

Manage Resource Inventory - DDP BA

TMF518_MRI

Version 1.2



September, 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

Notice	2
Table of Contents	3
List of Requirements	6
List of Use Cases	10
List of Figures.....	11
List of Tables	12
Executive Summary	13
1 Introduction	14
1.1 DDP Structure.....	14
1.2 Document Structure.....	14
1.3 Terminology Used In This Document	15
2 Business Problem Description, Project Scope	16
2.1 Project Scope	16
2.2 Benefits.....	17
2.2.1 Service Provider Benefits	17
2.2.2 Supplier Benefits	18
3 Business Processes.....	19
3.1 Business Requirements.....	19
3.2 Category I: Static and Structural Requirements	19
3.3 Category II: Normal Sequences, Dynamic Requirements.....	19
3.3.1 General Requirements	19
3.3.1.1 Management Domain Partitioning.....	19
3.3.1.2 Inventory Associated with a Subordinate OS	20
3.3.1.3 "Bulk" inventory retrieval	21
3.3.1.4 Multi-Object Inventory Update	25
3.3.1.5 Management Domain (MD) Inventory.....	29
3.3.1.6 Operations System (OS) Inventory	30
3.3.2 Requirements for Connection Oriented Technologies	30
3.3.2.1 Managed Element (ME) Inventory	30
3.3.2.2 Subnetwork Inventory	31
3.3.2.3 Topological Link (TL) Inventory.....	31
3.3.2.4 Subnetwork Connection (SNC) Inventory	32

3.3.2.4.1	Flexible SNC.....	32
3.3.2.4.2	Fixed SNC	33
3.3.2.5	Route Inventory	34
3.3.2.6	Termination Point (TP) Inventory	35
3.3.2.7	Termination Point Pool (TPPool) Inventory	36
3.3.2.8	Physical Termination Point (PTP) Inventory	36
3.3.2.9	Edge Termination Point (TP) Inventory.....	37
3.3.2.10	Contained Termination Point (TP) Inventory	37
3.3.2.11	Containing Termination Point (TP) Inventory	39
3.3.2.12	Floating Termination Point (FTP) Inventory.....	39
3.3.2.13	Associated Termination Point (TP) Inventory	39
3.3.2.14	Supported Termination Point (TP) Inventory	40
3.3.2.15	Supporting Termination Point (TP) Inventory	40
3.3.2.16	Equipment Inventory.....	40
3.3.2.17	Supporting Equipment Inventory	41
3.3.2.18	Supported Equipment Inventory	41
3.3.2.19	Traffic Descriptors (TD) Inventory	41
3.3.2.20	Transmission Descriptors (TMD) Inventory	42
3.3.2.21	Cross Connect (CC) Inventory	42
3.3.2.21.1	Flexible Cross Connect	42
3.3.2.21.2	Fixed Cross Connect.....	43
3.3.2.22	Group Termination Point (TP) Inventory.....	43
3.3.3	Requirements for ConnectionLess Technologies	44
3.3.3.1	Termination (TP) Inventory	44
3.3.3.2	Matrix Flow Domain Inventory	44
3.3.3.3	Flow Domain Inventory	46
3.3.3.4	Traffic Conditioning Inventory	47
3.3.3.5	Flow Domain Fragment (FDFr) Inventory	47
3.4	Category III: Abnormal or Exception Conditions, Dynamic Requirements.....	48
3.5	Category IV: Expectations and Non-Functional Requirements.....	51
3.6	Category V: System Administration Requirements	52
4	Use Cases	53
4.1	The requesting OS registers to receive network inventory updates from the target OS.....	54
4.2	The requesting OS resynchronizes its database with the target OS.....	56
4.3	The requesting OS discovers the target OS network inventory (fine grain approach)	57
4.4	The requesting OS queries the target OS concerning inventory (fine grain approach)	61

4.5	One OS notifies another OS of inventory change	65
4.6	Inventory OS Provides Names for installed network entities.....	66
4.7	Discovery OS Provides Names for installed network entities.....	68
4.8	Inventory OS Provides Names for planned network entities	69
4.9	Naming for installed network entities - No separate Discovery OS.....	71
4.10	Distributed stewarding.....	72
4.11	Bulk Inventory Retrieval Use Cases.....	73
4.11.1	Bulk Inventory Retrieval– Coarse grained approach, batched response with message communication style (asynchronous)	75
4.11.2	Bulk Inventory Retrieval– Coarse grained approach, RPC (synchronous) communication style	76
4.11.3	Bulk Inventory Retrieval – Coarse grained approach, file transfer.....	78
4.12	Effects of the suppression of OCNs during creation of an object	82
4.13	OS Requests that Target OS update its inventory	82
4.14	OS Informs Other OSs About Multiple Inventory Events	84
5	Traceability Matrices	86
6	Future Directions.....	94
7	References.....	95
7.1	References	95
7.2	IPR Releases and Patent Disclosure	95
8	Administrative Appendix	96
8.1	About this document	96
8.2	Use and Extension of a TM Forum Business Agreement	96
8.3	Document History	96
8.4	Company Contact Details.....	97
8.5	Acknowledgments.....	97

List of Requirements

R TMF518 MRI II 0001	17
R TMF518 MRI II 0002	18
R TMF518 MRI II 0003	19
R TMF518 MRI II 0004	19
R TMF518 MRI II 0005	25
R TMF518 MRI II 0006	26
R TMF518 MRI II 0007	26
R TMF518 MRI II 0008	27
R TMF518 MRI II 0010	27
R TMF518 MRI II 0012	27
R TMF518 MRI II 0013	27
R TMF518 MRI II 0014	27
R TMF518 MRI II 0015	27
R TMF518 MRI II 0017	27
R TMF518 MRI II 0019	28
R TMF518 MRI II 0020	28
R TMF518 MRI II 0021	28
R TMF518 MRI II 0023	28
R TMF518 MRI II 0024	28
R TMF518 MRI II 0026	28
R TMF518 MRI II 0027	28
R TMF518 MRI II 0029	29
R TMF518 MRI II 0030	29
R TMF518 MRI II 0032	29
R TMF518 MRI II 0034	29
R TMF518 MRI II 0036	30
R TMF518 MRI II 0037	30
R TMF518 MRI II 0038	30
R TMF518 MRI II 0040	30
R TMF518 MRI II 0042	31
R TMF518 MRI II 0043	31
R TMF518 MRI II 0044	31
R TMF518 MRI II 0045	31

<u>R TMF518 MRI II 0046</u>	32
<u>R TMF518 MRI II 0047</u>	33
<u>R TMF518 MRI II 0048</u>	33
<u>R TMF518 MRI II 0049</u>	33
<u>R TMF518 MRI II 0050</u>	33
<u>R TMF518 MRI II 0051</u>	33
<u>R TMF518 MRI II 0052</u>	34
<u>R TMF518 MRI II 0054</u>	34
<u>R TMF518 MRI II 0055</u>	34
<u>R TMF518 MRI II 0057</u>	34
<u>R TMF518 MRI II 0059</u>	34
<u>R TMF518 MRI II 0061</u>	35
<u>R TMF518 MRI II 0064</u>	35
<u>R TMF518 MRI II 0066</u>	35
<u>R TMF518 MRI II 0068</u>	35
<u>R TMF518 MRI II 0070</u>	35
<u>R TMF518 MRI II 0072</u>	36
<u>R TMF518 MRI II 0074</u>	36
<u>R TMF518 MRI II 0076</u>	37
<u>R TMF518 MRI II 0078</u>	37
<u>R TMF518 MRI II 0080</u>	37
<u>R TMF518 MRI II 0081</u>	38
<u>R TMF518 MRI II 0083</u>	38
<u>R TMF518 MRI II 0085</u>	38
<u>R TMF518 MRI II 0087</u>	38
<u>R TMF518 MRI II 0088</u>	38
<u>R TMF518 MRI II 0090</u>	39
<u>R TMF518 MRI II 0092</u>	39
<u>R TMF518 MRI II 0094</u>	39
<u>R TMF518 MRI II 0096</u>	39
<u>R TMF518 MRI II 0098</u>	40
<u>R TMF518 MRI II 0099</u>	40
<u>R TMF518 MRI II 0100</u>	40
<u>R TMF518 MRI II 0101</u>	41
<u>R TMF518 MRI II 0102</u>	41
<u>R TMF518 MRI II 0104</u>	41

<u>R TMF518 MRI II 0105</u>	41
<u>R TMF518 MRI II 0106</u>	42
<u>R TMF518 MRI II 0107</u>	42
<u>R TMF518 MRI II 0108</u>	43
<u>R TMF518 MRI II 0109</u>	43
<u>R TMF518 MRI II 0110</u>	43
<u>R TMF518 MRI II 0111</u>	43
<u>R TMF518 MRI II 0112</u>	43
<u>R TMF518 MRI II 0113</u>	43
<u>R TMF518 MRI II 0114</u>	43
<u>R TMF518 MRI II 0115</u>	44
<u>R TMF518 MRI II 0116</u>	44
<u>R TMF518 MRI II 0117</u>	44
<u>R TMF518 MRI II 0118</u>	44
<u>R TMF518 MRI II 0119</u>	45
<u>R TMF518 MRI II 0120</u>	45
<u>R TMF518 MRI II 0121</u>	45
<u>R TMF518 MRI II 0122</u>	45
<u>R TMF518 MRI II 0123</u>	45
<u>R TMF518 MRI II 0124</u>	46
<u>R TMF518 MRI II 0125</u>	46
<u>R TMF518 MRI II 0126</u>	46
<u>R TMF518 MRI II 0127</u>	46
<u>R TMF518 MRI II 0128</u>	46
<u>R TMF518 MRI II 0129</u>	46
<u>R TMF518 MRI II 0130</u>	46
<u>R TMF518 MRI II 0131</u>	46
<u>R TMF518 MRI II 0132</u>	47
<u>R TMF518 MRI II 0150</u>	31
<u>R TMF518 MRI II 0151</u>	40
<u>R TMF518 MRI II 0152</u>	41
<u>R TMF518 MRI II 0153</u>	32
<u>R TMF518 MRI II 0155</u>	34
<u>R TMF518 MRI II 0156</u>	49
<u>R TMF518 MRI III 0146</u>	47
<u>R TMF518 MRI IV 0147</u>	49

Manage Resource Inventory - DDP BA

R_TMF518_MRI_IV_0148	50
R_TMF518_MRI_IV_0149	49
R_TMF518_MRI_V_0154	51

List of Use Cases

<u>UC_TMF518_MRI_0001</u>	53
<u>UC_TMF518_MRI_0002</u>	55
<u>UC_TMF518_MRI_0003</u>	56
<u>UC_TMF518_MRI_0004</u>	60
<u>UC_TMF518_MRI_0005</u>	64
<u>UC_TMF518_MRI_0006</u>	65
<u>UC_TMF518_MRI_0007</u>	66
<u>UC_TMF518_MRI_0008</u>	68
<u>UC_TMF518_MRI_0009</u>	69
<u>UC_TMF518_MRI_0010</u>	70
<u>UC_TMF518_MRI_0011</u>	73
<u>UC_TMF518_MRI_0012</u>	74
<u>UC_TMF518_MRI_0013</u>	76
<u>UC_TMF518_MRI_0014</u>	79
<u>UC_TMF518_MRI_0015</u>	80
<u>UC_TMF518_MRI_0016</u>	82



List of Figures

Figure 2-1. Inputs to the TM Forum Integration Program 16

Figure 2-2. TM Forum Integration Program 17

Figure 3-1. Target OS Subnetwork Partitioning 21

Figure 3-2. Inventory Update Example 27

Figure 4-1. An Example of Network and Associated Management Architecture 54

Figure 4-2. Context Diagram for Bulk Inventory Use Cases 74

List of Tables

Table 3-1. Examples of the Inventory Filter.....	25
Table 3-2. Exceptions for the Multi-Object Update Request	49
Table 5-1. Use Case - Requirements Traceability Matrix.....	86
Table 5-2. Requirements - Use Cases Traceability Matrix.....	89

Executive Summary

This document entails the Business Agreement (BA) aspect of the MTOSI / MTNM Manage Resource Inventory (MRI) Document Delivery Package (DDP). As its name indicates, it covers requirements and use cases concerning the management of resource inventory.

The following management capabilities are covered:

- General Management such as (among others):
 - Bulk inventory retrieval (retrieving selected information in a single operation)
 - Multi-Object Inventory Update
- Inventory Management of Connection Oriented Technologies
- Inventory Management of Connectionless Technologies
- Inventory Notifications

This document generalizes and extends the inventory management requirements and use cases from TMF 513 V3.0/V3.1 and TMF 517 V1.2. 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 MTOSI / MTNM 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 Structure

The present document corresponds to the BA part of the MRI OM-DDP (Manage Resource Inventory).

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.

1.3 Terminology Used In This Document

Refer to the [SD0-1](#) supporting document.

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.

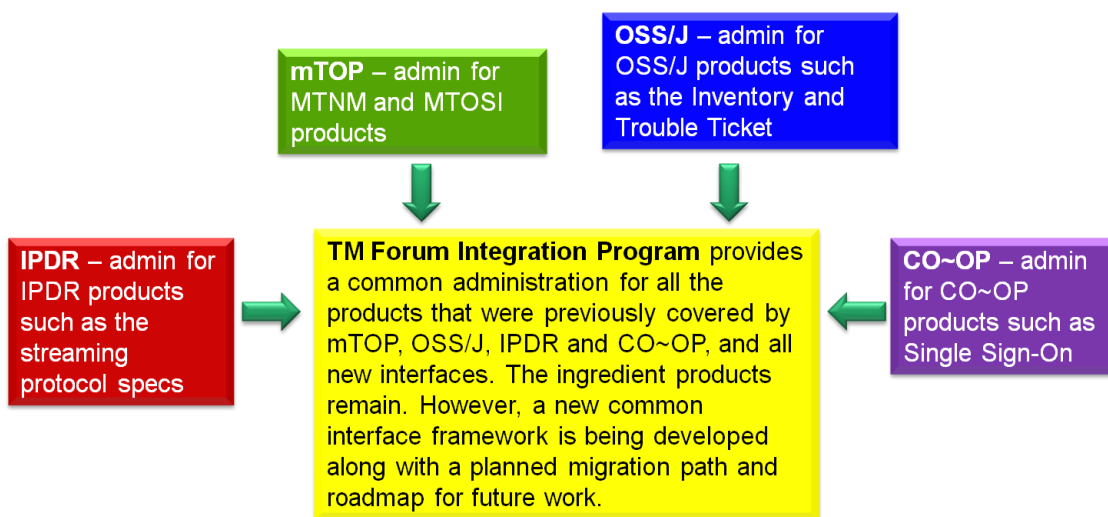


Figure 2-1. Inputs to the TM Forum Integration Program

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.

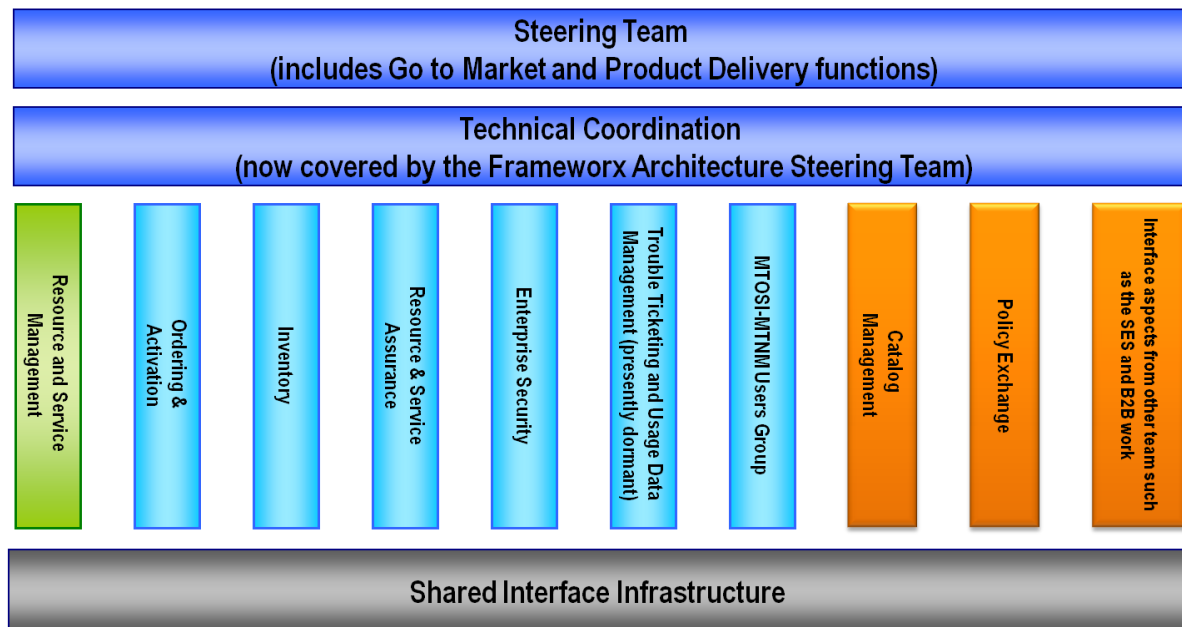


Figure 2-2. TM Forum Integration Program

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 Business Requirements

No requirements in this category have been identified

3.2 Category I: Static and Structural Requirements

Refer to the [TMF518_NRB](#) BA document

3.3 Category II: Normal Sequences, Dynamic Requirements

3.3.1 General Requirements

3.3.1.1 Management Domain Partitioning

An OS may want to discover or audit only part of another OS's inventory. For example,

1. (Network Size) The OS with the desired inventory may have a very large inventory and the client OS may only want to discover the inventory in one chunk (for lack of a better word) at a time.
2. (Network Size) In a vein similar to the point above, the client OS may want to audit only a part of the server OS's inventory each night because an audit of the entire inventory can not be done in one night.
3. (Geography) The client OS may be directed to discover the inventory of the server OSs based on geography. This could happen, for example, if the client OS can discover inventory for an area much larger than the management span for an inventory OS, and so multiple inventory OSs are needed for a single discovery OS. [This example also fits the Network Size case.]
4. (Technology) Some inventory OSs may only handle particular technologies, e.g., SONET/SDH or ATM. This is a difficult problem to address because many of the Next Generation NEs support multiple technologies. Even a particular card may support multiple technologies.
5. (Vendor Type) An inventory OS may be directed to request the inventory concerning NEs from a particular equipment supplier. NEs from other equipment suppliers could be managed by another inventory OS. This situation could happen, for example, if the NE supplier also provides inventory OSs.

