Framework - DDP BA

TMF518_FMW

Version 1.2

September, 2011

©TM Forum 2011

Notice

No recipient of this document and code shall in any way interpret this material as representing a position or agreement of TM Forum or its members. This material is draft working material of TM Forum and is provided solely for comments and evaluation. It is not “Forum Approved” and is solely circulated for the purposes of assisting TM Forum in the preparation of final material in furtherance of the aims and mission of TM Forum.

Although it is copyrighted material of TM Forum:

- Members of TM Forum are only granted the limited copyright waiver to distribute this material within their companies and may not make paper or electronic copies for distribution outside of their companies.
- Non-members of the TM Forum are not permitted to make copies (paper or electronic) of this draft material other than for their internal use for the sole purpose of making comments thereon directly to TM Forum.
- If this material forms part of a supply of information in support of an Industry Group Liaison relationship, the document may only be used as part of the work identified in the Liaison and may not be used or further distributed for any other purposes

Any use of this material by the recipient, other than as set forth specifically herein, is at its own risk, and under no circumstances will TM Forum be liable for direct or indirect damages or any costs or losses resulting from the use of this material by the recipient.

This material is governed, and all recipients shall be bound, by all of the terms and conditions of the Intellectual Property Rights Policy of the TM Forum (http://www.tmforum.org/Bylaws/1094/home.html) and may involve a claim of patent rights by one or more TM Forum members or by non-members of TM Forum.

Direct inquiries to the TM Forum office:

240 Headquarters Plaza,
East Tower – 10th Floor,
Morristown, NJ 07960 USA
Tel No. +1 973 944 5100
Fax No. +1 973 944 5110
TM Forum Web Page: www.tmforum.org

Table of Contents

List of Requirements

 ©TM Forum 2011

List of Use Cases

List of Figures

List of Tables

Executive Summary

This document entails the Business Agreement (BA) aspect of the MTNM / MTOSI Framework Document Delivery Package (DDP). It covers requirements and use cases concerning both the interface communication mechanisms and the general network resources aspects.

The following items are covered:

- Resource identification
- Functional modeling
- Resources data retrieval mechanisms
- Notification mechanisms

This document generalizes and extends the management requirements and use cases from TMF 513 v3.0 and TMF517. TMF 513 focuses exclusively on the NML-EML interface. However, this document considers the more general scenario of OS-OS communications with NML-EML as a special case.

1 Introduction

1.1 DDP Structure

In order to allow for more efficient release delivery, the previous monolithic BA, IA and SS documents have been partitioned into smaller self-contained (though not independent) units called Document Delivery Packages (DDPs).

This is similar to the 3GPP concept of Integration Reference Point (IRP). The basic idea is that the Interface, which is specified by the entire document set (of a release), is partitioned into DDPs where each DDP specifies “a certain aspect” of the Interface, which needs to be very clearly scoped.

There are three kinds of DDPs:

- the FrameWork DDP (FMW) – this DDP contains the generic artifacts that are applicable to all the other DDPs.
- Data Model DDP (DM-DDP) – a DDP that concerns a data model (entities, data structures, attributes, state, but no operations)
- Operation Model DDP (OM-DDP) – a DDP that concerns a computational model (operations, notifications, transactions) for a given functional area (such as resource inventory management)

The unified deliverables structure for any given MTNM / MTOSI product release is as follows:

- Product Release Notes:
  - a scope specification for the type and extent of the delivered product,
  - the partitioning of the release into DDPs (i.e., definitions of various aspects of the release),
  - and an overview of the release’s (delta) deliverables;
- For each DDP:
  - Business Agreements (Bas): a business view specification
  - Information Agreements (Ias): a system view specification
  - Interface Implementation Specifications (ISSs): implementation and deployment view specification per supported enabling technology (mapping of the IA to either CORBA (IDL, services usage) or XML (WSDL, XSD, bindings…)
  - Supporting Documentation: normative and informative supporting documents.
- Reference Implementation (optional) of core IIS fragments for selected interfaces and enabling technologies.

1.2 Document Overview

The purpose of the Framework Document Delivery Package (DDP) is to capture all the basic principles shared by the other DDPs of MTOSI and MTNM, and to describe them according to the underlying business requirements and design decision aspects in support of them.

The TM Forum Integration Program (which includes the MTOSI and MTNM work) is committed to the development of interfaces for managing network resources and connectivity and the associated services.

The Interface model consists of a generic view which is derived and developed at different abstraction levels with different interactions patterns, in order to cover both the OS-OS fine grained interactions and the coarse grained services, focusing on network resources management as well as on services management. Note that an OS in this context is any management system that exhibits Element Management Layer (EML), Network Management (NML) and/or Service Management Layer (SML) functionality. Such innovative modeling concepts allow the single interface using one model and consistent operations paradigm to manage many disparate network technologies (SDH/SONET, WDM, ATM, DSL, Ethernet, Wireless, etc.) even where intertwined within single network devices.

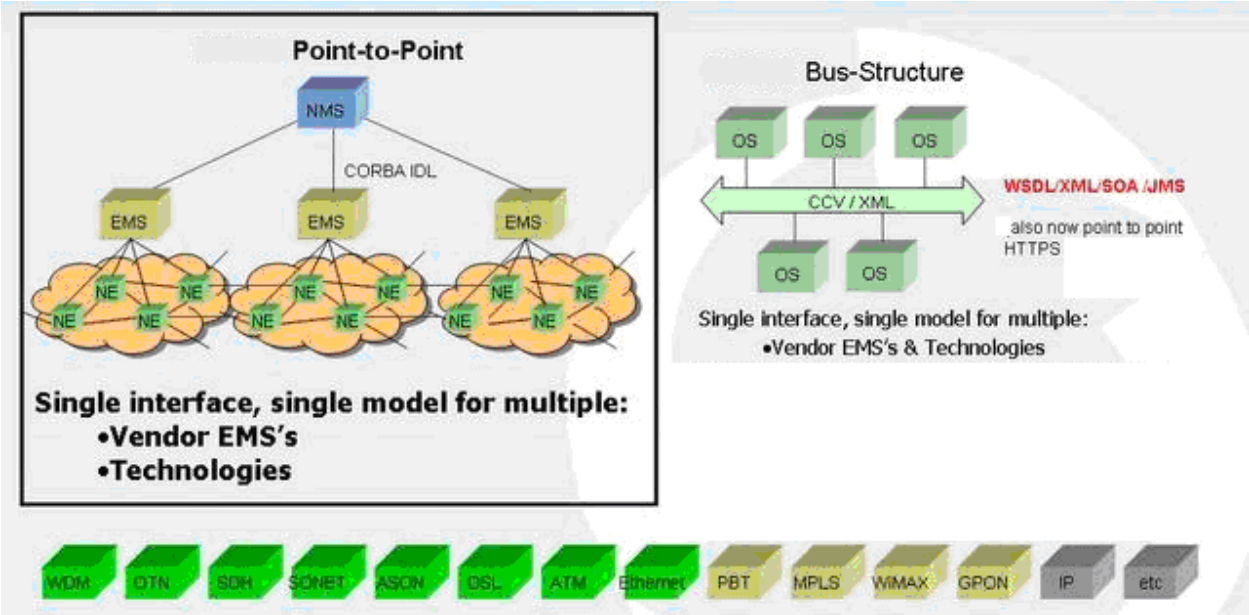


Figure 1-1. MTNM and MTOSI high level communication architecture

1.3 Document Structure

This document corresponds to the BA part of the framework DDP, containing all the aspects common to every MTNM / MTOSI interface.

The following sections are included in this document:

- Section 1 is this introduction.
- Section 2 defines the business problem and project scope
- Section 3 defines the requirements and associated descriptive text.
- Section 4 contains the use cases.
- Section 5 has traceability matrices between the use cases and the requirements.
- Section 6 provides a list of open issues to be considered in later versions of this document.
- Section 7 lists references and states IPR claims, if any.
- Section 8 provides administrative details such as document history and acknowledgements.

1.4 Terminology Used In This Document

Refer to the SD0-1 supporting document.

A complete set of examples relevant to the physical ports and to the logical inventory entities (PTPs, CTPs, FTPs) can be found in SD1-18, Functional Modeling Concepts and the related SD1-22, Modeling Components supporting documents.

2 Business Problem Description, Project Scope

2.1 Project Scope

The TM Forum Integration Program is responsible for all of the interface and business services work within the TM Forum. In some cases, interface work is delegated to other teams but the final verification for technical uniformity and integrity is the responsibility of the TM Forum Integration Program.

Initially, the TM Forum Integration Program was formed to coordinate the various existing TM Forum interfaces activities (as shown in **Figure 2-1**). In particular, the responsibility for maintaining MTOSI and MTNM is now covered by the MTOSI-MTNM Users Group which is a team within the TM Forum Integration Program. The long term plan (which is already well under progress) is to migration the various input work to a single harmonized suite of interfaces.

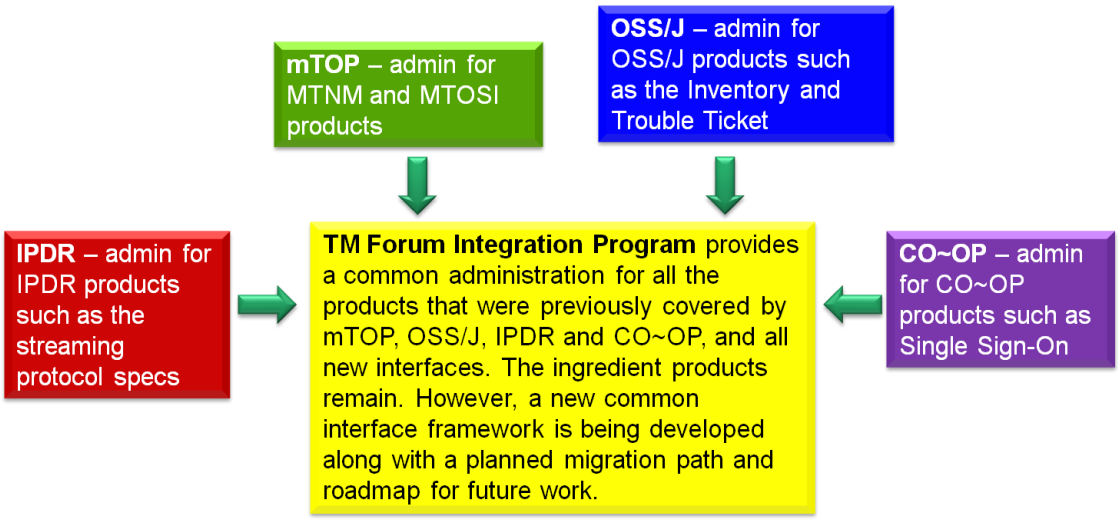


Figure 2-1. Inputs to the TM Forum Integration Program

Figure 2-2 provides a summary of the team within the TM Forum Integration Program as well as a few teams outside of the program but which also do some interface work. In terms of MTOSI and MTNM, the main input for updates come from the Resource and Service Management Team.

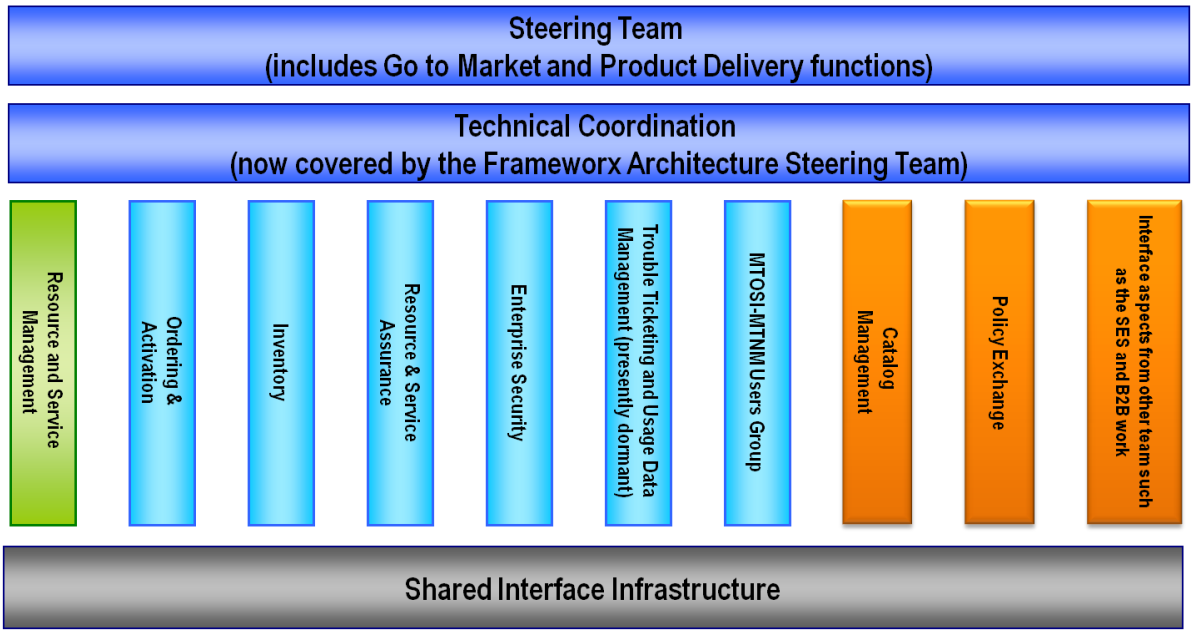


Figure 2-2. TM Forum Integration Program

2.2 Benefits

MTOSI and MTNM provide a set of Interface specifications that allow for resource and service management (with only MTOSI covering service management, but with MTOSI and MTNM both covering resource management, using very much the same information model).

