signed 8 bit Integer
UOCTET	8	Unsigned 8 bit Integer
SHORT	9	Signed 16 bit Integer
USHORT	10	Unsigned 16 bit Integer
INTEGER	11	Signed 32 bit Integer
UINTEGER	12	Unsigned 32 bit Integer
LONG	13	Signed 64 bit Integer
ULONG	14	Unsigned 64 bit Integer
STRING	15	Text. It is a variable length, unbounded, Unicode string.
TIME	16	Absolute date-time to millisecond resolution.
FINETIME	17	Absolute date-time to picosecond resolution.
URI	18	Uniform Resource Identifier (address). It is a variable-length, unbounded Unicode string.
OBJECTREF	19	Object Reference
DIRECTION	129	MPS Direction
POSITION	130	MPS Position

4.3.2 MPS POSITION AND DIRECTION DATA TYPES [OPTIONAL]

4.3.2.1 Introduction

This subsection defines MPS Position and Direction data types and support types required in the definition of pointing constraints. These are consistent with those used within CCSDS Navigation data format Recommended Standards, and specifically the Pointing Request Message (PRM) (reference [D6]), but for MPS, to enable use of the MO Framework, they are defined explicitly in terms of MAL Attributes.

MPS Position and Direction data types are only required in the context of the following MPS data structures:

– Geometric Constraints (see 4.3.6.2.8);

– Triggers of type PositionTrigger and DirectionTrigger (see 4.3.7);

– Repetitions of type PositionRepetition and PointingRepetition (see 4.3.8).

As all of these are considered optional elements of the MPS information model, MPS Position and Direction data types are themselves optional.

Coordinate System

Some sub-types of MPS Position and Direction require the specification of the coordinate reference frame used. The set of allowed coordinate system values is specified in the PRM (reference [D6]) annex B2 or in the SANA registry as per reference [D6] annex E2, or a mission specific frame.

To allow for evolution, both of the set of standard coordinate systems defined within this registry and through mission specific extension, coordinate systems are not defined as an enumeration but represented as a MAL::String.

4.3.2.2 Position Data Type

Position

Name	***Position***	**Extends**	MAL::Composite	**SFP**	Abstract

CartesianPosition

Name	**CartesianPosition**		**Extends**	*Position*	**SFP**	2
Attribute	**Type**	**Nullable**	**Description**			
x	MAL::Double	No	Cartesian x coordinate defined in the given frame and with values of the given unit.			
y	MAL::Double	No	Cartesian y coordinate defined in the given frame and with values of the given unit.			
z	MAL::Double	No	Cartesian z coordinate defined in the given frame and with values of the given unit.			
frame	MAL::String	No	One of the coordinate reference frames as defined in reference [D6] annex B2, or in the SANA registry as per reference [D6] annex E2 or a mission specific frame.			
units	MAL::String	Yes	The distance unit name, as defined in reference [D6] annex D. Default = ‘km’.			

SurfacePosition

The unit may be defined here. Typically used to specify a coordinate on the surface of a celestial body. Optionally, the altitude above the surface (as defined by a reference ellipsoid) can also be specified.

Name	SurfacePosition		Extends	*Position*	SFP	3
Attribute	**Type**	**Nullable**	**Description**			
longitude	MAL::Double	No	Angular coordinate. May also represent azimuth.			
latitude	MAL::Double	No	Angular coordinate. May also represent elevation.			
frame	MAL::String	No	One of the coordinate reference frames as defined in reference [D6] annex B2, or in the SANA registry as per reference [D6] annex E2 or a mission specific frame.			
units	MAL::String	Yes	To be one of the Angle units as defined in reference [D6] annex D. Default = 'deg'.			
altitude	MAL::Double	Yes	Altitude above a reference ellipsoid (negative values allowed). Default = 0			
altitudeUnits	MAL::String	Yes	The distance unit name, as defined in reference [D6] annex D. Default = 'm'.			

OrbitFilePosition

Name	OrbitFilePosition		Extends	*Position*	SFP	4
Attribute	**Type**	**Nullable**	**Description**			
orbitFile	MAL::String	No	Name of or reference to a file containing an ODM (reference [D12]).			

OrbitalPosition

Name	OrbitalPosition		Extends	*Position*	SFP	5
Attribute	**Type**	**Nullable**	**Description**			
orbitNumber	MAL::Integer	No	Orbit number. Depending on the relativeOrbit flag, the orbit number may be absolute (since start of mission) or relative (to the orbital repeat cycle).			
relativeOrbit	MAL::Boolean	No	Flag indicating if the orbit number is absolute or relative to the orbital repeat cycle.			
orbitAngle	MAL::Double	No	Angle within orbit.			
units	MAL::String	Yes	The units used for orbitAngle, as defined in reference			

			[D6] annex D.

ObjectPosition

Name	ObjectPosition		Extends	*Position*	SFP	6
Attribute	**Type**	**Nullable**	**Description**			
object	MAL::String	No	Name or identifier of a celestial body as per reference [D6] annex E2, or a mission specific object.			

PositionReference

Name	PositionReference		Extends	*Position*	SFP	7
Attribute	**Type**	**Nullable**	**Description**			
reference	MAL::String	No	Name of a mission specifc position definition.			

4.3.2.3 Direction Data Type

Direction

Name	***Direction***	Extends	MAL::Composite	SFP	Abstract

CartesianDirection

Dimensionless unit vector. Either a direction in the base frame or in a secondary frame may be defined.

Name	CartesianDirection		Extends	*Direction*	SFP	8
Attribute	**Type**	**Nullable**	**Description**			
x	MAL::Double	No	Cartesian x coordinate defined in the given frame.			
y	MAL::Double	No	Cartesian y coordinate defined in the given frame.			
z	MAL::Double	No	Cartesian z coordinate defined in the given frame.			
frame	MAL::String	No	One of the coordinate reference frames as defined in reference [D6] annex B2, or in the SANA registry as per reference [D6] annex E2 or a mission specific frame.			

SphericalDirection

Based on azimuth and elevation, the unit may be defined here. Typically used to define a direction in a secondary frame. When used to specify a surface coordinate, this actually represents a {longitude, latitude} pair.

Name	SphericalDirection	Extends	*Direction*	SFP	9

Attribute	Type	Nullable	Description
azimuth	MAL::Double	No	Angular coordinate. May also represent longitude.
elevation	MAL::Double	No	Angular coordinate. May also represent latitude.
frame	MAL::String	No	One of the coordinate reference frames as defined in reference [D6] annex B2, or in the SANA registry as per reference [D6] annex E2 or a mission specific frame.
units	MAL::String	Yes	To be one of the Angle units as defined in reference [D6] annex D. Default = 'deg'.

RADecDirection

Based on celestial angular coordinates of right ascension and declination, the unit may be defined here.

Name	RADecDirection	Extends	*Direction*	SFP	10

Attribute	Type	Nullable	Description
ra	MAL::Double	No	Right Ascension: Celestial angular coordinate, measured eastward along the celestial equator.
dec	MAL::Double	No	Declination: Celestial angular coordinate, north or south of the celestial equator.
frame	MAL::String	No	One of the coordinate reference frames as defined in reference [D6] annex B2, or in the SANA registry as per reference [D6] annex E2 or a mission specific frame.
units	MAL::String	Yes	To be one of the Angle units as defined in reference [D6] annex D. Default = 'deg'.

RevolutionDirection

Based on fixed rotation about an axis, the direction is defined by an angle within a single revolution.

Name	RevolutionDirection	Extends	*Direction*	SFP	11

Attribute	Type	Nullable	Description
revolutionAngle	MAL::Double	No	Angle within a revolution.
units	MAL::String	Yes	To be one of the Angle units as defined in reference [D6] annex D. Default = 'deg'.

NamedTargetDirection

Name	**NamedTargetDirection**		Extends	*Direction*	SFP	12
Attribute	**Type**	**Nullable**	**Description**			
namedTarget	MAL::String	No	Name or identifier of a catalogued celestial object or a mission specific object.			

DirectionReference

Name	**DirectionReference**		Extends	*Direction*	SFP	13
Attribute	**Type**	**Nullable**	**Description**			
reference	MAL::String	No	Name of a mission specific direction definition.			

4.3.2.4 Physical Value Data Types

Physical Value

PhysicalValue is an abstract base type for the specific value types defined below. Only specific value types are used in the pointing constraint definitions below.

Name	***PhysicalValue***		Extends	MAL::Composite	SFP	Abstract
Attribute	**Type**	**Nullable**	**Description**			
value	MAL::Double	No	Physical value.			
units	MAL::String	Yes	Optional unit. The unit type depends on the specific value type, as defined in reference [D6] annex D.			

For MPS, three specific physical value types (as defined in reference [D6] annex D) are used in the context of geometric constraints on position or pointing:

Angle

Has units of type Angle.

Name	**Angle**		Extends	*PhysicalValue*	SFP	14

AngularVelocity

Has units of type AngularVelocity.

Name	AngularVelocity	Extends	*PhysicalValue*	SFP	15

Distance

Has units of type Distance.

Name	Distance	Extends	*PhysicalValue*	SFP	16

4.3.3 EXPRESSIONS

Expression

When entering MPS data, it is often not possible to provide an absolute value for a required attribute. Instead, it is necessary to provide a calculation to be performed at run time that supplies the value. These calculations are defined as **expressions** of a specified data type. The data type can be any defined ArgType (see 4.3.1), which may be any MAL Attribute type, Position, or Direction.

The expressions are themselves text strings which comprise a sequence of operands and operators. Operands may be literals or references to objects and their attributes or arguments, as defined within the MPS information model.

Name	Expression		Extends	MAL::Composite	SFP	17
Attribute	**Type**	**Nullable**	**Description**			
type	ArgTypeEnum	No	Enumeration specifying the data type of the result of the expression.			
value	MAL::Attribute	No	Providing the ArgType is a MAL Attribute type, this field may be used to hold a simple literal value or the evaluated result of the expression.			
expressionLanguage	MAL::String	No	Defines the expression language used to specify the expression.			
expression	MAL::String	No	The text of the expression.			

4.3.4 ADDITIONAL MPS DATA TYPES

Slider

Used to indicate a relative position with respect to an MPS object, such as a **planning activity** where 0 represents the start and 1 the end of the activity. The slider is a real number that can represent any point between these two extremes.

Name	**Slider**		**Extends**	MAL::Composite	**SFP**	18
Attribute	**Type**	**Nullable**	**Description**			
position	MAL::Float	No	Relative point between the start and end of an MPS object, where 0 represents the start and 1 represents the end.			

StateDef

Status values may be represented as enumerated Integers, but the enumeration is not defined by the Recommended Standard, but in the context of planning configuration data. StateDefs hold the definitions of the text labels associated with specific status values.

Name	**StateDef**		**Extends**	MAL::Composite	**SFP**	19
Attribute	**Type**	**Nullable**	**Description**			
value	MAL::Integer	No	Enumerated value of the Status			
state	MAL::String	No	Text label associated with the enumerated value.			

TimeWindow

Represents a specific period of time, specified as two Expressions of type Time defining the start and end of the TimeWindow.

Name	**TimeWindow**		**Extends**	MAL::Composite	**SFP**	20
Attribute	**Type**	**Nullable**	**Description**			
start	Expression <MAL::Time>	No	Start time of the time window.			
end	Expression <MAL::Time>	No	End time of the time window.			

EventWindow

Represents a specific period relative to two events that mark the start and end of the EventWindow.

Name	EventWindow		Extends	MAL::Composite	SFP	21
Attribute	**Type**	**Nullable**	**Description**			
startEvent	Expression <MAL::ObjectRef>	No	Object Type: EventInstance. The start of the event window is relative to the referenced startEvent.			
startOffset	Expression <MAL::Duration>	No	The start of the event window is offset by the defined time period from the startEvent.			
endEvent	Expression <MAL::ObjectRef>	No	Object Type: EventInstance. The end of the event window is relative to the referenced endEvent.			
endOffset	Expression <MAL::Duration>	No	The end of the event window is offset by the defined time period from the endEvent.			