In order to support the selective inventory discovery and auditing described in the above items, an ME retrieval operation shall be defined that supports filtering based on location, manufacturer, productName (intended as the identifier of the managed element product/type name), and resourceState.

R_TMF518_MRI_II_0001	The Interface should support requests to retrieve MEs based on any combination of location, manufacturer, productName and (if supported) resourceState. For example, an OS should be able to request all MEs at "locationA", from "manufacturerB", of type "productC" and with resourceState equal to Planning (or vice versa equal to Installing).
Source	TMF 517 Version 1.2, Requirement II.8

Note: There is no notion at the interface of a pre-defined list of allowed manufacturer, or productNames, and so on. Their definition is left to the local agreements (i.e. Implementation Statement documents).

3.3.1.2 Inventory Associated with a Subordinate OS

In the object naming scheme, a subordinate OS (such as an Target OS) does not contain any other objects. However, the subordinate OS may have a management relationship with the transmission descriptor, managed element, subnetwork and topological link objects. The Interface should allow for the retrieval of all managed elements and all transmission descriptors associated with a subordinate OS (these capability allow a discovering OS to discover resources on a per subordinate OS basis).

It may also be useful to discover the topological links under the management of a particular subordinate OS. This can be done by

1. retrieving all the managed elements and associated PTPs on a per subordinate OS basis,
2. retrieving all the topological links in the management domain, and
3. for each topological link, determine whether the associated PTPs are in managed elements that are managed by the same subordinate OS.

This approach is more complex for the discovering OS. Alternately, the top-level OS could offer an operation to retrieve topological links on a per subordinate OS basis (this is the recommended approach).

Retrieval of all subnetworks managed by a subordinate OS may be problematic if the top-level OS uses a different subnetwork partitioning than the subordinate OS. In the situation shown in **Figure 3-1**, the ADMs and DCSs (that interconnect the rings) come from different equipment suppliers and, as such, could be managed by different subordinate OSs. The subordinate OS managing the DCSs will have 3 singleton subnetworks, and the subordinate OS managing the ADMs will have many open ring subnetworks to manage (6 in all). Since the top-level OS has a view of both the DCSs and ADMs, it may represent all four of the rings as a single subnetwork. Clearly, this is a much simpler view than using the combined subnetwork partitioning of the two subordinate OSs. So, if the top-level OS exposes a different subnetwork decomposition than the subordinate OSs, it is recommended that

retrieval of subnetwork managed by the subordinate OSs not be offered; otherwise this operation may be offered.

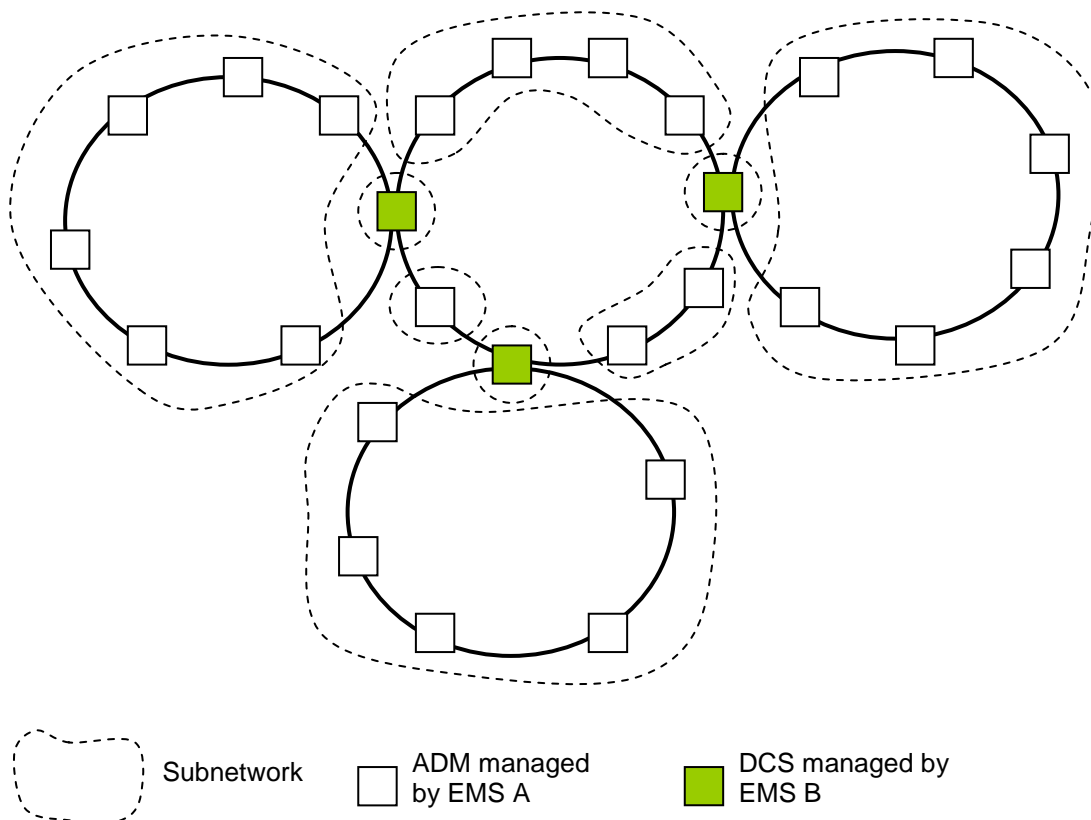


Figure 3-1. Target OS Subnetwork Partitioning

Based on the above discussion, [R_TMF518_MRI_II_0013](#) is stated in Section “Operations System (OS) Inventory”.

3.3.1.3 “Bulk” inventory retrieval

It is often required for an OS to retrieve information in bulk from another OS. This is the case periodically for initial alignment and subsequent auditing purposes. The selected mechanism is named “bulk inventory retrieval”, with support for a *difference* retrieval capability (i.e., the retrieval of all inventory that has changed from a given point in the past).

R_TMF518_MRI_II_0002	<p>The Interface shall support the capability for a requesting OS to retrieve the inventory from a target OS via <i>bulk inventory retrieval</i>.</p> <p>The retrieved inventory information may be partitioned in any manner provided that:</p>
----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> Individual object instances (but not necessarily the fully containment hierarchy) are completely contained in a single partition The full details of an object instance may only appear in one partition but to set context, it may be necessary to include the object instance with just its name in several partitions. The receiver can reconstruct the document as the various partitions are received. So, for example, if the hierarchy of an ME is to be sent in multiple partitions, the context of each partition needs to be present: in this example, the MD and ME objects with just their names and whatever part of the ME hierarchy that is to be transported in the given partition. <p>See also the requirement R_TMF518_MRI_IV_0149 which specifies different retrieval styles.</p>
Source	TMF 517 Version 1.2, Requirement II.9

R_TMF518_MRI_II_0003	Each of the bulk inventory mechanisms listed in R_TMF518_MRI_II_0002 may optionally support the ability to retrieve only the inventory that has been modified after a Diff Date and Time provided by the requesting OS.
Source	TMF 517 Version 1.2, Requirement II.10

R_TMF518_MRI_II_0004	<p>Each of the bulk inventory mechanisms listed in R_TMF518_MRI_II_0002 shall support the ability to provide a subset of the inventory known to the target OS. The subset is defined by a filter provided by the requesting OS.</p> <p>The filter shall contain two parts:</p> <ul style="list-style-type: none"> Base Instances List: It is a list of names of instances present in the inventory repository, used to determine a set of disjoint sub-trees of instances which they are the roots of. (Scope Qualifier, Selection Qualifier) pairs list: <ul style="list-style-type: none"> Scope Qualifier Not all the instances present in the set of disjoint sub-trees determined by the list of Base Instances should be considered to construct the bulk retrieval response. A Scope Qualifier designates, within this set of sub-trees, those instances which should be considered in the scope of the bulk retrieval response. It can be of four kinds which are mutually exclusive: <ol style="list-style-type: none"> empty: in this case all the instances of all the sub-trees are considered.
----------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>2. an Included Object Type: in this case all the instances of all the sub-trees which correspond to this object type are considered; the list of valid object types is given after this requirement table.</p> <p>3. an attribute matching filter: it is a logical expression conformant to the requirements R_TMF518_FMW_IL_0023 and R_TMF518_FMW_IL_0024 from the TMF518_FMW_BA. All the instances of all the sub-trees matching this logical expression are considered.</p> <p>4. an object name: it will determine the unique instance among all the instances of all the sub-trees that needs to be considered; note that this case is also covered by the item 3 above; reason to keep it in this requirement is for backward compatibility.</p> <ul style="list-style-type: none"> – if “empty” is used as a Scope Qualifier there must be a single (Scope Qualifier, Selection Qualifier) pair in the list; – no two different pairs in the list should have the same Scope Qualifier. – in case when the different Scope Qualifiers in the list are in mutual contradiction, the return result may be inconsistent (e.g. contradiction in two matching filter expressions). – the Scope Qualifiers in the list of pairs (case of the “included object type”) need to be picked such that each selected base class has a single (connected) sub-tree below it. For example, if all the EHs under a given MD where needed, the requesting OS would need to list the given MD as a base instance and put ME and EH in the Included Object Types list. <p>○ Selection Qualifier The information to return from each instance in the scope is determined from the Selection Qualifier that can be one of the three kinds:</p> <ul style="list-style-type: none"> – NAME: the name attribute only – ATTRS: all the non-relationship pointers attributes (including the name) – FULL: all the attributes (including the name) and the relationship pointers attributes
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	Note that there is one Selection Qualifier per Scope Qualifier.
Source	Adapted and enhanced from TMF 517 Version 1.2, Requirement II.11

The valid Object Type strings correspond to the following abbreviations used for the object type names:

- MD – management domain
- ME – managed element
- EH – equipment holder
- EQ – equipment
- PTP, FTP, CTP, TPPool
- EPG – equipment protection group
- FD – Flow Domain
- FDF – Flow Domain Fragment
- MFD – Matrix Flow Domain
- MFDfr – Matrix Flow Domain Fragment
- PG – protection group
- SNC – subnetwork connection
- RT – route
- CC – cross connection
- SN – subnetwork
- TCP – Traffic Conditioning Profile
- TL – topological link
- OS – operations system
- TMD – transmission descriptor

Some examples are listed in the following table using an intuitive pseudo concrete syntax.

[The “;” is used as a separator, and “=” is used to associate a value with a type of object. For example, MD=md1;ME=ME123;EH=ACME7 identifies an equipment holder within managed element ME123 within management domain md1. The relative name of the equipment holder is ACME7.

Also attribute matching filter use XPATH expressions.]

Filter Constraint	Simple Filter
-------------------	---------------

Given an ME name (e.g., md1/ME123), get all the EHs and Equipment Attributes	BaseInstanceList: { (MD=md1;ME=ME123) } ScopeAndSelection: { (EH, ATTRS), (EQ, ATTRS) }
Given an ME name (e.g., md1/ME123), get all the contained Ehs Attributes and the ME attributes	BaseInstanceList: { (MD=md1;ME=ME123) } ScopeAndSelection: { (ME, ATTRS), (EH, ATTRS) }
Get all MEs and associated attributes (within the entire inventory known to the target OS)	BaseInstanceList: { } ScopeAndSelection { (ME, ATTRS) }
Get all ME, OS and MD names (within the entire inventory known to the target OS)	BaseInstanceList: { } ScopeAndSelection: { (ME, NAME), (MD, NAME), (OS, NAME) }
Given an EH name (e.g., md1/ME123/ACME7), get all the contained Equipment (and relationship pointers)	BaseInstanceList: { (MD=md1;ME=ME123;EH=ACME7) } ScopeAndSelection: { (EQ,FULL)}
Given a PTP name (e.g., md1/ME123/PTPxyz), get the PTP Attributes	BaseInstanceList: { (MD=md1;ME=ME123;PTP=PTPxyz) } ScopeAndSelection: { (PTP, ATTRS) }
Query all the MEs under MD=md1 which attribute "version" equals to "2.0". Get their Names	BaseInstanceList: { (MD=md1) } ScopeAndSelection: { { {http://www.w3.org/TR/xpath} {/meAttrs[softwareVersion='2.0']} }, {NAME} }

Table 3-1. Examples of the Inventory Filter

3.3.1.4 Multi-Object Inventory Update

The capability known as **Multi-Object Inventory Update** entails an OS requesting that another OS (referred to as the target OS) update its inventory based on a provided collection of updates. The expectation is that the target OS update its inventory as requested, but no other side-effects are expected (e.g., creating an SNC in the network). This is a key point concerning this capability.

The inventory update request can involve addition (new object), modification (change to an existing object) or deletion (removal of an object).

The inventory update request is delivered in the form of one or more subtrees. The root of each proposed subtree update shall already exist in the target OS's inventory. The subtrees can be integrated into the inventory of the target OS in several ways (these are referred to as integration modes):

1. Replace – for each proposed subtree update, everything from the root and below in the target OS's inventory is replaced by the proposed subtree update.
2. Merge without Overwrite – each proposed subtree update is merged with the associated subtree in the target OS's inventory. For each proposed subtree update, branches and leaves are added to the associated subtree in the target OS's inventory to achieve alignment with the proposed update. In the case of overlapping objects (in the proposed update and in the target OS's inventory), the object in the target OS remains unchanged.
3. Merge with Overwrite – this is the same as Merge without Overwrite except for the treatment of overlapping objects. In this case,
 - values in the existing object instances are overwritten with new values from the update subtree
 - new attribute-value pairs are added for optional attributes that were not present in the existing graph but which are present in the update subtree
 - values that are not covered in the update subtree but which are present in the object instances of the existing graph are left unchanged (note that the update subtree may only update a subset of the attributes for a given object instance).

In addition to the integration modes of operation stated above, it is also possible to stipulate that an inventory update request be “best effort” or “atomic” (these are referred to as effort modes). “Best effort” means that the target OS shall make as many of the requested updates as possible and report on the ones that failed. “Atomic” means that the target OS must make all the requested updates or completely roll back the operation. Either effort mode can be applied to any of the integration modes.

Figure 3-2 shows the effect of an updated inventory request for both the Merge without Overwrite and Merge with Overwrite modes of the inventory update capability.

- The Replace mode is not shown since the result is the same as the proposed update from the Requesting OS.
- The resulting subtree structure (in terms of objects and containment structure) is the same for both the merge with and without overwrite modes. This is true in general.

The handling of deletions can be covered most directly by using the Replace mode. Another option is to use the Merge with Overwrite mode and set the resourceState to “Retiring/unavailable” for objects that are to be deleted. This would likely involve some policy for the target OS concerning how long it should retain an object in the retiring resourceState before the object is removed from its inventory.

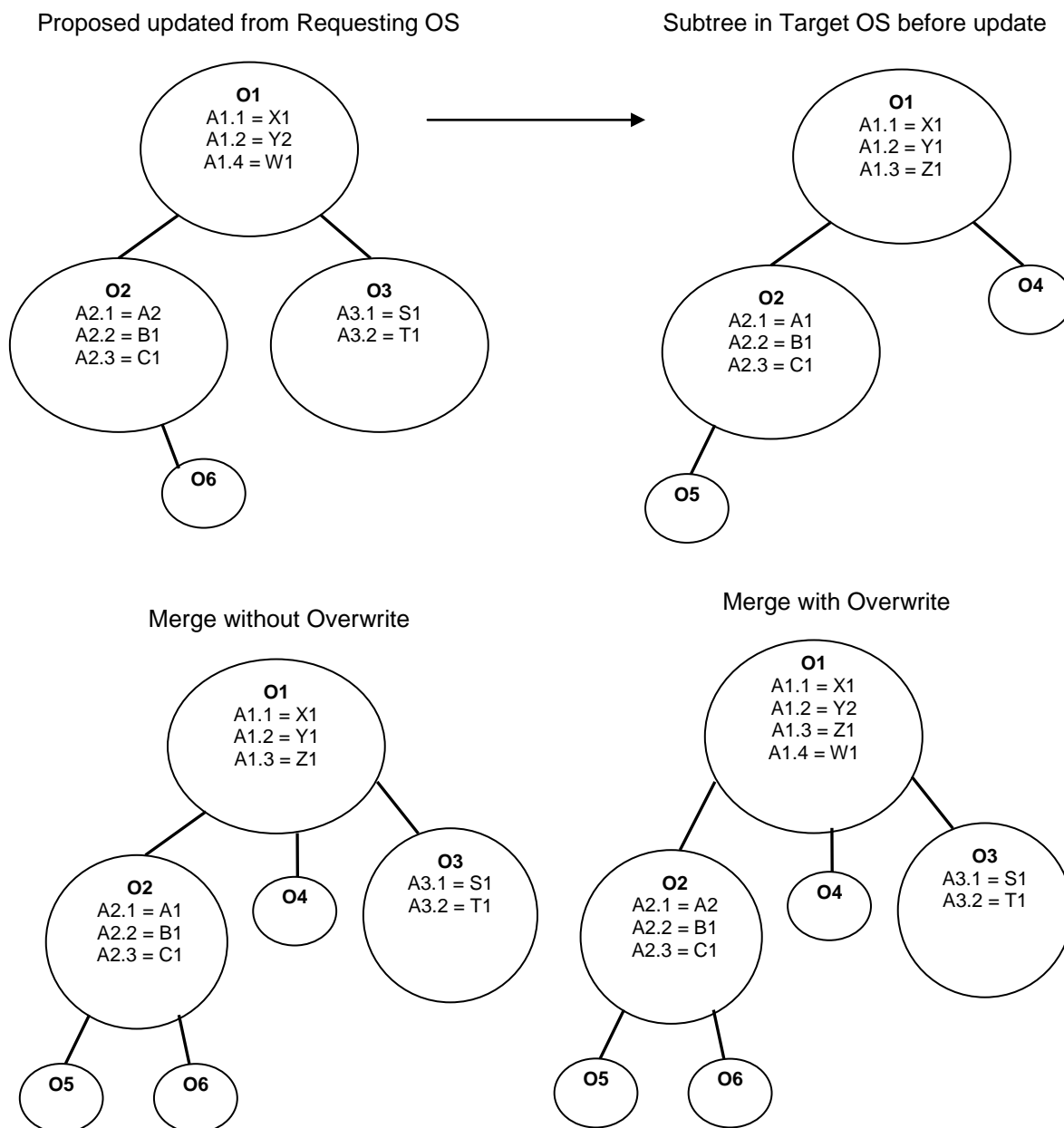


Figure 3-2. Inventory Update Example

The following example may help further clarify *inventory update* capability.