These specifications are intended to lower design, implementation, Verification Validation & Testing (VVT), and maintenance costs for management interfaces. These Interfaces are intended for use by service providers, suppliers of equipment and OSS suppliers. The intention is to also encourage system integrator usage of management systems that make use of the Interfaces.

In particular, the followed approach tends to minimize the cost of integration, provide access to all necessary information and control, and support all vendor/operator differentiation. The intent of the interface is to provide compatibility among different version, for a detailed description see SD2-6_VersioningAndExtensibility.

2.2.1 Service Provider Benefits

The service provider benefits are as follows:

- One stop shopping concerning feature requests for much of the TM Forum contract specification work is part of the defined Change Control Group (CCG) process that TM Forum makes available in order to control the interface.
- The technical deliverables are also of high value to the service provider. The Interface specifications allow for an open, multi-supplier environment, shorten delivery times and lower integration costs.
- The MTOSI and MTNM products provide an integrated, multi-technology interface with support for most key layer 1 and layer 2 transport technologies. This is in contrast to earlier approaches where

each technology-specific forum provided a single-technology management interface. The service provider was faced with having to use many different, uncoordinated management interfaces.

- These products are not bound to any one middleware, transport or computing language. So, the service provider will be able to evolve to new technologies as they arise.

2.2.2 Supplier Benefits

The supplier benefits are as follows:

- Fewer Adapters leads to Lower Costs – in as much as MTOSI and MTNM gain market penetration (and there has already been significant market acceptance of these interfaces), the supplier is faced with the need to build fewer adapters between their products and the products of their partners. A supplier can also directly see cost savings in the use of the Interfaces among its own products (as the need for an open interface arises).
- Lower Middleware Transitions Costs – the Interfaces are defined to be middleware and transport independent. So, the supplier can migrate from one middleware or transport technology to another without changing the supporting business logic in the code.
- Increase Usage by System Integrators (Sis) – a supplier's support of their own "open" interfaces goes only so far to encourage Sis. Clearly, an SI would like to make use of supplier products (both equipment and OSS suppliers) that make use of well supported standard interfaces rather than supplier specific interfaces. The latter case forces the SI into a situation characterized by many pair-wise negotiations between various suppliers.
- Lower Training Cost – in as much as a supplier re-uses the Interfaces for multiple products and for multiple customers, the various training costs are lower because the designers, system engineers, developers and testers are using the same Interfaces over and over again.

3 Business Processes

3.1 Category I: Static and Structural Requirements

3.1.1 Common Information

The requirements for the management of network or service resources have been specified in terms of the entities that are visible across the interface and the service requests that may be invoked across the interface. The data model detailing these entities and the associated service requests is detailed in the relevant IA DDP sections, while the BA DDP sections contain the requirements for the visible data entities and the associated service requests.

Among its functions, the Common Communications Vehicle (CCV) is used to publish new entities. In the following paragraphs, it should be noted that the OS managed entity can be used to represent an EMS, NMS or SMS. A top-level OS has direct access to the CCV. Subordinate Oss access the CCV indirectly via the top-level OS.

For each entity instance published on the CCV, there is a publishing OS and a naming OS. The publishing OS first announces the existence of the entity instance on the CCV. The naming OS stores and names the entity instance. The “name” of the entity could be the same as the “discovered name” (the name used to announce the entity) but it does not have to be.

For any particular entity, the discovering OS and the naming OS need not be the same OS. Furthermore, an entity may be stored by more than one OS. Note that the use of the word “store” is used to distinguish from the concept of “steward” (or owner).The usage of “store” in this context means that the OS has information about an entity. Multiple Oss on the CCV may store information (maybe even the same exact set of information) about the same network entity. However, it may be desirable to have a single steward for each network or service resource.

Note: An attribute is mandatory unless otherwise stated, therefore it shall always be found at the interface within relevant resources, independently from its value set (i.e., an empty value could be allowed). When an attribute is described as optional, this means that the attribute need not be present at the interface within the relevant resource.

R_TMF518_FMW_I_0001	All entities visible across the Interface shall have the following attributes: • ***name –*** this attribute represents a unique identifier for the entity instance on the CCV. Once set by the naming OS, the value of the name is invariant for the life of the entity and should be used when referring to the entity over the CCV. The name attribute of an entity instance shall always be assigned a value, unless the entity instance is first discovered by an OS that is not the naming OS for the entity instance. In such a case, the discovering OS shall provide a value for the Discovered Name attribute of the entity. The naming OS shall provide a value for the Name attribute of the entity once it becomes aware of the new

	entity instance. • ***discovered name*** – this is the name of the entity when its existence is first published on the CCV, but only in the case where the OS that publishes the entity on the CCV is not the naming OS. If the naming OS first publishes the entity on the CCV, this attribute may be left empty. The discovered name is assumed to be unique from the perspective of the publishing OS. The discovered name published on the CCV does not necessarily need to be the name of the entity as discovered by the OS from the subtending network. There are no specific rules concerning the relationship between the "name" attribute and "discovered name" attribute of the entity. They could be set to the same value or they could be different. • ***namingOS*** – this attribute represents an identifier for the steward of the entity, intended as the OS that sets the name of the entity. The namingOS attribute is set by the OS that is responsible for setting the "name" of the entity. It is assumed that each entity (that is published on the CCV) has a unique naming OS. This attribute is optional in a point to point NML-EML paradigm. • ***aliasNameList*** – this attribute contains a list of aliases for the entity. The attribute is a list of name-value pairs. The name refers to the type of alias (e.g., nativeEMSname) and the value component holds the alias itself. There are a number of other names that can be applied to a network entity, such as *nativeEMSname*, intended as the name of the entity as presented on an EMS GUI, or the *nativeNEname*, intended as the name of the resource as displayed on a (possibly remote) terminal connection to an NE. Another case could be the Transport Network Assigned Address (TNA), intended as the name of an access point as represented to a client of a control planed based network. It allows users to identify UNI interfaces for call setup without the service provider exposing the internal addresses of the network. Some or all of these names may need to be associated with an MTOSI entity, in the aliasNameList. • **user label** – this attribute represents the "user friendly" name for the entity. The user label is filled by the naming OS. This attribute is optional. • **owner**: this attribute represents an identifier for the owner of the entity. This attribute is optional. • ***vendorExtensions*** – this attributes allows vendors to further qualify and extend an entity's characteristics. The use of this attribute is detailed in Section 3.4.3.
Source	TMF517 rel.1.1 merged requirements; vendorExtensions is new

3.1.2 Functional modeling concepts

For the network resources exposed at the Interface, the layering concept of the ITU-T abstract network model has been efficiently combined through encapsulation with the real behaviour of network nodes, in order to optimize the information transfer across the Interface.

⇒ *All the relevant information, the rules and the detailed examples of the followed approach are available in the SD1-18 Supporting Document.*

3.1.3 Coarse grained approach

A coarse grained approach has been developed for the potentially higher abstraction level needed to manipulate the network resources and their structural grouping at different management layers. This basic choice has been motivated by both efficiency and performance goals and by loosely coupling objectives in between the Interface and the OS internal data model.

3.1.4 Resource naming requirements

These requirements are to address the situation where an OS that discovers a network resource is not the one responsible for naming the object used to model the resource.

Many of the current OS products are split along the lines of network inventory management, and network discovery/activation.

In some cases, a service provider will have a network inventory management OS from one supplier and a network discovery/activation OS from another supplier.

In case of this multiple OS environment, the following steps may be used to plan, inventory and discover new network entities such as managed elements and topological links:

1. A new network entity is planned and then entered into the database of the network inventory OS. At this point, the network entity has not been installed in the network. For this example, the network inventory OS will be the naming OS of the network entity and as such will get to set the “name” attribute for the network entity.
2. At some later point, the planned network entity is actually installed.
3. The network discovery OS then discovers the newly installed network entity and publishes the information on the CCV (using the discovered name of the network entity).
4. The network inventory OS retrieves the announcement off the CCV.
5. The network inventory OS matches the newly announced network entity with a network entity already in its database. This point depends on a particular solution implementation based on service provider naming conventions. It should be noted that this naming approach is only one that could be taken. For example the matching could be based on the values of several object attributes that model unique physical properties of the resource (e.g., part # and serial #). Anyway, and it is not in the scope of Interface, but it is reported here because it is a significant aspect of the decision of having a name and a discovered name for network resources.
6. The network inventory OS publishes the new network entity on the CCV, using the name provided by the service provider when the network entity was initially planned.
7. From this point on, the other Oss on the CCV will use the name of the network entity provided by the naming OS.

R_TMF518_FMW_I_0002	Network resource names shall be unique across the name spaces of all the top-level Oss associated with a particular instance of a CCV.
Source	TMF517 version 1.1

R_TMF518_FMW_I_0003	Each entity on the CCV is given a unique name (represented by the value of the “name” attribute) by exactly one OS. This OS is referred to as the “naming OS” for the entity.
Source	TMF 517 v 1.1 Requirement I. 8

The Management Domain is an entity which has been introduced solely for the purpose of guaranteeing uniqueness of names in the Interface space.

Figure 3-1 represents the relationship among Management Domain, OS and other managed entities (i.e., Managed Element, TMD, Subnetwork and Topological Link).

In cases where the top-level OS does expose the same subnetwork partitioning as the various subordinate Oss, it makes sense to have a “manages” relationship between the OS and the subnetwork. When the top-level OS exposes a different subnetwork partitioning than the subordinate OS, then the “manages” relationship between the subordinate Oss and the subnetworks would not be provided.

The top-level OS manages zero or more management domains, and a single management domain may be managed by one or more Oss. The relationship between the top-level OS and the subtending Managed Elements (Mes) and Topological Links (TLs) can be inferred from the ME containment under management domain.

If one wants to know the relationship between the subordinate Oss and the Mes, subnetworks (only if the top-level OS and subordinate Oss have the same subnetwork partitioning), Transmission Descriptors (TMDs) and TLs that they manage, then a “manages” relationship needs to be made explicit (as shown in the **Figure 3-1**).

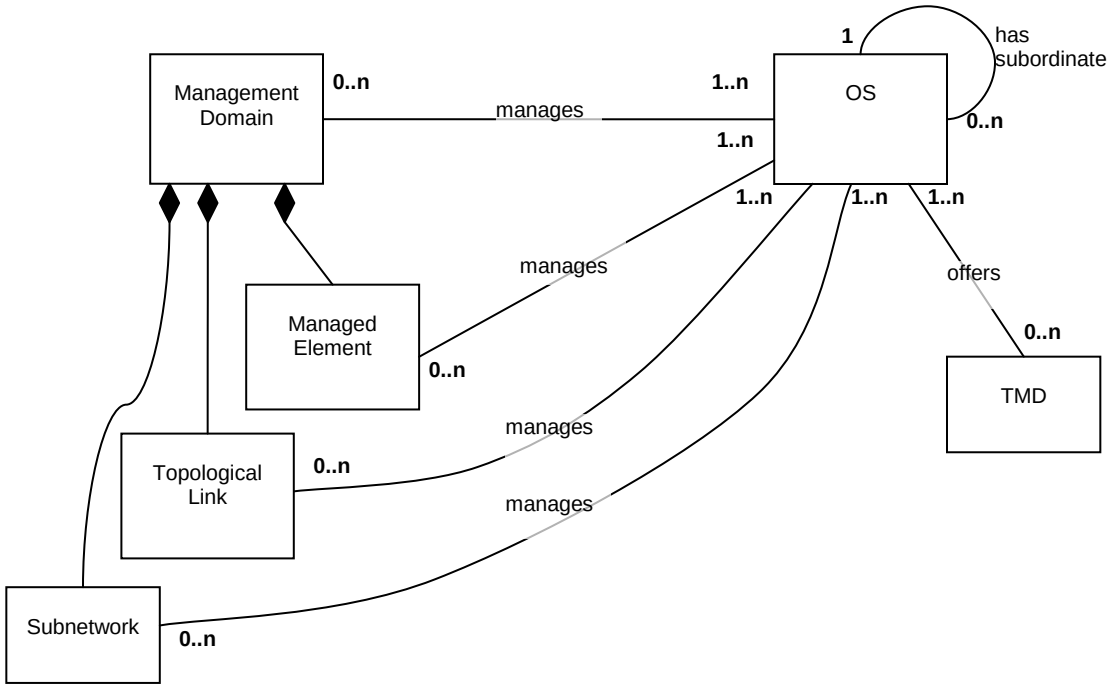


Figure 3-1. "Manages" relationships for OS

R_TMF518_FMW_I_0004	One or more management domains are used to represent the managed network of a service provider in the context of a given CCV. Each network resource is contained under exactly one management domain. Each OS on the CCV may store a part or the whole inventory associated with one or more management domains.
Source	TMF517 version 1.1

In addition, the network Administrator can decide which resources are administratively assigned to which administrative domain. The interface shall allow for retrieval of this attribute, which can be assigned also to a Management Domain. A particular case is the managing of Virtual Private Network services.

R_TMF518_FMW_I_0005	Network Access Domain. It represents a domain to which the Management Domain resources are associated.
Source	TMF517 version 1.1

Management Domain at root of Naming Tree: Several alternatives for the root of the naming tree were discussed. Initially, it was proposed to have an OS entity as the root. In this scheme, the OS entity would represent both the OS as a system and the network managed by the OS. This is similar to the approach

taken in the NML-EML naming scheme, where EMS is at the top of the hierarchy. However, it was noted that this approach does not fit well in cases where a backup OS in a shared bus communication architecture may also manage the same network (objects representing network entities would have 2 names on the CCV in this case). A variation of this approach was also considered, i.e., OS at the top of the naming tree with EMS below OS. All naming below the EMS would be exactly the same as the MTNM naming scheme (an advantage of this approach). Unfortunately, this approach has the same issue as the aforementioned approach.