DefListEntry

Used in the context of the MPS Plan Information Management service, this holds a list of definitions for a specified type of MPS data item, together with their definitions.

Name	DefListEntry		Extends	MAL::Composite	SFP	22
Attribute	**Type**	**Nullable**	**Description**			
definitionID	MAL::ObjectRef	No	Object Type: ActivityDefinition | EventDefinition | *Resource* | RequestDefinition Item Definition (key and version)			
description	MAL::String	No	Description of the item			

4.3.5 ARGUMENTS

4.3.5.1 General

The **instance** objects of several MPS data items have associated arguments that can be used to parameterize the object. The set of arguments is defined in the associated **definition** object, with the argument values forming part of the **instance** object. Arguments apply to ActivityInstance, EventInstance, RequestInstance and Functions.

ArgDef

The **definition** of an argument is an ArgDef, a set of which may be contained within the definition MO object of a **planning event**, **planning activity**, or **planning request**. This defines the name and data type of the argument. Depending on the data type, the ArgDef may require additional type specific attributes to support data validation. SubTypes are identified for Numeric, String, and Status arguments.

It should be noted that if the argument is an array, then all values of the array are of the same type, as defined in argType.

Name	ArgDef		Extends	MAL::Composite	SFP	23
Attribute	**Type**	**Nullable**	**Description**			
argName	MAL::Identifier	No	Name of the argument			
description	MAL::String	No	Extended description of the argument			
argType	ArgTypeEnum	No	Enumeration specifying the data type of the argument.			
argUnits	MAL::String	Yes	Units that the argument value is expressed in, as defined in reference [D6] annex D.			
isArray	MAL::Boolean	No	If True, indicates that the argument is an array of values of type ArgType.			
validationData	ValidationDetails	Yes	Optional. Specifies the allowed range of values for the Argument, with concrete subtypes specific to the data type of the Argument.			

Argument

The **instance** of an argument is an Argument, a set of which may be contained within the instance MO object of a **planning event** or **planning activity** or within a **planning request**. This comprises the name and value of the argument, corresponding to the set of arguments defined in the ArgDef. Argument values are represented as a MAL Attribute of appropriate data type. As there is no equivalent MAL::Attribute type for Position or Direction, values of these types are represented as a MAL::String containing a literal in the defined expression format for these types.

It should be noted that if the argument is an array (count > 1) then all values of the array are of the same type, as defined by the argType of the associated ArgDef.

Name	Argument		Extends	MAL::Composite	SFP	24
Attribute	**Type**	**Nullable**	**Description**			
argName	MAL::Identifier	No	Name of the argument			
count	MAL::Integer	Yes	If argument is an array, count of the number of elements in the array.			
argValues	List <MAL::Attribute>	No	Argument value (or values if it is an array). MAL Attribute type must match the ArgType of the ArgDef. Position and Direction values are represented as a MAL::String.			

ArgSpec

In the case of the **planning activity**, there is also an ArgSpec, a set of which may be contained within the ActivityDetails structure embedded within a **planning request** or parent **planning activity** definition. The ArgSpec defines how to derive the value of an Argument when instantiating it at run-time. The ArgSpec attribute is an Expression, the result of which must be a value matching the defined ArgType.

Name	ArgSpec		Extends	MAL::Composite	SFP	25
Attribute	**Type**	**Nullable**	**Description**			
argName	MAL::Identifier	No	Name of the argument			
count	MAL::Integer	Yes	If argument is an array, count of the number of elements in the array.			
argSpecs	List <Expression>	No	Expression that can be evaluated at run-time to provide argument value(s) of appropriate data type.			

4.3.5.2 Validation Details

ValidationDetails

Name	*ValidationDetails*	Extends	MAL::Composite	SFP	Abstract

NumericRange

Concrete sub-type of *ValidationDetails* that provides additional attributes to support data validation for numeric data types.

Name	NumericRange		Extends	*ValidationDetails*	SFP	26
Attribute	**Type**	**Nullable**	**Description**			
min	MAL::Double	No	Minimum value of the argument			
max	MAL::Double	No	Maximum value of the argument			
precision	MAL::Short	No	Precision of the argument.			

StringPatterm

Concrete sub-type of *ValidationDetails* that provides additional attributes to support data validation for the string data type.

Name	StringPattern		Extends	*ValidationDetails*	SFP	27
Attribute	**Type**	**Nullable**	**Description**			
maxLength	MAL::Integer	No	Maximum length of the string (characters).			

regex	MAL::String	No	A 'regular expression' or sequence of characters defining a character pattern that the string value must match.

StatusValues

Concrete sub-type of *ValidationDetails* that provides additional attributes to support data validation and interpretation for integer type arguments that are effectively enumerated Statuses.

Name	**StatusValues**		**Extends**	*ValidationDetails*	**SFP**	28
Attribute	**Type**	**Nullable**	**Description**			
allowedValues	List <StateDef>	No	Set of allowed State definitions (see 4.3.4), comprising the enumerated value and an associated text label.			

4.3.6 CONSTRAINTS

4.3.6.1 General

Constraint

Name	***Constraint***	**Extends**	MAL::Composite	**SFP**	Abstract

ConstraintNode

Multiple planning constraints can be combined using a ConstraintNode. The ConstraintNode specifies the logical operation (AND or OR) to be used when combining a set of constraints together. As the ConstraintNode is itself defined as a sub-type of Constraint, it is possible to construct a tree of ConstraintNodes using different logical operators.

Name	**ConstraintNode**		**Extends**	*Constraint*	**SFP**	29
Attribute	**Type**	**Nullable**	**Description**			
operator	LogicOpEnum	No	Enumeration specifying the logic for combining multiple Boolean conditions together. One of {AND, OR}. Default = AND			
negate	MAL::Boolean	No	Specifies whether the result of combining the Constraints is to be inverted (NOT function). Default = False			
constraints	List <Constraint>	No	The set of Constraints to be combined.			

LogicOpEnum

Name	LogicOpEnum		SFP	30
Enumeration	**Value**	**Description**		
AND	1	Logical AND		
OR	2	Logical OR		

4.3.6.2 Conditional Constraints

4.3.6.2.1 General

ConditionalConstraint

Name	*ConditionalConstraint*	Extends	*Constraint*	SFP	Abstract

4.3.6.2.2 Constraint Expression

ConstraintExpression

All types of constraint can be considered conditions that are either met or not met when a **planning activity** is placed in a Plan. They can therefore be specified as a potentially complex Boolean expression that combines references to the arguments and attributes of objects in the MPS information model using operators of various types (arithmetic, comparative, logical, string, temporal, and geometric). The expression must evaluate to TRUE for the constraint to be met.

As introduced in 4.3.3, this Recommended Standard does not define a full expression language capable of supporting such complex Boolean expressions. It does, however, support the use of externally defined expression languages. The ConstraintExpression type allows for the use of such an expression language to define any type of constraint, providing communicating entities all have the capability to evaluate that expression language.

Name	ConstraintExpression		Extends	*ConditionalConstraint*	SFP	31
Attribute	**Type**	**Nullable**	**Description**			
constraint	Expression <MAL::Boolean>	No	Potentially complex conditional expression that must evaluate to TRUE for the constraint to be met.			

4.3.6.2.3 Temporal Constraints [Optional]

TemporalConstraint

Temporal constraints impose a restriction on when a **planning activity** can appear in a plan. The abstract type *TemporalConstraint* identifies the **planning activity** that is subject to the constraint, while concrete sub-types allow the specification of three different types of temporal constraint:

- TimeConstraint: the time at which the **planning activity** is to be planned;
- TimeWindowConstraint: a time window within which the **planning activity** is to be planned;
- DurationConstraint: a restriction on the duration of the **planning activity** in the plan.

Name	***TemporalConstraint***		**Extends**	*ConditionalConstraint*	**SFP**	Abstract
Attribute	**Type**	**Nullable**	**Description**			
activityRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance. Identifies the constrained **planning activity.** If omitted the activity containing the constraint is assumed.			

TimeConstraint

Name	**TimeConstraint**		**Extends**	*TemporalConstraint*	**SFP**	32
Attribute	**Type**	**Nullable**	**Description**			
time	Expression <MAL::Time>	No	The time at which the **planning activity** must be planned.			
timeRef	Slider	No	The point in the duration of the **planning activity** that is time constrained. 0: the start of the **planning activity** 1: the end of the **planning activity**			

TimeWindowConstraint

Name	**TimeWindowConstraint**		**Extends**	*TemporalConstraint*	**SFP**	33
Attribute	**Type**	**Nullable**	**Description**			
startRef	Slider	No	The point in the duration of the activity that is constrained to be after the start time of the time window. Although typically the start of the activity (0), this can be any point up to the end of the activity (1).			
endRef	Slider	No	The point in the duration of the activity that is			

			constrained to be before the end time of the time window. Although typically the end of the activity (1), this can be any point up to the start of the activity (0).
timeWindows	List <TimeWindow>	No	The [set of] TimeWindows within which the activity must be placed on the Plan.

DurationConstraint

Name	**DurationConstraint**		**Extends**	*TemporalConstraint*	**SFP**	34
Attribute	**Type**	**Nullable**	**Description**			
minDuration	Expression <MAL::Duration>	No	Specifies the minimum duration of the **planning activity**			
maxDuration	Expression <MAL::Duration>	No	Specifies the maximum duration of the **planning activity**			

4.3.6.2.4 Sequential Constraint [Optional]

Sequential Constraint

Sequential constraints impose a restriction on the order of **planning activities** in a Plan with respect to both other **planning activities** and **planning events**.

Two objects are identified: the predecessor which must be followed in the Plan by the successor. While either the predecessor or successor may be a **planning event**, it is not possible to specify a sequential constraint between two **planning events**: one or both must be a **planning activity**.

Name	**SequentialConstraint**		**Extends**	*ConditionalConstraint*	**SFP**	35
Attribute	**Type**	**Nullable**	**Description**			
predecessor	Expression <MAL::ObjectRef>	No	Object Type: ActivityInstance | EventInstance Identifies the **planning activity** or **planning event** that must occur first on the Plan.			
successor	Expression <MAL::ObjectRef>	No	Object Type: ActivityInstance | EventInstance Identifies the **planning activity** or **planning event** that must follow the predecessor on the Plan.			
predecessorRef	Slider	No	Point on the predecessor that must be followed by the successor			
successorRef	Slider	No	Point on the successor that must follow the predecessor			
minOffset	Expression <MAL::Duration>	No	Minimum period between the specified points on the predecessor and successor.			
maxOffset	Expression	No	Maximum period between the specified points on the			

	<MAL::Duration>		predecessor and successor.

4.3.6.2.5 Exclusion Constraint [Optional]

Exclusion Constraint

An exclusion constraint specifies a set of 2 or more **planning activities** or **planning events** that cannot occur concurrently in a Plan. As only **planning activities** can be excluded, at least one of the set must be a **planning activity**. Excluded objects are specified by **definition** [class] rather than **instance**, the exclusion applying to all instances of the class. Exclusion implies no overlap between the excluded items.

Name	**ExclusionConstraint**		**Extends**	*ConditionalConstraint*	**SFP**	36
Attribute	**Type**	**Nullable**	**Description**			
classRefs	List <MAL::ObjectRef>	No	Object Type: ActivityDefinition | EventDefinition Specifies the **definition** (class) of excluded **planning activities** and **planning events**.			

4.3.6.2.6 Resource and Argument Constraints [Optional]

ArgumentConstraint

An argument constraint may be associated with a **planning activity** to restrict when it can be planned, based on the value of an argument of the **planning activity** itself or a related **planning event**.

Name	**ArgumentConstraint**		**Extends**	*ConditionalConstraint*	**SFP**	37
Attribute	**Type**	**Nullable**	**Description**			
activityRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance. Identifies the **planning activity** for which the argument constraint applies. If omitted the activity containing the constraint is assumed.			
objectRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance | EventInstance Identifies the Object (**planning activity** or **planning event**) whose argument is to be referenced. If omitted the activity containing the constraint is assumed.			
argName	MAL::Identifier	No	Identifies the specific argument of the referenced Object whose value is to be compared			
comparator	ExpressionOperatorEnum	No	Comparisson operator, which may be one of: =, !=, >, >=, <, <=, contains, icontains The contains operator only applies to strings and may be case sensitive or insensitive.			

value	MAL::Attribute	No	Value (of same type as the referenced Argument) to be compared against.

