MEF Standard

MEF 95.0.1

Amendment to MEF 95: Policy Driven Orchestration

October 2022

Disclaimer

The information in this publication is freely available for reproduction and use by any recipient and is believed to be accurate as of its publication date. Such information is subject to change without notice and MEF Forum (MEF) is not responsible for any errors. MEF does not assume responsibility to update or correct any information in this publication. No representation or warranty, expressed or implied, is made by MEF concerning the completeness, accuracy, or applicability of any information contained herein and no liability of any kind shall be assumed by MEF as a result of reliance upon such information.

The information contained herein is intended to be used without modification by the recipient or user of this document. MEF is not responsible or liable for any modifications to this document made by any other party.

The receipt or any use of this document or its contents does not in any way create, by implication or otherwise:

a) any express or implied license or right to or under any patent, copyright, trademark or trade secret rights held or claimed by any MEF member which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor

b) any warranty or representation that any MEF members will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor

c) any form of relationship between any MEF member and the recipient or user of this document.

Implementation or use of specific MEF standards, specifications, or recommendations will be voluntary, and no Member shall be obliged to implement them by virtue of participation in MEF Forum. MEF is a non-profit international organization to enable the development and worldwide adoption of agile, assured and orchestrated network services. MEF does not, expressly or otherwise, endorse or promote any specific products or services.

Table of Contents

1 Abstract

This amendment provides enhancements to MEF 95 [1] that enables policies to be grouped and defines the semantics of this grouping. It also enables execution order across (not within) policies to be specified for any type of policy that is being used (i.e., imperative, declarative, or intent).

2 Introduction

This amendment introduces the notion of groups of policies to MEF 95 [1]. This results in the following changes to MEF 95 [1]:

- In the terminology section, add definitions of Atomic, Composite, and Priority of a policy.
- In section 8.7, change Figure 10 to show new classes for grouping policies. In addition, update the text immediately below Figure 10 to reflect the above changes.
- In section 8.7.1, add a new attribute to the MPMPolicyStructure class. Update Figure 11 showing the new attribute and its associated operations. Update the two associated tables with definitions of the new attribute and its operations.
- In section 8.7.2, define a new relationship (MPMPolicyStructureHasPolicy).
- In section 8.7.3, two new subclasses are defined (MPMPolicyStructureComposite and MPMPolicyStructureAtomic), and the existing three subclasses (MPMImperativePolicy, MPMDeclarativePolicy, and MPMIntentPolicy) are redefined to inherit from MPMPolicyStructureAtomic instead of MPMPolicyStructure. Also, update Figure 12.

In this amendment, changes are shown as follows:

- Instructions for how to apply the amendment are shown in *blue italics*
- Text to be removed is shown with ~~red strikethrough~~
- Text to be added is shown in red

3 Changes to Section 3

Add the following three new definitions to Table 2:

Term	Definition	Reference
Atomic	A type of Object that cannot contain any other types of Objects.	THIS DOCUMENT
Composite	A type of Object that can contain other Atomic or Composite Objects including their subclasses.	THIS DOCUMENT
Priority	Defines the execution order of actions within an Atomic or Composite Imperative Policy.	THIS DOCUMENT

Table 2 – Terminology and Abbreviations

4 Changes to Section 8.7

Replace Figure 10 with the following figure:

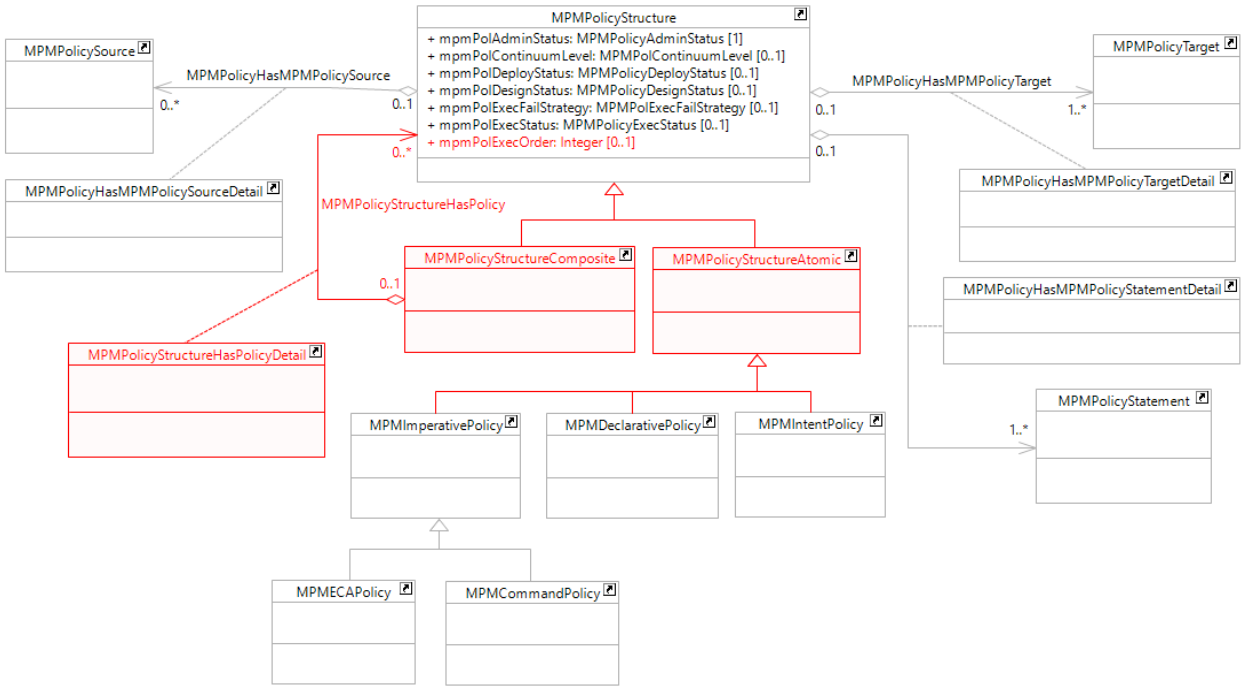

Figure 10 – The MPMPolicyStructure Class Hierarchy

Replace the paragraph immediately following Figure 10 with the following:

An MPMPolicy is a set of concrete subclasses of the MPMPolicyStructure class. In Figure 10, all subclasses of MPMPolicyStructure except MPMECAPolicy and MPMCommandPolicy are abstract. The Composite pattern [6] [8] is used to define two subclasses of MPMPolicyStructure. MPMPolicyStructureAtomic is used to represent different types of individual MPMPolicies (i.e., those that do not contain other MPMPolicies), while MPMPolicyStructureComposite enables trees of MPMPolicies to be defined.

5 Changes to Section 8.7.1

Add the following attribute and its definition to Table 4:

Attribute Name	Description
mpmPolExecOrder: Integer[0..1]	This is an optional non-negative integer whose value defines the desired order of execution of this MPMPolicy Object (compared with all other MPMPolicy Objects). The MPMPolicy whose mpmPolExec-Order attribute has the highest value is executed first. **[A1-O1]** A default value of 0 **MAY** be defined. **[A1-O2]** MPMPolicies that have the same value for their mpmPol-ExecOrder attributes **MAY** be executed in any order.

Table 4 – Attributes of the MPMPolicyStructure Class

Change Figure 11 to add 1 attribute (mpmPolExecOrder) and its get and set operations as follows:

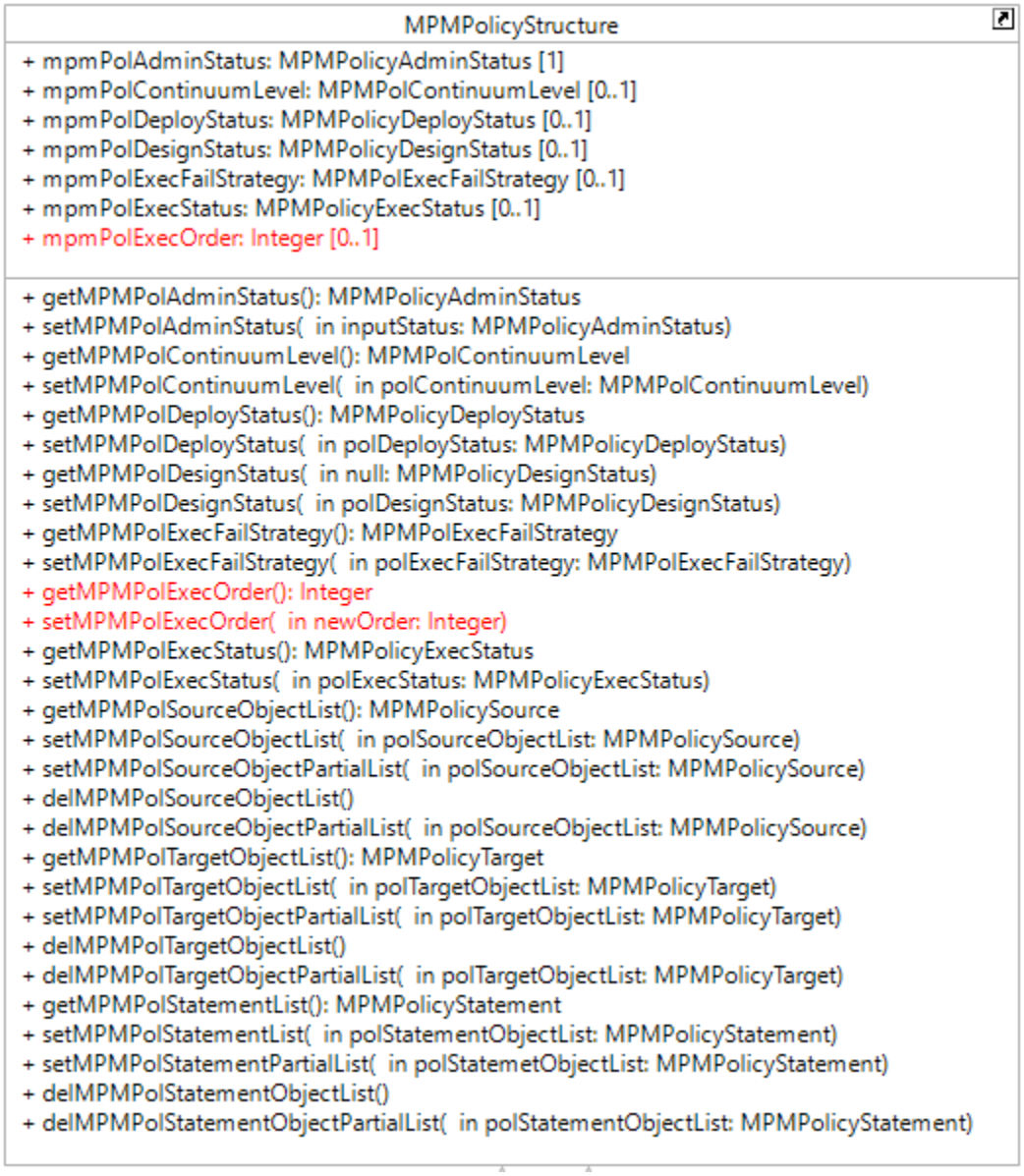

Figure 11 – Operations of the MPMPolicyStructure Class

Update Table 5 to add definitions for the get and set operations as follows:

Operation Name	Description
getMPMPolExecOrder() : Integer[1..1]	This operation returns the current execution order of this MPMPolicy Object. This operation takes no input parameters.
setMPMPolExecStatus(in newOrder : Integer[1..1])	This operation sets the current execution order of this MPMPolicy Object. This operation takes a single input parameter, called newOrder, which defines the new value for the mpmPolExecOrder attribute.

Table 5 – Operations of the MCMPolicyStructure Class

6 Changes to Section 8.7.2

Change text in section 8.7.2 as shown, and add a fourth sub-section, also as shown.

8.7.2 MPMPolicyStructure Relationships

The MPMPolicyStructure class defines ~~three~~ four aggregation relationships, as shown in Figure 10.

Add a new subsection as shown:

8.7.2.4 The MPMPolicyStructureHasPolicy Aggregation

The MPMPolicyStructureHasPolicy aggregation is an optional aggregation and defines the set of MPMPolicyStructure Objects that are attached to this particular MPMPolicyStructureComposite Object. The semantics of this aggregation are defined by the MPMPolicyStructureHasPolicyDetail association class.

The multiplicity of this aggregation is 0..1 - 0..*. This means that it is an optional aggregation (i.e., the "0" part of the 0..1 cardinality). If this aggregation is instantiated (i.e., the "1" part of the 0..1 cardinality), then zero or more (0..*) MPMPolicyStructure Objects can be contained in this particular MPMPolicyStructureComposite Object. The 0..* cardinality enables an MCMPolicyStructureComposite Object to be defined without having to first define an associated MPMPolicyStructure Object for it to aggregate. The semantics of this aggregation are defined by the MPMPolicyStructureHasPolicyDetail association class. This enables the management system to control which set of concrete subclasses of MCMPolicyStructureComposite can aggregate which types of MPMPolicyStructure Objects.