Using management domain as the root of the naming tree was then considered. This approach works in the case of multiple management systems (primary and backup) for the same network. This approach is also extensible to the case where an OS has knowledge of (or manages) more than one service provider network (there may be one instance of management domain for each service provider network). It should be emphasized, however, that multi-provider management is not within the scope of the initial version of the Interface.

It was decided to have one or more instances of “Management Domain” at the top of the naming tree with the following exceptions, which are also at the top of the naming tree:

- Operations System (OS); since an OS can manage part or all of several Management Domains, and a Management Domain can be managed by several Oss, it is not possible to relate OS and Management Domain via containment.
- Transmission Descriptors (TMD), Alarm Severity Assignment Profiles (ASAP) and, generally speaking, all the profile type entities; since there are several contexts under which a profile can be classified, it is not possible to constrain it to a Management Domain, but instead the relevant context shall be associated to this network resource. Examples of profile contexts are:
    1. (ME Product) The equipment vendor product or product line is one possible context. For example, a TMD may be associated with a given ME product type (e.g., SDH ADM XYZ) from vendor ABC.
    2. (Service Provider Product) Another context is based on service provider product. For example, a given set of TMDs (perhaps related to multiple ME products) might be used for a given product (e.g., Premium Ethernet LAN) provided by a service provider.
- Connectivity partitioning entities, i.e. subnetworks; since the MD is a grouping concept with respect to naming and subnetwork is a grouping concept with respect to connectivity, the two grouping concepts should not limit or restrict one another.

Entity Naming Directly Under Management Domain: An instance of the management domain entity resides at the root of the naming tree. All the entities that are directly named under “EMS” in the MTNM product are named under Management Domain in the MTOSI product.

Enhancements have been done to the naming hierarchy in the new (MTOSI) scheme with respect to the previous (MTNM) scheme.

R_TMF518_FMW_I_0006	1. The naming scheme for the entities represented over the MTOSI / MTNM interface follow the same containment hierarchy at the OS-OS level and at the EML-NML level, with the following exception(s): 2. The highest entities in the naming tree are the Management Domain and the OS (they replace the “EMS” entity used in the MTNM old containment tree). The name value of a management domain is a free

	format string. The OS naming hierarchy is completely separated from the Management Domain naming hierarchy. There is no conventional naming relationship between them. The assigned name (i.e., the value in the "name" attribute) of a network entity must be used on the CCV every time that an OS refers to that resource. 3. A given network entity is only named under a single management domain. This follows from the assumption that each network entity has a unique name of the CCV. 4. Topological link, subnetwork, Flow Domain and managed element are named directly under Management Domain. The MatrixFlowDomain is named under the ManagedElement. 5. Operations System entities are not named under Management Domain because one OS can manage more than one Management Domain. 6. TMD entities are not named under Management Domain because TMDs and in general all profiling mechanisms are related more to a specific vendor instead of a specific management domain or management system.
Source	TMF 517 v 1.1 Requirement I. 9

Note: additional information and pictorial representations related to the inventory layout structure are presented in SD2-12_Resource_Inventory_Layout.

3.1.5 Interface versioning

Backward and forward compatibilities can be defined as follows:

- Backward compatibility means that a newer version of a client OS can be rolled out in a way that does not break existing server Oss. A server OS can send an older version of a message to a client OS that understands the new version and still have the message successfully processed.
- Forward compatibility means that a newer version of a server OS can be rolled out in a way that does not break existing client Oss. Of course the older client Oss will not implement any new behavior, but a server OS can send a newer version of an instance and still have the instance successfully processed.

Note that backward and forward compatibility are defined from the perspective of an OS playing the client role.

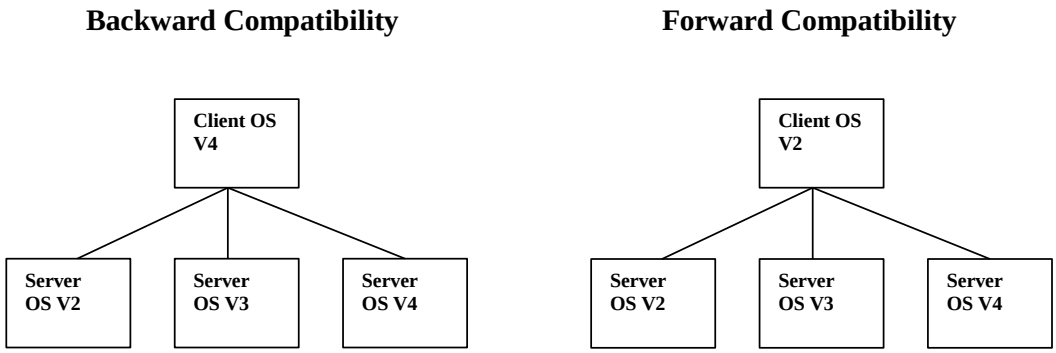


Figure 3-2. Interface Compatibility

A major update is a change for which no meaningful communication is possible with respect to the change. A message processor will not be able to natively process a message after a major update. After a major update the service will loose its backward compatibility.

A minor update is a change after which it is still possible for a processor to natively process a message. After a minor change the service will still be backward compatible.

Interface version identifiers are of the form “N.M” (or N_M), where “N” represents the major version number and “M” represents the minor version one: each time a major update is introduced in an interface the major version number should be incremented.

The interface shall be developed in order to fully support backward and forward compatibility for any minor version. It should not prevent support of multiple major versions.

R_TMF518_FMW_I_0007	A unique identifier, referred as the Interface Version Number, shall be assigned to a specific version of the interface.
Source	TMF513 Requirement I.030

R_TMF518_FMW_I_0008	The Interface shall fulfill the following requirements: • Support of multiple versions of the interface, i.e. allowing interacting Oss to be upgraded independently; • Support of Naming Context conventions.
Source	TMF513 Requirement I.031

Note: The only statement that can be made here is that the Interface will not be the piece that will prevent the upgrade scenarios. In addition, when changes are made to the Interface and those changes are not supported in an application, it will mean that the full functionality of the Interface is not available or applicable.

3.1.6 Notifications

R_TMF518_FMW_I_0009	The Interface shall support a reliable mechanism for an OS to distribute notifications.
Source	TMF513 v3.0, Requirement I.041

R_TMF518_FMW_I_0010	All event notifications shall have the following attributes: • Identifier – this attribute shall represent an identifier for the event. The value of this identifier is not guaranteed to be unique. • Source Time – this attribute shall represent the time at which the event occurred at its source
Source	Amended from TMF513 v3.0, Requirement I.068

R_TMF518_FMW_I_0011	In addition to those attributes identified in R_TMF518_FMW_I_0010 a number of notifications may also have the following attributes: • Object Name – this attribute shall represent the name of the entity against which the event notification is generated • Object Type – this attribute shall represent the type of the entity against which the event notification is generated • OS Time – this attribute shall represent the time at which the event occurred at the notifying OS.
Source	Amended from TMF513 v3.0, Requirement I.093

Note: Source Time always retains the same value until the original alarm is just forwarded by several OS in the bus. In cases where an alarm may be transformed, if the original alarm is changed in a new event a new Source Time for the new event is created.

3.1.6.1 Notification Types

The requirements in this section pertain to the simple event notifications (to be clearly distinguished from Multi-Event Inventory Notifications), e.g., the creation of a single object.

MTOSI / MTNM products in fact provide both simple event notifications, e.g., creation of a particular object or attribute value changes for several attributes of a given object, as well as mechanisms for efficiently conveying all the changed information in single notification, using an inventory layout instance (the Multi-Event Notifications, see 3.2.7).

R_TMF518_FMW_I_0012	An ***Object Creation Notification*** is an event used across the Interface to indicate that an object has been created.

	An ***Object Deletion Notification*** is an event used across the Interface to indicate that an object has been deleted. An ***Object Discovery Notification*** is an event used across the interface to indicate that an object has been discovered (but not yet named). An ***Attribute Value Change (AVC) Notification*** is an event used across the Interface to indicate that one or more of the attribute values of an object have changed. A ***State Change Notification*** is an event used across the Interface to indicate that a state transition has occurred. A ***File Transfer Status Notification*** is an event used across the Interface to indicate the status of file transfer. A ***Heartbeat Notification*** is an event used across the Interface to indicate the state of the notification delivery mechanism between the sending and receiving Oss. A ***Software Backup Status Notification*** is an event used across the Interface to indicate that status of the backup of Managed Element (ME) data.
Source	TMF513 v3.0, Requirement I.042

Rationale for Design Decisions Concerning Object Discovery Notification (OdscN)

An Object Discovery Notification (OdscN) is an event used across the Interface to indicate that an object has been discovered but not yet named by the naming OS, in case the discovery and naming OS are different Oss. This OdscN is provided for communication between a discovery OS (i.e., an OS that first announces a new network entity on the CCV and a naming OS (the OS which provides a unique MTOSI name for the object) in order to let the naming OS know that a new entity is present.

The example scenario shown in **Figure 3-3** is used to clarify how the OdscN can be used. Within this example, the following assumptions are made:

1. The Discovery OS provides adaptation between the Interface used on the CCV and the management protocols used to communicate with the network. The Discovery OS could be an EMS (for example).
2. Only the Discovery OS has access to the network.
3. The Activation OS creates SNCs via requests to the Discovery OS which translates and forwards the request to the subtending network.
4. The Fault Management OS, for example, can turn alarm reporting “on” or “off” for particular entities via a request to the Discovery OS.
5. There is only one management domain for the entire network.
6. The physical inventory Oss is meant to store network resources inventory.

It should be noted that the Oss shown in the figure are just examples. The specifications do not prescribe OS types they specify an Interface that can be used by a variety of OS types for resource management and service management..

The new network entity can be announced on the CCV in several different ways. In the first method, a planned network entity is named and announced on the CCV before the network entity is actually installed. The steps are as follows:

1. A network entity is first entered into the Inventory OS.
2. The Inventory OS then creates an object to represent this network entity and puts the object into the “Planning” resource state. At this point, the network entity has not yet been installed.
3. The Inventory OS sends an Object Creation Notification (OCN) to (at least) the Discovery OS, which previously registered for OCNs.
4. At some later point, the network entity is actually installed.
5. The installed entity is detected by the Discovery OS which issues an AVC notification on the resourceState of the network entity (change from “Planning” to “Installing”).

This approach can be followed only if the Discovery OS is able to keep track of the planned entities, and moreover if the network entities are planned and named at the Interface before they are discovered.

Alternately, the new network entity is first announced on the CCV by the Discovery OS **before** it is given a unique name. The steps are as follows:

1. A network entity has not yet been announced by the Inventory OS at the Interface (it may or may not be in the Inventory OS database), and it is directly detected by the Discovery OS, which it is not authorized to name it.
2. The Discovery OS should send a notification, but it can’t be an OCN (an OCN should only be sent by the naming OS). Therefore, the Discovery OS sends an OdscN to the notification service.
3. The OdscN is received by the Inventory OS (which also plays the role of naming OS for the object in question). Assuming that only the Inventory OS has subscribed to receive OdscNs.
4. Assume that the Inventory OS matches the OdscN notification with a planned object in its inventory.
5. The Inventory OS provides a name for the new object, and then issues an OCN.

The OCN is received and processed by the Discovery, Fault Management and Activation Oss. All Oss on the CCV (from this point on) will use the name provided by the Inventory OS to refer to the newly installed object.

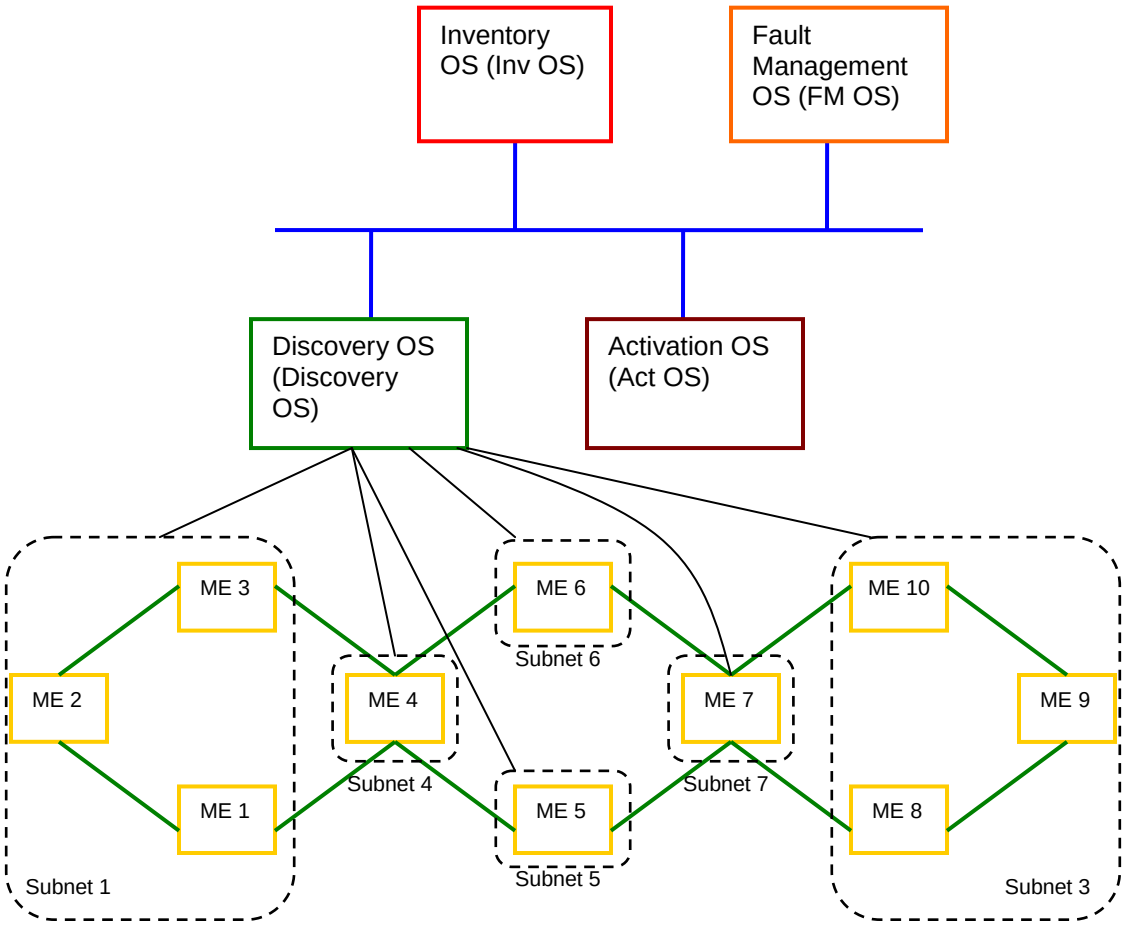


Figure 3-3. Network and Associated Management Architecture

R_TMF518_FMW_II_0056	The Interface shall support subscription to notifications by interested parties related to • the creation of Managed Domain instances • the deletion of Managed Domain instances • the discovery of Managed Domain instances • the attribute value change of Managed Domain instances
Source	TMF518_FMW, Version 1.1