ExpressionOperatorEnum

Name	**ExpressionOperatorEnum**		**SFP**	38
Enumeration	**Value**	**Description**		
EQUAL	1	=		
DIFFER	2	!=		
GREATER	3	>		
GREATER_OR_EQUAL	4	>=		
LESS	5	<		
LESS_OR_EQUAL	6	<=		
CONTAINS	7	Case sensitive containment (Strings only)		
ICONTAINS	8	Case insensitive containment (Strings only)		

ResourceConstraint

Name	***ResourceConstraint***		**Extends**	*ConditionalConstraint*	**SFP**	Abstract
Attribute	**Type**	**Nullable**	**Description**			
activityRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance. Identifies the **planning activity** for which the resource constraint applies. If omitted the activity containing the constraint is assumed.			
resourceRef	MAL::ObjectRef	No	Object Type: *Resource* Identifies the **planning resource** that is constrained for the duration of the **planning activity**.			
comparator	ExpressionOperatorEnum	No	comparison operator, which may be one of: =, !=, >, >=, <, <=, contains, icontains The contains operator only applies to strings and may be case sensitive or insensitive.			

SimpleResourceConstraint

The simple resource constraint must be satisfied for the duration of the referenced **planning activity**.

Name	**SimpleResourceConstraint**		**Extends**	*ResourceConstraint*	**SFP**	39
Attribute	**Type**	**Nullable**	**Description**			

value	MAL::Attribute	No	Value (of same type as the referenced Resource) to be compared against.

ComplexResourceConstraint

In the [simple] resource constraint, the value of the referenced **planning resource** is constrained against a single value for the entire duration of the referenced **planning activity**.

With the complex resource constraint, the period over which the constraint applies can be customized relative to the referenced **planning activity**; and the value against which the referenced **planning resource** is constrained can be specified as a relative resource profile which evolves over time.

The attributes of the complex resource constraint extend or modify those of the [simple] resource constraint as follows:

Name	**ComplexResourceConstraint**		**Extends**	ResourceConstraint	**SFP**	40
Attribute	**Type**	**Nullable**	**Description**			
startRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the start of the constraint period relates.			
endRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the end of the constraint period relates.			
startOffset	Expression <MAL::Duration>	No	Offset from startRef that specifies the start of the constraint period.			
endOffset	Expression <MAL::Duration>	No	Offset from endRef that specifies the end of the constraint period.			
valueProfile	RelativeResourceProfile	No	ResourceProfile specifying an evolving value over time against which the value of the **planning resource** is to be compared. (See 4.2.4.3.)			

4.3.6.2.7 Function Constraint [Optional]

Function Constraint

Function constraints make use of an external **custom function** to determine whether or not a constraint is satisfied. Available functions must be pre-defined (see 4.2.9) to allow them to be referenced in a function constraint.

As for complex resource constraints, the period over which the function constraint applies is specified relative to the referenced **planning activity**.

Name	**FunctionConstraint**		Extends	*ConditionalConstraint*	SFP	41
Attribute	**Type**	**Nullable**	**Description**			
activityRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance. Identifies the **planning activity** for which the function constraint applies. If omitted the activity containing the constraint is assumed.			
startRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the start of the constraint period relates.			
endRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the end of the constraint period relates.			
startOffset	Expression <MAL::Duration>	No	Offset from startRef that specifies the start of the constraint period.			
endOffset	Expression <MAL::Duration>	No	Offset from endRef that specifies the end of the constraint period.			
function	FunctionDetails	No	Specifies the Function to be applied and its set of input arguments.			

4.3.6.2.8 Geometric Constraints [Optional]

GeometricConstraint

Name	***GeometricConstraint***		Extends	*ConditionalConstraint*	SFP	Abstract
Attribute	**Type**	**Nullable**	**Description**			
activityRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance. Identifies the **planning activity** for which the geometric constraint applies. If omitted the activity containing the constraint is assumed.			
startRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the start of the constraint period relates.			
endRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the end of the constraint period relates.			
startOffset	Expression <MAL::Duration>	No	Offset from startRef that specifies the start of the constraint period.			
endOffset	Expression <MAL::Duration>	No	Offset from endRef that specifies the end of the constraint period.			

PositionConstraint

Sub-type of geometric constraint expressed in terms of a specified Position and a tolerance. The tolerance is defined as a sphere around the specified position, expressed as a distance or angle. It should be noted that the position itself can be expressed using any of the concrete position sub-types, including orbital and surface positions. The use of a constraint expressed by OrbitalPosition is particularly relevant for Earth observation satellites with a repetitive ground track and on-board position based scheduler. The position can also specified as an expression.

Name	**PositionConstraint**		**Extends**	*GeometricConstraint*	**SFP**	42
Attribute	**Type**	**Nullable**	**Description**			
position	Expression <Position>	No	Specifies the required position expressed using any concrete position type.			
tolerance	Expression <MAL::Double>	No	Specifies the maximum distance or angle from the required position that satisfies the constraint, effectively defining a sphere around the required position.			
units	MAL::String	Yes	Optional. The tolerance unit name, as defined in reference [D6] annex D. Default = 'km', but 'deg' is more relevant for an OrbitalPosition.			

PointingConstraint

Pointing constraints impose a restriction on a **planning activity** appearing in a Plan, based on the pointing direction of a physical object, such as a spacecraft or instrument.

As with the Direction data types (see 4.3.2.3), pointing constraints are consistent with the pointing templates defined for use within CCSDS Navigation data format Recommended Standards, and specifically the Pointing Request Message (PRM) (reference [D6]). PointingConstraint is a concrete sub-type of *GeometricConstraint* that includes attributes common to all pointing templates. The pointing template itself is then identified as an attribute and any additional arguments applicable to the template are provided as a list of name-value pairs.

Name	**PointingConstraint**		**Extends**	*GeometricConstraint*	**SFP**	43
Attribute	**Type**	**Nullable**	**Description**			
pointingFrame	MAL::String	Yes	Optional. Frame to which the pointing constraint applies. One of the spacecraft body frames defined in reference [D6] annex B2, or in the SANA registry as per reference [D6] annex E2, section 1.7 reference [19] or a mission specific frame. Default frame is the spacecraft frame or any other mission specific default frame.			

boresight	Direction	No	Direction in any spacecraft frame.
boresightMargin	Angle	Yes	Defines an optional cone region around the boresight, allowing a margin for application of the pointing constraint. Default = 0.0
phaseAngleMargin	Angle	Yes	Defines an optional rotation around the boresight, w.r.t. the default phase angle, allowing a margin for application of the pointing constraint. Default = 0.0
unconstrainedPhaseAngle	MAL::Boolean	Yes	If TRUE no constraint will apply to the phaseAngle. The phaseAngleMargin attribute will be ignored in this case. Default = FALSE
pointingTemplate	MAL::String	No	One of the pointing templates defined in the PRM with the XML available in the SANA registry reference [D6] annex E2, or a mission specific pointing template.
pointingArguments	List <MAL::NamedValue>	Yes	The argument list is consistent with the referenced template by name. Physical values are represented as a pair of arguments containing the value and units respectively. Position and Direction type arguments are represented as strings containing the literal value.

The following table summarizes currently defined pointing templates and their additional arguments. Not all templates require additional arguments.

Pointing Template	Argument	Type
Inertial Pointing	target phaseAngle offsetAngle angularRate	Direction Angle Angle AngularVelocity
Sun Pointing	phaseAngle offsetAngle angularRate	Angle Angle AngularVelocity
Track with Inertial Direction Yaw Steering	targetBody phaseAngle	Position Angle
Track with Power Optimized Yaw Steering	targetBody	Position
Nadir with Power Optimized Yaw Steering		
Nadir with Ground Track Aligned Yaw Steering		
Nadir with Orbital Pole Aligned Yaw Steering		
Limb Pointing with Power Optimized Yaw Steering	surface dirVector height	MAL::String Direction Distance

Limb Pointing with Inertial Direction Yaw Steering	surface dirVector height phaseAngle	MAL::String Direction Distance Angle
Velocity Pointing with Orbital Pole Yaw Steering	phaseAngle	Angle

RevolutionConstraint

Specifies a range of revolution angles for a rotating spacecraft.

Name	**RevolutionConstraint**		**Extends**	*GeometricConstraint*	**SFP**	44
Attribute	**Type**	**Nullable**	**Description**			
revolutionAngle	Expression <MAL::Double>	No	Angle of revolution			
tolerance	Expression <MAL::Double>	No	Tolerance in the angle of revolution			
units	MAL::String	Yes	Optional. The angular unit name, as defined in reference [D6] annex D. Default = 'deg'.			

DistanceConstraint

Specifies a range of distances between two physical objects (the observer and the target).

Name	**DistanceConstraint**		**Extends**	*GeometricConstraint*	**SFP**	45
Attribute	**Type**	**Nullable**	**Description**			
observer	Expression <Position>	No	Position of the observer [Object1]			
target	Expression <Position>	No	Position of the target [Object2]			
minDistance	Expression <MAL::Double>	No	Minimum distance between observer and target			
maxDistance	Expression <MAL::Double>	No	Maximum distance between observer and target			
units	MAL::String	Yes	Optional. The distance unit name, as defined in reference [D6] annex D. Default = 'km'.			

AngleConstraint

Specifies a range of values for the angle subtended between three physical objects. The constrained angle is that subtended at the central object by target objects 1 and 2.

Name	**AngleConstraint**		Extends	*GeometricConstraint*	SFP	46
Attribute	**Type**	**Nullable**	**Description**			
centreObject	Expression <Position>	No	Position of the central object.			
targetObject1	Expression <Position>	No	Position of target object 1.			
targetObject2	Expression <Position>	No	Position of target object 2.			
minAngle	Expression <MAL::Double>	No	Minimum angle subtended at the central object by target objects 1 and 2.			
maxAngle	Expression <MAL::Double>	No	Maximum angle subtended at the central object by target objects 1 and 2.			
units	MAL::String	Yes	Optional. The angular unit name, as defined in reference [D6] annex D. Default = 'deg'.			

4.3.6.3 Effects [Optional]

Effect

Name	***Effect***		Extends	*Constraint*	SFP	Abstract
Attribute	**Type**	**Nullable**	**Description**			
activityRef	Expression <MAL::ObjectRef>	Yes	Object Type: ActivityInstance | ActivityDefinition. Identifies the **planning activity** for which the resource effect applies. May be either an ActivityDefinition or an ActivityInstance. If omitted the activity containing the effect is assumed.			
resourceRef	MAL::ObjectRef	No	Object Type: *Resource* Identifies the **planning resource** that is constrained for the duration of the **planning activity**.			

Simple Effect

A simple effect applies the defined operation on the specified **planning resource** at the time relative to the **planning activity** defined by timeRef+timeOffset.

Name	**SimpleEffect**		Extends	*Effect*	SFP	47
Attribute	**Type**	**Nullable**	**Description**			
timeRef	Slider	No	The point in the duration of the **planning activity** to which the time of the Effect is relative. 0: the start of the **planning activity**			

			1: the end of the **planning activity**
timeOffset	Expression <MAL::Duration>	No	Offset from timeRef that specifies the time at which the Effect is to be applied.
operator	EffectOperationEnum	No	Operation to be performed on the **planning resource**. One of: SET, INCREMENT, DECREMENT. Increment and decrement are only applicable to numeric data types.
value	MAL::Attribute	No	The value that the **planning resource** is to be set to if the Effect operator is SET; or to be incremented/decremented by if it is INCREMENT or DECREMENT.