1. Assume that the target OS currently has an SNC which is in the “PENDING” SNC state and resourceState = “Planning”.
2. Another OS sends an inventory update request concerning the same SNC (there could be other updates in the request) with the SNC state listed as “PARTIAL” and the resourceState = “Installing”.

- a. The integration mode is “replace” and the effort mode is “best effort”.
- b. The only expectation is that the target OS update its inventory to reflect this change to the SNC. The target OS is not required to make any changes to the network to reflect this requested update.

There are at least two wrinkles to this scenario:

1. After the operation is complete, there is no reason to keep the target OS from actually making changes to the network to reflect the update indicated by the requesting OS. This is a policy decision that does not conflict with the intent of the inventory update operation.
2. On the same token, it is acceptable for the target OS to reject the request (or some component of the overall request) if the requested update violates a policy that has been established for the target OS. For example, the target OS may be designed not to fulfill inventory update requests if the inventory suggested in the update request is known to be different from the current view of the network as known to the target OS.

In some cases, the inventory update involves a repetitive pattern, e.g., the addition of 1000 DSL SNCs with the same traffic characteristics (only the names, aEnd list and zEnd are different). This can be handled in several ways, i.e.,

1. Simply list all the updates in the inventory layout and send the layout to the target OS in the response to an update inventory request.
2. Use some type of shorthand notation in the inventory layout and send to the target OS in the response to an update inventory request.
3. Create a separate operation for various types of repetitive inventory layout updates.

Option #1 is currently supported and that Option #2 may be considered for a future release (after some implementation experience has been gained). However, Option #3 is not recommended.

It must be emphasized that the inventory update operation described here is different from the createSNC or a multi-object version of createSNC. The createSNC operation requests that the target OS create a new SNC in the pending state under the extensive conditions associated with the createSNC operation. The behavior conditions associated with createSNC do not apply to the inventory update operation when a new SNC in the pending state is reported. The inventory update operation provides **uniform and generic behavior** to all the requested updates whereas the createSNC and other similar operations have very **specific behavior** and error responses. In a given situation, one needs to select between the two options.

R_TMF518_MRI_II_0005	<p>The Interface shall support the ability for one OS (Requesting OS) to request that another OS (Target OS) update its inventory.</p> <ul style="list-style-type: none"> • The update request may apply to multiple objects of possibly different types. • The update request may involve a combination of object additions, object modifications and object deletions. • Only the data kept by the Target OS is affected. There is not expectation that the underlying network is modified.
----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<ul style="list-style-type: none"> One of the following integration modes shall be associated with each update request: <ul style="list-style-type: none"> Replace – for each proposed subtree update, everything from the root and below in the target OS's inventory is replaced by the proposed subtree update. Merge without Overwrite – each proposed subtree update is merged with the associated subtree in the target OS's inventory. For each proposed subtree update, branches and leaves are added to the associated subtree in the target OS's inventory to achieve alignment with the proposed update. In the case of overlapping objects (in the proposed update and in the target OS's inventory), the object in the target OS remains unchanged. Merge with Overwrite – this is the same as Merge without Overwrite except for the treatment of overlapping objects. In this case, the proposed object replaces its associated object instance in the target OS's inventory. In addition to the integration modes of operation stated above, it is also possible to stipulate that an inventory update request be “best effort” or “atomic” (these are referred to as effort modes). “Best effort” means that the target OS shall make as many of the requested updates as possible and report on the ones that failed. “Atomic” means that the target OS must make all the requested updates or completely roll back the operation. Either effort mode can be applied to any of the integration modes.
Source	TMF518_MRI, Version 1.0

3.3.1.5 Management Domain (MD) Inventory

R_TMF518_MRI_II_0006	The Interface shall support requests to retrieve all the attributes of a given Management Domain.
Source	TMF 517 Version 1.2, Requirement II.12
R_TMF518_MRI_II_0007	The Interface shall support requests to retrieve the objects instances that are named (contained) under a given management domain. The possible object types include the following: subnetwork, topological link and managed element.
Source	TMF 517 Version 1.2, Requirement II.13

3.3.1.6 Operations System (OS) Inventory

R_TMF518_MRI_II_0008	The Interface shall support requests to retrieve all the attributes of a given Operations System.
Source	TMF 517 Version 1.2, Requirement II.14

R_TMF518_MRI_II_0010	The Interface shall support requests to retrieve all the Management Domains to which the top-level OS has a “manages” relationship.
Source	TMF 517 Version 1.2, Requirement II.16

R_TMF518_MRI_II_0012	The Interface shall support requests to retrieve all the subordinate Operations Systems to which the top-level OS has a “manages” relationship.
Source	TMF 517 Version 1.2, Requirement II.18

The motivation for [R_TMF518_MRI_II_0013](#) is explained in Section 3.3.1.1.

R_TMF518_MRI_II_0013	The Interface shall support the request to retrieve all the managed elements, topological links and subnetworks to which a given subordinate OS has a “manages” relationship. Note that there will not be any “manages” relationships with subnetworks unless the top-level and subordinate OSs uses the same subnetwork partitioning.
Source	TMF 517 Version 1.2, Requirement II.19

R_TMF518_MRI_II_0014	The Interface shall support requests to retrieve all the TMDs to which a given OS (top-level or subordinate) has an “offers” relationship.
Source	TMF 517 Version 1.2, Requirement II.20

3.3.2 Requirements for Connection Oriented Technologies

3.3.2.1 Managed Element (ME) Inventory

R_TMF518_MRI_II_0015	The Interface shall allow the requesting OS to retrieve the attributes of all the Managed Elements (ME)s that are contained in a management domain.
Source	TMF 513 Version 3.0, Requirement II.002

R_TMF518_MRI_II_0017	The Interface shall allow the requesting OS to retrieve the
----------------------	-------------------------------------------------------------

	attributes of all the Managed Elements (ME)s that are contained within a Subnetwork specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.004

R_TMF518_MRI_II_0019	The Interface shall allow the requesting OS to retrieve the attributes of a Managed Elements (ME)s which name is specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.006

3.3.2.2 Subnetwork Inventory

R_TMF518_MRI_II_0020	The Interface shall allow the requesting OS to retrieve the attributes of a Subnetwork, given a Subnetwork name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.8

R_TMF518_MRI_II_0021	The Interface shall allow the requesting OS to retrieve the names of the containing Subnetwork(s) given a Managed Element name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.9

R_TMF518_MRI_II_0023	The Interface shall allow the requesting OS to retrieve the attributes of all the top level Subnetwork (s).
Source	TMF 513 Version 3.0, Requirement II.11

3.3.2.3 Topological Link (TL) Inventory

R_TMF518_MRI_II_0024	The Interface shall allow the requesting OS to retrieve the attributes of all the top level Topological Links between the Subnetwork (s).
Source	TMF 513 Version 3.0, Requirement II.12

R_TMF518_MRI_II_0026	The Interface shall allow the requesting OS to retrieve the attributes of a top level Topological Link, given a top level TL name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.14

R_TMF518_MRI_II_0027	The Interface shall allow the requesting OS to retrieve the attributes of all the Topological Links between NEs and internal
----------------------	------------------------------------------------------------------------------------------------------------------------------

	to NEs given a Subnetwork specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.15

R_TMF518_MRI_II_0029	The Interface shall allow the requesting OS to retrieve the attributes of a Topological Link, given a TL name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.17

R_TMF518_MRI_II_0150	<p>The Interface shall allow the requesting OS to retrieve the attributes of all Topological Links (TLs) that are terminated at a specific Flow Domain (FD) given by the requesting OS.</p> <p>Note:</p> <p>This also includes the internal Topological Links.</p>
Source	TMF 513 Version 3.1, Requirement II.380

3.3.2.4 Subnetwork Connection (SNC) Inventory

3.3.2.4.1 Flexible SNC

A In the following requirements, an SNC shall be considered a flexible SNC unless explicitly stated otherwise.

R_TMF518_MRI_II_0030	The Interface shall allow the requesting OS to retrieve the attributes of all the Subnetwork Connection (SNC)s that are contained within a Subnetwork specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.018

R_TMF518_MRI_II_0032	The Interface shall allow the requesting OS to retrieve the attributes of all the Subnetwork Connection (SNC)s that are contained within a Subnetwork and for specific SNC layer rate(s) specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.020

R_TMF518_MRI_II_0034	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Subnetwork Connection (SNC)s that contain a Termination Point (TP) and for specific connection rates(s) specified by the requesting OS .</p> <ol style="list-style-type: none"> 1. If a Physical Termination (PTP) is specified, then all SNC(s) that pass through the contained Connection Termination Point (CTP)s are returned. The SNCs returned include the CTPs at either end or as part of their route.
----------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>2. If a CTP is specified, then all the SNC(s) that pass through the specified CTP are returned. The SNCs returned include the CTPs at either end or as part of the route.</p> <p>If the CTP provides the source of a point-to-multipoint SNC then all the SNCs of that multipoint configuration that connect to that CTP will be returned.</p>
Source	TMF 513 Version 3.0, Requirement II.022

R_TMF518_MRI_II_0036	The Interface shall allow the requesting OS to retrieve the attributes of a Subnetwork Connection (SNC)s given a SNC name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.024

R_TMF518_MRI_II_0037	The Interface shall allow the requesting OS to retrieve the attributes of a Subnetwork Connection (SNC)s given a SNC user label specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.025

R_TMF518_MRI_II_0153	The Interface shall allow the requesting OS to retrieve the subnetwork connection management mode of operation.
Source	TMF 513 Version 3.0, Requirement II.100

3.3.2.4.2 Fixed SNC

A fixed SNC shall be an SNC in which all of its cross connects are fixed.

R_TMF518_MRI_II_0038	The Interface shall allow the requesting OS to retrieve the attributes of all the fixed Subnetwork Connections contained within a Subnetwork specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.179

R_TMF518_MRI_II_0040	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the fixed Subnetwork Connections that contain the Termination Point and connection rate(s) specified by the requesting OS.</p> <p>1) If a Physical Termination Point is specified, then all SNC(s) that pass through the contained Connection Termination Point (CTP)s are returned. The SNCs returned include the CTPs at either end or as part of</p>
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>their route.</p> <p>2) If a CTP is specified, then all the SNC(s) that pass through the specified CTP are returned. The SNCs returned include the CTPs at either end or as part of the route.</p> <p>If the CTP provides the source of a point-to-multipoint SNC then all the SNCs of that multipoint configuration that connect to that CTP will be returned.</p>
Source	TMF 513 Version 3.0, Requirement II.181

3.3.2.5 Route Inventory

R_TMF518_MRI_II_0042	<p>The Interface shall allow the requesting OS to retrieve the attributes of a Route given a Subnetwork Connection (SNC) name specified by the requesting OS.</p> <p>If the SNC has alternative routes, then the Interface shall allow the requesting OS to retrieve the intended route (if the SNC is the Pending or Partial state), the active route otherwise.</p>
Source	TMF 513 Version 3.0, Requirement II.26

R_TMF518_MRI_II_0043	<p>The Interface shall allow the requesting OS to retrieve the attributes of a Route and the attributes of all the Topological Link (TL)s given a Subnetwork Connection (SNC) name specified by the requesting OS.</p> <p>If the SNC has alternative routes, then the Interface shall allow the requesting OS to retrieve the intended route (if the SNC is the Pending or Partial state), the active route otherwise.</p>
Source	TMF 513 Version 3.0, Requirement II.218

R_TMF518_MRI_II_0044	The Interface shall allow the requesting OS to retrieve the attributes of the backup Route given a Subnetwork Connection (SNC) name and Route identifier specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.256

R_TMF518_MRI_II_0045	The Interface shall allow the requesting OS to retrieve the attributes of all the Routes (intended and backup) given a Subnetwork Connection (SNC) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.260

R_TMF518_MRI_II_0046	The Interface shall allow the requesting OS to retrieve the attributes of the intended Route given a Subnetwork Connection (SNC) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.261

3.3.2.6 Termination Point (TP) Inventory

R_TMF518_MRI_II_0047	The Interface shall allow the requesting OS to retrieve the attributes of a Termination Point (TP) given a TP name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.27

R_TMF518_MRI_II_0048	The Interface shall allow the requesting OS to retrieve the attributes of all the Connection Termination Point (CTP) that are associated with a Traffic Descriptor (TD) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.29

R_TMF518_MRI_II_0155	The Interface shall allow the requesting OS to retrieve the mapping mode of a Connection Termination Point (CTP) name specified by the requesting OS.
Source	TMF 513 v3.0, Requirement II.071

R_TMF518_MRI_II_0049	<p>The Interface shall allow the requesting OS to retrieve the names of all the Termination Point Pool (TP Pool)s that are associated with a Termination Point (TP) specified by the requesting OS.</p> <p>Descriptions of the associations can be obtained from the respective TPPool attributes for the description of use.</p> <p>In case of ATM VP TP Pool administration the specified TP is an ATM NI CTP representing a real user network interface that gets partitioned into virtual user network interfaces represented by TP pools. Each TP pool contains ATM VP CTPs that are clients of the specified ATM NI CTP.</p>
Source	TMF 513 Version 3.0, Requirement II.30

R_TMF518_MRI_II_0050	The Interface shall allow the requesting OS to retrieve the attributes of all the Termination Point (TP)s that are associated with a Transmission Descriptor (TMD) name specified by the requesting OS.
----------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Source	TMF 513 Version 3.0, Requirement II.186
--------	-----------------------------------------

3.3.2.7 Termination Point Pool (TPPool) Inventory

R_TMF518_MRI_II_0051	The Interface shall allow the requesting OS to retrieve the attributes of a Termination Point Pool (TP Pool) given a TP Pool name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.268

R_TMF518_MRI_II_0052	The Interface shall allow the requesting OS to retrieve the attributes of all the Termination Point Pool (TP Pool)s given a Subnetwork name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.031

R_TMF518_MRI_II_0054	The Interface shall allow the requesting OS to retrieve the names of all the Termination Point (TP)s and Group Termination Point (GTP)s that have been grouped by the Termination Point Pool (TP Pool) specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.28

3.3.2.8 Physical Termination Point (PTP) Inventory

R_TMF518_MRI_II_0055	<p>The Interface shall allow the requesting OS to retrieve the attributes all the Physical Termination Point (PTP)s (ports) given a Managed Element (ME) name and one or more layer rate(s) specified by the requesting OS.</p> <p>The Interface shall allow the requesting OS to specify one or more layer rates to set the scope of the retrieval of PTPs from the requesting OS.</p> <p>Example physical TP layer rates covering SONET and SDH are: Electrical STS1 / STM0; Electrical STS3 / STM1; Optical OC1 / STM0; Optical OC3 / STM1</p>
Source	TMF 513 Version 3.0, Requirement II.33

R_TMF518_MRI_II_0057	The Interface shall allow the requesting OS to retrieve the attributes of all the Physical Termination Point (PTP)s (ports) given a Managed Element (ME) name and connection layer rate(s) specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.35

R_TMF518_MRI_II_0059	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Physical Termination Point (PTP)s (ports) and Floating Termination Point (FTP)s given a Managed Element (ME) name and one or more layer rate(s) specified by the requesting OS.</p> <p>The Interface shall allow the requesting OS to specify one or more layer rates to set the scope of the retrieval of PTPs and FTPs from the requesting OS.</p>
Source	TMF 513 Version 3.0, Requirement II.214

R_TMF518_MRI_II_0061	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Physical Termination Point (PTP)s (ports) and Floating Termination Point (FTP)s given a Managed Element (ME) name that are capable of supporting the connection layer rate(s) specified by the requesting OS.</p>
Source	TMF 513 Version 3.0, Requirement II.216

3.3.2.9 Edge Termination Point (TP) Inventory

R_TMF518_MRI_II_0064	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Edge Termination Point (Edge TP)s (ports) given a Subnetwork name and one or more layer rate(s) specified by the requesting OS.</p>
Source	TMF 513 Version 3.0, Requirement II.37

R_TMF518_MRI_II_0066	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Edge Termination Point (Edge TP)s (ports) given a Subnetwork name and that are capable of supporting connection layer rate(s) specified by the requesting OS.</p>
Source	TMF 513 Version 3.0, Requirement II.39

3.3.2.10 Contained Termination Point (TP) Inventory

R_TMF518_MRI_II_0068	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the potentially present Termination Point (TP)s contained by (served by) a given TP identified by a TP name and layer rate specified by the requesting OS.</p> <p>Some examples:</p> <p>If the requesting OS specifies an OC3 physical TP (port) name that can be mapped to STS1s or VT1.5s, then the target OS response would include 3 STS1 TPs (CTPs) and 84 VT1.5 TPs</p>
----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>(CTPs), respectively.</p> <p>If the requesting OS specifies an STS1 TP name (physical or logical TP, i.e., a physical port or a logical channel (CTP) that can be mapped to VT1.5s), then the target OS response would include 28 VT1.5 TPs (CTPs).</p>
Source	TMF 513 Version 3.0, Requirement II.41

R_TMF518_MRI_II_0070	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the currently cross-connectible or cross-connected Termination Point (TP)s that are contained by (served by) a given TP identified by a specified TP name and layer rate specified by the requesting OS.</p> <p>Some examples:</p> <p>If the requesting OS specifies an OC3 physical TP (port) name that can be mapped to STS1s or VT1.5s, then</p> <ul style="list-style-type: none"> ▪ If none of the STS1 CTPs are terminated and mapped the target OS response would include only the 3 STS1 TPs (CTPs). ▪ If all 3 STS1 CTPs are terminated and mapped, then 84 VT1.5 TPs (CTPs), are returned. <p>If the requesting OS specifies a STS1 TP name (physical or logical TP, i.e., a physical port or a logical channel (CTP) that can be mapped to VT1.5s), then the target OS response would include:</p> <ul style="list-style-type: none"> ▪ 28 VT1.5 TPs (CTPs) if the STS1 TP is terminated and mapped. ▪ No TPs if the STS1 TP is neither terminated nor mapped.
Source	TMF 513 Version 3.0, Requirement II.43