R_TMF518_FMW_II_0057	Each top-level OS is expected to publish a creation notification for itself and its associated management domain/s when it is first associated with the CCV.
Source	TMF 517 Version 1.2, Requirement II.29

3.2 Category II: Normal Sequences, Dynamic Requirements

3.2.1 General Interface aspects

R_TMF518_FMW_II_0014	The Interface shall be used between Operations Systems (Oss) within the same administration (i.e., service provider). • Use of the Interface between administrations is for further study. • "OS" refers to any management system covering SML, NML, and/or EML functionality.
Source	TMF517 rel.1.1 BR1

R_TMF518_FMW_II_0015	The Interface among Oss shall function independently of how the relevant Oss manage the underlying network. The Interface between Oss shall be independent of the interface(s) between the Oss and the network resources.
Source	TMF517 rel.1.1 BR3

Figure 3-3 shows some possible management arrangements among a collection of Oss using the Interface. Each OS manages part or all of one or more Management Domains, i.e. a collection of network resources such as EMSs, Nes, equipment and subnetworks. There is a single Information Model that contains a Data component and an Operations component. The Common Communication Vehicle (CCV) allows different (management) systems to communicate. An instantiation of the CCV could be a JMS bus or a point-to-point HTTP association. Although not shown in the figure, an EMS could also be directly attached to the CCV.

A CCV instance represents a physical association among Oss. In particular, the Oss residing on a CCV are capable of communicating with each other. A subset of the OS on a CCV that share a common business purpose and support a common (possibly distributed) information model instance form a relationship referred to as an MTOSI / MTNM Business Association (MBA), which is strongly based on the web services standards.

Multiple orthogonal MBAs can be supported by the same CCV instance. However, Oss belonging to an MBA must reside on the same CCV so that they can communicate. The MBA can be seen as a logical overlay partition on top of a CCV.

R_TMF518_FMW_II_0016	The CCV shall fulfill the NGOSS requirements for a common communication mechanism (for reference, see TMF 053, The NGOSS Technology-Neutral Architecture).
Source	TMF517 rel.1.1 BR4

R_TMF518_FMW_II_0017	The CCV shall support both a Publish-Subscribe and a Point-to-

	Point communication mechanism where: ⇒ A Publish-Subscribe communication mechanism delivers published messages to all subscribers on the CCV. ⇒ A Point-to-Point communication mechanism delivers each message to exactly one consumer on the CCV.
Source	TMF517 rel.1.1 BR5

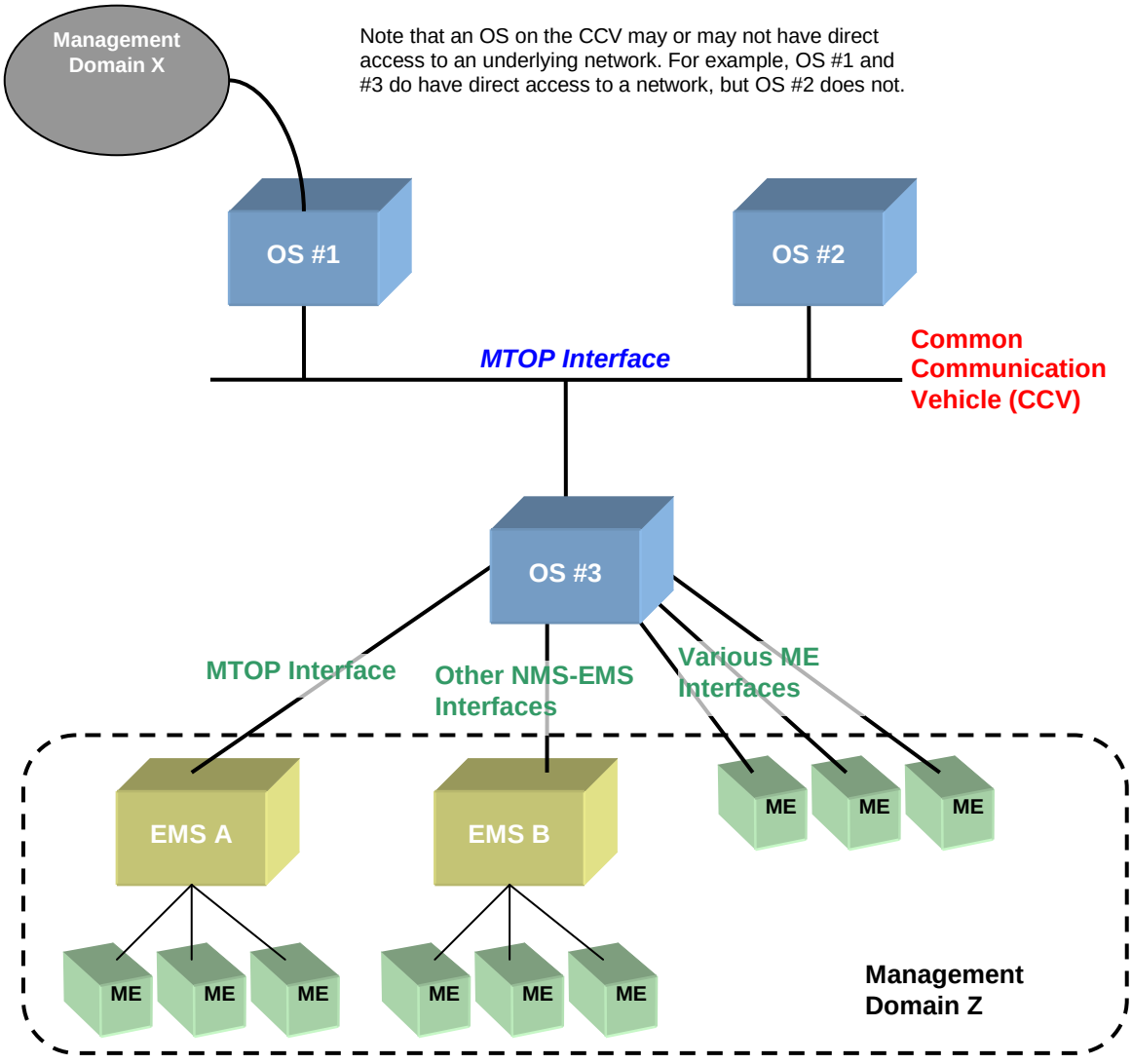


Figure 3-4. MTOSI / MTNM Reference Model

Figure 3-4 provides an example of an OS communications network configuration reference model. As noted previously, it should be emphasized that the specifications do not to define particular OS types. The operations and notifications defined in the Interface specification can be used or offered by various OS types, but the Interface specifications do not define or constrain the specific types of Oss that can use or offer the Interface.

Figure 3-4 shows several Oss and EMSs interacting via the Interface over a CCV inside the corresponding MBA. Only the management systems connected to the CCV would interact directly via the Interface. As shown in the figure, some of the management systems on the CCV also manage underlying EMSs and Mes via management protocols using many different interfaces (i.e. SNMP, TL1 etc.).

Typically:

- Fault Management OS or Activation OS retrieves inventory from the Inventory OS,
- Inventory OS synchronizes inventory with another Inventory OS,
- Inventory OS retrieve inventory from an EMS,
- Fault Management OS receives alarms (perhaps root cause alarms) from another Fault Management OS.

It is also possible to have several systems of the same type, e.g., fault management. So, for example, one fault management OS may handle DWDM and SDH equipment, and the other fault management system may handle ATM equipment. Another possibility for having more than one OS of the same type could be redundancy. Yet another possibility is to have an OS be a Manager of Managers (MoM) with respect to other OS, e.g., a generic fault management OS could manage several technology specific fault management Oss via the MTOSI / MTNM.

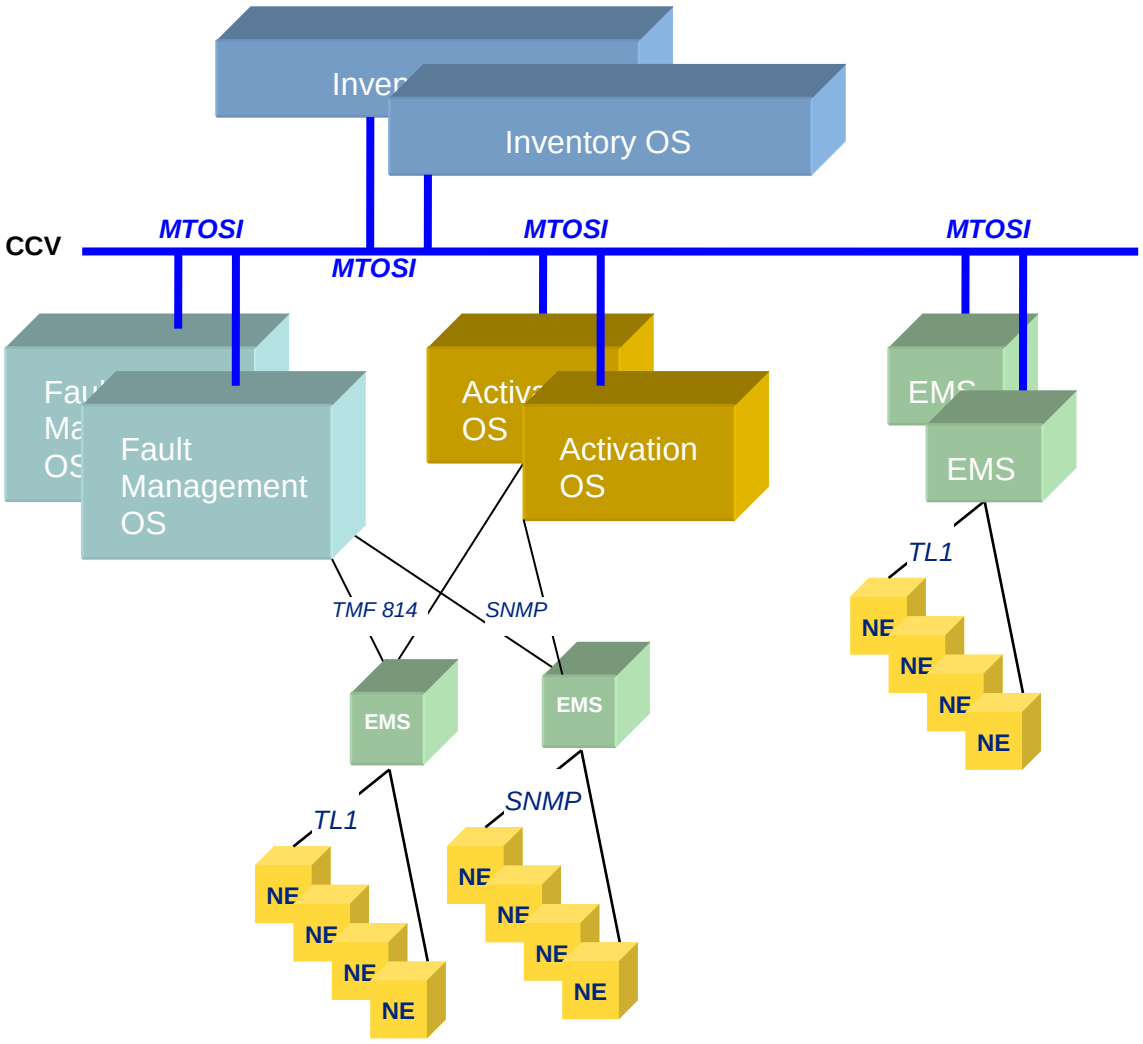


Figure 3-5. Example of the MTOSI / MTNM Reference Architecture

3.2.2 Interface transport independence and transport protocol bindings

R_TMF518_FMW_II_0018	The Interface specification shall be transport protocol independent. Being transport independent will allow replacing the underlying transport without changing the application code (and the application logic) of both the OS client and OS server.
Source	SD2-5_Communication_Styles

3.2.3 Communication architecture

R_TMF518_FMW_II_0019	The Interface must take into account scalability and performance issues. Real world scenarios shall be applied to check for information scale (MIB instances) and operations intensities.
Source	TMF513, IV.001

3.2.3.1 Interaction models

The Interface needs to support both point to point interactions as well as a multi-point to multi-point approach.

The point-to-point interaction is reflected in the Interface throughout the session model, i.e., emsSession, nmsSession and emsSessionFactory.

The management of multiple Clients access to the interface could involve the concept of a session. In a stateful kind of interaction between Client and Server, in which each subsequent Client request depends on the state of the previous one, a session should be maintained for each Client. In a stateless kind of interaction, there is no need to manage a session, as each Client request is totally independent of the previous one.