EffectOperationEnum

Name	**EffectOperationEnum**	**SFP**	48
Status	**Value**	**Description**	
SET	1	Set to specified value	
INCREMENT	2	Increment by specified value	
DECREMENT	3	Decrement by specified value	

ComplexEffect

In the simple effect, the value of the impacted **planning resource** is set to the specified value at a single point in time.

With the complex effect, the value of the impacted **planning resource** can be evolved over a specified time period in accordance with a defined RelativeResourceProfile.

Name	**ComplexEffect**	**Extends**	*Effect*	**SFP**	49
Attribute	**Type**	**Nullable**	**Description**		
startRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the start of the effect period relates.		
endRef	Slider	No	Identifies the point in the duration of the referenced **planning activity** to which the end of the effect period relates.		
startOffset	Expression <MAL::Duration>	No	Offset from startRef that specifies the start of the effect period.		
endOffset	Expression <MAL::Duration>	No	Offset from endRef that specifies the end of the effect period.		
operator	EffectOperationEnum	No	Operation to be performed on the **planning resource**. One of: SET, INCREMENT, DECREMENT. Increment and decrement are only applicable to		

			numeric data types.
Value	RelativeResourceProfile	No	Resource profile specifying an evolving value to which the value of the **planning resource** is to be set. (See 4.2.4.3.)

4.3.7 TRIGGERS

4.3.7.1 General

Trigger

All sub-classes of Trigger include the time at which they are predicted to occur (in advance of execution); and, where applicable, the time at which they actually occurred (post execution).

Name	***Trigger***		**Extends**	MAL::Composite	**SFP**	Abstract
Attribute	**Type**	**Nullable**	**Description**			
time	MAL::Time	No	Predicted or actual time of Trigger. The predicted time may evolve during the planning process up to the time of execution. The actual time is only available post execution, and hence can only be provided by a plan execution function.			

4.3.7.2 Temporal Triggers

TimeTrigger

Sub-type of Trigger based on time. The trigger time is the specified constraint, and will usually match the predicted time on the base class during the planning process, but the actual time could still be slightly different post-execution.

Name	**TimeTrigger**		**Extends**	*Trigger*	**SFP**	50
Attribute	**Type**	**Nullable**	**Description**			
triggerTime	MAL::Time	No	Planned time of Trigger.			

4.3.7.3 Position Triggers [Optional]

PositionTrigger

Sub-type of Trigger based on position. Depending on the coordinate type of position used, a margin may be specified in terms of distance from the specified position. This is not relevant for an OrbitalPosition, as an orbiting spacecraft would pass through the specified angle.

Name	PositionTrigger		Extends	*Trigger*	SFP	51
Attribute	**Type**	**Nullable**	**Description**			
triggerPosition	Position	No	Planned position of Trigger.			
distanceMargin	Distance	Yes	Defines a sphere around the trigger position within which a position is considered to meet the trigger condition.			

4.3.7.4 Direction Triggers [Optional]

DirectionTrigger

Sub-type of Trigger based on pointing. Depending on the coordinate type of direction used, a margin may be specified in terms of angle from the specified direction. This is not relevant for a RevolutionDirection, as a rotating spacecraft or instrument would pass through the specified angle.

Name	DirectionTrigger		Extends	*Trigger*	SFP	52
Attribute	**Type**	**Nullable**	**Description**			
triggerDirection	Direction	No	Planned direction of Trigger.			
angleMargin	Angle	Yes	Defines a circle around the trigger direction within which a direction is considered to meet the trigger condition.			

4.3.7.5 Angle Trigger [Optional]

Angle Trigger

Sub-type of Trigger based on the angle subtended between three physical objects. The trigger angle is that subtended at the central object by target objects 1 and 2.

Name	AngleTrigger		Extends	*Trigger*	SFP	53
Attribute	**Type**	**Nullable**	**Description**			
centreObject	Expression <Position>	No	Position of the central object. The trigger angle is that subtended at the central object by target objects 1 and 2.			
targetObject1	Expression <Position>	No	Position of target object 1.			
targetObject2	Expression <Position>	No	Position of target object 2.			
angleMargin	Angle	No	The trigger occurs if the angle is within ±			

			angleMargin of the trigger angle.

4.3.7.6 Event Triggers

Event Trigger

Sub-type of Trigger based on **planning event**.

Name	**EventTrigger**		**Extends**	*Trigger*	**SFP**	54
Attribute	**Type**	**Nullable**	**Description**			
triggerEvent	MAL::ObjectRef <EventInstance>	No	Reference to an EventInstance			
timeOffset	MAL::Duration	No	Time offset from the EventInstance			

4.3.8 REPETITIONS

4.3.8.1 General

Repetition

A **repetition** is used to specify the repeated instantiation of a [set of] **planning activities**. Multiple sub-types of Repetition are defined to support the specification of repeat cycles by different criteria. It can be used in the context of a **planning request** to specify a standing order for repeated execution of the [set of] planning activities.

In the context of an ActivityNode embedded within a **planning request** (see 4.2.2.3), it is possible to nest one Repetition inside another, enabling the specification of complex repetitive sequences of activities.

All sub-types have the following attributes:

Name	***Repetition***		**Extends**	MAL::Composite	**SFP**	Abstract
Attribute	**Type**	**Nullable**	**Description**			
count	MAL::Integer	Yes	Maximum number of repeat cycles/instances [optional]. If not specified there is no limit to the number of repetitions.			
timeWindow	TimeWindow	Yes	Time period over which the repetition is applicable [optional]. If not specified repetition continues indefinitely.			
separationType	SeparationTypeEnum	No	Specifies whether the repetition interval is **Relative** to the previous occurrence, or **Absolute** for all occurrences.			

SeparationTypeEnum

Name	SeparationTypeEnum		SFP	55
Enumeration	**Value**	**Description**		
RELATIVE	1	Tolerance on separation is only considered between any two occurrences		
ABSOLUTE	2	Tolerance on separation applies to a multiple of the separation from the initial occurrence.		

4.3.8.2 Temporal Repetition

TemporalRepetition

A sub-type of Repetition based on time.

Name	TemporalRepetition		Extends	*Repetition*	SFP	56
Attribute	**Type**	**Nullable**	**Description**			
initialTime	Expression <MAL::Time>	No	Nominal time of first occurrence.			
separation	Expression <MAL::Duration>	No	The required time interval between occurrences.			
tolerance	Expression <MAL::Duration>	No	The allowed tolerance (+/-) in the required time between occurrences, the interpretation of which is dependent on the separationType.			

4.3.8.3 Location Repetition [Optional]

LocationRepetition

A sub-type of Repetition based on Position. Separate concrete sub-types provide for repetitions based on generic position and orbital position.

Name	*LocationRepetition*	Extends	*Repetition*	SFP	Abstract

PositionRepetition

Name	PositionRepetition		Extends	*LocationRepetition*	SFP	57
Attribute	**Type**	**Nullable**	**Description**			
initialPosition	Expression <Position>	No	Nominal position of first occurrence.			
repetitionDirection	Expression <Direction>	No	Direction of repetition.			

separation	Expression <MAL::Double>	No	The required Distance between occurrences.
tolerance	Expression <MAL::Double>	No	The allowed tolerance (+/-) in the required distance between occurrences, the interpretation of which is dependent on the separationType.
units	MAL::String	Yes	The units used for separation and tolerance, as defined in reference [D6] annex D.

OrbitRepetition

A sub-type of Repetition based on the orbital cycle.

Name	**OrbitRepetition**	**Extends**	*LocationRepetition*	**SFP**	58
Attribute	**Type**	**Nullable**	**Description**		
orbitNumber	Expression <MAL::Long>	No	Orbit number for the first occurrence. Depending on the relativeOrbit flag, the orbit number may be absolute (since start of mission) or relative (to the orbital repeat cycle).		
relativeOrbit	MAL::Boolean	No	Flag indicating if the orbit number is absolute or relative to the orbital repeat cycle.		
orbitSeparation	Expression <MAL::Long>	No	The required number of orbits separation between occurrences. If orbitNumber is Relative and the required repetition is once per repeat cycle, this is the number of orbits in the repeat cycle, but the value 0 may also be used.		
angleSeparation	Expression <MAL::Double>	No	The required angular separation between occurrences. This allows for multiple repetitions within an orbit. The value 0 indicates only one occurrence within the orbit.		
orbitAngle	Expression <MAL::Double>	No	The required position of the first occurrence within the orbit expressed as an angle.		
tolerance	Expression <MAL::Double>	No	The allowed tolerance (+/-) in the required orbital angle.		
units	MAL::String	Yes	The units used for orbitAngle, angularSeparation, and tolerance, as defined in reference [D6] annex D.		

4.3.8.4 Pointing Repetition [Optional]

PointingRepetition

A sub-type of Repetition based on Pointing. Concrete sub-types provide for repetition based on direction and revolutions.

Name	***PointingRepetition***	**Extends**	*Repetition*	**SFP**	Abstract

DirectionRepetition

A sub-type of Repetition based on direction, which supports the specification of astronomical surveys.

Name	DirectionRepetition		Extends	*PointingRepetition*	SFP	59
Attribute	**Type**	**Nullable**	**Description**			
initialDirection	Expression <Direction>	No	Nominal direction of first occurrence.			
targetDirection	Expression <Direction>	No	Specifies the direction of repetition as line connecting the initial and target directions.			
separation	Expression <MAL::Double>	No	The required angle between occurrences.			
tolerance	Expression <MAL::Double>	No	The allowed tolerance (+/-) in the required angle between occurrences, the interpretation of which is dependent on the separationType.			
units	MAL::String	Yes	The units used for separation and tolerance, as defined in reference [D6] annex D.			

RevolutionRepetition

A sub-type of Repetition based on the revolutions of a rotating spacecraft or instrument.

Name	RevolutionRepetition		Extends	*PointingRepetition*	SFP	60
Attribute	**Type**	**Nullable**	**Description**			
revsSeparation	Expression <MAL::Long>	No	The required number of revolutions between occurrences.			
revsTolerance	Expression <MAL::Long>	No	The allowed tolerance (+/-) in the required number of revolutions between occurrences, the interpretation of which is dependent on the separationType.			
revAngle	Expression <MAL::Double>	Yes	Specifies the angle within a revolution.			
units	MAL::String	Yes	The units used for revAngle, as defined in reference [D6] annex D.			

4.3.8.5 Angle Repetition [Optional]

AngleRepetition

A sub-type of Repetition based on the angle subtended between three physical objects. The repetition angle is that subtended at the central object by target objects 1 and 2.

Name	AngleRepetition		Extends	*Repetition*	SFP	61
Attribute	**Type**	**Nullable**	**Description**			
centreObject	Expression <Position>	No	Position of the central object.			
targetObject1	Expression <Position>	No	Position of target object 1.			
targetObject2	Expression <Position>	No	Position of target object 2.			
initialAngle	Expression <MAL::Double>	No	Initial angle subtended at the central object by target objects 1 and 2.			
separation	Expression <MAL::Double>	No	The required angle between occurrences. If this is zero, this implies that repetition is between multiple occurrences of the initialAngle.			
tolerance	Expression <MAL::Double>	No	The allowed tolerance (+/-) in the required angle between occurrences, the interpretation of which is dependent on the separationType.			
units	MAL::String	Yes	The units used for separation and tolerance, as defined in reference [D6] annex D.			

4.3.8.6 Event Repetition

Event Repetition

A sub-type of Repetition based on **planning events**.

Name	**EventRepetition**		**Extends**	*Repetition*	**SFP**	62
Attribute	**Type**	**Nullable**	**Description**			
eventRef	Expression <MAL::ObjectRef>	No	Object Type: EventDefinition. Reference to an EventDefinition (type of event)			
separation	Expression <MAL::Long>	No	Number of occurrences of the planning event required between occurrences of the planning activity.			
tolerance	Expression <MAL::Long>	No	The allowed tolerance (+/-) in the number of occurrences of the planning event between occurrences of the planning activity, the interpretation of which is dependent on the separationType.			

5 ERROR CODES

Standard error codes defined by the MAL are applicable to the MPS service operations. In particular it is noted that this includes errors associated with delivery issues and authorization failure.

