SWIM TI Message Exchange Patterns: Identification Guidelines (v0.2)

# Message Exchange Patterns

## MEP Introduction

A message exchange pattern (MEP) identifies a repeatable sequence of messages exchanged between two systems, specifying the order, direction and cardinality of these messages (e.g. Request/Reply, Publish/Subscribe)

MEPs can only be understood in relation to a particular level of abstraction. Concretely, at TI protocol level the MEPs are rather basic, however, more sophisticated MEPs can be constructed based on these to enable the exchange needs required at application level.

* Primitive MEPs are those that are directly related to the capability of the lower level protocols of the SWIM TI. These protocols integrate the SWIM TI Service Interface Bindings. (i.e. HTTP, AMQP). The primitive MEPs provided by the SWIM TI are:
	+ Fire and Forget
	+ Synchronous Request/Reply
* Application MEPs. These are the MEPs that describe the information interactions at application level and that are implemented using SWIM TI primitive MEPS. They include[[1]](#footnote-1):
	+ Fire and Forget
	+ Synchronous Request/Reply
	+ Asynchronous Request/Reply
	+ Fan-out
	+ Publish/Subscribe Push
	+ Publish/Subscribe Pull
	+ Brokered Publish/Subscribe Push
	+ …

## MEP Definition

As defined by the W3C [2], a message exchange pattern (MEP) is a template, devoid of application semantics, that describes a generic pattern for the exchange of messages between agents. It describes relationships (e.g. temporal, causal, sequential, etc.) of multiple messages exchanged in conformance with the pattern, as well as the normal and abnormal termination of any message exchange conforming to the pattern.

## MEP Characteristics

Message Exchange Patterns can be classified[[2]](#footnote-2) according to certain characteristics of the message exchanges and the interaction between the systems involved.

### Cardinality

Cardinality describes the number of systems participating in the exchange of messages. We distinguish three classes of MEPs according to their cardinality:

* 1-1: The Message Exchange Pattern involves the exchange of messages between two systems.
* 1-many: The Message Exchange Pattern involves the exchange of messages of one system with one or more systems.
* Many-many: The Message Exchange Pattern involves the exchange of messages of multiple systems with multiple systems.

### Decoupling

#### Space

Space decoupling describes the characteristic of a MEP where the systems participating in the message exchange do not need knowledge of the network address of the other systems [1].

#### Time

Time decoupling characteristic describes a MEP where the systems participating in a message exchange do not have to be simultaneously active for the message exchange to proceed [1].

#### Process

Process decoupling characteristic describes the degree of decoupling of the system processes involved in the MEP. According to this characteristic MEPs are classified in two classes [1]:

* Synchronous: The processes of the systems participating in the exchange of messages stay blocked while the exchange occurs.
* Asynchronous: The processes of the systems participating in the exchange of messages do not stay blocked while the exchange occurs.

### Symmetry

Symmetry characteristic classifies MEPs taking into account the separation of roles placed into the systems and the reversibility of the MEP:

* Asymmetric: In asymmetric MEPs the systems participating in the message exchange have differentiated roles which cannot be exchanged. Asymmetric MEPs are most often used in Client-Server architectures.
* Symmetric: Symmetric MEPs do not enforce a strict separation of roles on the systems participating in the message exchange. Symmetric MEPs are most often used in peer-to-peer architectures.

## MEP Catalogue

This section provides a detailed view to the most common MEPs in the scope of the SWIM-TI; either as primitive MEPs directly supported by the SWIM TI protocols, or compositions on top of these in the form of application MEPs.

### Primitive MEPs (SWIM TI MEPs)

Primitive Message Exchange Patterns are the Message Exchange Patterns provided by the lowest level protocol that compose the service interface bindings. Primitive MEPs are used as basic building blocks to build more sophisticated application MEPs. These MEPs are also referred in the document as “SWIM TI MEPs” for correlating directly to the protocols that are part of the SWIM TI.

#### SWIM-TI Fire-and-Forget

Other names: “1-way”.

**Definition:**



Figure 5: Fire-and-Forget

1. A message is sent from SWIM-TI A to SWIM-TI B.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-1 |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Process Decoupling | Asynchronous |
| Symmetry | Symmetric |

#### SWIM-TI Synchronous Request/Reply

Other names: “Synchronous Request/Response”, “Request/Reply”, “Request/Response”.

**Definition:**



Figure 6: Request/Reply

1. A message (request) is sent from SWIM-TI B (Client) to SWIM-TI A (Server).
2. SWIM-TI B remains blocked awaiting for a response, SWIM-TI A remains blocked processing the response.
3. A message (reply) is sent from SWIM-TI A to SWIM-TI B.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-1 |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Synchronicity | Synchronous |
| Symmetry | Asymmetric |

### Application MEPs

Application message exchange patterns are defined in relation to the information exchange behaviour perceived at application level. These patterns relate to the message exchanges between an application and an information service and are built using the SWIM-TI MEPs as building blocks.

The following is a list of some of the most common application MEPs, service designers may customize or create their own application MEPs to satisfy their specific message exchange needs.

In order to simplify the description of the MEPs, the concept of SWIM Enabled Application is used to represent an entity that exchanges information in conformance with SWIM specifications (e.g. YP Spec).