[A1-O3] An MPMPolicyStructureComposite Object, or any of its subclasses, **MAY** aggregate zero or more MPMPolicyStructure Objects.

The MPMPolicyStructureHasPolicyDetail class is a concrete association class and defines the semantics of the MPMPolicyStructureHasPolicy aggregation. The attributes and relationships of this class can be used to define which MPMPolicyStructure Objects can be attached to which particular set of MPMPolicyStructureComposite Objects. These semantics can be further enhanced by using the Policy Pattern to define policy rules that constrain which part Objects (i.e., MPMPolicyStructure) are attached to which aggregate Object. Note that MCMPolicyStructure is an abstract class that is the superclass of imperative, declarative, and intent policy rules.

7 Changes to Section 8.7.3

Change text in section 8.7.3 as shown.

8.7.3 MPMPolicyStructure Subclasses

The MPMPolicyStructure class defines ~~three~~two subclasses, which are described in the following subsections. The MPMPolicyStructureAtomic class currently defines three subclasses. The MPMImperativePolicy class currently defines two subclasses. Figure 12 will be used to describe each of these ~~three~~seven subclasses.

Replace Figure 12 with the following figure:

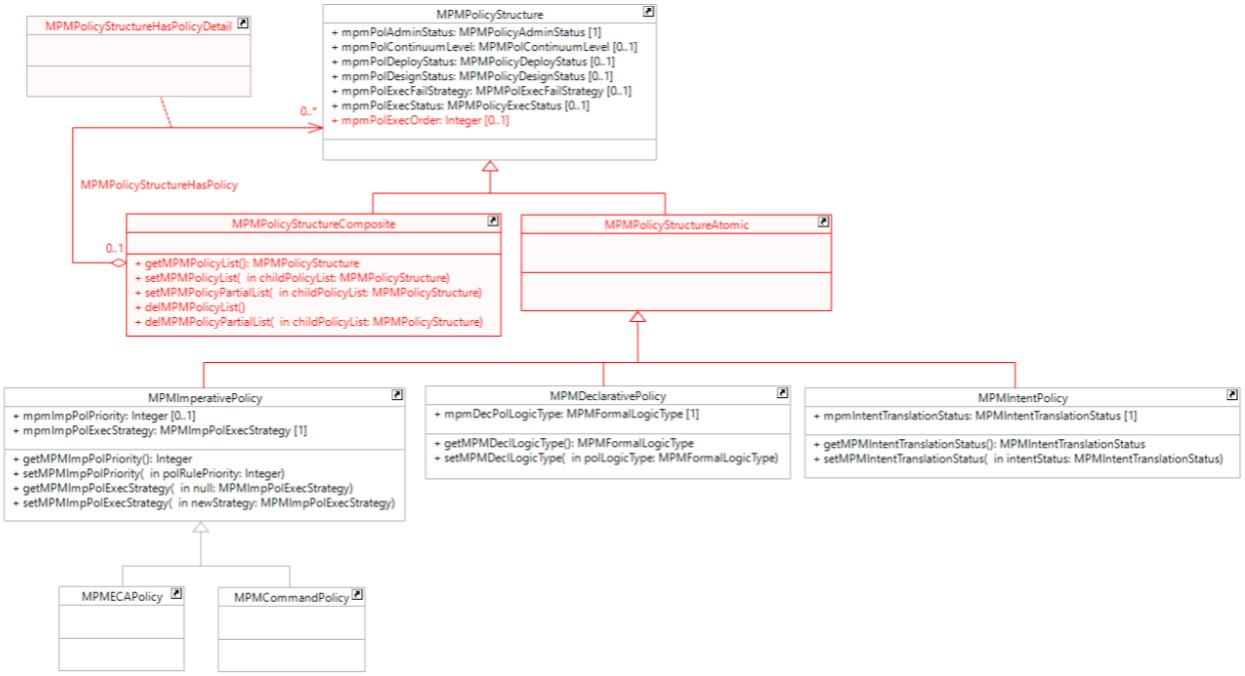

Figure 12 – MPMPolicyStructure Subclasses

Insert two new subsections as follows, and subsequently renumber the existing subsections 8.7.3.1-3 that are in MEF 95 [1] to 8.7.3.3-5.