The following error codes shall apply to this specification.

Error	#	Description	ExtraInfo Type	ExtraInfo Description
INVALID	0	One or more fields in the message contain invalid values.	List <MAL::Pair>	ExtraInfo comprises a list of structures, each identifying an invalid field, comprising: 1. A String giving a dot separated nested index for the field, to allow for fields that are themselves a structure, of the form '3.2' meaning the 2nd field of the composite structure that is the 3rd field of the message. 2. A UInteger giving a secondary error code that details the reason for invalidity.
CANCEL_FAILED	1	The cancelRequest operation failed to cancel the referenced RequestInstance.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.
UPDATE_FAILED	2	The update operation (to Request, PlanStatus, Activity, Event or Resource) failed to update the referenced object.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.
REVOKE_FAILED	3	The revokePlan operation failed to revoke the referenced Plan, for example because it has already started executing.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.
INSERT_FAILED	4	The insertActivity or insertEvent operation failed to insert the requested object.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.
DELETE_FAILED	5	The deleteActivity or deleteEvent operation failed to delete the requested object.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.
ACTIVATE_FAILED	6	The activatePlan operation failed as the activation was outside the validity period of the Plan, or the start of the planPeriod had already passed.	MAL::String	ExtraInfo indicates 'Validity' or 'Expired' as appropriate.

UNSUPPORTED	7	An optional data structure used in the message is not supported by the service provider.	List <MAL::String>	ExtraInfo comprises a list of Strings giving a dot separated nested index for the unsupported field(s), to allow for fields that are themselves a structure, of the form '3.2' meaning the 2nd field of the composite structure that is the 3rd field of the message.
ACTIVATE_SUBPLAN_FAILED	8	The activateSubPlan operation failed.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.
DEACTIVATE_SUBPLAN_FAILED	9	The deactivateSubPlan operation failed.	MAL::String	ExtraInfo provides additional information on the reason for failure as a free format string.

SecondaryErrorCodeEnum

For the INVALID error, the secondary error code is defined as a UInteger that allows for deployment specific extensibility. The following standard secondary error codes are defined:

Name	**SecondaryErrorCodeEnum**	**SFP**	63
Error	**Error #**	**Description**	
UNKNOWN	0	Referenced MO object is not available to the service provider.	
UNDEFINED	1	Undefined value for enumeration field.	
OUT_OF_RANGE	2	A numeric value is outside the supported range.	
UNRECOGNIZED	3	Value of type MAL::Identifier or MAL::String (referencing a named item) does not correspond to a known item.	
BAD_TIME	4	A date-time value is outside the supported time period.	
BAD_POSITION	5	A position value is outside the supported position range.	
BAD_DIRECTION	6	A direction value is outside the supported direction range.	
INCONSISTENT	7	A value is inconsistent with that of another field within the message. This indicates violation of a constraint rule.	

6 SERVICE SPECIFICATION XML

6.1 OVERVIEW

The MO MAL specification (reference [2]) defines a normative XML Schema Definition (XSD) for validating MO service specifications and the MAL XML specification. The use of XML for service specification provides a machine-readable format rather than the text-based document format (reference [5]).

The MPS service specification defined in this document is also represented as an XML specification that follows the MAL defined schema.

The published XML Schema Definition (XSD) and the service specifications are held in an online SANA registry (reference [6]) located at:

https://sanaregistry.org/r/moschemas/

6.2 XML SCHEMA DEFINITION (XSD) FOR MO SERVICES

The XML Schema Definition (XSD) that is used to validate the actual XML service specifications has a filename with the structure "ServiceSchema-vBBB.xsd", where 'BBB' is replaced with the issue number of the corresponding document.

The normative XML for an MO service specification has a filename with the structure "areaAAA-vBBB-AREA" where 'AAA' is replaced with the area number, 'BBB' is replaced with the area version which shall match the issue number of the corresponding document and 'AREA' is replaced by the area name.

6.2.1 For this specification the following version of the XML Schema Definition (XSD) is applicable:

https://sanaregistry.org/r/moschemas/ServiceSchema-v003.xsd

6.2.2 The latest version of the XML Schema Definition is directly available from the address:

https://sanaregistry.org/r/moschemas/ServiceSchema.xsd

6.3 MAL XML

6.3.1 The normative XML for the MAL specification, validated against the XML Schema Definition (XSD), is located at:

https://sanaregistry.org/r/moschemas/area001-v003-MAL.xml

where 001 corresponds to the area number for MAL and 003 corresponds to the area version on which this specification is based.

6.3.2 The latest version of the MAL specification is directly available from the address:

https://sanaregistry.org/r/moschemas/ServiceDefMAL.xml

6.4 MPS XML

6.4.1 The normative XML for the MPS specification, validated against the XML Schema Definition (XSD), is located at[1]:

https://beta.sanaregistry.org/r/moschemas/area005-v001-MPS.xml

where 005 corresponds to the area number for MPS and 001 corresponds to the area version which shall match the issue number of this document.

6.4.2 The latest version of the MPS specification is directly available from the address:

https://beta.sanaregistry.org/r/moschemas/ServiceDefMPS.xml

[1] Note that this Red book contains references to the beta SANA registry and will be changed to the formal SANA registry on publication as a Blue Book

7 XML FILE FORMATS

7.1 INTRODUCTION

Some MPS deployments, including those based on legacy systems, may not be designed to use service-based interaction, using instead the transfer of files with a defined data structure. In order to support the exchange of information using the MPS Information Model structures, the following standardized file formats are defined:

– Planning Request;

– Planning Response (to a Planning Request);

– Plan.

These are based on the MPS information model defined in section 4, expressed as XML schemas, restricted to the data structures required to support Planning Requests and Plans respectively.

This section describes the approach taken to define the normative XML schemas and provides the location of the schemas in an online SANA registry. Detailed definition of the data structures referenced and the interpretation of their constituent fields can be found in section 4, except for additional or modified types specific to the file formats that are defined in this section.

7.2 XML SCHEMA NAMESPACE

The schema for the MPS File Structures are defined within a single namespace with the following URN:

urn:ccsds:schema:mo:mps

The schema for the MALTypes is defined within a separate namespace for the MAL with the following URN:

urn:ccsds:schema:mo:mal

7.3 XML SCHEMA ENCODING

Data structures of the MPS information model are ultimately composed of fields defined as one of the MAL::Attribute types. The encoding of these data types is fixed in the context of the XML file structures and associated XML schema, and follows the approach already defined for XML encoding of the MAL in reference [D10].

The following table defines the mapping of MAL::Attribute data types to XSD schema types used in the context of the MPS XML file structures.

Table 7-1: Mapping of MAL Attribute Types to XSD Types

MAL Attribute Type	XSD Type	Comment
MAL::Boolean	boolean	
MAL::Octet	byte	
MAL::UOctet	unsignedByte	
MAL::Short	short	
MAL::UShort	unsignedShort	
MAL::Integer	int	
MAL::UInteger	unsignedInt	
MAL::Long	long	
MAL::ULong	unsignedLong	
MAL::Float	float	
MAL::Double	double	
MAL::Duration	duration	
MAL::Time	mal:Time	Corresponds to XSD dateTime but with format constrained to CCYY-MM-DDThh:mm:ss.sss
MAL::FineTime	mal:FineTime	Corresponds to XSD dateTime but with format constrained to CCYY-MM-DDThh:mm:ss.sssssssss
MAL::String	string	
MAL::Blob	hexbinary	
MAL::Identifier	mal:Identifier	Corresponds to XSD string but with its format restricted to that of a MAL Identifier
MAL::URI	anyURI	
MAL::ObjectRef	mal:ObjectRef	Corresponds to an XSD complex type

The majority of MAL::Attribute types correspond to the specified XSD simple types.

The remainder are defined within the MALTypes schema that is imported into the MPS schema. These are shown in the table above with the 'mal:' namespace prefix and either imply constraints to be imposed on their encoding as XSD simple types or define MAL specific XSD complex types.

MAL::Time and MAL::FineTime correspond to the XSD dateTime simple type but with constraints on the time format to be used.

NOTE – In the specific case of an Expression of type Time defined in the MPSTypes schema, the value field is encoded as dateTime and can be used to represent either MAL type, according to the rules below.

Req_4.3.8.E.66 Fields of type MAL::Time shall be encoded using the XSD dateTime simple type with the format CCYY-MM-DDThh:mm:ss.sss.

Req_4.3.8.E.67 Fields of type MAL::FineTime shall be encoded using the XSD dateTime simple type with the format CCYY-MM-DDThh:mm:ss.sssssssss.

NOTE – MAL::Identifier corresponds to the XSD string simple type, but the string must follow the format defined for MAL Identifiers.

Req_4.3.8.E.68 Fields of type MAL::Identifier shall be encoded using the XSD string simple type with any format constraint imposed by the MAL (reference [2]).

NOTE – MAL::ObjectRef and MAL::StaticObjectRef corresponds to an XSD complex types defined as part of the MPS XML schema hierarchy. The MALTypes schema also includes an XSD complex type for the MAL::ObjectIdentity composite.

Req_4.3.8.E.69 Fields of type MAL::ObjectRef shall be encoded using the XSD MALObjectRef complex type. If a field is defined in the information model as being of type MAL::ObjectRef<T>, with a specified concrete object type T, then the *area* and type fields of the reference shall be omitted.

NOTE – Lists are encoding directly as an XSD element with multiple occurrences.

Req_4.3.8.E.70 Fields defined as a list of items of defined type shall be encoded as an XSD element of defined type with minimum and maximum occurrences.

7.4 XML SCHEMA STRUCTURE

The MPS XML file structures are defined as a hierarchy of XML schemas.

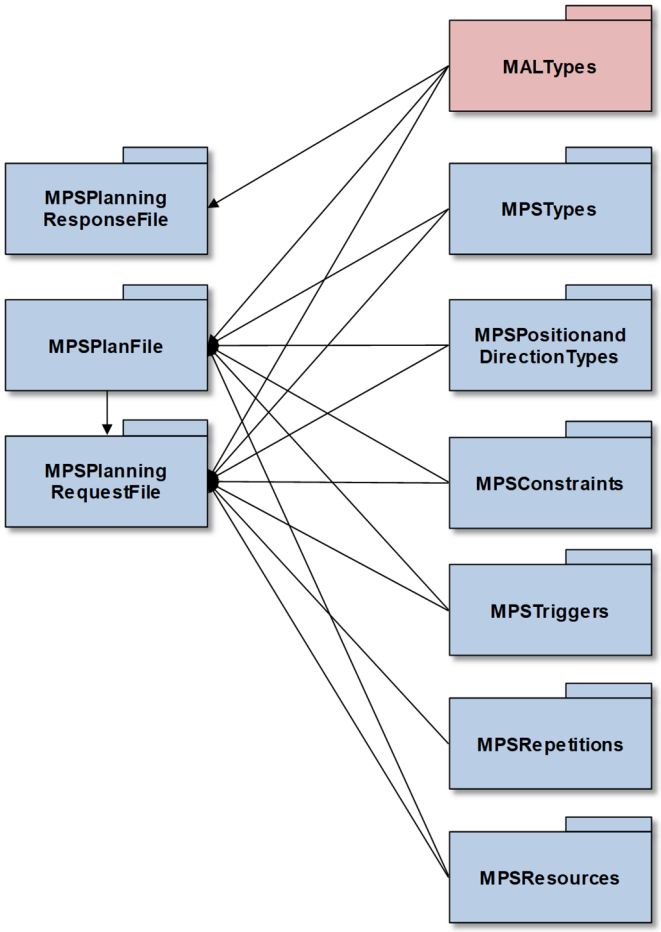

Figure 7-1: XML Schema Hierarchy

Each MPS XML file structure has a root XSD schema that includes:

– the top level element for the file;

– XSD representations of each MPS Data Item that may be included as part of the file.

It should be noted that these root XSD schemas do not include all defined data structures relating to an MPS data item, but only those required for the specific file structure. Those not required, such as Definition data structures and service specific data structures, are omitted.