The Interface messaging paradigm is oriented towards stateless interactions and as such, it has no need for a session management.

R_TMF518_FMW_II_0020	The MTOSI / MTNM shall allow a Server OS to support multiple Client Oss simultaneously
Source	TMF517, Requirement IV.1

R_TMF518_FMW_II_0021	The MTOSI / MTNM shall allow a Client OS to use the services of multiple Server Oss simultaneously.
Source	TMF517, Requirement IV.2

3.2.3.2 Communication styles

⇒ *For a complete analysis, see* SD2-5_Communication_Styles.

3.2.4 Interface capability discovery

R_TMF518_FMW_II_0022	The Interface shall allow an OS to retrieve the, operations, notifications, alarms and other capabilities supported by a given OS.
Source	TMF513 v3.0, Requirement II.146

3.2.5 Filtering and attribute matching

The Interface is required to support querying of information based on a general matching attribute capability.

Therefore the Interface shall support the ability to allow the specification of attribute matching criteria.

R_TMF518_FMW_II_0023	The following attribute matching evaluations criteria shall be supported: 1. PRESENCE – a given attribute can be evaluated as to whether it appears or not. This evaluation can be applied to any attribute type. Evaluation criterion evaluates to TRUE if and only if such an attribute is present. 2. EQUALITY – the attribute value can be evaluated for a matching against an established value. This evaluation can be applied to any attribute type. Evaluation criterion evaluates to TRUE if and only if the value supplied in the filter is equal to the value of the attribute 3. COMPARISON – the attribute value can be evaluated for comparison to a requested value (i.e., greater than, greater than or equal to, less than, less than or equal to). This evaluation can be applied to real numbers. Evaluation criterion evaluates to TRUE if and only if the predicate is satisfied. 4. SUBSTRINGS – the attribute value can be evaluated for the presence of substrings. This evaluation should be applied to strings and can be qualified to specify where the corresponding substring should exist in the value of attributes respectively. Evaluation criterion evaluates to TRUE if and only if the criterions described as below are satisfied: a. If the evaluation is qualified with the location of any string, it shall appear in the corresponding attribute value; b. If the evaluation is qualified with the location of

	initial string, it shall match the first element in the corresponding attribute value; c. If the evaluation is qualified with the location of final string, it shall match the last element in the corresponding attribute value; 5. SETOF– the attribute value can be evaluated for matching any items provided in a set. It can be qualified with SUPERSET, SUBSET or NON-NULL INTERSECTION value to specify what comparison rule should be adopted. This evaluation can be applied to real numbers or strings. Evaluation Criterion evaluates to TRUE if and only if the criterions described as below are satisfied: a. If the evaluation is qualified with SUPERSET, all members of the attribute should be present in the given set in the filter; b. If the evaluation is qualified with SUBSET, all members of the given set in the filter should be present in the attribute; c. If the evaluation is qualified with NON-NULL INTERSECTION, at least one member of the given set in the filter should be present in the attribute;
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0024	Individual attribute matching evaluations reported in R_TMF518_FMW_II_0023s can be combined via the following logical operators: AND, OR and NOT. These logical operators should comply with the following priority: not, and, or.
Source	TMF518_FMW, Version 1.0

3.2.6 Multiple Action Requests and Transactions (MART)

The Interface is required to support the ability to allow a Client OS to perform multiple operations on multiple objects using a single request to a Server OS.

The requirements in this section pertain to a massive and orchestrated operational scenario that has two main objectives:

1. Efficiently apply the same operation to a large number of objects of the same type, e.g. create 500 ATM SNCs, where each action/object pair is regarded as an independent transaction.
2. Apply sets of operations to sets of objects as a single transaction. The transactional logic is delegated to the server OS. This is useful in performance tasks such as the configuration of a managed element and all associated equipment and ports.

Although these objectives have distinct behaviors, they share common characteristics:

- The request may contain multiple actions and/or objects.
- The request may have a long duration and so require additional control and status facilities.

The requirement is to be able to group a set of operations (actions and objects) into a single request. The request shall also include an indication of the required execution and transactional behavior. The server OS shall return a single response containing a list of the results, one result per operation.

In principle, the net results of a successful MART request will be the same as if the individual operations had been requested separately in the equivalent order. Of course, this is not an Interface requirement, but it is more an expectation on the behavior of the Server OS.

R_TMF518_FMW_II_0025	The Interface shall allow a server OS that receives a MART request from a client OS to report the results of each operation within a MART response to the client OS. The client will be informed of the outcome of each individual contained operation within a MART request.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0026	The Interface shall allow uniquely identifying and referring to a MART request. The Interface shall allow a client OS to query a server OS for the current status of a request.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0027	The Interface shall support the ability to allow an OS to request that an outstanding MART operation be terminated. The resulting situation could be separately retrieved.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0028	The Interface shall support a message exchange pattern (ref to SD2-5_Communication_Styles) that shall allow for a MART request to take several hours to complete. The maximum time allowed for any single MART request shall be 24 hours.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0029	The Interface shall support the out of band (e.g. file transfer) message exchange pattern (ref to SD2-5_Communication_Styles).
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0030	The Interface shall support a message based request and response message exchange pattern (ref to SD2-5_Communication_Styles).

Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0031	The Interface shall allow in a MART request to let the client define the execution order of subsets of operations. This may be execution sequential in a specified order or delegate order and concurrency decisions to the server.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0032	The Interface shall allow in a MART request to let the client define the failure behavior (halt, rollback or continue) for each operation and sets of operations.
Source	TMF518_FMW, Version 1.0

3.2.7 Multi-Event Inventory Notifications

The requirements in this section pertain to single notifications that report changes of multiple entities.

The Interface shall provide the capability for both single-event inventory notifications, e.g., creation of a particular entity or attribute value changes for several attributes of a given entity, as well as multi-event inventory notifications, i.e. a direct method for getting bulk inventory updates from the source OS to other interested Oss.

R_TMF518_FMW_II_0033	The Interface shall allow an OS to report on a collection of (not necessarily related) inventory events using an inventory layout structure that is carried in one or more Multi-Event Inventory (MEI) notifications. The following types of inventory events shall be supported: • Object Creation (OC) • Object Discovery (Odsc) • Object Deletion (OD) • Attribute Value Change (AVC) • State Change (SC)
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0034	The inventory events listed in R_TMF518_FMW_II_0033 shall be mapped to a MEI notification as described in **Table 3-1**.
Source	TMF518_FMW, Version 1.0

Inventory Event	Information Mapping to Inventory Layout
Object Creation (OC)	Include the data structure for newly created object with all known attribute values and relationships. The containing objects should be included in the layout (just the object names in cases where the containing object has already been reported).
Object Discovery (Odsc)	Include the data structure for the discovered object with all known attribute values and relationships. The *discovered name* would be included in lieu of the object's *name*. The containing objects should be included in the layout (just the object names in cases where the containing object has already been reported).
Object Deletion (OD)	Include only the deleted object's name and resourceState (set to Retiring) would be reported. The containing objects should be included in the layout (just the object names in cases where the containing object has already been reported). None of the contained objects would be reported (the receiving OS should assume all contained objects are also in the "Retiring" state).
Attribute Value Change (AVC)	Include the object with at least the attributes whose values have changed. However, it is acceptable to include the entire data structure (i.e., all attributes) for the object. The containing objects should be included in the layout (just the object names in cases where the containing object has already been reported).
State Change (SC)	Similar treatment to AVCs, except that the source OS may want to send state change notifications immediately (via individual notifications as opposed ot the MEI notification) when the state change is related to a fault condition.

Table 3-1. Mapping of Inventory Events to the Inventory Layout carried in a MEI notification

R_TMF518_FMW_II_0035	The Interface shall allow an OS to support both single-event inventory notifications and multi-event inventory notifications mechanisms; the event shall be reported only once, choosing the most appropriate mechanism.
Source	TMF518_FMW, Version 1.0

The mapping between the parameters of the single-event inventory notifications, their required status (M=mandatory, O=optional or, if empty, not meaningful) in the single-event notification and the proposed MEI notification use is shown in Table **3-2**.

Attributes	OC	OD	AVC	SC	Odsc	Multi-event Inventory (MEI) Notification
notificationId	M	M	M	M	M	There is one notification Id for each MEI notification.
objectName	M	M	M	M		This is reflected in the inventory layout of the MEI notification.

objectType	M	M	M	M	M	This is reflected in the inventory layout of the MEI notification.
osTime	M	M	M	M	M	This parameter appears once in the MEI notification and applies to all the information reported by the given notification.
sourceTime	M	M	M	M	M	It was decided not to support this parameter in the MEI notification.
edgePointRelated	M	M	M	M	M	It seems that one can make this determination from the edgePoint attribute of the PTP.
attributeList			M	M		This is reflected in the inventory layout of the MEI notification.
discoveredName					M	This is reflected in the inventory layout of the MEI notification.
vendorExtensions	O	O	O	O	O	This is used by the OS supplier to append additional parameters to the MEI notification.

Table 3-2. Multi-event Inventory Notification Attributes

R_TMF518_FMW_II_0036	The MEI notification shall have the following parameters: • ***notificationID*** – an identifier for the notification. The uniqueness and the sequence of the notificationId are not guaranteed. • ***osTime*** – the time at which the sending OS generated the notification. • ***events*** – this data structure details the collection of events. • ***vendorExtensions*** – this is used by the OS supplier to append additional parameters to the MEI notification.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0037	The following attributes appear in individual objects that comprise the inventory layout in an MEI notification: • ***SourceTime*** – (if known) the time at which the NE detected the inventory change. This only applies to the ME and objects within since an NE would not typically know about containing objects such as subnetworks and management domains. • ***eventIndication*** – this attribute indicates the type of change that is being reported for a given object.

	Possible values are: ○ Object Creation (OC) ○ Object Discovery (Odsc), ○ Object Deletion (OD), ○ Attribute Value Change (AVC), ○ State Change (SC) and ○ AVC_SC_and_Childern. The following conventions apply: ○ If an object creation or object discovery indicator is applied to a parent object, it implicitly applies to all the children, i.e., the OS sending the notification only needs to set the indicator attribute in the parent object. ○ If an object deletion is applied to a parent object, it is assumed that all the children are also deleted. The ttribut should not be listed. ○ The AVC_SC_and_Children indication is applied to a parent object to imply that there are some AVC or SC changes to the parent and some or all of the children
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0038	In the case that multiple MEI notifications are used to convey a single inventory layout structure, the partitioning of the inventory layout structure is constrained by the following atomic (i.e., indivisible) object structures including their full containment hierarchy; ME, TL, OS, TMD, and MLSN.
Source	TMF518_FMW, Version 1.0

R_TMF518_FMW_II_0039	In the case that multiple MEI notifications are used to convey a single inventory layout structure, the inventory layout structure should be divided in such a way that the receiver can reconstruct the document as the various segments are received. Each MEI notification shall contain the following information (in addition to a portion of the inventory layout structure): • ***CorrelationID*** – used by the receiving Oss to associate the notifications • ***SequenceNumber*** – used by the receiving Oss to reassemble the inventory layout structure • ***endOfSequence*** – this is set to “true” in the last notification of the sequence.
Source	TMF518_FMW, Version 1.0

3.3 Category III: Abnormal or Exception Conditions, Dynamic Requirements

3.3.1 Server availability detection

R_TMF518_FMW_III_0040	The Interface shall allow a client OS to detect when a server OS is no longer available.
Source	TMF517, Requirement III.1

3.3.2 Interface examples and implementation user guide line

Examples of exceptions and abnormal conditions are reported in the interface examples and in the implementation user guide lines.

3.4 Category IV: Expectations and Non-Functional Requirements

R_TMF518_FMW_IV_0041	The Interface must take into account scalability and performance issues. Real-world scenarios shall be applied to check for object instance scale (MIB instances) and operations intensity.
Source	Requirement IV.001 TMF513 v3.5

3.4.1 Security Management

Security Management involves mainly: authentication, authorization, single sign-on and encryption.

As a prerequisite to any type of relationship, an OS needs to be authenticated for access to the CCV.

R_TMF518_FMW_IV_0042	In order to obtain access to the CCV, an OS must be authenticated. It should also be possible to gain selective access to a subset of the messages on the CCV. Such access should be governed by an access control mechanism.
Source	TMF517 Requirement II.1 Requirement II.139 TMF513 v3.0 Requirement II.140 TMF513 v3.0

As an example of how security requirements can be met, each Client OS could be identified and authenticated by a credential mechanism, i.e. an authentication and service authorization server. A Client OS will use the credentials that it gets from the authentication server when making requests of a Server OS. Based on the identity and access rights of the Client OS as well as other policies supported by the Server OS, the Server OS may grant or deny the Client's request (this can be done on a per request basis, if needed).

NML-EML paradigm security items	**OS-OS paradigm security items**
Session	There is no equivalent. A client OS can attempt to request service from a Server OS at any time. The Server OS can always deny service.
Access to EMS (user and password)	Each Client OS get authenticated by an authentication server. A Client OS will use the credentials that it gets from the authentication server when making requests of a Server OS. Based on the identity and access rights of the Client OS as well as other policies supported by the Server OS, the Server OS may grant or deny the Client's request (this can be done on a per request basis, if needed).
Release of resources when the NMS informs the EMS that it wishes to discontinue usage of its services (i.e., end the session)	A similar step is needed for a Client OS to inform a Server OS that it wishes to discontinue usage of its services. In the XML, there will need to be an operation that a Client OS uses to tell a Server OS that it wishes to discontinue usage of its services. When receiving such a request, the Server OS will delete any remaining iterators that have been created on behalf of the Client OS, stop the process of creating any PM or inventory files requested by the Client OS, and delete any PM or inventory files created on behalf of the Client OS. Note that this is different from endSession, since the Client OS may follow-up (perhaps immediately) with another service request to the Client OS (and the Client may deny or fulfill the request). There is no session to start or end, but there is a need to inform the Client OS that resource can now be released. This would typically be done when a Client OS needs to be shut-down (gracefully).