R_TMF518_MRI_II_0072	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the “in use” (actual) Termination Point (TP)s that are contained by (served by) a given TP identified by a TP name and layer rate specified by the requesting OS.</p> <p>This operation will be used when there are a large number of potential TPs (e.g., in ATM). All of the potential TPs are not returned. The TPs can be scoped on layer rate. If no layer rate is specified, then actual TP at all the contained layers are returned. If layer rate(s) are specified, then only actual TPs at the specified layer rates are returned.</p>
----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Source	TMF 513 Version 3.0, Requirement II.45
--------	----------------------------------------

3.3.2.11 Containing Termination Point (TP) Inventory

R_TMF518_MRI_II_0074	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Termination Point (TP)s that contain (serves) the TP identified by a TP name specified by the requesting OS.</p> <p>Some examples:</p> <ol style="list-style-type: none"> If the requesting OS specifies an STS1 TP (CTP) name that is a channel of an OC3, then the target OS would respond with the OC3 physical TP. If the requesting OS specifies an OC3 physical TP (port) name, then the target OS would respond with no containing TPs.
Source	TMF 513 Version 3.0, Requirement II.47

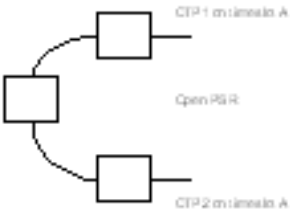
3.3.2.12 Floating Termination Point (FTP) Inventory

R_TMF518_MRI_II_0076	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Floating Termination Point (FTP)s given a Managed Element (ME) name and one or more layer rate(s) specified by the requesting OS.</p> <p>The Interface shall allow the requesting OS to specify one or more layer rates to set the scope of the retrieval of FTPs from the target OS.</p>
Source	TMF 513 Version 3.0, Requirement II.210

R_TMF518_MRI_II_0078	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Floating Termination Point (FTP)s given a Managed Element (ME) name that are capable of supporting the connection layer rate(s) specified by the requesting OS.</p>
Source	TMF 513 Version 3.0, Requirement II.212

3.3.2.13 Associated Termination Point (TP) Inventory

R_TMF518_MRI_II_0080	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Termination Point (TP)s associated with a given TP identified a TP name specified by the requesting OS.</p> <p>Some examples of associated TPs to be returned by the target</p>
----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>OS:</p> <p>In the case of an open Path Switched Ring (PSR) topology, if the requesting OS specifies a CTP (timeslot) on one end of the open PSR, the target OS shall return the CTP on the other end of the open PSR corresponding to the specified timeslot, associated with relationship (as illustrated in the figure below showing the associated CTP relationship in an open ring)</p> 
Source	TMF 513 Version 3.0, Requirement II.49

3.3.2.14 Supported Termination Point (TP) Inventory

R_TMF518_MRI_II_0081	The Interface shall allow the requesting OS to retrieve the attributes of all of the Physical Termination Point (PTP)s that are supported by the Equipment identified by the Equipment name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.50

3.3.2.15 Supporting Termination Point (TP) Inventory

R_TMF518_MRI_II_0083	The Interface shall allow the requesting OS to retrieve the attributes of all the Equipment that support the Physical Termination Point (PTP) identified by the PTP name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.52

3.3.2.16 Equipment Inventory

R_TMF518_MRI_II_0085	The Interface shall allow the requesting OS to retrieve the attributes of all the Equipment given a Managed Element (ME) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.56

R_TMF518_MRI_II_0087	The Interface shall allow the requesting OS to retrieve the attributes of all the Equipment contained directly in the Equipment Holder identified by an Equipment Holder name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.58

R_TMF518_MRI_II_0088	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Equipments and Equipment Holders given an Equipment Holder name specified by the requesting OS.</p> <p>The target OS shall return all Equipments and Equipment Holders at all levels of containment beneath the specified Equipment Holders.</p>
Source	TMF 513 Version 3.0, Requirement II.279

3.3.2.17 Supporting Equipment Inventory

R_TMF518_MRI_II_0090	The Interface shall allow the requesting OS to retrieve the attributes of all the Equipment that support an Equipment identified by an Equipment name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.225

3.3.2.18 Supported Equipment Inventory

R_TMF518_MRI_II_0092	The Interface shall allow the requesting OS to retrieve the attributes of all the Equipment that are supported by an Equipment identified by an Equipment name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.227

3.3.2.19 Traffic Descriptors (TD) Inventory

R_TMF518_MRI_II_0094	The Interface shall allow the requesting OS to retrieve the attributes of all the Traffic Descriptors that are managed by the target OS.
Source	TMF 513 Version 3.0, Requirement II.60

R_TMF518_MRI_II_0096	The Interface shall allow the requesting OS to retrieve the attributes of a Traffic Descriptors given a TD name specified by the requesting OS.
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------

Source	TMF 513 Version 3.0, Requirement II.62
--------	----------------------------------------

3.3.2.20 Transmission Descriptors (TMD) Inventory

R_TMF518_MRI_II_0151	The Interface shall allow the requesting OS to retrieve the attributes of all the Transmission Descriptors (TMDs) that are being managed by the target OS.
Source	TMF 513 Version 3.1, Requirement II.187

R_TMF518_MRI_II_0098	The Interface shall allow the requesting OS to retrieve the attributes of a Transmission Descriptor for a TMD name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.189

R_TMF518_MRI_II_0152	<p>The Interface shall allow the requesting OS to retrieve the layered transmission parameters of a Transmission Descriptor (TMD) by using a filter that is based on the layer rates and the groupings defined in the supporting document SD1-16_LayeredParameters.</p> <p>It shall also be possible to retrieve all layered transmission parameters defined in the TMD.</p> <p>For a list of the currently defined set of supported transmission parameters refer to the supporting document SD1-16_LayeredParameters.</p>
Source	TMF 513 Version 3.1, Requirement II.352

3.3.2.21 Cross Connect (CC) Inventory

3.3.2.21.1 Flexible Cross Connect

A Cross Connect shall be considered a flexible CC unless explicitly stated otherwise.

R_TMF518_MRI_II_0099	The Interface shall allow the requesting OS to retrieve the attributes of all of the Cross Connects given a Managed Element name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.63

3.3.2.21.2 Fixed Cross Connect

A fixed Cross Connect is an CC that cannot be deleted by the requesting OS.

R_TMF518_MRI_II_0100	The Interface shall allow the requesting OS to retrieve the attributes of all the fixed Cross Connect (CC)s given a Managed Element (ME) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.183

R_TMF518_MRI_II_0101	The Interface shall allow the requesting OS to retrieve the attributes of all the potentially fixed Cross Connect (CC)s that are associated with the Termination Point (TP) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.185

3.3.2.22 Group Termination Point (TP) Inventory

R_TMF518_MRI_II_0102	The Interface shall allow the requesting OS to retrieve the attributes of all the Group Termination Point (GTP)s that are managed by the target OS.
Source	TMF 513 Version 3.0, Requirement II.171

R_TMF518_MRI_II_0104	The Interface shall allow the requesting OS to retrieve the attributes of a Group Termination Point (GTP) for a GTP name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.173

R_TMF518_MRI_II_0105	The Interface shall allow the requesting OS to retrieve the attributes of a Group Termination Point (GTP) that contains a Connection Termination Point (CTP) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.184

R_TMF518_MRI_II_0106	The Interface shall allow the requesting OS to retrieve the names of all the Termination Point Pool (TP Pool)s that are associated with the Group Termination Point (GTP) name specified by the requesting OS.
Source	TMF 513 Version 3.0, Requirement II.272

3.3.3 Requirements for ConnectionLess Technologies

3.3.3.1 Termination (TP) Inventory

R_TMF518_MRI_II_0107	<p>The Interface shall allow the requesting OS to retrieve the layered transmission parameters of a Termination Point (TP) by using a filter that is based on the Layer Rates and the groupings defined in the supporting document SD1-16_LayeredParameters.</p> <p>It shall also be possible to retrieve all layered transmission parameters defined in the TP.</p> <p>For a list of the currently defined set of supported TP transmission parameters refer also to the supporting document SD1-16_LayeredParameters.</p>
Source	TMF 513 Version 3.1, Requirement II.291

R_TMF518_MRI_II_0108	The Interface shall allow the requesting OS to retrieve the attributes of the Matrix Flow Domain (MFD) that is assigned to a requesting OS specified Connectionless Port TP (CPTP).
Source	TMF 513 Version 3.1, Requirement II.292

3.3.3.2 Matrix Flow Domain Inventory

R_TMF518_MRI_II_0109	<p>The Interface shall allow the requesting OS to retrieve the attributes of all Matrix Flow Domain (MFD)s supported within a specified Managed Element (ME).</p> <p>It is irrelevant whether or not the MFDs are actually associated to a Flow Domain (FD).</p>
Source	TMF 513 Version 3.1, Requirement II.295

R_TMF518_MRI_II_0110	The Interface shall allow the requesting OS to retrieve the attributes of a Matrix Flow Domain (MFD) given a specified MFD name
Source	TMF 513 Version 3.1, Requirement II.296

R_TMF518_MRI_II_0111	The Interface shall allow the requesting OS to retrieve the attributes of all ConnectionLess Port TP (CPTP)s assigned to a given Matrix Flow Domain (MFD).
Source	TMF 513 Version 3.1, Requirement II.297

R_TMF518_MRI_II_0112	The Interface shall allow the requesting OS to retrieve the
----------------------	-------------------------------------------------------------

	attributes of the Matrix Flow Domain (MFD)s that are supported by a specific Equipment; given a specified Equipment name.
Source	TMF 513 Version 3.1, Requirement II.300

R_TMF518_MRI_II_0113	The Interface shall allow the requesting OS to retrieve all the Matrix Flow Domain (MFD)s that are associated with a specified Transmission Descriptor (TMD).
Source	TMF 513 Version 3.1, Requirement II.301

R_TMF518_MRI_II_0114	<p>The Interface shall allow the requesting OS to retrieve the layered transmission parameters of a Matrix Flow Domain (MFD) by using a filter that is based on the Layer Rate and the groupings defined in the supporting document SD1-16_LayeredParameters.</p> <p>It shall also be possible to retrieve all layered transmission parameters defined in the MFD.</p> <p>For a list of the currently defined set of supported MFD transmission parameters refer also to the supporting document SD1-16_LayeredParameters.</p>
Source	TMF 513 Version 3.1, Requirement II.302

R_TMF518_MRI_II_0115	The Interface shall allow the requesting OS to retrieve the attributes of the Flow Domain (FD) that is associated to a specified Matrix Flow Domain (MFD).
Source	TMF 513 Version 3.1, Requirement II.303

R_TMF518_MRI_II_0116	<p>The Interface shall allow the requesting OS to retrieve the attributes of all Connectionless Port TP (CPTP)s which are potentially able to be assigned to a specified Matrix Flow Domain (MFD).</p> <p>Notes:</p> <p>Potentially means: The CPTPs are on the same equipment or same rack with backplane connectivity as the specified MFD.</p> <p>It is irrelevant whether the CPTPs are already assigned to an MFD or not.</p>
Source	TMF 513 Version 3.1, Requirement II.304

3.3.3.3 Flow Domain Inventory

R_TMF518_MRI_II_0117	The Interface shall allow the requesting OS to retrieve the attributes of all Flow Domain (FD)s managed by the target OS.
Source	TMF 513 Version 3.1, Requirement II.314

R_TMF518_MRI_II_0118	The Interface shall allow the requesting OS to retrieve the attributes of all Connectionless Port TP (CPTP)s associated to the Flow Domain (FD). It shall be possible to filter this request to retrieve only “fdEdge” (UNI and E-NNI sides), only “fdInternal” or all CPTPs.
Source	TMF 513 Version 3.1, Requirement II.315

R_TMF518_MRI_II_0119	The Interface shall allow the requesting OS to retrieve the attributes of all Matrix Flow Domain (MFD)s associated with a specified Flow Domain (FD).
Source	TMF 513 Version 3.1, Requirement II.316

R_TMF518_MRI_II_0120	<p>The Interface shall allow the requesting OS to retrieve the attributes of all the Flow Domain (FD)s given a specified FD user label.</p> <p>Note:</p> <p>The user label may or may not be unique within the target OS domain. Non-unique user labels can be used by the Requesting OS for example to associate several FDs from the same target OS, typically to indicate that those FDs participate in the same service provider's application (corporate customer, etc.). This situation may occur for example in case the Target OS supports only singleton FDs.</p>
Source	TMF 513 Version 3.1, Requirement II.317

R_TMF518_MRI_II_0121	The Interface shall allow the requesting OS to retrieve the attributes of a Flow Domain (FD) given a specified FD name
Source	TMF 513 Version 3.1, Requirement II.318

R_TMF518_MRI_II_0122	The Interface shall allow the requesting OS to retrieve the layered transmission parameters of a Flow Domain (FD) by using a filter that is based on the layer rates and the groupings defined in the supporting document SD1-16_LayeredParameters .
----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>It shall also be possible to retrieve all layered transmission parameters defined in the FD.</p> <p>For a list of the currently defined set of supported FD transmission parameters refer also to the supporting document SD1-16_LayeredParameters.</p>
Source	TMF 513 Version 3.1, Requirement II.319

3.3.3.4 Traffic Conditioning Inventory

R_TMF518_MRI_II_0123	The Interface shall allow the requesting OS to retrieve the attributes of all the Traffic Conditioning (TC) Profiles that are managed by the target OS.
Source	TMF 513 Version 3.1, Requirement II.330

R_TMF518_MRI_II_0124	The Interface shall allow the requesting OS to retrieve the attributes of a Traffic Conditioning (TC) Profile for a specified TC Profile name.
Source	TMF 513 Version 3.1, Requirement II.331

R_TMF518_MRI_II_0125	The Interface shall allow the requesting OS to retrieve the attributes of all the Termination Point (TP)s that are associated with a specified Traffic Conditioning (TC) Profile name.
Source	TMF 513 Version 3.1, Requirement II.332

3.3.3.5 Flow Domain Fragment (FDFr) Inventory

R_TMF518_MRI_II_0126	The Interface shall allow the requesting OS to retrieve the attributes of all the Flow Domain Fragment (FDFr)s contained within a specified Flow Domain (FD). It shall be possible to filter the FDFrs to be retrieved based on their connectivity rates.
Source	TMF 513 Version 3.1, Requirement II.338

R_TMF518_MRI_II_0127	The Interface shall allow the requesting OS to retrieve the attributes of all the Flow Domain Fragment (FDFr)s that connect or pass through client Flow Point (FP)s of the specified Connectionless Port TP (CPTP).
Source	TMF 513 Version 3.1, Requirement II.339

R_TMF518_MRI_II_0128	The Interface shall allow the requesting OS to retrieve the
----------------------	-------------------------------------------------------------

	attributes of a Flow Domain Fragment (FDFr) given a specified FDFr name.
Source	TMF 513 Version 3.1, Requirement II.340

R_TMF518_MRI_II_0129	The Interface shall allow the requesting OS to retrieve the attributes of all the Flow Domain Fragment (FDFr) given a specified FDFr user label.
Source	TMF 513 Version 3.1, Requirement II.341

R_TMF518_MRI_II_0130	The Interface shall allow the requesting OS to retrieve the attributes of the Flow Domain Fragment (FDFr) connecting a Flow Point (FP) given a specified FP name.
Source	TMF 513 Version 3.1, Requirement II.342

R_TMF518_MRI_II_0131	The Interface shall allow the requesting OS to retrieve the Matrix Flow Domain Fragment (MFDFr)s of a Flow Domain Fragment (FDFr) given a specified FDFr name.
Source	TMF 513 Version 3.1, Requirement II.343

R_TMF518_MRI_II_0132	<p>The Interface shall allow the requesting OS to retrieve the layered transmission parameters of a Flow Domain Fragment (FDFr) by using a filter that is based on the groupings defined in the supporting document SD1-16_LayeredParameters.</p> <p>It shall also be possible to retrieve all layered transmission parameters defined in the FDFr.</p> <p>For a list of the currently defined set of supported FDFr transmission parameters refer also to the supporting document SD1-16_LayeredParameters.</p>
Source	TMF 513 Version 3.1, Requirement II.344

3.4 Category III: Abnormal or Exception Conditions, Dynamic Requirements

See section 4.1.1 in the [TMF518 FMW](#) for the complete list of exceptions that may be raised by a target OS in response to a requesting OS.