A set of XSD schemas are also defined that are included by the above MPSPlanFile and MPSPlanningRequestFile to define referenced data types within the MPS information model. It should be noted that these are not required by the MPSPlanningResponseFile schema. Separate schema are defined to cover:

- MAL Data Types: MAL Attribute Types that do not map directly to XSD simple types:
  - Time, FineTime, and Identifier,
  - ObjectRef and ObjectIdentity,
  - NamedValue;
- MPS Data Types covering:
  - MPS Base Data Types (see 4.3.1),
  - Expressions (see 4.3.3),
  - Additional MPS Data Types (see 4.3.4),
  - Arguments (see 4.3.5);
- MPS Position and Direction Types (see 4.3.2) [Optional]:
  - MPS Constraints (see 4.3.6),
  - MPS Triggers (see 4.3.7),
  - MPS Repetitions (see 4.3.8),
  - MPS Resources (see 4.2.4.3).

The MPSRepetitions schema is only included by the MPSPlanningRequestFile schema.

The MPSPlanningRequestFile schema also includes the MPSPlanFile schema to support the direct embedding of a plan within a planning request.

The MPSResources schema defines resource profiles used within the MPSPlanFile and MPSConstraints schema.

7.5 XML FILE STRUCTURE

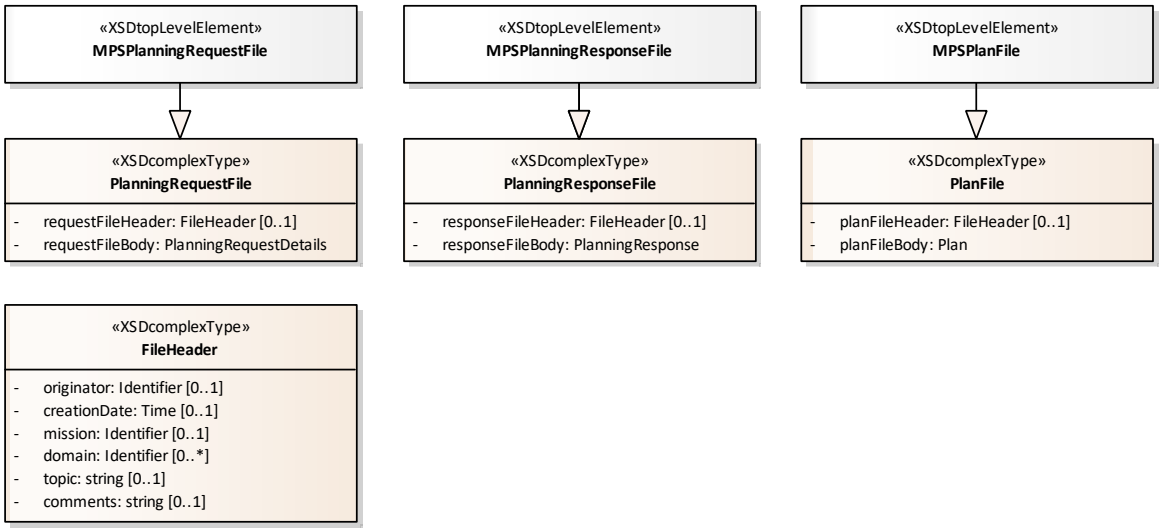

Figure 7-2: MPS XML File Structure

The general structure of the MPS Files comprises an optional common FileHeader, followed by a file body that corresponds to:

- PlanningRequestDetails (see 4.2.5.3) in the case of a PlanningRequestFile;
- A dedicated structure, defined below, in the case of a PlanningResponseFile that is a combination of a PlanningRequestResponse and a RequestStatusUpdate (see 4.2.5.3);
- Plan (see 4.2.6.1) in the case of the PlanFile.

This is illustrated in figure 7-2 above. There is an XSD top level element for each MPS File.

The FileHeader is an additional concrete data structure not defined in the context of the MPS Information Model, but defined below. This is included in the MPSTypes schema.

PlanningRequestFile

Name	PlanningRequestFile		
Attribute	**Type**	**Nullable**	**Description**
requestFileHeader	FileHeader	Yes	File header giving details of the source and scope of the MPS file.
requestFileBody	PlanningRequestDetails	No	File body corresponding to a the PlanningRequestDetails structure.

PlanningResponseFile

Name	PlanningResponseFile		
Attribute	**Type**	**Nullable**	**Description**
responseFileHeader	FileHeader	Yes	File header giving details of the source and scope of the MPS file.
responseFileBody	PlanningResponse	No	File body corresponding to a combination of PlanningRequestResponse and RequestStatusUpdate (defined in 7.6 below)

PlanFile

Name	PlanFile		
Attribute	**Type**	**Nullable**	**Description**
planFileHeader	FileHeader	Yes	File header giving details of the source and scope of the MPS file.
planFileBody	Plan	No	File body corresponding to a Plan structure.

FileHeader

In the absence of any standard MO message headers, the MPSFileHeader provides optional information about the origin, creation date, and scope of the file.

Name	FileHeader		
Attribute	**Type**	**Nullable**	**Description**
originator	Identifier	Yes	Identity of the entity responsible for generation of the file.
creationDate	Time	Yes	File creation date.
mission	Identifier	Yes	The space mission to which the file relates.
domain	List<Identifier>	Yes	Mission domain to which the file relates.
topic	string	Yes	Mission specific information that further defines the scope of the file.
comments	string	Yes	Field for any additional information or comments.

7.6 PLANNING REQUEST XML FILE FORMATS

Two XML file formats are defined for use in conjunction with MPS planning requests:

– MPSPlanningRequestFile;

– MPSPlanningResponseFile.

The body of the MPSPlanningRequestFile corresponds to a PlanningRequestDetails structure from the MPS information model (see 4.2.5.3).

The MPSPlanningRequestFile schema includes the definition of XSD complex types for the following data structures:

– PlanningRequestDetails;

– ActivityDetails (ActivityNode and SimpleActivityDetails).

The XSD representation of these data structures omits some attributes that are only meaningful where they are dynamically updatable. Similarly the set of allowed values for status attributes may be restricted to those relevant to the context of the message, omitting those relevant only during or post execution of the planned activities.

In the specific case of the Planning Request File, the option to include an embedded requested Plan is supported by inclusion of the MPSPlanFile schema.

In order to be able to interpret a **plan** and correlate a **planning request** with the resultant **planning activities** contained in the plan, the identity of the planning request is required. This identity is assigned by the planning function, not the originator of the planning request. This may be returned through ad-hoc mechanisms; however, the MPSPlanningResponseFile is provided as a standard file format to supply this feedback. The file may also optionally include the result of the planning request. The body of the MPSPlanningRespnseFile is based on the PlanningRequestResponse and RequestStatusUpdate structures from the MPS information model, but uses the dedicated hybrid structure detailed below.

Although the PlanningResponse message combines the information contained in both the PlanningRequestResponse and the RequestStatusUpdate, this does not imply that the corresponding operations within the planning system have to be combined. It is implementation-dependent when the file is generated and whether the status information is included. There is no definition of the service interface for deployments using the file formats.

Name	**PlanningResponse**		
Attribute	**Type**	**Nullable**	**Description**
instance	ObjectRef <RequestInstance>	No	Reference to the RequestInstance created in response to a submitted MPSPlanningRequestFile.
userReference	Identifier	No	User supplied reference for the planning request. This is distinct from the instanceID of the RequestInstance that is assigned by the planning function.
status	RequestStatusEnum	Yes	Current status of the planning request.
outputPlanRef	ObjectRef <Plan>	Yes	Reference to the Plan that contains the planned activities resulting from the planning request. It should be noted that this is only available once the planning request has been processed and successfully

			planned. The outputPlanRef may be updated following iterative planning cycles or re-planning.
returnData	List<NamedValue>	Yes	Optional return data from the planning process, provided as a list of ID-Value pairs. This can be used to provide additional information required by the User to interpret the planned operations.
statusInfo	string	Yes	StatusInfo provides the reason for termination and is customizable, but includes: - Completed (all constituent activities completed successfully) - Expired (constituent activities expired prior to execution) - Failed (constituent activities failed during execution) - Deleted (constituent activities were deleted)
errorCode	integer	Yes	Error Code optional in the case of a failure status for the planning request (for example Terminated state with statusInfo Failed). The codes are implementation specific.
errorInfo	string	Yes	Supplementary error information.

This allows the return of both the reference to the created RequestInstance and current status of the request in a single PlanningResponseFile. No timestamp is included within the structure above, but it is recommended that the creation date field of the file header is used, particularly if the status field is provided.

The point at which any PlanningResponseFile is generated by a planning function is deployment-specific. This may limit the set of possible status values, and whether or not the other nullable fields are meaningful in context. The outputPlanRef is only relevant if the planning process has generated an output Plan addressing the planning request. The last three fields may be used to supply a reason for rejection, but are most relevant post execution of the Plan.

NOTE – In the equivalent Planning Request Service operation, an error may be returned if the planning request is INVALID, and no RequestInstance is created. In the case of an invalid PlanningRequestFile, this error reporting mechanism is not available. Providing the PlanningRequestFile is not so badly formed that its identity cannot be interpreted, it would be possible to provide a PlanningResponseFile in return. In this case, a RequestInstance would need to be created and assigned the status REJECTED, the statusInfo attribute set to ‘INVALID’ and the errorCode and errorInfo attributes set consistently with the error code defined in section 4 above.

7.7 PLAN XML FILE FORMAT

A single XML file format is defined for distribution of MPS plans:

– MPSPlanFile.

The body of the MPSPlanFile corresponds to a Plan structure from the MPS information model (see 4.2.6.1).

The MPSPlanFile schema includes definition of XSD complex types for the following data structures:

– Plan (together with subsidiary data structures for PlanInformation, PlannedItems, PlanRevisions, and PlanResources);

– EventInstance;

– ActivityInstance;

– ResourceProfile.

The XSD representation of these data structures omits some attributes that are only meaningful where they are dynamically updatable, such as statusInfo. Similarly the set of allowed values for status attributes may be restricted to those relevant to the context of the message, omitting those relevant only during or post execution of the planned activities. The following details the specific omissions for each data structure:

Plan:	PlanStatusEnum restricted to Draft and Released; statusInfo omitted
EventInstance:	eventStatus and statusInfo omitted
ActivityInstance:	status, execInst, returnData, statusInfo, errorCode, and errorInfo omitted

7.8 MPS XML FILE FORMAT SCHEMA LOCATION

The published XML schemas for MPS file formats are held in an online SANA registry, located at the following URL:

https://sanaregistry.org/r/mpsfileschemas/

The following schemas are located within this registry:

MPSPlanningRequestFile.xsd
MPSPlanningResponseFile.xsd
MPSPlanFile.xsd
MPSTypes.xsd
MPSPositionandDirectionTypes.xsd
MPSConstraints.xsd
MPSTriggers.xsd
MPSRepetitions.xsd
MPSResources.xsd

MALTypes.xsd

It is not possible to add versions to schema files because this would invalidate the file references within the schemas. As such, all the above files are contained within a single .zip file with the following naming convention: "MPSFileFormats-v.v.zip", where the 'v.v' part is replaced with the issue number of the corresponding document and an incremental patch number counting from zero.

The file schemas defined by this document can be found at[5]:

https://beta.sanaregistry.org/r/mpsfileschemas/MPSFileFormats-1.0.zip

The latest version of any specification may always be directly addressed by removing the '-v.v' part from the URL; for example:

https://beta.sanaregistry.org/r/mpsfileschemas/MPSFileFormats.zip

[5] Note that this Red book contains references to the beta SANA registry and will be changed to the formal SANA registry on publication as a Blue Book

ANNEX A

PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT (PICS) PROFORMA

(NORMATIVE)

A1 INTRODUCTION

A1.1 OVERVIEW

This annex provides the Protocol Implementation Conformance Statement (PICS) Requirements List (RL) for an implementation of the Mission Operations MPS Services Recommended Standard. The PICS for an implementation is generated by completing the RL in accordance with the instructions below. An implementation claiming conformance must satisfy the mandatory requirements referenced in the RL.