8.7.3.1 MPMPolicyStructureAtomic Class Definition

This is an abstract class and specializes MPMPolicyStructure. This class represents stand-alone MPMPolicy Objects.

[A1-R1] An MPMPolicyStructureAtomic Object **MUST NOT** contain another MPMPolicyStructure Object.

8.7.3.2 MPMPolicyStructureComposite Class Definition

This is an abstract class and specializes MPMPolicyStructure. This class represents a set of related MCMPolicyStructure Objects that are organized into a tree structure.

[A1-O4] Each MPMPolicyStructureComposite Object **MAY** contain zero or more MPMPolicyStructureAtomic and/or zero or more MPMPolicyStructureComposite Objects.

Table [A1-6] defines the operations for this class.

Operation Name	Description
getMPMPolicyList() : MPMStructure[1..*]	This operation returns the set of all MPMPolicyStructure Objects that are contained in this specific MPMPolicyStructureComposite Object. There are no input parameters to this operation. This operation returns a list of zero or more MPMPolicy-Structure Objects (i.e., the list is made up of MPMPolicyStructureAtomic and/or MPMPolicyStructureComposite Objects). **[A1-D1].** If this Object does not contain any MPMPolicyStructure Objects, then a NULL MPMPolicyStructure Object **SHOULD** be returned.
setMPMPolicyList (in child-ObjectList : MPMPolicyStructure [1..*])	This operation defines a set of MPMPolicyStructure Objects that will be contained by this particular MPMPolicyStructureComposite Object. This operation takes a single input parameter, called childObjectList, which is an array of one or more MPMPolicyStructure Objects (i.e., one or more MPMPolicyStructureAtomic and/or MPMPolicyStructureComposite Objects). This operation first deletes any existing contained MPMPolicyStructure Objects (and their aggregations and association classes), and then instantiates a new set of MPMPolicyStructure Objects; in doing so, each MPMPolicyStructure Object is contained within this particular MPMPolicyStructure Composite Object by creating an instance of the MCMPolicyStructureHasPolicy aggregation. **[A1-D2].** Each created aggregation **SHOULD** have an association class (i.e., an instance of the MCMPolicyStructureHasPolicy association class).
setMPMPolicyPartialList (in childObjectList : MPMPolicyStructure[1..*])	This operation defines a set of one or more MPMPolicyStructure Objects that should be contained within this particular MPMPolicyStructureComposite Object WITHOUT affecting any other existing contained

	MPMPolicyStructure Objects or the Objects that are contained in them. This operation takes a single input parameter, called childObjectList, which is an array of one or more MPMPolicyStructure Objects. This operation creates a set of aggregations between this particular MPMPolicy-StructureComposite Object and each of the MPMPolicyStructure Objects identified in the childObjectList. **[A1-D3].** Each created aggregation **SHOULD** have an association class (i.e., an instance of the MPMPolicyStructureHasPolicy association class).
delMPMPolicyList()	This operation deletes all contained MPMPolicyStructure Objects of this particular MPMPolicyStructureComposite Object. This has the effect of first, removing the association class, and second, removing the aggregation, between this MPMPolicyStructureComposite Object and each MPMPolicyStructure Object that is contained in this MPMPolicyStructureComposite Object. This operation has no input parameters.
delMPMPolicyPartialList (in childObjectList : MPMPolicyStructure[1..*])	This operation deletes a set of MPMPolicyStructure Objects from this particular MPMPolicyStructureComposite Object. This operation takes a single input parameter, called childObjectList, which is an array of one or more MPMPolicyStructure Objects. This has the effect of first, removing the association class and second, removing the aggregation, between each MPMPolicyStructure Object specified in the input parameter and this MPMPolicyStructureComposite Object. **[A1-R2]**. All other aggregations between this MPMPolicyStructureComposite and other MPMPolicyStructure Objects that are not identified in the input parameter **MUST NOT** be affected.

Table A1-6. Operations of the MPMPolicyStructureComposite Class

There is one relationship defined for the MPMPolicyStructureComposite class. It was defined in section 8.7.2.4.

8 References

[1] MEF 95, *MEF Policy Driven Orchestration*, July 2021