The definition of additional security (beyond that mentioned in R_TMF518_FMW_IV_0042) such as XML digital signature and/or encryption is possible, but beyond the scope of the interface Business Agreement (see http://xml.apache.org/security/ and http://www.w3.org/TR/xmlenc-core/ for more information). Security may also be a characteristic of the selected transport technology for the solution set.

3.4.2 Interface Extensibility

The Interface implementation is defined in terms of service interfaces and data entities. The service interfaces consist of operations and notifications. The operations and notifications have parameters. The data entities are comprised of attributes. The following requirements relate to allowable extensions of the Interface with respect to service interfaces and data entities, and their associated ttributes.

Some of the requirements in this section involve additions to the Interface. Clearly, additional service interfaces (operations and notifications) are needed as the Interface is extended. The other requirements in this section involve deletions and modifications;. The expectation is that Interface elements such as operations will not be deleted passing from one version to the next unless an error with an existing element is identified.

“Deprecate” is used instead of “delete” with respect to parameters for operations and notifications, since it may be easier to leave the operation/notification signature alone and just deprecate usage of a particular parameter (perhaps some default value is inserted). A similar remark can be made concerning the deprecation of entity attributes.

R_TMF518_FMW_IV_0043	The versioning methodology shall allow for the addition of service interfaces and data entities in going from one version to the next.
Source	TMF517, Requirement IV.003

R_TMF518_FMW_IV_0044	The versioning methodology shall allow for the deletion of service interfaces and data entities in going from one version to the next.
Source	TMF517, Requirement IV.004

R_TMF518_FMW_IV_0045	The versioning methodology shall allow for the addition of operations and notifications to existing service interfaces.
Source	TMF517, Requirement IV.005

R_TMF518_FMW_IV_0046	The versioning methodology shall allow for the deletion of operations and notifications from existing service interfaces.
Source	TMF517, Requirement IV.006

R_TMF518_FMW_IV_0047	The versioning methodology shall allow for the addition of parameters to operations and notifications. The versioning methodology shall provide formal mechanisms in order to understand if the added parameters can be ignored if not yet supported at the receiving side or must be understood.
Source	TMF517, Requirement IV.007

R_TMF518_FMW_IV_0048	The versioning methodology shall allow for the deprecation of parameters with respect to an operation or notification.

Source	TMF517, Requirement IV.008

R_TMF518_FMW_IV_0049	The versioning methodology shall allow for the addition of attributes to data entities. The versioning methodology shall provide formal mechanisms in order to understand if the added attributes can be ignored if not yet supported at the receiving side or must be understood
Source	TMF517, Requirement IV.009

R_TMF518_FMW_IV_0050	The versioning methodology shall allow for the deprecation of data entity attributes.
Source	TMF517, Requirement IV.010

R_TMF518_FMW_IV_0051	The versioning methodology shall allow for modifications to the relationships (including containment) among data entities.
Source	TMF517, Requirement IV.011

⇒ *For the extensibility principles analysis and the relevant rules, see* SD2-6_VersioningAndExtensibility.

3.4.3 Information Value Extensibility

The requirements in this section allow the MTOSI / MTNM team and in some cases vendors to qualify, replace or extend the values for some discretely valued characteristic information. It should be emphasized that the value set for some MTOSI / MTNM defined information are fixed and no changes by vendors are allowed.

In some cases a user of the MTOSI / MTNM will need to further qualify some information. This is already supported for the probableCause attribute which can be qualified by the probableCauseQualifier.

There are therefore cases where the MTOSI / MTNM team has decided that the value set for some information is likely to need qualification, replacement and/or extension. For example, the resourceState has been assigned a few values by the MTOSI / MTNM team, but it is generally recognized that each service provider is likely to have a different set of values for resouceState (depending on how they track inventory).

For this reason, the MTOSI team has designated resourceState as an information whose set of possible values can be qualified, replaced (all or in part) or extended.

The MTOSI / MTNM team encourages interested parties to submit contributions to the TM Forum MTOSI / MTNM team if they require additional entities and notifications that are not present in the MTOSI / MTNM model. But anyway it is recognized that the MTOSI / MTNM team may not always be able to make additions in time for a particular vendor or service provider to meet market demands. In such cases, the interface is enough flexible in order to add vendor-specific entities and/or notifications using the mechanisms described by SD2-6_VersioningAndExtensibility.

R_TMF518_FMW_IV_0052	A user of the MTOSI / MTNM shall be able to further qualify a value of an attribute (assuming the attribute is designated as “qualifiable” in the MTOSI specification). The values of the qualifier shall be defined by the organization using the MTOSI / MTNM.
Source	TMF517, Requirement IV.012

R_TMF518_FMW_IV_0053	It shall be possible to extend a discrete attribute with additional values. It shall be possible to remove or replace an attribute value. This type of extension is done as part of the MTOSI / MTNM specifications (as opposed to an extension done by the user of the MTOSI / MTNM).
Source	TMF517, Requirement IV.013 and Requirement IV.014

R_TMF518_FMW_IV_0054	It shall be possible for a user of the MTOSI / MTNM to extend the value set of a discrete attribute.
Source	TMF517, Requirement IV.015

The following requirement applies concerning the addition of vendor-specific entities and notifications.

R_TMF518_FMW_IV_0055	The MTOSI / MTNM model shall allow for the addition of vendor-specific entities and notifications. It is understood that these should only be temporary extensions to address market demands and that the vendor should make every attempt to have the extensions included in the MTOSI / MTNM model as some later point in time
Source	TMF517, Requirement IV.016

⇒ *Detailed rules and examples are available in SD2-6 VersioningAndExtensibility*

3.5 Category V: System Administration Requirements

Not applicable.

4 Use Cases

For the following use cases, the Actor is an OS. Also, the system being “acted upon” is also one or more Oss. According to the different contexts, the Oss in the use cases plays different roles. The following roles are used in the use cases:

1. Requesting OS – this role is played by a Client OS (i.e. an OS that is responsible for invoking service requests and receiving service responses) that invokes the services of one or potentially more than one target OS. The requesting OS has a “contains” and “uses” relationships with one or more instances of Client OS, Server OS or both.
2. Target OS – this is role is played by a Server OS (i.e. an OS that is responsible for receiving service requests and providing service responses to a Client OS) that provides services to a requesting OS. The target OS has a “contains” and “uses” relationships with one or more instances of Client OS, Server OS or both.
3. Publishing OS – this is role is played by an OS that generates notifications and publishes them on the CCV via a notification service.
4. Subscriber OS – this role is played by an OS that subscribes to a notification service in order to get notifications of a specific Topic.

Some of the use cases involve more specific roles (e.g., Inventory OS) which are explained in the use cases that follow.

4.1.1 Exceptions

Table 4-1 identifies the complete list of exceptions that may be raised by a target OS in response to a requesting OS. In the “Description” section of each Use Case the specific exceptions that may be raised are identified. The target OS can qualify each exception by indicating further details about the exception in a free format string attached to the exception. This is specifically necessary if more than one step in the description leads to the same exception. The following exceptions from Table **4-1** are considered to be “general” exceptions, in that an OS may raise these exceptions at any time:

- Internal Error
- Not Implemented
- Unable To Comply
- Comm Loss

The Unable To Comply exception may be raised whenever the OS cannot respond to a request, some Use Cases may identify specific conditions that will result in this exception.

The Comm Losss exception may be raised for two different reasons:

- When communication between a top-level OS and a subordinate OS is necessary to fulfill the requests in the Use Case and communication to the subordinate OS(s) is lost;
- When communication with at least one NE is necessary to fulfill the requests in a Use Case and communication to the NE(s) is lost. This also applies to the case where communication to a “Control Plane” entity is required to fulfill a request. A specific step for this exception will not be specified in the description of the use cases.

The “error reason” parameter may be used to supply more information.

Table 4-1. Use Case Exceptions

ID	Name	General Description
1	Internal Error	The request has resulted in an OS internal error.
2	Not Implemented	The entire request is not supported by the target OS or the request with the specified input parameters is not supported.
3	Invalid Input	The request contains an input parameter that is syntactically incorrect or identifies an object of the wrong type or is out of range.
4	Entity Not Found	The specified object instance does not exist.
5	Object In Use	The object identified in the request is currently in use.
6	User Label In Use	The user label uniqueness constraint can not be met; the specified user label is currently being used.
7	Unable To Comply	The target OS cannot respond to the request.
8	Unsupported Routing Constraints	The target OS is unable to satisfy the requested routing constraints.
9	Access Denied	The requesting OS is not permitted to perform the request.
10	Capacity Exceeded	The request will result in resources being created or activated beyond the capacity supported by the NE or target OS.
11	Not In Valid State	The state of the specified object is such that the target OS cannot perform the request.
12	Protection Effort Not Met	The level of protection effort in the request cannot be met by the target OS.
13	Timeslot In Use	A timeslot is already in use.
14	TP Invalid Endpoint	The specified TP does not exist or cannot be created.
15	Comm Loss	The target OS (which is a top-level OS) is unable to communicate: - either with the subordinate OS, - or with the NE and communication is required to complete the request. The “error reason” parameter may be used to supply more information.
16	Policy Violation	A policy of the target OS has been violated; it may happen when requesting to create or delete a given object or to modify specific attributes. The target OS should indicate the specific policy that has been violated.

 ©TM Forum 2011

4.2 OS Initialization Use Cases

According to the different Oss contexts, we can distinguish at the interface between a point-to-point session based interaction, which is typical of the NML-EML scenario, and a multipoint to multipoint stateless interaction, which can be extended to the Oss space. The initialization scenarios can therefore be different according to the chosen interaction model: they can be parallelized according to the **Table 4-2**.

For the first column use cases, refer to TMF513 NMS-EMS Session Management Use Cases section.

The second column use cases are reported in this document.

Table 4-2. Use Case Parallelism

NML-EML Use Case	**OS-OS Use Case**
EMS (Re) Starts	OS (Re)starts
NMS creates a session with EMS	Client OS Prepares to Use the Services of a Server OS
NMS retrieves the interface version used by the EMS	This is included as part of “Client OS Prepares to Use the Services of a Server OS”
NMS closes a session with an EMS	No corresponding Use Case. See TMF854A.
EMS closes a session with an NMS	Server OS No Longer Allows Client Oss to Use Its Services
NMS detects that an EMS is unavailable	Client OS Detects that Server OS is Unavailable
EMS detects that an NMS is unavailable	In a Service-Oriented Architecture (SOA), the server should not keep track of the clients. If a client crashes while waiting for a server response, it is the client’s responsibility to recover and if needed repeat the service request.

Use Case Id	UC_TMF518_FMW_0001
Use Case Name	Target OS Restarts
Summary	The OS starts up and creates an entry in the Directory according to the rules that are out of the scope of this document. So far, if the Directory requires authentication, the OS shall authenticate, and so on. The Directory could also be populated with information concerning the service interfaces and notifications offered by the OS. In the case of a restart, the OS may only need to modify some of the existing Directory information. The OS may also enable itself to send notifications. **Notes**:

	1. The Directory term here is generically used and does not imply any specific implementation. In fact, implementation will likely vary depending on the selected Distributed Interface-Oriented Architecture (DIOA – see TMF053) technology. 2. A Universal Description, Discovery and Integration (UDDI) directory could support the directory service. The WSDL can be annotated to contain all the information about the interfaces that it offers. In the case of an IDL solution, only one service interface, the EMS SessionFactory_I would be registered with the OMG Naming Service, and the EMS would serve as the directory for the other interface information. So, the OMG Naming Service and EMS would collectively support the Directory function noted in the summary above.
Actor(s)	OS
Pre-Conditions	All required communication services are running on the OS and Notification Service hosts.
Begins When	The OS starts (or restarts)
Description	1. When the OS starts it creates an entry with associated interface and notification information (including handles for the interfaces that it offers) in the Directory. In case the Directory can't be found the middleware informs the OS and the operation stops. • In the case of a restart, it may be just a matter of updating the Directory information. • Exchanging interface references and notification information via another mechanism (e.g. file, FTP, email) is acceptable if agreed to by OS implementation groups. 2. If notifications are enabled and if not already done at a previous startup, the OS registers with the Notification Service as a publisher and/or subscriber of notifications, if needed. In case the Notification Service can't be found, the middleware informs the OS and the operation stops. 3. Depending on the notification technology, the OS may need to (re)connect to the Notification Service. In case the Notification Service can't be found, the middleware informs the OS and the operation stops. 4. The OS will could send a state change notification in case it is a Publishing OS to indicate that its service state is IN_SERVICE. This lets potential Requesting Oss know that they can use the services of the OS. **Note:** The OS may not be completely initialized when the Directory entry is made available. This is because the OS will have to do internal initialization as well as initializing its interfaces. It is an OS implementation decision on how to

	handle requests arriving during initialization.
Ends When	The OS has made its entry available in the Directory, and is able to send out notifications (in cases where the OS is a publisher of notifications).
Post-Conditions	The OS is ready to receive requests on its interface and to send out notifications (in cases where the OS is a publisher of notifications).
Exceptions	See **Table 4-1**
Traceability	R_TMF518_FMW_I_0009

Use Case Id	UC_TMF518_FMW_0002
Use Case Name	Client OS prepares to use the services of a Target OS
Summary	The Client OS looks up in the Directory a particular Server OS, determines information about the service interfaces and notifications offered by the Server OS, and then may start to invoke operations and/or register for notifications supported by the Server OS. **Note:** The Server OS may not be completely initialized when the Directory entry is made available. This is because the Server OS will have to do internal initialization as well as initializing its interfaces. It is an OS implementation decision on how to handle requests arriving during initialization.
Actor(s)	Client OS
Pre-Conditions	1. Both the client and server Oss have successfully started (refer to UC_TMF518_FMW_0001) 2. The Client OS has authenticated itself with the Authentication Service and has the capability to communicate with the Directory Service 3. The Client OS knows the Server OS name with whom it needs to communicate
Begins When	Client OS (re)starts – in which case this use case is performed for every OS that acts in the server role to the Client OS.
Description	1. The Client OS sends a request to retrieve the capabilities of the Server OS from the Directory. This would include information about service interfaces/managers, operations and notifications supported with their version number indication. In case the authentication of the client OS is not validated by the server OS, an Access Denied exception shall be raised. In case the version number indication does not match with the managed ones, the Unable to Comply exception shall be raised with Version Mismatch present in the error reason parameter. **Note**: In the case of an IDL solution a client OS not able to request service interface references until after a session has been established with the Server OS that is providing part of the Directory function.