The MO MPS Services Recommended Standard includes optional elements, as outlined in 2.6. These comprise:

- Optional services and service capability sets as summarized in table 2-7. A compliant deployment can support any combination of MO MPS services. If a service is supported, then it must support any mandatory service capability sets; other capability sets are optional.
- Optional data structures corresponding to optional elements of the MPS Information Model, as summarized in table 2-6.

An implementation's completed RL is called the PICS. The PICS states which protocol features have been implemented. The following entities can use the PICS:

- the protocol implementer, as a checklist to reduce the risk of failure to conform to the Recommended Standard through oversight;
- the supplier and acquirer or potential acquirer of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
- the user or potential user of the implementation, as a basis for initially checking the possibility of interworking with another implementation (while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSes);
- a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A1.2 NOTATION

A1.2.1 Status Column Symbols

The following are used in the RL to indicate the status of features:

Symbol	Meaning
M	Mandatory
O	Optional

A1.2.2 Support Column Symbols

The support of every item as claimed by the implementer is stated by entering the appropriate answer (Y, N, or N/A) in the support column.

Symbol	Meaning
Y	Yes, supported by the implementation
N	No, not supported by the implementation
N/A	Not applicable

A2 GENERAL INFORMATION

A2.1 IDENTIFICATION OF PICS

Ref	Question	Response
1	Date of Statement (DD/MM/YYYY)	
2	CCSDS document number containing the PICS	
3	Date of CCSDS document containing the PICS	

A2.2 IDENTIFICATION OF IMPLEMENTATION UNDER TEST (IUT)

Ref	Question	Response
1	Implementation name	
2	Implementation version	
3	Machine name	
4	Machine version	
5	Operating System name	
6	Operating System version	
7	Special Configuration	
8	Other Information	

A2.3 USER IDENTIFICATION

Supplier	
Contact Point for Queries	
Implementation name(s) and Versions	
Other Information Necessary for full identification, for example, name(s) and version(s) for machines and/or operating systems; System Name(s)	

A2.4 INSTRUCTIONS FOR COMPLETING THE RL

An implementer shows the extent of compliance to the protocol by completing the RL; the resulting completed RL is called a PICS.

A3 MPS SERVICES PICS

The MPS RL has an entry for each service, service capability set, file format, and information model element set.

There are 5 separate MPS services defined, each of which has multiple capability sets. The service is shown as a top level item, with subsidiary items for each capability set, using the format *s.c* in the Item column, where *s* is the service number, and *c* is the capability set number. All services are optional, but if implemented, some capability sets are mandatory. Compliance with a service implies compliance with the service operations and message structures defined in 0, together with applicable high-level, functional, and structural requirements defined therein. For capability sets, the service operations it comprises are listed in the Protocol Feature column.

Additional items are shown for each of the standard file formats that may be supported without services. These are indicated in the Item column by F.*n.* Compliance with a file format implies compliance with the file structures defined in 7 and formally expressed in the referenced XML schemas for MPS file formats.

The MPS services use an information model that itself has optional elements, which implies some features may not be supported within service messages.

While a compliant deployment shall support the full structure of messages exchanged at the service interface for supported services and capability sets, it is not required to support optional data structures within those messages at application level.

Support for optional elements of the information model is shown as a separate RL item ‘D’, with subsidiary items for each optional element, corresponding to those identified in table 2-6. This section should be completed in conjunction with both services and file based items.

Item	Protocol Feature	Reference	Status	Support
1	Planning Request Service	2.6 and 3.2	O	
1.1	submitRequest getRequestSummaries getRequestStatus	2.6 and 3.2	M	
1.2	cancelRequest	2.6 and 3.2	O	
1.3	updateRequest	2.6 and 3.2	O	
1.4	monitorRequestStatus	2.6 and 3.2	O	
1.5	getRequest	2.6 and 3.2	O	
2	Plan Distribution Service	2.6 and 3.3	O	
2.1	getPlanSummaries getPlan getPlanStatus	2.6 and 3.3	M	
2.2	monitorPlanStatus	2.6 and 3.3	O	
2.3	monitorPlan	2.6 and 3.3	O	
2.4	queryPlan	2.6 and 3.3	O	
2.5	getPartialPlan	2.6 and 3.3	O	
3	Plan Execution Control Service	2.6 and 3.4	O	
3.1	submitPlan revokePlan getPlanStatus	2.6 and 3.4	M	
3.2	activatePlan deactivatePlan	2.6 and 3.4	O	
3.3	monitorPlanExecution	2.6 and 3.4	O	
3.4	monitorPlanExecutionDetail	2.6 and 3.4	O	
3.5	activateSubPlan deactivateSubPlan getSubPlanStatus	2.6 and 3.4	O	
3.6	monitorSubPlanExecution	2.6 and 3.4	O	
3.7	suspendActivity resumeActivity	2.6 and 3.4	O	
3.8	getActivityStatus	2.6 and 3.4	O	
4	Plan Information Management Service	2.6 and 3.5	O	
4.1	listRequestDefs getRequestDefs	2.6 and 3.5	O	
4.2	listEventDefs getEventDefs	2.6 and 3.5	O	
4.3	listActivityDefs	2.6 and 3.5	O	

	getActivityDefs			
4.4	listResourceDefs getResourceDefs	2.6 and 3.5	O	
4.5	getSystemConfig	2.6 and 3.5	O	
5	Plan Edit Service	2.6 and 3.6	O	
5.1	updatePlanStatus	2.6 and 3.6	M	
5.2	insertActivity insertEvent deleteActivity deleteEvent	2.6 and 3.6	M	
5.3	updateActivity updateEvent	2.6 and 3.6	O	
5.4	updateResource	2.6 and 3.6	O	
5.5	updateResourceProfile	2.6 and 3.6	O	
5.6	applyTimeShift	2.6 and 3.6	O	
F	MPS File Formats	7	O	
F.1	Planning Request File Format	7.6	O	
F.2	Planning Response File Format	7.6	O	
F.3	Plan File Format	7.7	O	
D	MPS Information Model	2.6 and 4	M	
D.1	Core Features	2.6 and 4	M	
D.2	Basic Constraints	2.6, 4.3.6.2.3, 4.3.6.2.4, 4.3.6.2.5	O	
D.3	Plan Revisions	2.6 and 4.2.6.1.4	O	
D.4	Resources	2.6, 4.2.4, and 4.2.6.1.5	O	
D.5	Resource Constraints (requires D.4)	2.6, 4.3.6.2.6, and 4.3.6.3	O	
D.6	Position & Direction	2.6, 4.3.2.2, 4.3.2.3, 4.3.7.3, 4.3.7.4, 4.3.7.5, 4.3.8.3, 4.3.8.4, 4.3.8.5	O	
D.7	Geometric Constraints (requires D.6)	2.6, 4.3.6.2.8	O	
D.8	Functions	2.6, 4.2.9, 4.3.6.2.7	O	

ANNEX B

SECURITY, SANA, AND PATENT CONSIDERATIONS

(INFORMATIVE)

B1 SECURITY CONSIDERATIONS

B1.1 SYSTEM SECURITY REQUIREMENTS

Security requirements are specific to the deployed mission system and can vary significantly between different mission systems. The MPS services or file formats support a limited subset of the interactions supported by a typical mission system, and as such must be capable of deployment in the context of a mission or organization specific security architecture that supports multiple services.

The mission security architecture shall address the following:

- Protection of the communications link between MPS service consumer and provider to ensure data integrity and confidentiality. This may or may not include the encryption of service messages, depending on mission specific requirements.

- Control of access to specific MPS services, service operations and service data through the management of access rights associated with registered service users.

- Authentication to ensure only genuine registered service users have access to MPS services and to ascertain their level of access rights.

For the MPS services, the security considerations of this specification are the same as those of the MAL in reference [2]. Specifically, authentication and authorization of a participating consumer or provider is provided by the MAL access control concept and is covered in subsections 3.6, 5.2, and 5.3 of the Reference Model (reference [1]).

Security of the communications link carrying the MPS services is delegated to the implementation of the underlying Transport Layer.

For MPS file formats, this specification only addresses the format of the files, not the method of transfer. All security considerations relating to the transfer of these files must therefore be addressed by the actual file transfer service and security architecture of the deployed mission system.

B1.2 POTENTIAL THREATS

In many mission systems, mission planning is the principal or nominal way of controlling the mission. Unauthorized access to the MPS services can therefore be a means of sabotaging the mission:

- the Planning Request Service could be used to inject dangerous planning requests into the system. This is mitigated by the Planning function itself, the implementation of which can be used to detect and flag potentially dangerous requests either for automatic rejection or to obtain authorization from the mission planner.

- the Planning Request and Plan Distribution Services could be used to access confidential information about submitted planning requests or planned activities.

- The Plan Execution Control Service could be used to submit unauthorized plans for automated execution, or to stop execution of the currently authorized plans.

- The Plan Edit Service could be used to make unauthorized changes to the currently authorized plans, by deleting or modifying planned activities and events, or by inserting new activities or events into the currently authorized plans.

B1.3 ACCESS CONTROL

The MPS services are closely tied to the Access Control aspect of the MAL where returned authentication identifiers are used in the MAL message header to authenticate and authorise messages via Access Control.

Registered users are assigned roles (access rights) that may limit their access to MPS services. The set of access control roles is specific to the service deployment. An implementation of the MPS service provider can than restrict access to services, service capability sets, and individual service operations based on the assigned roles. Similarly roles can be used to restrict access to a subset of service data, either by data class or **domain**.

Which access control roles are supported is specific to the mission deployment and depends on the access control requirements for the mission. Typical MPS roles include access to:

- Individual MPS Services: access to Plan Execution Control and Plan Edit Services is likely to be more restricted than to the Planning Request and Plan Distribution Services.

- Restricted Capbility Sets or individual Operations of an MPS Service: more sensitive operations may require a special access control role.

- Data class or **domain** of contained information (for example to restrict access to a specific spacecraft, subsystem or payload in terms of available planning requests and planning activities)

- Information pertaining to other users: access for some users may be restricted to their own planning requests and planned activities.

- Submission of custom planning requests: access for some users may be restricted to predefined planning requests.

It is the responsibility of the implementation of the MPS service provider to enforce access control based on the assigned user roles.

B1.4 DATA INTEGRITY

As stated previously, the confidentiality and integrity of MPS service messages is delegated to the implementation of the underlying transport layer.

This is dependent on the technology used for the implementation of the transport layer and the corresponding MAL technology binding. It may include the encryption of the service messages,.

B1.5 AUTHENTICATION

Authentication for the MPS Services, as for all MO Services, may be supported through the MO Common Login Service (reference [E3]).

The Login service allows a service user to provide authentication information to the system. It takes the user's credentials and uses a deployment-specific mechanism to authenticate the user; the result of this is used by the MAL during access control.

The Login service and the access control provided by the MAL are fully dependent on a deployment-specific security architecture (for example, the authentication protocol Kerberos). Both layers (Common and MAL) provide access to, and use of, this security service; they do not implement it themselves.

B1.6 CONFIDENTIALITY

For some missions, there may be commercial or security considerations that result in a need for confidentiality of planning requests and resultant planned activites.

Where this is the case, the MPS service provider may be required to implement an access control filter on the return of information to users. In particular:

- Visibility of planning requests may be restricted (based on a specific access control role) to those raised by the user currently accessing the service.

- Distributed plans may similarly be filtered to restrict visibility of the planned activites resulting from restricted planning requests. A **partial plan** would then be returned in place of the full unrestricted plan.

It is mission and deployment specific how this is implemented, but it is expected that this would make use of special access control roles assigned to users. This may typically restrict access to information based on a level of access rights:

1) All Information

2) User's Own + Unrestricted Information

3) Only User's Own Information

B1.7 AUDITING

The MPS Services include the potential to access the evolving state of planning requests and plans (together with their contained planning activities, planning events and planning resources) through the delivery of status **updates** pertaining to the corresponding MPS data items.