R_TMF518_MRI_III_0146	The Multi-Object Inventory Update request shall support the following behavior:
-----------------------	---------------------------------------------------------------------------------

	<ul style="list-style-type: none"> In the case of failure when the effort mode is set to “atomic”, the target OS shall indicate the reason for failure using the exceptions described in Table 3-2. In this case, the target OS may stop processing when it encounters the first problem that causes the request to fail. In the case of failure when the effort mode is set to “best effort”, the target OS shall indicate the reason for failure (on a per object basis) using the exceptions described in Table 3-2. In this case, the target OS needs to be exhaustive in reporting failed aspects of the request. It is possible, however, to use some shortcuts. In particular, the target OS may report that it could not update a given object instance as well as all children of the object instance.
Source	TMF518_MRI, Version 1.0

Table 3-2. Exceptions for the Multi-Object Update Request

Create	Modify	Delete
<p>EXCPT_INVALID_INPUT</p> <ul style="list-style-type: none"> Creation of given Object type is not supported <i>[target OS should also supply the name of the object type]</i> Attempting to create an object whose containing object (possibly several layers up) does not exist <i>[target OS should supply the name of the containing object that it claims does not exist]</i> 	<p>EXCPT_INVALID_INPUT</p> <ul style="list-style-type: none"> Modification of the given attribute is not supported <i>[target OS should supply a list of the (object::attribute) that are not supported]</i> Given attribute can not be set to the provided value, i.e., provided value is not supported <i>[target OS should supply a list of the (object::attribute::value) that are not supported]</i> 	<p>If a request is made to delete an object that does not exist, the target OS does not generate an exception.</p>
	<p>EXCPT_OBJECT_IN_USE</p> <ul style="list-style-type: none"> Attempting to modify an object that is in use (this may depend on the policies of the target OS) <i>[target OS should supply the name of the object instances in question that are already in use]</i> 	<p>EXCPT_OBJECT_IN_USE</p> <ul style="list-style-type: none"> Attempting to delete an object when an associated object (possibly several layers up) that is currently in use, e.g., attempting to delete a CTP when an associated SNC is still in use may violate a policy of the target OS <i>[target OS should supply the name of the containing object that is in use]</i>

<p>EXCPT_POLICY_VIOLATION</p> <ul style="list-style-type: none"> Creation of a given object violates a policy of the target OS <i>[target OS should indicate the specific policy that has been violated]</i> 	<p>EXCPT_POLICY_VIOLATION</p> <ul style="list-style-type: none"> Modification of the given attribute violates a policy of the target OS <i>[target OS should indicate the specific policy that has been violated]</i> 	<p>EXCPT_POLICY_VIOLATION</p> <ul style="list-style-type: none"> Deletion of a given object violates a policy of the target OS <i>[target OS should indicate the specific policy that has been violated]</i>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.5 Category IV: Expectations and Non-Functional Requirements

R_TMF518_MRI_IV_0149	<p>Support for bulk inventory retrieval can be done in any of the following ways:</p> <ul style="list-style-type: none"> - Batched responses – in this approach the server (target OS) sends the requested information back to the client (requesting OS) in a series of responses (batches). For example, using asynchronous communication over JMS. - Batched retrieval – in this approach the client (requesting OS) retrieves the requested information from the server (target OS) via a series of requests. This is analogous to the MTNM iterator approach. For example, using http/s with synchronous communication. - Batched File transfer – in this approach the requested inventory is delivered in the form of file(s). The approach allows the client (requesting OS) to designate the location of the file(s) (by providing a URI). For example, using FTP. <p>All of the above bulk inventory retrieval methods shall support the same Inventory output data structure schema, as well as, its batching capabilities.</p>
Source	TMF 517 Version 1.2, Requirement II.9

R_TMF518_MRI_II_0156	<p>Support for Multi-Object Inventory Update can be done in any of the two following ways:</p> <ul style="list-style-type: none"> - single request response – in this approach the requesting OS sends the information to be used for the update as input parameter in a single request - file transfer – in this approach the requesting OS delivers the information to be used for the update in a specific file and sends as input parameter of the request the location of this file (by providing a URI). Upon reception of the update request the target OS accesses the file using a file transfer protocol (for example, using FTP).
Source	TMF518_MRI, Version 1.1

R_TMF518_MRI_IV_0147	<p>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 (non-</p>
----------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	divisible) XML object structures including their full containment hierarchy; ME, TL, OS, TMD, and MLSN.
Source	TMF518_MRI, Version 1.0

R_TMF518_MRI_IV_0148	<p>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):</p> <ul style="list-style-type: none"> • 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_MRI, Version 1.0

3.6 Category V: System Administration Requirements

R_TMF518_MRI_V_0154	<p>The NML-EML Interface shall allow the NMS to retrieve the connectivity awareness capability (with regard to the connectionless technologies) provided by the EMS.</p> <p>The EMS works either in the “connectivity-aware” or in the “connectivity-unaware” mode:</p> <ul style="list-style-type: none"> • “connectivity aware” indicates that the EMS has some capability to detect the connectivity between the MFDs within an FD • “connectivity-unaware” indicates that the EMS has no capability at all to detect connectivity between the MFDs within an FD.
Source	TMF 513 Version 3.1, Requirement 290

4 Use Cases

For the following use cases, the Actor is an OS. Also, the system being “acted upon” is also an OS. The OSs in the use cases play different roles. The following roles are used in the use cases:

- a) Client OS – this role is played by an OS that is responsible for invoking service requests and receiving service responses.
- b) Server OS – – this role is played by an OS that is responsible for receiving service requests and providing service responses to a Client OS
- c) Requesting OS – this role is played by an OS that uses the services of one or more target OS. The requesting OS has a “contains” and “uses” relationships with one or more instances of either Client OS, Server OS or both.
- d) Target OS – this is role is played by an OS that provides services to a requesting OS. The target OS has a “contains” and “uses” relationships with one or more instances of either Client OS, Server OS or both.
- e) Publishing OS – this is role is played by an OS that generates notifications and publishes them on the CCV vis a notification service. A target OS can also play the role of a publishing OS.
- f) Subscriber OS – this role is played by an OS that subscribes to a notification service in order to get notifications of a specific Topic. The requesting OS may also play the role of a subscriber OS.
- g) Naming OS – this role is play by an OS with respect to a given network resource. When playing this role, the OS provides a unique name for the network resource that is to be used by all OSs on the CCV when referring to the given network resource.
- h) Discovering OS – this role is played by an OS with respect to a given network resource. When playing this role, the discovering OS is the first OS to announce the given network resource on the CCV. A discovered name is provided for the network resource as part of the announcement. For a given network resource, the naming OS and discovering OS need not be the same OS. Also, the “discovered name” can be different from the “name” of the network resource.

From this point onwards, network synchronization and reconciliation is achieved with the usual fine grained mechanisms (from MTNM) or via bulk discovery (from MTOSI).

The example scenario shown in the figure below is used in several use cases.

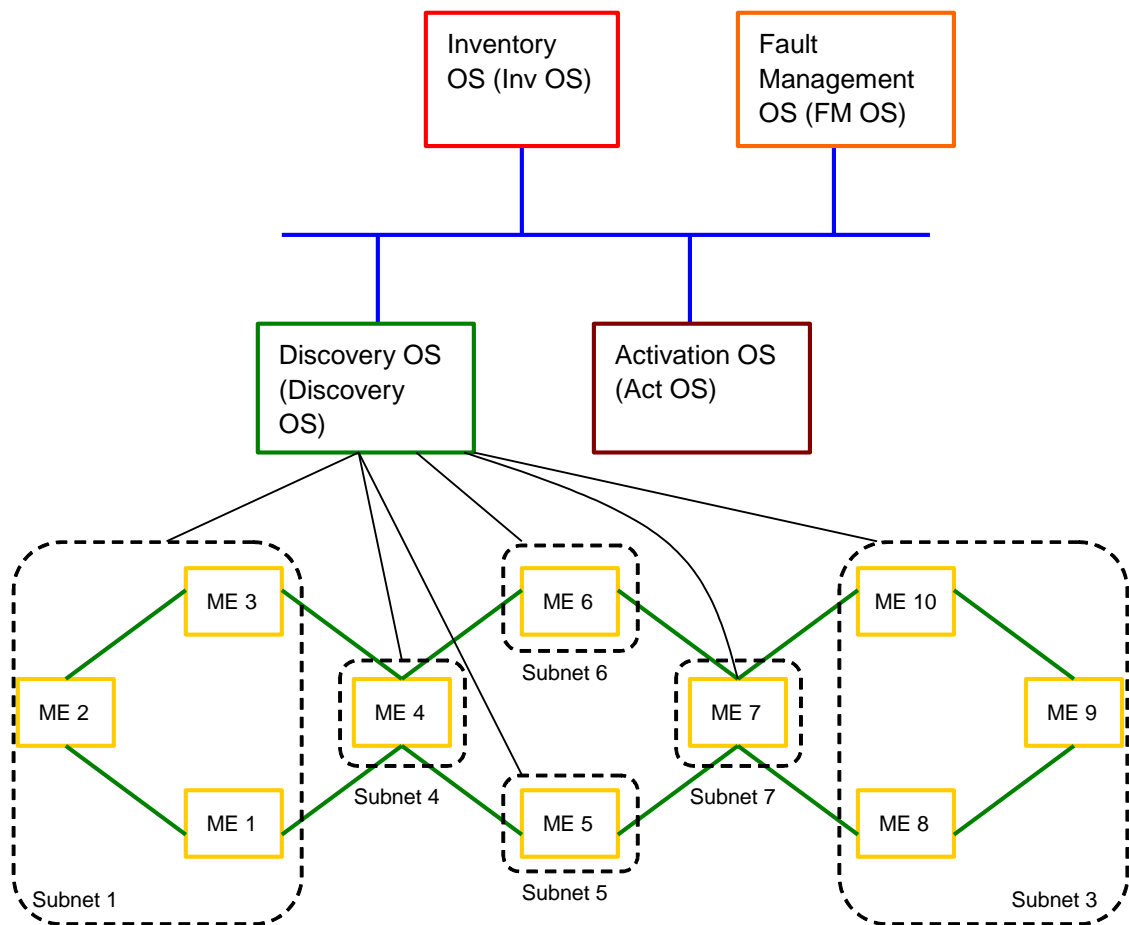


Figure 4-1. An Example of Network and Associated Management Architecture

4.1 The requesting OS registers to receive network inventory updates from the target OS

Use Case Id	UC_TMF518_MRI_0001
Use Case Name	The requesting OS registers to receive network inventory updates from the target OS
Summary	The requesting OS registers with the notification service related to the target OS, sets the appropriate filter to receive network inventory update notifications, and connects to the notification service.

Actor(s)	The requesting OS
Pre-Conditions	<ol style="list-style-type: none"> 1. The requesting OS and target OS have executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document. 2. The requesting OS has a reference for the notification service used by the target OS.
Begins When	The requesting OS sends a request to register itself at the notification service related to the target OS.
Description	<ol style="list-style-type: none"> 1. The requesting OS registers at the notification service related to the target OS as a consumer of notifications (if this has not been done earlier). 2. The requesting OS sets the filter criteria needed to receive inventory updates from the target OS via the notification service. 3. The requesting OS connects to the notification service and thus is able to receive notifications matching the filter conditions specified (if this has not been done earlier). <p>Note:</p> <p>The requesting OS can request that the inventory notifications be filtered based on resource type (e.g., PTPs) and/or notification type (i.e., creations, attribute value changes, deletions).</p> <p>All the inventory notifications are time stamped.</p>
Ends When	<p>In case of success:</p> <p>The requesting OS receives a positive acknowledgement to its connection request to the notification service.</p> <p>In case of failure:</p> <p>The requesting OS receives a negative acknowledgement to its registration request, an invalid filter specified or the request times out.</p>
Post-Conditions	<p>In case of success:</p> <p>The requesting OS is connected to the Notification Service with filtering criteria to receive network inventory update notifications.</p> <p>In case of failure:</p> <p>The requesting OS is not connected to the Notification Service with filtering criteria to receive network inventory update notifications.</p>
Exceptions	<p>Filter creation:</p> <ul style="list-style-type: none"> • Invalid grammar <p>Filter building</p> <ul style="list-style-type: none"> • Invalid constraint

	Connection phase: <ul style="list-style-type: none"> • Illegal consumer type • Consumer already connected.
Traceability	This is use case is a generalization of Use Case 7.4.1 from TMF 513 v3.1

4.2 The requesting OS resynchronizes its database with the target OS

Use Case Id	UC_TMF518_MRI_0002
Use Case Name	The requesting OS resynchronizes its database with the target OS
Summary	The requesting OS sends a series of queries to the target OS, with the intent of re-synchronizing its understanding of the target OS's network inventory with the actual target OS inventory.
Actor(s)	The requesting OS
Pre-Conditions	<ol style="list-style-type: none"> 1. The requesting OS and target OS have executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document. 2. The requesting OS has determined that it wants to resynchronize its database with that of the target OS.
Begins When	The requesting OS sends the first inventory query to the target OS.
Description	<ol style="list-style-type: none"> 1. This Use Case is basically UC_TMF518_MRI_0003 in a different context. 2. It is also possible to do a partial re-synchronization; In this case UC_TMF518_MRI_0004 is re-used (multiple times).
Ends When	<p>In case of success:</p> <p>The requesting OS has received the last response to the requests that it sent the target OS.</p> <p>In case of failure:</p> <p>The requesting OS has determined all requests have been acknowledged or have timed out, and at least one of the requests has timed out or was acknowledged in the negative (the target OS could not or would not return the requested information).</p>
Post-Conditions	<p>In case of success:</p> <p>The requesting OS has collected all the requested inventory</p>

	<p>information from the target OS.</p> <p>In case of failure:</p> <p>The requesting OS has not collected all the inventory information that it requested.</p>
Exceptions	<ol style="list-style-type: none"> 1. Communication failure between the requesting OS and the target OS. 2. The requesting OS may include incorrect or unknown information in its queries to the target OS. 3. Entity not found: when an input parameter references an object that does not exist 4. Query unknown or not supported. 5. Unable to comply
Traceability	<p>Refer to UC_TMF518_MRI_0003 and UC_TMF518_MRI_0004</p> <p>This use case is a generalization of Use Case 7.4.2 from TMF 513 v3.1</p>

4.3 The requesting OS discovers the target OS network inventory (fine grain approach)

Use Case Id	UC_TMF518_MRI_0003
Use Case Name	The requesting OS discovers the target OS network inventory
Summary	The requesting OS sends a series of queries to the Target OS, with the intent of discovering the network inventory managed by the Target OS.
Actor(s)	The requesting OS
Pre-Conditions	<ol style="list-style-type: none"> 1. The requesting OS and target OS have executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document. 2. The requesting OS has not yet discovered the inventory managed by the target OS.
Begins When	The requesting OS sends the first inventory query to the target OS.
Description	<p>There are many possible ways for the Requesting OS to obtain the Target OS's inventory. Two possibilities are given below, however the queries can be sent in many different combinations.</p> <p>Although the requests themselves are synchronous, the Requesting OS may send subsequent inventory requests to the Target OS before</p>

	<p>receiving a response to a pending request.</p> <p>Although not mandated by this use case, the Requesting OS may want to start receiving Target OS inventory updates at the time the discovery process begins. Once the Requesting OS has received responses to all the inventory queries, the Requesting OS can analyze the inventory update notifications that it received during the discovery process (if any) and updates its inventory accordingly.</p> <p>(Approach A: Discovery of all subnetworks' details)</p> <ol style="list-style-type: none"> 1. The Requesting OS sends a request to retrieve Target OS information from the Target OS. The Target OS returns the name, user label, native Target OS name, owner, type and software version of the Target OS. 2. The Requesting OS sends a request to retrieve all Subnetworks from the Target OS. The Target OS returns the names and associated information for all the subnetworks that it manages. 3. The Requesting OS sends a request to retrieve all Managed Elements from the Target OS. For each request, the Target OS returns the names and associated information concerning the Managed Elements. 4. For each Managed Element, the Requesting OS sends a request to retrieve the Physical Termination Points (including FTPs) from the Target OS. For each request, the Target OS returns the names of the Physical Termination Points (PTPs) and all information associated with the PTP, including a indication of whether the PTP is on the edge of the containing subnetwork. Alternatively for each termination point (PTP or CTP), the Target OS sends a request to retrieve all the contained Performance Monitoring Points (PMPs) within a specified PTP or CTP from the Target OS. For each request, the Target OS returns the names of the contained PMPs and all associated information. 5. For each subnetwork, the Requesting OS sends a request to retrieve all the Topological Links within a specified subnetwork from the Target OS. The Target OS returns all the names and associated information for all the topological links associated with a subnetwork. 6. The Requesting OS sends a request to retrieve all the top-level Topological Links from the Target OS. The Target OS returns the names and associated information for each of the top-level topological links. 7. For each PTP, the Target OS sends a request to retrieve all the contained CTPs within a specified PTP or CTP from the Target OS. For each request, the Target OS returns the names of the contained CTPs and all associated information. If the CTP contains other CTPs, this information will also be returned. 8. For each identified subnetwork, the Requesting OS sends a request to retrieve all the Subnetwork Connections from the
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>Target OS. For each request, the Target OS returns the names of the subnetwork connections and all associated information.</p> <ol style="list-style-type: none"> 9. For each subnetwork connection, the Requesting OS sends a request to retrieve all the Routes. 10. For every Managed Element the Requesting OS sends a request to retrieve all the Protection Groups from the Target OS. For each request, the Target OS returns the names of the protection groups and all associated information for all the protection groups it manages within the Managed Element specified. 11. For every Managed Element the Requesting OS sends a request to retrieve all Cross-Connects from the Target OS. For each request, the Target OS returns the associated information for all the crossconnects it manages within the Managed Element specified. 12. For every subnetwork the Requesting OS sends a request to retrieve all TP Pools from the Target OS. For each request, the Target OS returns the names and associated information for all the TP Pools it manages within the subnetwork specified. 13. If supported by the Target OS, the Requesting OS sends a request to retrieve all Traffic Descriptors from the Target OS. The Target OS returns the names and associated information for all the traffic descriptors it manages. 14. If supported by the Target OS, for every Managed Element the Requesting OS sends a request to retrieve all Equipment Holders and Equipment from the Target OS. For each request, the Target OS returns the names and associated information for all the equipment holders and equipment it manages in the ME specified. 15. Refer to Use Case UC_TMF518_RPM_0002 "Requesting OS retrieves PM capabilities of a Managed Element (ME)" from TMF518_RPM. 16. The Requesting OS sends a request to retrieve all Flow Domains from the Target OS. The Target OS returns the names and associated information for all the Flow Domains that it manages. 17. For each identified Flow Domain, the Requesting OS sends a request to retrieve all the Flow Domain Fragments from the Target OS. For each request, the Target OS returns the names of the Flow Domain Fragments and all associated information. 18. For each Flow Domain Fragment, the Requesting OS sends a request to retrieve its Flow Domain Fragment Route. 19. For every Managed Element the Requesting OS sends a request to retrieve all Matrix Flow Domains from the Target OS. For each request, the Target OS returns the names and associated information for all Matrix Flow Domains it manages within the ME specified. 20. For each Matrix Flow domain, the Requesting OS sends a request to retrieve all Matrix Flow Domain Fragments. For each request, the Target OS returns the names and associated information for all Matrix Flow Domain Fragments it manages within the ME
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>specified.</p> <p>(Approach B: Discovery of subnetworks with their edge points only)</p> <ol style="list-style-type: none"> 1. The Requesting OS sends a request to retrieve all the Target OS information from the Target OS. The Target OS returns the name, user label, native Target OS name, owner, type and software version of the Target OS. 2. The Requesting OS sends a request to retrieve all Subnetworks from the Target OS. The Target OS returns the names and associated information for all the subnetworks that it manages. 3. For each subnetwork, the Requesting OS sends a request to retrieve from the Target OS the Physical Termination Points (PTPs) which are the edges of a specified subnetwork. For each request, the Target OS returns the names of the edge points and all information associated with it. 4. The Requesting OS sends a request to retrieve all the top-level Topological Links from the Target OS. The Target OS returns the names and associated information for each of the top-level topological links. 5. For each PTP, the Target OS sends a request to retrieve all the contained CTPs within a specified PTP or CTP from the Target OS. For each request, the Target OS returns the names of the contained CTPs and all associated information. If the CTP contains other CTPs, this information will also be returned. 6. For each identified subnetwork, the Requesting OS sends a request to retrieve all the Subnetwork Connections from the Target OS. For each request, the Target OS returns the names of the subnetwork connections and all associated information. 7. If supported by the Target OS, the Requesting OS sends a request to retrieve all Traffic Descriptors from the Target OS. The Target OS returns the names and associated information for all the traffic descriptors it manages.
Ends When	<p>In case of success:</p> <p>The requesting OS has received the last response to the requests that it sent the target OS.</p> <p>In case of failure:</p> <p>The requesting OS has determined all requests have been acknowledged or have timed out, and at least one of the requests has timed out or was acknowledged in the negative (the target OS could not or would not return the requested information).</p>
Post-Conditions	<p>In case of success:</p> <p>The requesting OS has collected all the requested inventory information from the target OS.</p>