	2. If not done previously, the client OS subscribes to notifications of interest via the Notification Service. 3. If the Client OS is restarting, i.e. if it was registered earlier, the Notification Service may send out all notifications that occurred while the Client OS was unavailable (durable subscription). In this case, the Client OS may synchronize its database by evaluating these notifications. 4. If the Client OS is starting for the first time or if it decides to discard all previous notifications, it may (re)discover network inventory of the Server OS (assuming the Server OS keeps any inventory information). 5. The Client OS periodically monitors the availability of the Server OS. **Note:** Access control is done at the Server OS. The Server OS decides to accept (and respond) or not accept a request from a Client OS based on access control policies. Access control can be performed on a per request basis since each request will contain the credentials of the Client (i.e., requesting) OS.
Ends When	One of the following events occurs: 1. The Client OS stops using the services of the Server OS 2. The Client OS detects that the Server OS is unavailable
Post-Conditions	1. The Client is able to access the services offered by the Server OS. 2. The Client is initialized and ready to communicate with the Server OS as well as to receive notifications.
Exceptions	See **Table 4-1**
Traceability	R_TMF518_FMW_II_0020 R_TMF518_FMW_II_0021 R_TMF518_FMW_II_0022 R_TMF518_FMW_IV_0042

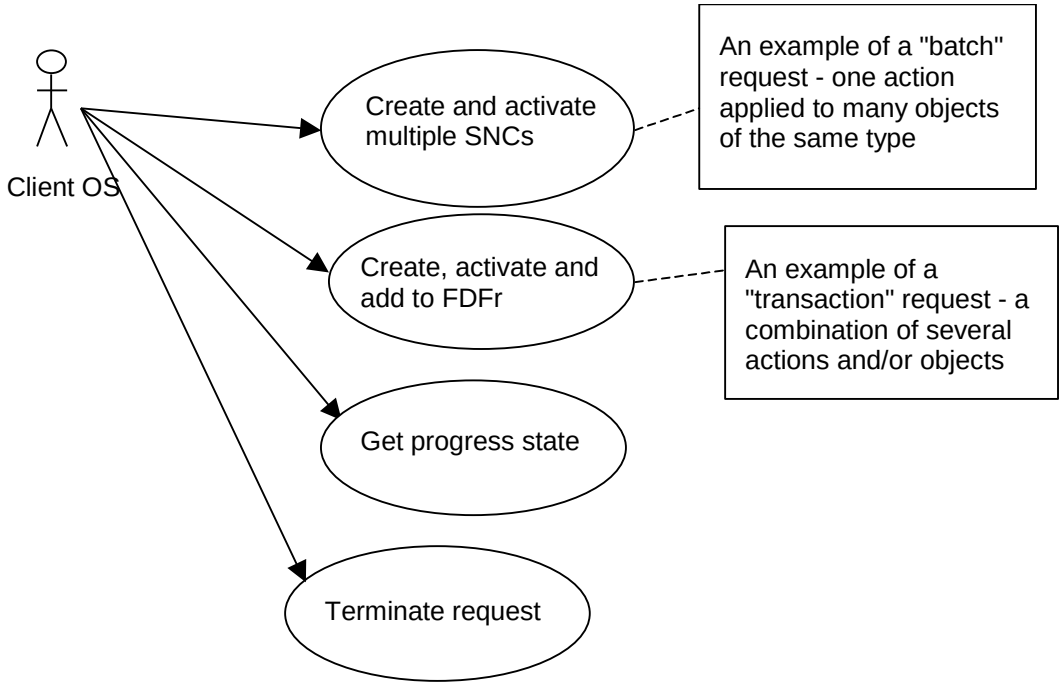
Use Case Id	UC_TMF518_FMW_0003
Use Case Name	Server OS no longer allows client OS to use its services
Summary	The Server OS denies service requests from all client Oss. This use cases is typically executed when a Server OS is being taken out of service gracefully.
Actor(s)	Server OS
Pre-Conditions	The Client OS has already performed at least UC_TMF518_FMW_0002
Begins When	Server OS changes its Resource Fulfillment State to OUT_OF_SERVICE
Description	1. The Server OS changes its service state from IN_SERVICE to OUT_OF_SERVICE (the OS is entirely not capable of performing

	its provisioned functions and is not restricted by administrative actions). 2. The Server OS does appropriate clean up, i.e. will delete existing iterators, stop processes on existing PM and inventory file creation requests. 3. The Client Oss may deregister from notifications offered by the Server OS. 4. The Server OS changes the access control policy to deny access to the Client Oss. **Notes** 1. There is no guarantee (to the Client Oss) the service interface references to the Server OS will be the same when the Server OS re-enters the IN_SERVICE state. 2. The Server OS shall make the appropriate shutdown operations. 3. The Server OS may de-register from the directory service (this is not mandatory).
Ends When	The Server OS stops communicating with the Client Oss
Post-Conditions	1. The Client OS has taken the appropriate internal action and has released all resources associated with the Server OS. 2. The Server OS has released all resources associated with the Client Oss. 3. Client Oss are no longer communicating with the Server OS. 4. The Notification Service is not trying to forward notifications from the Server OS to the Client Oss any longer if the Client OS decided to de-register.
Exceptions	None
Traceability	R_TMF518_FMW_III_0040

Use Case Id	UC_TMF518_FMW_0004
Use Case Name	Client OS Detects that Server OS is Unavailable
Summary	The Client OS detects that a Server OS has become unavailable. This might be detected due to a neglected request (i.e., a request to which the Server OS does not respond), an unacknowledged ping, or a failure on the part of the Server OS to send an expected heartbeat notification. This situation may be caused by a Server OS crash, e.g., the Server OS hardware is powered off before a controlled shut down can be accomplished, or a communication failure on the CCV.
Actor(s)	Client OS
Pre-Conditions	The Client OS has executed at least UC_TMF518_FMW_0002
Begins When	The Client OS receives a failure on a ping, or on heartbeat or on a

	request to the Server OS.
Description	1. The Client OS tries to release resources associated with the Server OS. 2. The Client OS takes appropriate internal actions and frees up any resources associated with the connection to the Server OS. 3. The Client OS initiates a failover procedure if available.
Ends When	The Client OS has recovered from the failure with the Server OS
Post-Conditions	The Client OS process is no longer communicating with the Server OS.
Exceptions	None
Traceability	R_TMF518_FMW_III_0040

4.3 Multiple Action Requests and Transactions (MART)



Use Case Id	UC_TMF518_FMW_0005
Use Case Name	Batch request - E.g. Create and activate multiple SNCs
Summary	The client OS requires to create and activate a large number (e.g. >1000) of SNCs which will be managed by the server OS. The request will take significant time (e.g. >1hour) to perform the request. Each

	SNC operation is a separate action, that is the failure of an SNC operation does not affect the processing of the other operations (i.e. it is a best effort operation). In this case the request runs to completion – see other cases for termination, progress requests etc. No particular MEP is assumed in this case. For the specific single operation behavior description, please refer to Use Case 7.7.3: NMS creates and activates a Subnetwork Connection (SNC) of TMF513.
Actor(s)	Client OS, Server OS
Pre-Conditions	Both Oss have executed UC_TMF518_FMW_0002
Begins When	Client OS wishes to create and activate as a "batch" request a set of SNCs.
Description	1. The Target OS validates the request: a) If the Target OS does not support the request, a Not Implemented exception is raised, and the Target OS stops serving the request; b) If the syntax is in error, an Invalid Input exception is raised and the Target OS stops serving the request; c) If the Server OS is unable to accomplish the operation, due to lack of internal resources, an Unable To Comply exception is raised and the Target OS stops serving the request; d) Any other severe error in the Server OS shall cause an Internal Error exception and the Target OS stops serving the request. 2) The Target OS, after the checks of point 1, starts processing the request: a) The degree of concurrent processing and order of actions is decided by the server OS as it believes appropriate for efficiency and load balancing. b) Server OS completes the batch of operations and responds to the client OS with the result data set. This indicates the success or failure for each operation and in the case of failure the reason for failure. For each requested operation, in fact, the same checks that should be done for a single "create and activate SNC" are performed: i) If a specified object is not of the right type, an Invalid Input exception is raised for the specific wrong operation request. ii) If a specified object is not known to the Target OS, an Entity Not Found exception is raised for the specific wrong operation request. iii) If a SNC object with the Name specified already exists, then an Object In Use exception is raised for the specific wrong operation request. iv) If uniqueness of the user label is required, the Target OS checks the user label for uniqueness; i.e., if a SNC object with the same user label exists already, then a User Label In Use exception is raised for the specific wrong operation

	request. v) If the provided layered parameters are inconsistent for example where two parameters at one TP conflict or where a parameter at one end of the SNC has a value that is illegal with respect to a parameter value at the other end, then an Unable to Comply exception is raised for the specific wrong operation request. vi) If one or more resources requested to be used for creating the SNC, are already in use, then an Object in Use exception is raised for the specific wrong operation request. vii) Each SNC operation has the same set of exceptions as for a single create and activate SNC request, as reported in the Use Case 7.7.3: NMS creates and activates a Subnetwork Connection (SNC) of TMF513.
Ends When	Client OS has received the results data. Or Client OS times out the request. (The time out is an implementation decision agreed between the parties).
Post-Conditions	Some, all or none of the SNCs have been created and activated. The client OS knows the outcome of each operation.
Exceptions	See **Table 4-1**
Traceability	R_TMF518_FMW_II_0025 R_TMF518_FMW_II_0028 R_TMF518_FMW_II_0029 R_TMF518_FMW_II_0030 R_TMF518_FMW_II_0031

Use Case Id	UC_TMF518_FMW_0006
Use Case Name	Transaction request – create, activate and add to FDFr
Summary	The client OS requires to create and activate an FDFr and then add CPTPs to it. This is one create and activate operation, as described in Use Case 9.6.1: NMS creates and activates a Flow Domain Fragment in TMF513, followed by one modify operation for the list of CPTPs to add, according to the Use Case 9.6.3: NMS modifies a Flow Domain Fragment in TMF513.. .The client OS wishes to perform this action as single request either as a: • "Atomic" transaction – result is either completely successful or is rolled-back to initial state (if possible), or, • "Best effort" transaction – request is progressed as far as possible and the resulting state is reported.
Actor(s)	Client OS, Server OS
Pre-Conditions	Both Oss have executed UC_TMF518_FMW_0002
Begins When	Client OS wishes to create and activate the FDFr
Description	The client OS requests the creation and activation of an FDFr and the

	addition of CPTPs to the FDFr. 1. The Target OS validates the request: • If the Target OS does not support the request, a Not Implemented exception is raised, and the Target OS stops serving the request; • If the syntax is in error, an Invalid Input exception is raised and the Target OS stops serving the request; • If the Server OS is unable to accomplish the operation, due to lack of internal resources, an Unable To Comply exception is raised and the Target OS stops serving the request; • Any other severe error in the Server OS shall cause an Internal Error exception and the Target OS stops serving the request. 2. The Target OS creates and activates the FDFr. If this fails, then in: • case a (atomic) the whole request ends and the state is returned to the client OS. • case b (best effort) the process continues, though in this example all following steps will also fail. 3. The server OS adds the CPTPs to the FDFr, as described in Use Case 9.6.3: NMS modifies a Flow Domain Fragment of TMF513. Server OS completes the set of operations and responds to the client OS with the resulting data set. This indicates the success or failure for each operation and in the case of failure the reason for failure.
Ends When	Client OS has received the results data. Or Client OS times out the request. (The time out is an implementation decision agreed between the parties).
Post-Conditions	Some, all or none of the operations have been implemented. The client OS knows the outcome of each operation.
Exceptions	See **Table 4-1**
Traceability	R_TMF518_FMW_II_0025 R_TMF518_FMW_II_0031 R_TMF518_FMW_II_0032

Use Case Id	UC_TMF518_FMW_0007
Use Case Name	Get progress state
Summary	The client OS has one or more MART requests which have been accepted by server OS's. The client OS requests one of the server OS for the progress state of a specific outstanding MART request.
Actor(s)	Client OS, Server OS
Pre-Conditions	The client OS has one or more MART requests which have been accepted by the server OS. The requests have not completed in the requested timeframe.
Begins When	Client OS wishes to know the progress of a request

Description	The client OS send a request for the progress state of a particular outstanding MART request. 1. The Target OS validates the request: • If the Target OS does not support the request, a Not Implemented exception is raised, and the Target OS stops serving the request; • If the syntax is in error, an Invalid Input exception is raised and the Target OS stops serving the request; • If the Server OS is unable to accomplish the operation, due to unexisting MART request reference, an Entity Not Found exception is raised and the Target OS stops serving the request; • Any other severe error in the Server OS shall cause an Internal Error exception and the Target OS stops serving the request. 2. The server OS returns the state of the MART request. Each operation within the MART request may be in one of the following states: • Completed successfully • Failed (+reason) • Not started • In progress
Ends When	Client OS has received the results data
Post-Conditions	Client OS knows the state of progress at the time of requesting
Exceptions	See **Table 4-1**
Traceability	R_TMF518_FMW_II_0026