The full set of **updates** provides a comprehensive audit trail on the execution of MPS service operations at the level of planning requests and plans.

While storage of and access to historical **updates** is not directly supported by the MPS Service specification, this provides the potential for a service provider to implement such an audit trail.

B1.8 AVAILABILITY

Availability requirements are mission specific. The required availability of an MPS service provider implementation will impact its design, both in terms of the physical deployment architecture and its software implementation.

B2 SANA CONSIDERATIONS

The recommendations of this document request SANA populate the registry specified in reference [2] with the schema and XML detailed in section 6 of this document.

As stated in reference [2], the registration rule for change to this registry requires an engineering review by a designated expert. The expert shall be assigned by the MPS WG Chair, or in absence, MOIMS Area Director.

Specifically, this applies to the following registries:

http://sanaregistry.org/r/moschemas/ for entries relating to the MPS Service Specifications.

https://sanaregistry.org/r/mpsfileschemas/ for entries relating to MPS File Format Schemas.

B3 PATENT CONSIDERATIONS

The recommendations of this document have no patent issues.

ANNEX C

DEFINITION OF ACRONYMS

(INFORMATIVE)

Acronym	Definition
AOS	Acquisition Of Signal
COM	Common Object Model
CSS	Cross Support Services
LOS	Loss of Signal
MAL	Message Abstraction Layer
MO	Mission Operations
MOIMS	Mission Operations and Information Management Systems [CCSDS Area]
MPS	Mission Planning and Scheduling
NAV	CCSDS Navigation Working Group
NEM	Navigation Event Message
ODM	Orbit Data Message
OMG	Object Management Group
PDS	Plan Distribution Service
PECS	Plan Execution Control Service
PES	Plan Edit Service
PI	Principal Investigator
PICS	Protocol Implementation Conformance Statement
PIMS	Plan Information Management Service
PRL	(PICS) Requirements List
PRM	Pointing Request Message
PRS	Planning Request Service
RASDS	Reference Architecture for Space Data Systems
RL	Requirements List
SANA	CCSDS Space Assigned Numbers Authority
SFN	Short Form Number
SFP	Short Form Part
TOO	Target Of Opportunity
UML	Unified Modelling Language
URI	Uniform Resource Identifier

Acronym	Definition
URL	Uniform Resource Locator
URN	Uniform Resource Name
XML	eXtensible Markup Language

ANNEX D

INFORMATIVE REFERENCES

(INFORMATIVE)

[D1] *Mission Operations Services Concept*. Issue 3. Report Concerning Space Data System Standards (Green Book), CCSDS 520.0-G-3. Washington, D.C.: CCSDS, December 2010.

[D2] *Mission Planning and Scheduling*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 529.0-G-1. Washington, D.C.: CCSDS, June 2018.

[D3] *Mission Operations—Common Services*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 522.0-B-1. Washington, D.C.: CCSDS, May 2020.

[D4] *Mission Operations Monitor & Control Services*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 522.1-B-1. Washington, D.C.: CCSDS, October 2017.

[D5] *Navigation Data Messages Overview*. Issue 3. Informational Report (Green Book), CCSDS 500.2-G-3. Washington, D.C.: CCSDS, March 2023.

[D6] *Pointing Request Message*. Issue 1. Recommendation for Space Data Systems Standards (Blue Book), CCSDS 509.0-B-1. Washington, D.C.: CCSDS, October 2023 (incorporating Technical Corrigenda 1 and 2).

[D7] *Cross Support Service Management—Simple Schedule Format Specification*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 902.1-B-1. Washington, D.C.: CCSDS, May 2018.

[D8] *Space Engineering—Telemetry and Telecommand Packet Utilization*. ECSS-E-ST-70-41C. Noordwijk, The Netherlands: ECSS Secretariat, 15 April 2016.

[D9] "XML Path Language (XPath) 3.1." Version 3.1, 21 March 2017. W3C. https://www.w3.org/TR/xpath-31/.

[D10] *Mission Operations—Message Abstraction Layer Binding to HTTP Transport and XML Encoding*. Issue 1. Recommendation for Space Data System Standards (Blue Book), CCSDS 524.3-B-1. Washington, D.C.: CCSDS, June 2018.

[D11] *Application and Support Layer Architecture*. Issue 1. Report Concerning Space Data System Standards (Green Book), CCSDS 371.0-G-1. Washington, D.C.: CCSDS, November 2020.

[D12] *Orbit Data Messages*. Issue 3. Recommendation for Space Data Systems Standard (Blue Book), CCSDS 502.0-B-3. Washington, D.C.: CCSDS, April 2023

ANNEX E

LITERAL FORMATS IN EXPRESSIONS

(INFORMATIVE)

E1 INTRODUCTION

When entering MPS data, it is often not possible to provide an absolute value for a required attribute, but instead a dynamic value can be specified as an **expression** (see 4.3.3) for derivation at run time. The expression can be provided as:

- a literal value;
- a reference to an attribute or **argument** of an MO object;
- a calculated value including one or more references to attributes or **arguments** of MO objects.

The language used to represent the **expression** does not form part of this Recommended Standard; any expression language can be used within a particular deployment. This annex specifies an optional simple XSD-based syntax for the specification of literal values and references that can be used in the absence of a complete expression language.

If used in an MPS Expression, the expressionLanguage field should be set to 'MPSliteral'.

MPS Expressions can be of any MAL::Attribute type plus the MPS position and direction types, as detailed in the following table.

Table E-1: MPS Expression Types

#	Type	Expression Subclass	XSD Type	Comment
1	Blob	Binary	hexbinary	
2	Boolean	Boolean	boolean	
3	Duration	Duration	duration	
4	Float	Real	float	
5	Double		double	
6	Identifier	String	string	Format restricted to that of a MAL Identifier. Cannot contain the characters :.()* as these have special meaning in the literal representation of an ObjectRef
7	Octet	Integer	byte	
8	UOctet		unsignedByte	

9	Short		short	
10	UShort		unsignedShort	
11	Integer		int	
12	UInteger		unsignedInt	
13	Long		long	
14	ULong		unsignedLong	
15	String	String	string	
16	Time	Time	dateTime	Format constrained to CCYY-MM-DDThh:mm:ss.sss
17	FineTime		dateTime	Format constrained to CCYY-MM-DDThh:mm:ss.sssssssss
18	URI	String	anyURI	
19	ObjectRef	Object	mal:ObjectRef	Corresponds to an XSD complex type
129	Direction	Direction	mps:Direction	
130	Position	Position	mps:Position	

The # and Type columns correspond to the possible values of the ArgTypeEnum that defines the data type of the expression. This specifies the type of the result value of the evaluated expression.

The Expression Subclass column groups these types based on the operations (expression operators and syntax) relevant to them.

The XSD Type column maps each data type to its representation in XML schema.

This is presented in the following subsections:

E2 Literals for Expression Types corresponding to XSD simple types

E4 Literals for Object References

E5 Literals for Position and Direction Types

Within the specification of the literals for ObjectRef, Position, and Direction, the names of the attributes of the corresponding composite data types are shown in *italics*. These attributes can be represented by the literal form of the corresponding XSD simple type for the attribute.

E2 LITERALS FOR EXPRESSION TYPES CORRESPONDING TO XSD SIMPLE TYPES

For those expression types that can be mapped directly to XSD data types, as shown in table E-1 above, literal values can be supplied according to the rules defined for XML Schema in https://www.w3.org/TR/xmlschema-2/.

This applies to all types with the exception of ObjectRef, Direction, and Position, for which special rules are defined in the following sections.

A literal value for these data types can be supplied directly for the Expression.value field, or included within the expression string itself.

MAL Identifiers are essentially represented as XSD strings, but have additional constraints as follows:

- The following characters cannot be used within a MAL Identifier, as they are used as delimiters in the literal representation of ObjectRefs (colon, period, or dot and parentheses) :.();
- The '*' character cannot be used within a MAL Identifier, as this is used to indicate a wildcard **domain**.

Times and FineTimes are both represented as XSD dateTimes, but with restricted format rules as outlined in the table above.

E3 LITERALS FOR OBJECT REFERENCES

A reference to an MO object comprises multiple elements, all of which may need to be expressed in its literal representation:

- domain an ordered list of MAL::Identifiers
- area MAL::Identifier
- type MAL::Identifier
- key MAL::Identifier
- version MAL::UInteger

The area and type can be omitted where a field is constrained to be a reference to an MO object of specific type. The domain may be omitted where it is defined by context. The version may also be omitted where the current or latest version is assumed. In its simplest form a literal objectRef consists only of the **key** field, which is a MAL::Identifier represented as an XSD string.

The following literal format is defined for an ObjectRef:

objectref = [*area*:*type*:][*domain*[.*domain*]*.]*key*[(*version*)]

The above returns a value of type ObjectRef. MO object references may also be used to access the values of attributes or **arguments** of the object in expressions of any type. The value returned has the type of the corresponding attribute or argument. The literal format above is extended as follows:

objectref = *objectref.attribute*[\[*index*|*name*\]] | o*bjectref@argument*[[\[*index*\]]

where *attribute* and *argument* are the names of the attribute or argument of the object respectively.

If the attribute or argument is itself a set or array, then a specific item can be referenced using an integer *index* enclosed in square brackets. In the specific case that an attribute is itself a collection of objects, then the required object can be specified by name.

NOTE – The name (or *key*) used to reference a specific object in a collection is dependent on the type of object. For Resources it is the name of the Resource; but for instantiated objects such as ActivityInstance and EventInstance it is the name of the associated Definition.

It should be noted that where the returned value is itself an object reference, attribute references can be used iteratively to follow a sequence of references to the required value. For example:

[myActivityInst.source]@argA
returns the argument argA of the parent ActivityInstance of myActivityInst

[myActivityInst.source].children[myActivityDef]
returns an ObjectRef pointing to the sibling activity of myActivityInst that has the name myActivityDef.

Square brackets may be used to enclose an object reference to aid readability.

The special keyword ‘Me’ may be used to reference the current object in context. This gives a shorthand form of referencing the attributes and **arguments** of an object:

Me.*attribute*

Me@*argument*

E4 LITERALS FOR POSITION AND DIRECTION TYPES

As positions and directions are not supported as MAL::Attribute types, it is not possible to supply a literal value directly to the value attribute of an MPS expression. The literal value must be provided within the expression string. The expression language used is implementation dependent, but the following provides an optional simple XSD-based representation for positions and directions.

The position and direction types are specific to MPS and defined as abstract data types with multiple concrete subtypes representing different types of coordinates.

The literal representation of all types begins with the @ symbol, followed by two discriminator characters that indicate which coordinate subtype is represented.

For Positions this is as follows:

@PC(*x,y,z*)[*units*]:*frame*	CartesianPosition
@PS(*longitude,latitude*[*,altitude*])[*units*[,*altitudeUnits*]]:*frame*	SurfacePosition
@PF(*orbitFile*)	OrbitFilePosition
@PO(*orbitNumber*,R|A,*orbitAngle*)[*units*]	OrbitalPosition
@PN(*object*)	ObjectPosition
@PR(*reference*)	PositionReference

For Directions this is as follows:

@DC(*x,y,z*):*frame*	CartesianDirection
@DS(*azimuth,elevation*)[*units*]:*frame*	SphericalDirection
@DA(*ra,dec*)[*units*]:*frame*	RADecDirection
@DN(*namedTarget*)	NamedTargetDirection
@DR(*reference*)	PDirectionReference

Where the named attributes are represented as XSD literals for the corresponding data type, with the exception of the relativeOrbit Boolean flag in OrbitalPosition, which is represented as R (True for relative orbit) or A (False for absolute orbit). The *frame* attribute is represented by one of the defined string values for the coordinate system.

ANNEX F

OPEN ISSUES

(INFORMATIVE)

The following open issues apply to this draft version of the Recommended Standard:

a) Reference [2] to the MAL references the current published version of the MAL and will require updating when the revised MAL is published. This MP&S Red Book is fully in line with Issue 3 of the MAL currently in the process of being published. The reference will be updated before publication of the MP&S Blue Book.