	<p>In case of failure:</p> <p>The requesting OS has not collected all the inventory information that it requested.</p>
Exceptions	<ol style="list-style-type: none"> 1. Communication failure between the requesting OS and the target OS. 2. The requesting OS may include incorrect or unknown information in its queries to the target OS. 3. Entity not found: when an input parameter references an object that does not exist. See UC_TMF518_MRI_0002. 4. Query unknown or not supported. 5. Unable to comply.
Traceability	<p>R_TMF518_MRI_II_0015, R_TMF518_MRI_II_0023, R_TMF518_MRI_II_0024, R_TMF518_MRI_II_0027, R_TMF518_MRI_II_0030, R_TMF518_MRI_II_0052, R_TMF518_MRI_II_0055, R_TMF518_MRI_II_0068, R_TMF518_MRI_II_0070, R_TMF518_MRI_II_0085, R_TMF518_MRI_II_0094, R_TMF518_MRI_II_0117, R_TMF518_MRI_II_0123</p> <p>This use case is a generalization of Use Case 7.4.3 from TMF 513 v3.1.</p>

4.4 The requesting OS queries the target OS concerning inventory (fine grain approach)

Use Case Id	UC_TMF518_MRI_0004
Use Case Name	The requesting OS queries the target OS concerning inventory
Summary	<p>The requesting OS sends a request to the target OS concerning a particular network inventory item. The following network inventory queries are possible:</p> <ol style="list-style-type: none"> 1. List of all Subnetworks managed by the target OS (the names or all associated attributes). 2. List of all Managed Elements within a subnetwork (the names or all associated attributes). 3. List of all the Managed Elements that are managed by the target OS (the names or all associated attributes). 4. List of all PTPs associated with a Managed Element (just the names, or with all associated attributes). 5. List of all CTPs supported by a PTP (just the names, or with all

	<p>associated attributes).</p> <ol style="list-style-type: none"> 6. List of all CTPs supported by a CTP (just the names, or with all associated attributes). 7. List of all Topological Links within a subnetwork (the names or all associated attributes). 8. List of all top-level Topological Links (the names or all associated attributes), i.e., the links between subnetworks 9. List of all Subnetwork Connection associated with a subnetwork (just the names, or with all associated attributes). 10. The present value of the attributes associated with a particular Subnetwork, Managed Element, Topological Link, PTP, CTP, or Subnetwork Connection 11. The name of the containing subnetwork for a specified Managed Element 12. List of all Subnetwork Connections at a specified rate (just the names, or with all associated attributes). Example rates are VT1.5/TU11 and VT2/TU12. 13. List of Subnetwork Connections that use a specified TP (just the names, or with all associated attributes). The TP can be a CTP or a PTP. 14. The route(s) of a specified subnetwork connection 15. List of all the PTPs at a specified layer or set of layers (e.g., Electrical STS1/STM0, Optical OC3/STM1) and associated with a particular Managed Element. The request can be for just the names of the PTP, or all the associated attributes. 16. List of the all the PTPs at the edge of a subnetwork (just the names, or with all associated attributes). The request can be scoped to get only the PTPs at a specified layer or set of layers. 17. List of all TPs that contain a specified CTP (just the names, or with all associated attributes). 18. List of TPs associated with a specified TP (just the names, or with all associated attributes). 19. List of all cross-connections for a specified Managed Element. The request can be filtered to get only the cross-connections at a specified layer rate or set of layer rates. 20. List of Equipment and Equipment Holder objects contained in a specified Managed Element, or Equipment Holder, for all levels of the containment hierarchy (just the names, or with all associated attributes). 21. List of all Equipment and Equipment Holder objects directly contained in a specified Equipment Holder. <ul style="list-style-type: none"> • This method differs from the previous one in that it only looks at the next level of the containment hierarchy.
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>22. List of all PTPs supported by an Equipment (just the names, or with all associated attributes).</p> <ul style="list-style-type: none"> • The PTPs that are returned are those that share their physical layer with the primary equipment (i.e. that represent a port on the equipment or are connected by a fibre, wire, etc.). • When there is equipment protection, this operation reports PTPs for the primary equipment only. That is, when called on a protecting equipment (as opposed to the primary equipment), this operation returns an empty list, irrespective of the current switch status. <p>23. List of all Equipment objects which implement a PTP. (just the names, or with all associated attributes).</p> <ul style="list-style-type: none"> • The equipments that are returned are those which support the physical layer of the PTP (i.e. have the port on them or are connected by a fibre, wire, etc.). • For a particular PTP the Tx port and Rx port may be on different cards and in this case both should be returned. Equipment that are used by the PTPs, but that do not support them directly (such as a shared DEMUX card) are not reported. <p>24. List of all protection groups. Refer to the Use Case UC_TMF518_RTM_0004 from TMF518_RTM</p> <p>25. List of names of CTPs that can carry protected traffic (protected TPs).</p> <p>26. List of names of CTPs that can carry preemptible extra traffic (preemptible TPs).</p> <p>27. List of names of CTPs that are configured to carry Non-preemptible Unprotected extra Traffic (NUT TPs).</p> <p>28. List of all ASAPs of the target OS.</p> <p>29. List of all Matrix Flow Domains contained in a Managed Element (the names or all associated attributes).</p> <p>30. List of all Matrix Flow Domains within a Flow Domain.</p> <p>31. The Flow Domain that a Matrix Flow Domain is associated to.</p> <p>32. List of all CPTPs assigned to a Matrix Flow Domain.</p> <p>33. List of all CPTPs which are potentially able to be assigned to a Matrix Flow Domain.</p> <p>34. The Matrix Flow Domain to which a CPTP has been assigned.</p> <p>35. List of all Equipments supporting a Matrix Flow Domain.</p> <p>36. List of all Matrix Flow Domains supported by an Equipment.</p> <p>37. List of all Matrix Flow Domain names associated to a Transmission Descriptor.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<p>38. List of all Flow Domains managed by the Target OS (the names or all associated attributes).</p> <p>39. List of all Flow Domains having a specific User Label.</p> <p>40. List of all Edge CPTPs associated to a Flow Domain.</p> <p>41. List of all Matrix Flow Domains supporting a Flow Domain (the names or all associated attributes).</p> <p>42. List of all Bandwidth Profiles managed by the Target OS (the names or all associated attributes).</p> <p>43. List of all Flow Domain Fragments contained in a Flow Domain (the names or all associated attributes).</p> <p>44. The Flow Domain Fragment connecting a specific Flow Point (CTP).</p> <p>45. List of all Flow Domain Fragments connecting Flow Points of a specific CTP.</p> <p>46. List of all Flow Domain Fragments having a specific User Label.</p> <p>47. List of all TP names associated to a Traffic Conditioning Profile.</p> <p>48. List of all fdEdge FPs (the names or all associated attributes) used by a specified FDFr.</p> <p>49. List of all fdInternal FPs (the names or all associated attributes) used by a specified FDFr.</p> <p>50. List of all FPs (the names or all associated attributes) used by a specified FDFr.</p> <p>51. The capabilities, which includes connection management mode.</p>
Actor(s)	The requesting OS
Pre-Conditions	<p>1. The requesting OS and target OS have executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document.</p> <p>2. The requesting OS determines that it needs to query the target OS concerning a particular network inventory item.</p>
Begins When	The requesting OS sends an inventory query to the target OS.
Description	<p>1. The requesting OS sends a network inventory query to the target OS.</p> <p>2. The target OS provides the requested network inventory information to the requesting OS.</p>
Ends When	<p>In case of success:</p> <p>The requesting OS receives the requested network inventory information.</p>

	<p>In case of failure:</p> <p>The requesting OS receives a negative response from the target OS or the request times out.</p>
Post-Conditions	<p>In case of success:</p> <p>The requesting OS has received the requested network inventory information from the target OS.</p> <p>In case of failure:</p> <p>The target OS has not received the requested network inventory information.</p>
Exceptions	<ol style="list-style-type: none"> 1. Communication failure between the requesting OS and the target OS. 2. The requesting OS may include incorrect or unknown information in its queries to the target OS. See UC_TMF518_MRI_0002 3. Query unknown or not supported 4. Unable to comply.
Traceability	<p>R_TMF518_MRI_II_0013, R_TMF518_MRI_II_0017, R_TMF518_MRI_II_0019, R_TMF518_MRI_II_0020, R_TMF518_MRI_II_0023, R_TMF518_MRI_II_0024, R_TMF518_MRI_II_0027, R_TMF518_MRI_II_0030, R_TMF518_MRI_II_0032, R_TMF518_MRI_II_0034, R_TMF518_MRI_II_0036, R_TMF518_MRI_II_0042, R_TMF518_MRI_II_0055, R_TMF518_MRI_II_0074, R_TMF518_MRI_II_0081, R_TMF518_MRI_II_0085, R_TMF518_MRI_II_0087, R_TMF518_MRI_II_0109, R_TMF518_MRI_II_0111, R_TMF518_MRI_II_0112, R_TMF518_MRI_II_0116, R_TMF518_MRI_II_0117, R_TMF518_MRI_II_0127, R_TMF518_MRI_II_0131, R_TMF518_MRI_II_0153</p> <p>This use case is a generalization of Use Case 7.4.4 from TMF 513 v3.1</p>

4.5 One OS notifies another OS of inventory change

Use Case Id	UC_TMF518_MRI_0005
Use Case Name	One OS notifies another OS of inventory change
Summary	The notifying OS detects a change in the monitored network and notifies another OS.

Actor(s)	The notifying OS
Pre-Conditions	The notified OS has executed Use Case UC_TMF518_MRI_0001 “The requesting OS registers to receive network inventory updates from the target OS”
Begins When	The notifying OS detects a change in the monitored network.
Description	<ol style="list-style-type: none"> 1. The notifying OS detects a change in the monitored network and generates a notification to inform the notified OS. 2. The notified OS receives the notification from the Notification Service.
Ends When	<p>In case of success:</p> <p>The notified OS receives the notification.</p> <p>In case of failure:</p> <p>The notified OS does not receive the notification.</p>
Post-Conditions	<p>In case of success:</p> <p>The database of the notified OS remains aligned with the database of the notifying OS.</p> <p>In case of failure:</p> <p>The database of the notified OS remains is misaligned with the database of the notifying OS.</p>
Exceptions	
Traceability	This use case is a generalization of Use Case 7.4.5 from TMF 513 v3.1.

4.6 Inventory OS Provides Names for installed network entities

Use Case Id	UC_TMF518_MRI_0006
Use Case Name	Inventory OS Provides Names for installed network entities
Summary	<p>This use case involves the discovery and naming of a newly installed network. The discovery OS and the naming OS roles are played by different OSs (i.e., the Discovery OS and Inventory OS, respectively), and the Discovery OS can provide only a discovered name, while the Inventory OS has the role of Naming OS.</p> <p>Assumptions:</p> <ol style="list-style-type: none"> 1. The Discovery OS only discovers and announces new network entities, but does not name them (i.e., only a discovered name is supplied by the Discovery OS, and the Inventory OS will provide the MTOSI / MTNM name for the object). The Discovery OS is a simple adapter in this case.

	<ol style="list-style-type: none"> Subnet 1 and 3 are managed by different Target OSs (both using MTOSI / MTNM). Subnets 4, 5, 6, and 7 have no Target OS, and the Discovery OS needs to process TL1 messages directly from this MEs in this case. The Inventory OS will use a single naming scheme based on the MTOSI / MTNM naming rules. The underlying network is using two naming schemes, i.e., TMF 814 and TL1. The Discovery OS discovers MEs, EHs, EQs, PTPs, and subnetworks. The network topology is entered into the Inventory OS by means of some tool external to MTOSI / MTNM (i.e. GUI, file, and so on). The Inventory OS is provided with names to be used for the various network entities. The Inventory OS does not send OCNs for network resources before they are installed (i.e. for planned network resources).
Actor(s)	Inventory, Discovery, Fault Management and Activation OSs
Pre-Conditions	<ol style="list-style-type: none"> There is no network in place. All OSs involved in the use case have executed Use Case 0001 OS (Re) Starts as defined in the TMF518 FMW BA document.
Begins When	The network is installed and the Discovery OS starts to detect the installed network
Description	<ol style="list-style-type: none"> The Discovery OS announces the various network entities on the CCV, using the discovered name of the object. [In general (not just for this use case), the discovered name of an object is assumed to be unique on the CCV.] The announcement is made via an Object Discovery (ODscN) notification. Only the Inventory OS has subscribed to the ODscN notifications from the Discovery OS. The Inventory OS attempts to match each ODscN notification with a planned object in its inventory. Once a match has been achieved, the Inventory OS will send an OCN to the notification service, describing the installed resource. The new object will also be supplied with a name. The Act OS and Fault OS will receive and store some or all of the information from the OCN. An OCN will be sent for each newly installed network entity. If an announced entity does not match or conflicts with an entry in the Inventory OS's inventory, the Inventory OS shall not send an OCN to the CCV. The consequent behavior of the Inventory OS on mismatch between the real network and the planned resources is not in the scope of MTOSI / MTNM interface (i.e. exceptions could be raised at the GUI of the Inventory OS). As an alternate to step 3, the Activation OS and Fault Management OS (or perhaps just one of the two) may not process OCNs, but rather do an inventory retrieval of installed (not planned) resources once per day, or on some other time schedule.
Ends When	All interested OS have been informed about the newly installed

	network.
Post-Conditions	The Inventory, Discovery, FaultManagement and Activation OS all have an accurate view of the installed network.
Exceptions	None
Traceability	517 Use Case 7

4.7 Discovery OS Provides Names for installed network entities

Use Case Id	UC_TMF518_MRI_0007
Use Case Name	Discovery OS Provides Names for installed network entities
Summary	<p>This use case involves the discovery and naming of a newly installed network. The discovery OS and the naming OS roles are played by the same OS (i.e., the Discovery OS), i.e., the Discovery OS can provide both the name and the discovered name. The Inventory OS is a separate OS.</p> <p>Assumptions:</p> <ol style="list-style-type: none"> 1. The Discovery OS announces discovered network entities on the CCV and stores sufficient (containment and relationship) information about the discovered entities in order to provide a proper MTOSI / MTNM name. The Discovery OS is an intelligent adapter in this case. The Discovery OS may also provided a discovered name for each network entity but this is not necessary. This use case is based on the architecture and related description concerning Figure 4-1. 2. Subnet 1 and 3 are managed by different Target OSs (both using TMF 814). Subnets 4, 5, 6, and 7 have no Target OS, and the Discovery OS needs to process TL1 messages directly from this MEs in this case. 3. The Discovery OS will use a single naming scheme based on the MTOSI / MTNM naming rules. The underlying network is using two naming schemes, i.e., TMF 814 and TL1. The Discovery OS discovers MEs, EHs, EQs, PTPs, and subnetworks. 4. The network topology is entered into the Inventory OS by means of some tool external to the MTOSI / MTNM (i.e. GUI, file, and so on). The Inventory OS is provided with names to be used for the various network entities.
Actor(s)	Inventory, Discovery, Fault Management and Activation OSs
Pre-Conditions	<ol style="list-style-type: none"> 1. There is no network in place. 2. All OSs involved in the use case have executed Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document.