Use Case Id	UC_TMF518_FMW_0008
Use Case Name	Terminate request
Summary	The client OS has one or more MART requests which have been accepted by server OS. The client OS requests one of the server OS to terminate a specific MART request which has not completed. The server OS stops any further processing of the request and returns the full status of the request.
Actor(s)	Client OS, Server OS
Pre-Conditions	The client OS has one or more MART requests which have been accepted by the server OS. The requests have not completed and the time out has not yet been reached.
Begins When	Client OS wishes to terminate the outstanding request. For example, the period of “quiet time” has ended and the client OS must free the server OS resources.
Description	The client OS send a request to terminate the out standing MART request. The termination request must identify the specific request instance to be terminated. 1. The Target OS validates the request:

	• If the Target OS does not support the request, a Not Implemented exception is raised, and the Target OS stops serving the request; • If the syntax is in error, an Invalid Input exception is raised and the Target OS stops serving the request; • If the Server OS is unable to accomplish the operation, due to unexisting MART request reference, an Entity Not Found exception is raised and the Target OS stops serving the request; • Any other severe error in the Server OS shall cause an Internal Error exception and the Target OS stops serving the request. 3. The server OS stops processing the identified request. In case the request were atomic, the relevant roll-back procedure shall be started as soon as possible. 4. Note that there may be parts of the request that are in progress and beyond the control of the server OS to stop (e.g. being processed by an NE). These particular sub-tasks cannot be stopped and the server OS cannot determine there result at that point. The server OS does not wait for such sub-actions to complete and reports the operation with unknown result state. 5. The server OS returns the state of the MART terminate request as known at the time of terminating. The following states are foreseen: • Completed successfully • Failed (+reason) • Not started • Unknown
Ends When	Client OS has received the results answer with the state of the MART terminate request.
Post-Conditions	Even if some, all or none of the operations been completed, the Server OS is no longer processing the MART request.
Exceptions	See **Table 4-1**
Traceability	R_TMF518_FMW_II_0026 R_TMF518_FMW_II_0027

4.4 Multi-Events Inventory Notifications

Use Case Id	UC_TMF518_FMW_0009

Use Case Name	OS Informs Other Oss About Multiple Inventory Events
Summary	The Publishing OS sends Multi-event Inventory (MEI) notification to the Notification Service which forwards it to all other OS (Subscribers) on the CCV.
Actor(s)	Publishing OS
Pre-Conditions	1. The Publishing OS has gained access to the notification service and is able to send notifications. 2. Some of the other Subscribing Oss on the CCV has registered to receive MEI notifications concerning changes to the network inventory. 3. The Publishing OS and Subscribing OS have executed UC_TMF518_FMW_0001.
Begins When	The Publishing OS detects multiple changes in the underlying network that warrant the use of a MEI notification.
Description	1. The Publishing OS detects multiple changes in the underlying network and populates an inventory layout structure (this structure is explained in SD2-12_Resource_Inventory_Layout) concerning the inventory changes. 2. The Publishing OS decides on how many MEI notifications are to be used to transport the inventory layout structure. If multiple MEI notifications are to be used, each notification will contain: • ***CorrelationID*** – used by the Subscribing Oss to associate the notifications • ***SequenceNumber*** – used by the Subscribing Oss to reassemble the inventory layout structure • ***endOfSequence*** – this is set to "true" in the last notification of the sequence. 3. The Subscribing Oss on the CCV receive the MEI notifications and reassemble the inventory layout structure (in the case of multiple MEI notifications are used).
Ends When	In case of success: Subscribing Oss on the CCV have received all the MEI notifications associated with a particular inventory layout structure. In case of failure: Subscribing Oss on the CCV are not aligned, having not received all the MEI notifications associated with a particular inventory layout structure.
Post-Conditions	In case of success: The database of the subscribed Oss remains aligned with the publishing Oss database. In case of failure: The databases of (at least) some of the subscribed Oss are misaligned with the publishing Oss database.

Exceptions	None – this follows the MTNM approach of not having exceptions to notifications.
Traceability	R_TMF518_FMW_II_0033, R_TMF518_FMW_II_0034, R_TMF518_FMW_II_0036, R_TMF518_FMW_IV_0055

5 Traceability Matrices

Table 5-1. Use Cases – Requirements Traceability Matrix

Use Case Id	Use Case Name	Requirements
UC_TMF518_FMW_0001	Target OS Restarts	R_TMF518_FMW_I_0009
UC_TMF518_FMW_0002	Client OS prepares to use the services of a Target OS	R_TMF518_FMW_II_0020 R_TMF518_FMW_II_0021 R_TMF518_FMW_II_0022 R_TMF518_FMW_IV_0042
UC_TMF518_FMW_0003	Server OS no longer allows client OS to use its services	R_TMF518_FMW_III_0040
UC_TMF518_FMW_0004	Client OS Detects that Server OS is Unavailable	R_TMF518_FMW_III_0040
UC_TMF518_FMW_0005	Batch request - E.g. Create and activate multiple SNCs	R_TMF518_FMW_II_0025 R_TMF518_FMW_II_0028 R_TMF518_FMW_II_0029 R_TMF518_FMW_II_0030 R_TMF518_FMW_II_0031
UC_TMF518_FMW_0006	Transaction request – create, activate and add to FDFr	R_TMF518_FMW_II_0025 R_TMF518_FMW_II_0031 R_TMF518_FMW_II_0032
UC_TMF518_FMW_0007	Get progress state	R_TMF518_FMW_II_0026
UC_TMF518_FMW_0008	Terminate request	R_TMF518_FMW_II_0026 R_TMF518_FMW_II_0027
UC_TMF518_FMW_0009	OS Informs Other Oss About Multiple Inventory Events	R_TMF518_FMW_II_0033, R_TMF518_FMW_II_0034, R_TMF518_FMW_II_0036, R_TMF518_FMW_IV_0055

Table 5-2. Requirements – Use Cases Traceability Matrix

Requirement Id	Use Case Name	Use Case Id
R_TMF518_FMW_I_0001		
R_TMF518_FMW_I_0002		

R_TMF518_FMW_I_0003		
R_TMF518_FMW_I_0004		
R_TMF518_FMW_I_0005		
R_TMF518_FMW_I_0006		
R_TMF518_FMW_I_0007		
R_TMF518_FMW_I_0008		
R_TMF518_FMW_I_0009	Target OS Restarts	UC_TMF518_FMW_0001
R_TMF518_FMW_I_0010		
R_TMF518_FMW_I_0011		
R_TMF518_FMW_I_0012		
R_TMF518_FMW_I_0013		
R_TMF518_FMW_II_0014		
R_TMF518_FMW_II_0015		
R_TMF518_FMW_II_0016		
R_TMF518_FMW_II_0017		
R_TMF518_FMW_II_0018		
R_TMF518_FMW_II_0019		
R_TMF518_FMW_II_0020	Client OS prepares to use the services of a Target OS	UC_TMF518_FMW_0002
R_TMF518_FMW_II_0021	Client OS prepares to use the services of a Target OS	UC_TMF518_FMW_0002
R_TMF518_FMW_II_0022	Client OS prepares to use the services of a Target OS	UC_TMF518_FMW_0002
R_TMF518_FMW_II_0023		
R_TMF518_FMW_II_0024		
R_TMF518_FMW_II_0025	Transaction request – create, activate and add to FDFr Batch request - E.g. Create and activate multiple SNCs	UC_TMF518_FMW_0006 UC_TMF518_FMW_0005
R_TMF518_FMW_II_0026	Terminate request Get progress state	UC_TMF518_FMW_0008 UC_TMF518_FMW_0007
R_TMF518_FMW_II_0027	Terminate request	UC_TMF518_FMW_0008
R_TMF518_FMW_II_0028	Batch request - E.g. Create and activate multiple SNCs	UC_TMF518_FMW_0005
R_TMF518_FMW_II_0029	Batch request - E.g. Create and activate multiple SNCs	UC_TMF518_FMW_0005

R_TMF518_FMW_II_0030	Batch request - E.g. Create and activate multiple SNCs	UC_TMF518_FMW_0005
R_TMF518_FMW_II_0031	Transaction request – create, activate and add to FDFr Batch request - E.g. Create and activate multiple SNCs	UC_TMF518_FMW_0006 UC_TMF518_FMW_0005
R_TMF518_FMW_II_0032	Transaction request – create, activate and add to FDFr	UC_TMF518_FMW_0006
R_TMF518_FMW_II_0033	OS Informs Other Oss About Multiple Inventory Events	UC_TMF518_FMW_0009
R_TMF518_FMW_II_0034	OS Informs Other Oss About Multiple Inventory Events	UC_TMF518_FMW_0009
R_TMF518_FMW_II_0035		
R_TMF518_FMW_II_0036	OS Informs Other Oss About Multiple Inventory Events	UC_TMF518_FMW_0009
R_TMF518_FMW_II_0037		
R_TMF518_FMW_II_0038		
R_TMF518_FMW_II_0039		
R_TMF518_FMW_II_0056		
R_TMF518_FMW_II_0057		
R_TMF518_FMW_III_0040	Client OS Detects that Server OS is Unavailable Server OS no longer allows client OS to use its services	UC_TMF518_FMW_0004 UC_TMF518_FMW_0003
R_TMF518_FMW_IV_0041		
R_TMF518_FMW_IV_0042	Client OS prepares to use the services of a Target OS	UC_TMF518_FMW_0002
R_TMF518_FMW_IV_0043		
R_TMF518_FMW_IV_0044		
R_TMF518_FMW_IV_0045		
R_TMF518_FMW_IV_0046		
R_TMF518_FMW_IV_0047		
R_TMF518_FMW_IV_0048		
R_TMF518_FMW_IV_0049		
R_TMF518_FMW_IV_0050		
R_TMF518_FMW_IV_0051		
R_TMF518_FMW_IV_0052		
R_TMF518_FMW_IV_0053		

R_TMF518_FMW_IV_0054		
R_TMF518_FMW_IV_0055	OS Informs Other Oss About Multiple Inventory Events	UC_TMF518_FMW_0009

6 Future Directions

7 References

7.1 References

[1] TMF513, Multi-Technology Network Management (MTNM) Business Agreement, Version 3.1, March 2007.

[2] TMF517, Multi-Technology Operations System Interface (MTOSI) Business Agreement, Version 1.2, December 2006

[3] SD1-6, Examples for contained TPs in different states of usage

[4] SD1-14, Inverse Multiplexing Overview

[5] SD1-16, Layered Parameters

[6] SD1-17, Layer Rates

[7] SD1-18, Functional Modeling Concepts

[8] SD1-19, Location Identification

[9] SD1-20, Maintenance Commands

[10] SD1-22 Model Diagram Components

[11] SD1-28, Performance Parameters

[12] SD1-33, Specification of Probable Cause Strings

[13] SD1-36, SNC and Protection

[14] SD1-44, Connectionless Technology Management

[15] SD0-1, Dictionary

7.2 Source or use

7.3 IPR Releases and Patent Disclosure

There are no known IPR claims on the material in this document. As per the TM Forum bylaws, any TM Forum member company that has IPR claims on this or any TM Forum specification needs to make the claims known to the TM Forum membership immediately.

8 Administrative Appendix

This Appendix provides additional background material about the TM Forum and this document.

8.1 About this document

This document has been generated from the Template_BA.dot Word template,

8.2 Use and Extension of a TM Forum Business Agreement

This document defines the business problem and requirement model for Service Management. The Business Agreement is used to gain consensus on the business requirements for exchanging information among processes and systems in order to solve a specific business problem. The Business Agreement should feed the development of Information Agreement(s), which is a technology-neutral model of one or more interfaces. While the Business Agreement contains sufficient information to be a "stand alone" document, it is better read together with the Information Agreement document TMF612_SB when the Information Agreement is available. Reviewing the two documents together helps in gaining a full understanding of how the technology neutral information model solution is defined for this requirement model. An initial Business Agreement may only deal with a subset of the requirements. It is acceptable for subsequent issues of the document to add additional requirements not addressed by earlier releases of the BA. Business Agreements are the basis for requirement traceability for information models.

It is expected that this document will be used:

- As the foundation for a TM Forum Information Agreement(s)
- To facilitate requirement agreement between Service Providers and vendors
- As input to a service Provider's Request for Information / Request for Proposal (RFI/RFP—RFX)
- As input for vendors developing COTS products
- As a source of requirements for other bodies working in this area

8.3 Document History

Version	Date Modified	Description of changes
1.0	November 2007	This is the first version of the document and as such, there are no changes to report.
1.1	May 2008	Updated based on review and consolidation comments for the preparation of the MTOSI 2.0 release.

1.2	September 2011	Updated sections 1 and 2. Added section 8.2. Replaced mTOP by MTNM / MTOSI everywhere in the document

8.4 Company Contact Details

Team Member Representative	
Name: Elisabetta Gardelli *Email:* Elisabetta.Gardelli@nsn.com *Phone:* +39 02 24376301	*Name:* Michel Besson *Email:* Michel.Besson@amdocs.com *Phone:* +44 7717 692178

8.5 Acknowledgments

This document was prepared by the members of the TM Forum MTNM / MTOSI RM team.

- Keith Dorking, Ciena Corporation, document editor
- Steve Fratini, Telcordia Technologies, MTNM / MTOSI Program Director
- Bernd Zeuner, Deutsche Telekom AG
- Jérôme Magnet, Ciena