To avoid cluttering the diagrams and improve readability we will use when beneficial the implicit return notation for the Request/Reply MEPs. In this convention the Reply message is annotated implicitly in the Request with the use of “: Reply”. The following diagram shows the two notations, no distinction is implied between them:



#### Fire-and-Forget

**Definition:**



1. A SWIM-TI Fire-and-Forget is sent from SWIM Enabled Application A to SWIM Enabled Application B.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-1 |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Synchronicity | Asynchronous |
| Symmetry | Symmetric |

#### Synchronous Request/Reply

**Definition:**

Figure 7: Synchronous Request/Reply

A SWIM Enabled Application Synchronous Request/Reply can be implemented in two ways using different SWIM-TI Primitive MEPs.

Alternative 1 (using SWIM-TI Synchronous Request/Reply):

1. A SWIM-TI Request is sent from SWIM Enabled Application B (Client) to SWIM-TI Enabled Application A (Server).
2. SWIM-TI Enabled Application B remains blocked waiting for a response, SWIM-TI Enabled Application A remains blocked processing the response.
3. A SWIM-TI Reply is sent from SWIM-TI Enabled Application A to SWIM-TI Enabled Application B.

Alternative 2 (using SWIM-TI Fire-and-Forget):

1. A SWIM-TI Fire-and-Forget is sent from SWIM Enabled Application B (Client) to SWIM-TI Enabled Application A (Server).
2. SWIM-TI Enabled Application B remains blocked waiting for a response, SWIM-TI Enabled Application A remains blocked processing the response.
3. A SWIM-TI Fire-and-Forget is sent from SWIM-TI Enabled Application A to SWIM-TI Enabled Application B.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-1 |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Process Decoupling | Synchronous |
| Symmetry | Asymmetric |

#### Asynchronous Request/Reply

Other names: “Asynchronous Request/Response”.

**Definition:**

 

Figure 8: Asynchronous Request/Reply

1. A Fire-and-Forget (request) is sent from SWIM Enabled Application B (Client) to SWIM Enabled Application A (Server).
2. A Fire-and-Forget (reply) is sent from SWIM Enabled Application A to SWIM Enabled Application B.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-1 |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Process Decoupling | Asynchronous |
| Symmetry | Asymmetric |

#### Fan-Out

 

Figure 9: Fan-Out

1. A Fire-and-Forget is delivered sequentially to multiple SWIM Enabled Applications.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-many |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Process Decoupling | Asynchronous |
| Symmetry | Asymmetric |

#### Publish/Subscribe Push

Definition:



Figure 10: Publish/Subscribe Push

1. A SWIM-TI Request/Reply is performed by one or more systems to declare interest in the publications (subscribe) of System A. The SWIM-TI Request/Reply can be either Synchronous or Asynchronous.
2. A SWIM-TI Fire-and-Forget[[3]](#footnote-3) is sent to each subscribed system when a new publication is available containing the publication.
3. Step 2 is repeated whenever there are new publications available.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-many |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Process Decoupling | Asynchronous |
| Symmetry | Asymmetric |

#### Publish/Subscribe Pull

**Definition:**

****

Figure 11: Publish/Subscribe Pull

1. A SWIM-TI Request/Reply is performed by one or more systems to declare interest in the publications of System A. The Request/Reply can be either Synchronous or Asynchronous.
2. A SWIM-TI Fire-and-Forget[[4]](#footnote-4) is sent to each subscribed system when a new publication is available to notify the subscribers of the existence of new publications.
3. Subscribed systems can perform a Request/Reply at their convenience to retrieve the publication.
4. Steps 2 and 3 are repeated whenever new publications are available.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | 1-many |
| Space Decoupling | NO |
| Time Decoupling | NO |
| Process Decoupling | Synchronous/Asynchronous[[5]](#footnote-5) |
| Symmetry | Asymmetric |

#### Brokered Publish/Subscribe Push

**Definition:**



Figure 12: Brokered Publish/Subscribe Push

1. A SWIM-TI Request/Reply is performed by one or more systems to declare interest in the publications received by the Broker from one or more publishing systems. The Request/Reply can be either Synchronous or Asynchronous.
2. A SWIM-TI Fire-and-Forget is sent to each subscribed system when a new publication is available containing the publication.
3. Step 2 is repeated whenever new publications are available.

**Notes:**

* The Brokered Publish/Subscribe MEP introduces a layer of decoupling between publisher and subscriber systems by means of a broker.
	+ Space decoupling: Publishers and subscribers do not need to know each other’s physical network addresses or even how many of them are.
	+ Time decoupling: Publishers and providers do not need to be available at the same time for the exchange to proceed as the broker can relay the messages when they are available.
* Figure 12 shows that the subscription process may occur at any point and the broker will adapt the publication process to forward the publication to all subscribers.
* A Brokered Publish/Subscribe Pull variant can be defined in an analogous manner.

**Classification:**

|  |  |
| --- | --- |
| Cardinality | Many-many |
| Space Decoupling | YES |
| Time Decoupling | YES |
| Process Decoupling | Asynchronous |
| Symmetry | Asymmetric |

# References

1. The Many Faces of Publish/Subscribe, Eugster et al.
2. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C
1. The list of application MEPs is not exhaustive. Due to the compositional nature of application MEPs new ones can be created or existing ones may be altered to satisfy specific application needs. [↑](#footnote-ref-1)
2. This classification and the resulting taxonomy are not unique and additional or different criteria can be used to create a different taxonomy of MEPs. [↑](#footnote-ref-2)
3. The Publication can also be implemented with a SWIM-TI Request/Reply MEP. [↑](#footnote-ref-3)
4. The notification can also be implemented using a Request/Reply MEP. [↑](#footnote-ref-4)
5. Both implementations are possible. [↑](#footnote-ref-5)