Begins When	The network is installed and the Discovery OS starts to detect the installed network
Description	<ol style="list-style-type: none"> 1. The Discovery OS announces and names the various network entities on the CCV. The discovered name may also be included (the Inventory OS may need this for matching). In this case, the Discovery OS uses an OCN to make the announcement, and the resource is described as installed. 2. If an announced entity does not match or conflicts with an entry in the Inventory OS's database, the consequent behavior of the Inventory OS concerning the mismatch between the real network and the planned resources is not within the scope of MTOSI / MTNM (e.g., exceptions could be raised at the GUI of the Inventory OS). 3. The FaultManagement and Activation OSs also use the OCNs from the Discovery OS. 4. As alternative, if only the Inventory OS has subscribed to the Object Creation Notifications (OCNs) from the Discovery OS, the FaultManagement and Activation OS could retrieve the updates from the Inventory OS on some regular schedule as follows: 5. For each OCN, the Inventory OS attempts to find a match with an planned object in its inventory. If a match can be found, the object's resourceState is changed from Planned to Installed. 6. The FaultManagement and Activation OSs will (on some regular schedule) get inventory updates from the Inventory OS. This could be done, for example, via a bulk retrieval operation. For example, the FM and Act OSs may only want to retrieve resources that are in the Installed resourceState.
Ends When	All interested OS have been informed about the newly installed network.
Post-Conditions	The Inventory, Discovery, FaultManagement and Activation OS all have an accurate view of the installed network.
Exceptions	None
Traceability	517 Use Case 8

4.8 Inventory OS Provides Names for planned network entities

Use Case Id	UC_TMF518_MRI_0008
Use Case Name	Inventory OS Provides Names for planned network entities
Summary	This use case involves the discovery and naming of a newly installed network. The discovery OS and the naming OS roles are not played by the same OS. The Inventory OS names and sends OCNs for

	<p>planned resources. The Discovery OS can store information to match the planned resources with the discovered resources.</p> <p>Assumptions:</p> <ol style="list-style-type: none"> 1. The Inventory OS announces planned network entities on the CCV and the Discovery OS stores information about the announced entities in order to provide a match with the discovered entities. The Discovery OS is an intelligent adapter in this case. 2. Subnet 1 and 3 are managed by different Target OSs (both using TMF 814). Subnets 4, 5, 6, and 7 have no Target OS, and the Discovery OS needs to process TL1 messages directly from this MEs in this case. 3. The Inventory OS will use a single naming scheme based on the MTOSI / MTNM naming rules. The underlying network is using two naming schemes, i.e., TMF 814 and TL1. The Discovery OS discovers MEs, EHs, EQs, PTPs, and subnetworks. 4. The Inventory OS has its own resource database, and sends OCNs for network resources before they are installed (i.e., for planned network resources). 5. The network topology is entered into the Inventory OS by means of some tool external to the MTOSI / MTNM (i.e. GUI, file, and so on). The Inventory OS is provided with names to be used for the various network entities. 6. The Discovery OS stores sufficient information about the planned entities announced on the CCV and the discovered entities in order to do matching.
Actor(s)	Inventory, Discovery, Fault Management and Activation OSs
Pre-Conditions	<ol style="list-style-type: none"> 1. There is no network in place. 2. All OSs involved in the use case have executed Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document. 3. The Inventory OS has announced the planned resources to CCV and the Discovery OS has subscribed to the OCNs from the Inventory OS.
Begins When	The network is installed and the Discovery OS starts to detect the installed network.
Description	<ol style="list-style-type: none"> 1. The Discovery OS matches the names of the discovered resources with the names previously provided by the Inventory OS. In this case, the Discovery OS uses an AVC to make the announcement, and the resource is described as installed. 2. The Activation and FaultManagement OSs will (on some regular schedule) get inventory updates for installed resources from the Inventory OS. The could be done, for example, via a bulk retrieval operation.
Ends When	All interested OS have been informed about the newly installed network.
Post-Conditions	The Inventory, Discovery, Fault Management and Activation OSs all

	have an accurate view of the installed network.
Exceptions	None
Traceability	517 Use Case 9

4.9 Naming for installed network entities - No separate Discovery OS

Use Case Id	UC_TMF518_MRI_0009
Use Case Name	Naming for installed network entities - No separate Discovery OS
Summary	<p>This use case involves the discovery and naming of a newly installed network. The discovery OS and the naming OS roles are played the same OS. In fact, the Discovery and Inventory OS are the same OS.</p> <p>Assumptions:</p> <ol style="list-style-type: none"> 1. There is not a separate Discovery OS. The Inventory OS handles both discovery from the network, and naming. This use case is based on the architecture and related description concerning Figure 4-1. 2. Subnet 1 and 3 are managed by different Target OSs (both using TMF 814). Subnets 4, 5, 6, and 7 have no Target OS, and the Inventory OS needs to process TL1 messages directly from this MEs in this case. 3. The Inventory OS will use a single naming scheme based on the MTOSI / MTNM naming rules. The underlying network is using two naming schemes, i.e., TMF 814 and TL1. The Inventory OS discovers MEs, EHs, EQs, PTPs, and subnetworks. 4. The network topology is entered into the Inventory OS by means of some tool external to the MTOSI / MTNM (i.e. GUI, file, and so on). The Inventory OS is provided with names to be used for the various network entities. 5. The Inventory OS does not send OCNs for network resources before they are installed (i.e. for planned network resources).
Actor(s)	Inventory, Fault Management and Activation OSs
Pre-Conditions	<ol style="list-style-type: none"> 1. There is no network in place. 2. All OSs involved in the use case have executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document
Begins When	The network is installed and the Inventory OS detects the installed network.
Description	<ol style="list-style-type: none"> 1. The Inventory OS matches each discovered network entity with its

	<p>own inventory. If a match is found, an OCN is published to the notification service and is received by the FM and Act OSs. If a no match or a mismatch is found, no OCN is published to the notification service.</p> <p>2. As an alternate to the last step, the Activation OS and FaultManagement OS (or perhaps just one of the two) may not process OCNs, but rather do an inventory retrieval of installed (not planned) resources once per day, or on some other time schedule.</p>
Ends When	All interested OS have been informed about the newly installed network.
Post-Conditions	The Inventory, Fault Management and Activation OSs all have an accurate view of the installed network.
Exceptions	None
Traceability	517 Use Case 10

4.10 Distributed stewarding

Use Case Id	UC_TMF518_MRI_0010
Use Case Name	Distributed stewarding
Summary	<p>The use case illustrates the situation where there are several OSs on a CCV that share stewarding responsibilities.</p> <p>Assumptions</p> <ol style="list-style-type: none"> 1. Assume the Inventory OS stewards MEs, all contained entities in the MEs and the links that interconnect the MEs. The Activation OS stewards subnetworks and SNCs.
Actor(s)	Inventory, Discovery, Fault Management and Activation OSs
Pre-Conditions	The Inventory OS has an existing inventory of installed network resources.
Begins When	The Inventory OS announces installed network resources on the CCV.
Description	<ol style="list-style-type: none"> 1. The Activation and FaultManagement OSs retrieve the inventory offered by the Inventory OS. Note that retrieval would need to be an ME basis, since the Inventory OS has no information about subnetworks at this point. 2. The Activation OS requests all the known subnetworks and SNCs from the Discovery OS. 3. The Activation OS provides MTOSI / MTNM names for the discovered subnetworks and SNCs, if the Discovery OS only provided discovered names. 4. The Activation OS publishes OCNs for the subnetworks and SNCs

	<p>which are consumed by the Inv and FaultManagement OSs.</p> <p>Alternately, the Inventory and FaultManagement OSs could retrieve this information on a scheduled basis. Once the subnetworks are known to the Activation OS, the other OSs on the CCV can retrieve information from the Activation OS on a per subnetwork basis. For example, the <code>getAllManagedElements</code> operation in the <code>multiLayerSubnetwork</code> interface could be used by the FaultManagement OS to retrieve all the MEs in a given subnetwork from the Activation OS.</p> <p>5. As new SNCs and subnetworks are discovered by the Activation OS from the Discovery OS, they will be named and then reported to the Inventory and FaultManagement OSs.</p> <p>6. The Activation OS will also inform interested parties on the CCV concerning SNCs that it creates in the network. The Discovery OS acts as an adapter for the Act OS when SNC creation or deletion requests need to be sent to the network.</p>
Ends When	All interested OS have been informed about the newly installed network.
Post-Conditions	The Inventory, Discovery, Fault Management and Activation OSs all have an accurate view of the installed network.
Exceptions	None
Traceability	517 Use Case 11

4.11 Bulk Inventory Retrieval Use Cases

These Use Cases describe three scenarios for retrieving network inventory from an existing repository (*target* OS) by another operation system (*requesting* OS). See the context diagram in **Figure 4-2**. In all cases the *requesting* OS uses a single operation to retrieve all or part of the *target* OS inventory. The scope of the retrieval is determined by simple filters. These scenarios differ only in the type of communication style used to complete the retrieval.

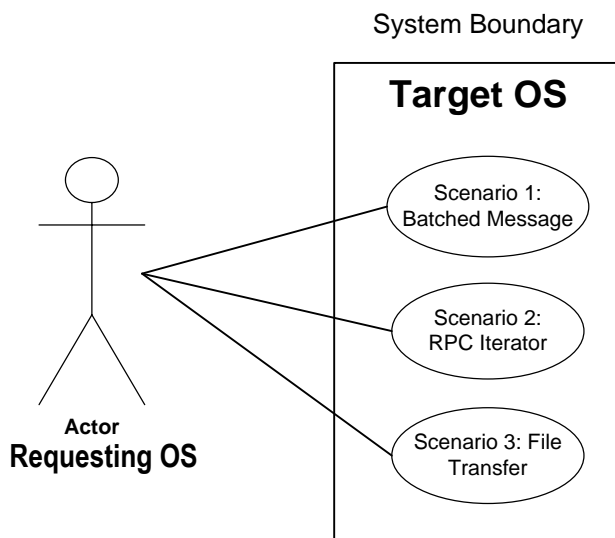


Figure 4-2. Context Diagram for Bulk Inventory Use Cases

Definition of Communication Styles:

The Interface allows for three styles for bulk inventory retrieval:

1. Batched responses (message communication style) – in this approach the server (*target OS*) sends the requested information back to the client (*requesting OS*) in a series of responses (batches). For example, using asynchronous communication over JMS.

Now, the batch size may provide a threshold indication in terms of the maximum number of Inventory units to be found in an XML document (response or file). Clearly, the inventory unit must provide a straight relation with the size of data generated by a `getInventory` query. Also, it must be relatively simple to manage. So, the ideal solution can only be that one unit corresponds to one MTOSI / MTNM object instance of the Inventory layout. The size of all MTOSI / MTNM objects is not significantly different. And, counting their output is also very straightforward. However, it is acceptable to say that a generated XML document may have more objects than specified by the batch size in the request. This is to allow complete data output for the top-level (atomic) object being exported.

For example, the batch size may be set to 500 units (MTOSI / MTNM objects). And, the threshold is crossed while generating data for a ME. All objects of the ME must be exported in the same XML document. Thus, the total number of objects in the document will most of the time be greater than the requested maximum size. But, it will never be by a significant difference.

2. Batched retrieval (RPC communication style) – in this approach the client (*requesting OS*) retrieves the requested information from the server (*target OS*) via a series of iterative requests, For example, using `http/s` with synchronous communication.
3. File transfer – in this approach the requested inventory is delivered in the form of a single file. The approach allows the client (*requesting OS*) to designate the location of the file. For example, using FTP.

4.11.1 Bulk Inventory Retrieval– Coarse grained approach, batched response with message communication style (asynchronous)

Use Case Id	UC_TMF518_MRI_0011
Use Case Name	Bulk Inventory Retrieval– Coarse grained approach, batched response with message communication style (asynchronous)
Summary	The <i>requesting</i> OS requests all or a specified portion of the inventory data from the <i>target</i> OS. The result is returned in one or more batches of XML over the CCV using the MSG message communication style (see the definition of <i>communication styles</i>). The <i>target</i> OS returns all objects satisfying the filter constraints of the <i>requesting</i> OS that have a modification date equal to or later than the Diff Date and Time .
Actor(s)	Requesting OS
Pre-Conditions	All OSs involved in the use case have successfully executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document.
Begins When	The <i>requesting</i> OS sends a request to the <i>target</i> OS to retrieve inventory data.
Description	<ol style="list-style-type: none"> 1. The <i>requesting</i> OS sends an inventory retrieval request to the <i>target</i> OS over the CCV. The request message has the following parameters: <ol style="list-style-type: none"> 3. Filter – attribute used to restrict the set of objects returned to the requesting OS (the allowable filters are defined in R_TMF518_MRI_II_0004) 4. Diff Date and Time – all objects with a modification date equal to or greater than this value and satisfying the filter constraints are returned (in batches) to the requesting OS. Generalized Time format is to be used. Note: This parameter is dependent on access to file modification dates for the inventory objects. It may be made optional by setting the value to NULL. In this case, inventory is retrieved subject only to filter constraints. 5. ReplyToAddress – address (e.g., URI or JMS topic name) where the target OS sends the responses. 6. RequestedBatchSize – the maximum number of units to be returned in each batch from the target OS. If the value is 0, it means that the entire inventory is returned in one batch. A unit is one of the following: <ol style="list-style-type: none"> 1. one management domain object 2. one transmission descriptor object 3. one OS object 4. one topologicalLink object 5. one managed element object and all contained objects passing the filter

	<p>6. one subnetwork object and all contained objects passing the filter.</p> <p>7. CorrelationID – this parameter is provided by the requesting OS and then returned in each response by the target OS to allow for correlation by the requesting OS.</p> <p>2. The <i>target</i> OS sends an acknowledgment message to the ReplyToAddress over the CCV, signifying successful validation of the input parameters. This allows the <i>requesting</i> OS to manage the returned batches more efficiently.</p> <p>3. The <i>target</i> OS prepares the specified inventory data and returns the information in a series of batches. Each batch has (at most) the number of items defined in the RequestedBatchSize parameter. Each batch has a CorrelationID, a SequenceNumber and a Boolean marker endOfReply which is set to true in the last batch.</p>
Ends When	<p>In case of success:</p> <p>The <i>requesting</i> OS receives all response messages up to the sequence number of the last batch with endOfReply set to true.</p> <p>In case of failure:</p> <p>The <i>requesting</i> OS receives an exception.</p>
Post-Conditions	<p>In case of success:</p> <p>The <i>requesting</i> OS has the requested inventory information from the <i>target</i> OS.</p> <p>In case of failure:</p> <p>The <i>requesting</i> OS does not have the entire requested inventory information from the <i>target</i> OS.</p>
Exceptions	<p>1. Not implemented: The <i>target</i> OS does not support this service.</p> <p>2. Invalid input: An input parameter is invalid (e.g., invalid filter syntax)</p> <p>3. Processing failure: The requested operation could not be performed.</p> <p>4. Communications failure: The target OS has lost communications with an entity that is needed to help fulfill the request.</p>
Traceability	517 Use Case 13

4.11.2 Bulk Inventory Retrieval– Coarse grained approach, RPC (synchronous) communication style

Use Case Id	UC_TMF518_MRI_0012
Use Case Name	Bulk Inventory Retrieval– Coarse grained approach, RPC (synchronous) communication style
Summary	The <i>requesting</i> OS requests that the <i>target</i> OS prepare a specified portion of its inventory for subsequent retrieval. The results will be

	retrieved over the CCV using the RPC communication style (see definition of <i>communication styles</i>). The target OS provides for iterated retrieval of all objects satisfying the provided filter constraints that have a modification date equal to or later than the Diff Date and Time .
Actor(s)	Requesting OS
Pre-Conditions	All OSs involved in the use case have executed the Use Case 0001 OS (Re) Starts as defined in the TMF518_FMW BA document.
Begins When	The requesting OS sends a request to the target OS
Description	<ol style="list-style-type: none"> The requesting OS sends an inventory retrieval request to the target OS over the CCV. The request message shall provide the following parameters: <ol style="list-style-type: none"> Filter – attribute used to restrict the set of objects returned to the requesting OS (the allowable filters are defined in R_TMF518_MRI_II_0004) Diff Date and Time – all objects with a modification date equal to or greater than this value and satisfying the filter constraints are returned (in batches) to the requesting OS. Generalized Time format is to be used. Note: This parameter is dependent on access to file modification dates for the inventory objects. It may be made optional by setting the value to NULL. In this case, inventory is retrieved subject only to filter constraints. RequestedBatchSize – the maximum number of units to be returned in each requested batch from the target OS. A unit is one of the following: <ol style="list-style-type: none"> one management domain object one transmission descriptor object one OS object one topologicalLink object one managed element object and all contained objects passing the filter one subnetwork object and all contained objects passing the filter. The <i>target</i> OS sends the first batch of inventory back to the requesting OS (the number of items is determined by the RequestedBatchSize). In the same response, the <i>target</i> OS also provides the IteratorAddress. This address (e.g., URI) is used by the <i>requesting</i> OS to control the batch retrieval. The <i>requesting</i> OS proceeds to retrieve the next batch of inventory from the iterator. The RequestedBatchSize can be different from the original value provided for RequestedBatchSize and can, in fact, vary with each request made by the requesting OS. Once the <i>requesting</i> OS has the address of the iterator, it can (at

	<p>any time) request the size of the iterator. Note that this is the total number of items in the iterator (minus the initial batch).</p> <p>5. The <i>target</i> OS should provide an endOfReply indication in the last batch retrieved by the <i>requesting</i> OS.</p> <p>6. Once the last batch is retrieved, the <i>requesting</i> OS could request that the <i>target</i> OS delete the iterator. The target OS could also automatically delete the iterator after the last batch.</p>
Ends When	<p>In case of success: The requesting OS has retrieved the last batch.</p> <p>In case of failure: The requesting OS receives an exception.</p>
Post-Conditions	<p>In case of success: The requesting OS has all the requested inventory information from the target OS.</p> <p>In case of failure: The requesting OS does not have the entire requested inventory information from the target OS.</p>
Exceptions	<ol style="list-style-type: none"> 1. Not implemented: The target OS does not support this service. 2. Invalid input: Any input parameter is invalid (e.g., invalid filter syntax) 3. Processing failure: The requested operation could not be performed. 4. Communications failure: The requesting OS can not reach the iterator.
Traceability	517 Use Case 14

4.11.3 Bulk Inventory Retrieval – Coarse grained approach, file transfer

Use Case Id	UC_TMF518_MRI_0013
Use Case Name	Bulk Inventory Retrieval – Coarse grained approach, file transfer
Summary	<p>The <i>requesting</i> OS requests all or a specified portion of the inventory of the <i>target</i> OS. The result is returned in an XML file (via ftp, http, or some other file transfer approach). The file destination is specified by the <i>requesting</i> OS. The <i>target</i> OS returns all objects satisfying the filter constraints of the <i>requesting</i> OS and that have a modification date equal to or later than the Diff Date and Time.</p> <p>The <i>requesting</i> OS provides through the request all the necessary instructions on processing the response into a desired inventory file format in terms of maximum file size (in terms of the number of object instances), file compression, file packing, and the rootname of the generated file(s). The <i>requesting</i> OS specifies in the request all the necessary instructions to allow transfer of the file(s) to a remote destination.</p>
Actor(s)	Requesting OS

Pre-Conditions	All OSs involved in the use case have executed the OS (Re)starts use case.
Begins When	The <i>requesting</i> OS sends a request to the <i>target</i> OS over the CCV to return inventory data as file(s).
Description	<ol style="list-style-type: none"> 1. The <i>requesting</i> OS sends the request command to the <i>target</i> OS over the CCV. The request message has the following parameters: <ol style="list-style-type: none"> 1. Filter – this attribute is used to restrict the set of objects returned to the requesting OS. (the allowable filters are defined in R_TMF518_MRI_II_0004) 2. Diff Date and Time – all objects with a modification date equal to or greater than this value and satisfying the filter constraints are returned (in batches) to the requesting OS. Generalized Time format is to be used. Note: This parameter is dependent on access to file modification dates for the inventory objects. It may be made optional by setting the value to NULL. In this case, inventory is retrieved subject only to filter constraints. 3. File Location – URI provided by the requesting OS indicating the rootname of the file(s) to be produced and location of where to place the retrieved XML inventory file(s). See RFC 3986 and RFC 1738 for details on URI syntax. 4. Batch Size - The maximum number of Inventory top level objects to be included in an inventory XML file (batch). These objects are atomic from an XML structure standpoint, which means that their complete structure including containment cannot be divided in two different batches. These objects are: ME, TL, OS, TMD, and MLSN. Therefore, depending on the inventory output size several files may be generated. Default behavior (when request parameter is omitted) is to generate the inventory in a single file (unbounded number of objects). 5. Compression Type – The type of compression to apply to the generated inventory file(s); the following options are available: NO_COMPRESSION, GZIP. Default behavior (when request parameter is omitted) is NO_COMPRESSION. Implementation of this file processing instruction by the Target OS is optional, and any incompatible request shall be handled with the appropriate exception. Vendor extension of this attribute shall be permitted. 6. Packing Type – The type of packing to apply to all the inventory file(s) generated from the same request. the following options are available: NO_PACKING, ZIP, TAR. Default behavior (when request parameter is omitted) is NO_PACKING. Implementation of this file processing instruction by the Target OS is optional, and any incompatible request shall be handled with the appropriate exception. Vendor extension of this attribute shall be permitted. 2. The <i>target</i> OS performs the upload of the file(s) to specified location. 3. The <i>target</i> OS uses the NT_FILE_TRANSFER_STATUS notification (defined in the MTNM OMGServicesUsage document) to indicate when the file transfer is complete or when a failure has occurred. Intermediate progress events may be indicated by setting the “transferStatus” to

			ON	rootname.xml	No packing (unnecessary)
			OFF	rootname.xml.gz	
		ON		rootname.[tar zip]	
			ON	rootname.xml.gz	
			OFF	rootname_<SN>.xml	The sequential number <SN> for a set of batch files
				rootname.[tar zip]	
		OFF	ON	rootname_<SN>.xml	The sequential number <SN> for a set of batch files
			OFF	rootname_<SN>.xml.gz	The sequential number <SN> for a set of batch files
	Multiple	ON	ON	rootname.[tar zip] rootname_<SN>.xml.gz	The sequential number <SN> for a set of batch files
Ends When	<p>In case of success: The requesting OS receives the response message signifying the file transfer is completed.</p> <p>In case of failure: The requesting OS receives an exception or gets an indication that the file transfer has failed.</p>				
Post-Conditions	<p>In case of success: The requesting OS has the requested inventory information from the target OS.</p> <p>In case of failure: The requesting OS does not have the requested inventory information from the target OS.</p>				
Exceptions	<ol style="list-style-type: none"> 1. Not implemented: The target OS does not support this service. 2. Invalid input: Any input parameter is invalid (e.g., invalid filter syntax) 3. Processing failure: The requested operation could not be performed. 4. Communications failure: The requesting OS can not reach the file location. 5. Unsupported Compression Format 6. Unsupported Packing Format 				
Traceability	517 Use Case 15				

4.12 Effects of the suppression of OCNs during creation of an object

Use Case Id	UC_TMF518_MRI_0014
Use Case Name	Effects of the suppression of OCNs during creation of an object
Summary	An OS (call it the target OS) works without sending OCNs for all object instances of a determinate object class (e.g., EQH, EQ and PTP) that are contained under any newly created objects of a given parent type (e.g., ME)
Actor(s)	Subscribing OS
Pre-Conditions	All OSs involved in the use case have been authenticated and have access to a CCV. In the Implementation Statement the OS stated the suppression of OCN during creation of MEs.
Begins When	An OS (call it the target OS) announce creation of a ME.
Description	<ol style="list-style-type: none"> 1. An instance of parent type ME is created (under the supervision of the target OS). 2. The target OS will send the corresponding ME OCN, but it will not send any OCNs related to all instance matching any of the types listed in the ChildTypeList (e.g., EQH, EQ and PTP). 3. The target OS sends a State Change Notification for the emsInSynchState attribute of the ME to True, when the current target OS data has been synchronized with the current NE data. 4. The subscriber OS (it could be any other OS on the CCV) retrieves all the contained objects in the contained ME.
Ends When	The subscriber OS knows all resources contained in the new ME.
Post-Conditions	<p>In case of success: The target OS has a reduced amount of “secondary” OCN to send when the “primary” is sent.</p> <p>In case of failure: This feature can not be dynamically turned on/off.</p>
Exceptions	
Traceability	517 Use Case 16

4.13 OS Requests that Target OS update its inventory

Use Case Id	UC_TMF518_MRI_0015
Use Case Name	OS Requests that Target OS update its inventory
Summary	<p>The requesting OS asks that another OS (referred to as the target OS) make specific updates to its inventory. There is no expectation that the updates go beyond the target OS. This use case covers both the atomic (all or nothing) and best effort scenarios.</p> <p>This use case can be used in several ways.</p> <ul style="list-style-type: none"> • A planning system (playing the role of the Requesting OS) could propose an inventory update to an inventory OS (playing the role of the Target OS) before the network has been updated. • On the other hand, a discovery OS (playing the role of the Requesting OS) may detect that the network has been updated and then request that an inventory OS (playing the role of the Target OS) update its inventory. Alternately, the discovery OS could just send a bulk inventory notification to the inventory OS. In this case, the discovery OS would not be sure if the inventory OS has aligned with the information in the bulk inventory notification. So, the former approach would make more sense if there was a desire to closely synchronize the inventories of the discovery and inventory OS. In any event, this is a policy decision on the part of the service provider.
Actor(s)	Requesting OS
Pre-Conditions	The requesting OS and target OS have been authenticated and have access to a CCV.
Begins When	The requesting OS sends an inventory update request to the target OS.
Description	<ol style="list-style-type: none"> 1. The requesting OS prepares an inventory update request and sends the request to the target OS. The request includes the inventory layout (which contains the inventory updates) and an indication of whether the operation is to be atomic or best effort. 2. The target OS analyzes the request and updates its inventory accordingly. This may involve object creations, modifications and deletions. 3. The target OS replies to the requesting OS when it has finished updating its inventory. In the atomic case, either all the updates succeed or everything is rolled back to the starting state. In the best effort case, some of the updates may succeed and others may fail. In this latter case, the target OS will provide an exception list for the requested updates that have failed.
Ends When	In case of success:

	<p>The requesting OS receives confirmation from the target OS that its inventory has been updated as requested (<i>atomic</i>) or partially updated (<i>best effort</i>).</p> <p>In case of failure:</p> <p>The requesting OS receives an overall exception from the target OS.</p>
Post-Conditions	<p>In case of success:</p> <p>The inventory of the target OS has been updated as requested (for <i>atomic</i>, and <i>best effort</i> when all updates succeed) or partially updated (<i>best effort</i> when not all the updates succeed).</p> <p>In case of failure:</p> <p>The inventory of the target OS remains unchanged as a result of the request from the requesting OS.</p>
Exceptions	See Table 3-2.
Traceability	R_TMF518_MRI_II_0005 , R_TMF518_MRI_III_0146

4.14 OS Informs Other OSs About Multiple Inventory Events

Use Case Id	UC_TMF518_MRI_0016
Use Case Name	OS Informs Other OSs About Multiple Inventory Events
Summary	<p>Server OS sends Multi-event Inventory (MEI) notification to the Notification Service (as Publisher) which forwards it to all other OS (Subscribers) on the CCV.</p> <p>[In this use case, the term “Subscribed OSs” is used to refer to the OSs on the CCV that have subscribed to receive network inventory notifications.]</p>
Actor(s)	Publishing OS
Pre-Conditions	<ol style="list-style-type: none"> 1. The publishing OS has gained access to the notification service and is able to send notifications. 2. Some of the other OSs on the CCV have registered to receive MEI notifications concerning changes to the network inventory.
Begins When	The publishing OS detects a multiple changes in the underlying network that warrant the use of a MEI notification. [The conditions under which an MEI is used versus a collection of single event notifications is an implementation issue.
Description	<ol style="list-style-type: none"> 1. The publishing OS detects multiple changes in the underlying network and populates an inventory layout structure (this structure is explained in SD2-12, Inventory Layout) concerning the inventory changes.

	<p>2. Based on an implementation agreement, the publishing OS decides on how many MEI notifications are to be used transport the inventory layout structure. If multiple MEI notifications are to be used, each notification will contain:</p> <p>7. CorrelationID – used by the receiving OSs to associate the notifications</p> <p>8. SequenceNumber – used by the receiving OSs to reassemble the inventory layout structure</p> <p>9. endOfSequence – this is set to “true” in the last notification of the sequence.</p> <p>3. The interested OSs on the CCV receive the MEI notifications and reassemble the inventory layout structure (in the case of multiple MEI notifications are used).</p>
Ends When	<p>In case of success: The subscribed OSs on the CCV have received all the MEI notifications associated with a particular inventory layout structure.</p> <p>In case of failure: The notification service has given up trying to deliver at least some of the MEI notification, before all of the subscribed OSs on the CCV have received them.</p>
Post-Conditions	<p>In case of success: The database of the subscribed OSs remains aligned with the publishing OS’s database.</p> <p>In case of failure: The databases of (at least) some of the subscribed OSs are misaligned with the publishing OS’s database.</p>
Exceptions	None
Traceability	None

5 Traceability Matrices

Table 5-1. Use Case - Requirements Traceability Matrix

Use Case Id	Use Case Name	Requirements
UC_TMF518_MRI_0001	The requesting OS registers to receive network inventory updates from the target OS	This use case is a generalization of Use Case 7.4.1 from TMF 513 v3.1
UC_TMF518_MRI_0002	The requesting OS resynchronizes its database with the target OS	Refer to UC_TMF518_MRI_0003 and UC_TMF518_MRI_0004 This use case is a generalization of Use Case 7.4.2 from TMF 513 v3.1
UC_TMF518_MRI_0003	The requesting OS discovers the target OS network inventory	R_TMF518_MRI_II_0015 , R_TMF518_MRI_II_0023 , R_TMF518_MRI_II_0024 , R_TMF518_MRI_II_0027 , R_TMF518_MRI_II_0030 , R_TMF518_MRI_II_0052 , R_TMF518_MRI_II_0055 , R_TMF518_MRI_II_0068 , R_TMF518_MRI_II_0070 , R_TMF518_MRI_II_0085 , R_TMF518_MRI_II_0094 , R_TMF518_MRI_II_0117 , R_TMF518_MRI_II_0123 This use case is a generalization of Use Case 7.4.3 from TMF 513 v3.1.

UC TMF518 MRI 0004	The requesting OS queries the target OS concerning inventory	R TMF518 MRI II 0013 , R TMF518 MRI II 0017 , R TMF518 MRI II 0019 , R TMF518 MRI II 0020 , R TMF518 MRI II 0023 , R TMF518 MRI II 0024 , R TMF518 MRI II 0027 , R TMF518 MRI II 0030 , R TMF518 MRI II 0032 , R TMF518 MRI II 0034 , R TMF518 MRI II 0036 , R TMF518 MRI II 0042 , R TMF518 MRI II 0055 , R TMF518 MRI II 0074 , R TMF518 MRI II 0081 , R TMF518 MRI II 0085 , R TMF518 MRI II 0087 , R TMF518 MRI II 0109 , R TMF518 MRI II 0111 , R TMF518 MRI II 0112 , R TMF518 MRI II 0116 , R TMF518 MRI II 0117 , R TMF518 MRI II 0127 , R TMF518 MRI II 0131 , R TMF518 MRI II 0153 This is use case is a generalization of Use Case 7.4.4 from TMF 513 v3.1
UC TMF518 MRI 0005	One OS notifies another OS of inventory change	This is use case is a generalization of Use Case 7.4.5 from TMF 513 v3.1.
UC TMF518 MRI 0006	Inventory OS Provides Names for installed network entities	517 Use Case 7
UC TMF518 MRI 0007	Discovery OS Provides Names for installed network entities	517 Use Case 8
UC TMF518 MRI 0008	Inventory OS Provides Names for planned network entities	517 Use Case 9
UC TMF518 MRI 0009	Naming for installed network entities - No separate Discovery OS	517 Use Case 10
UC TMF518 MRI 0010	Distributed stewarding	517 Use Case 11
UC TMF518 MRI 0011	Bulk Inventory Retrieval– Coarse grained approach, batched response with message communication style (asynchronous)	517 Use Case 13

UC_TMF518_MRI_0012	Bulk Inventory Retrieval– Coarse grained approach, RPC (synchronous) communication style	517 Use Case 14
UC_TMF518_MRI_0013	Bulk Inventory Retrieval – Coarse grained approach, file transfer	517 Use Case 15
UC_TMF518_MRI_0014	Effects of the suppression of OCNs during creation of an object	517 Use Case 16
UC_TMF518_MRI_0015	OS Requests that Target OS update its inventory	R_TMF518_MRI_II_0005 , R_TMF518_MRI_III_0146
UC_TMF518_MRI_0016	OS Informs Other OSs About Multiple Inventory Events	None

Table 5-2. Requirements - Use Cases Traceability Matrix

Requirement Id	Use Case Name	Use Case Id
R_TMF518_MRI_II_0001		
R_TMF518_MRI_II_0002		
R_TMF518_MRI_II_0003		
R_TMF518_MRI_II_0004		
R_TMF518_MRI_II_0005	OS Requests that Target OS update its inventory	UC_TMF518_MRI_0015
R_TMF518_MRI_II_0006		
R_TMF518_MRI_II_0007		
R_TMF518_MRI_II_0008		
R_TMF518_MRI_II_0010		
R_TMF518_MRI_II_0012		
R_TMF518_MRI_II_0013	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0014		
R_TMF518_MRI_II_0015	The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0003
R_TMF518_MRI_II_0017	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0019	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0020	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0021		
R_TMF518_MRI_II_0023	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0024	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0026		

R_TMF518_MRI_II_0027	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0029		
R_TMF518_MRI_II_0030	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0032	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0034	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0036	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0037		
R_TMF518_MRI_II_0038		
R_TMF518_MRI_II_0040		
R_TMF518_MRI_II_0042	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0043		
R_TMF518_MRI_II_0044		
R_TMF518_MRI_II_0045		
R_TMF518_MRI_II_0046		
R_TMF518_MRI_II_0047		
R_TMF518_MRI_II_0048		
R_TMF518_MRI_II_0049		
R_TMF518_MRI_II_0050		
R_TMF518_MRI_II_0051		
R_TMF518_MRI_II_0052	The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0003
R_TMF518_MRI_II_0054		
R_TMF518_MRI_II_0055	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0057		
R_TMF518_MRI_II_0059		

R_TMF518_MRI_II_0061		
R_TMF518_MRI_II_0064		
R_TMF518_MRI_II_0066		
R_TMF518_MRI_II_0068	The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0003
R_TMF518_MRI_II_0070	The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0003
R_TMF518_MRI_II_0072		
R_TMF518_MRI_II_0074	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0076		
R_TMF518_MRI_II_0078		
R_TMF518_MRI_II_0080		
R_TMF518_MRI_II_0081	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0083		
R_TMF518_MRI_II_0085	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0087	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0088		
R_TMF518_MRI_II_0090		
R_TMF518_MRI_II_0092		
R_TMF518_MRI_II_0094	The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0003
R_TMF518_MRI_II_0096		
R_TMF518_MRI_II_0098		
R_TMF518_MRI_II_0099		
R_TMF518_MRI_II_0100		
R_TMF518_MRI_II_0101		
R_TMF518_MRI_II_0102		
R_TMF518_MRI_II_0104		
R_TMF518_MRI_II_0105		
R_TMF518_MRI_II_0106		
R_TMF518_MRI_II_0107		

R_TMF518_MRI_II_0108		
R_TMF518_MRI_II_0109	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0110		
R_TMF518_MRI_II_0111	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0112	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0113		
R_TMF518_MRI_II_0114		
R_TMF518_MRI_II_0115		
R_TMF518_MRI_II_0116	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0117	The requesting OS queries the target OS concerning inventory The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0004 UC_TMF518_MRI_0003
R_TMF518_MRI_II_0118		
R_TMF518_MRI_II_0119		
R_TMF518_MRI_II_0120		
R_TMF518_MRI_II_0121		
R_TMF518_MRI_II_0122		
R_TMF518_MRI_II_0123	The requesting OS discovers the target OS network inventory	UC_TMF518_MRI_0003
R_TMF518_MRI_II_0124		
R_TMF518_MRI_II_0125		
R_TMF518_MRI_II_0126		
R_TMF518_MRI_II_0127	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0128		
R_TMF518_MRI_II_0129		
R_TMF518_MRI_II_0130		
R_TMF518_MRI_II_0131	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0132		
R_TMF518_MRI_II_0150		
R_TMF518_MRI_II_0151		
R_TMF518_MRI_II_0152		

R_TMF518_MRI_II_0153	The requesting OS queries the target OS concerning inventory	UC_TMF518_MRI_0004
R_TMF518_MRI_II_0155		
R_TMF518_MRI_II_0156		
R_TMF518_MRI_III_0146	OS Requests that Target OS update its inventory	UC_TMF518_MRI_0015
R_TMF518_MRI_IV_0147		
R_TMF518_MRI_IV_0148		
R_TMF518_MRI_IV_0149		
R_TMF518_MRI_V_0154		

6 Future Directions

7 References

7.1 References

- [1] [TMF518_FMW](#), Framework DDP BA
- [2] TMF513, Multi-Technology Network Management (MTNM) Business Agreement, Version 3.1, March 2007
- [3] TMF517, Multi-Technology Operations System Interface (MTOSI) Business Agreement, Version 1.2, December 2006
- [4] [TMF518_NRB](#), Network Resource Basic DDP BA
- [5] [TMF518_SB](#), Service Basic DDP BA
- [6] [SD0-1](#), Dictionary

7.2 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 [SD0-3 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 the Management of Resource Inventory. 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. 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	September 2007	This is the first version of the document and as such, there are no changes to report.
1.1	May 2008	Different corrections and updates following review comments.
1.2	September 2011	Updated sections 1.1 and 2. Replaced mTOP by MTNM / MTOSI everywhere in the document

8.4 Company Contact Details

Document Contact
Michel Besson Amdocs OSS division <i>Email: michel.besson@amdocs.com</i> <i>Phone: +44 7717 692178</i>

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