TMF518\_RTD Resource Test and Diagnostics

Version 1.0



September 2014

©TeleManagement Forum 2014

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](http://www.tmforum.org/)

Table of Contents

[Notice 2](#_Toc402264543)

[Table of Contents 3](#_Toc402264544)

[List of Requirements 6](#_Toc402264545)

[List of Use Cases 7](#_Toc402264546)

[Executive Summary 8](#_Toc402264547)

[1 Introduction 9](#_Toc402264548)

[1.1 Overview 9](#_Toc402264549)

[1.2 Interface Scope 10](#_Toc402264550)

[1.3 Document Structure 11](#_Toc402264551)

[1.4 Terminology Used In This Document 11](#_Toc402264552)

[1.4.1 TMF Terminology 11](#_Toc402264553)

[1.4.2 MEF Terminology 11](#_Toc402264554)

[1.4.3 IEEE Terminology 13](#_Toc402264555)

[1.4.4 ITU Terminology 15](#_Toc402264556)

[2 Business Problem Description, Project Scope 16](#_Toc402264557)

[2.1 Problem Statement 16](#_Toc402264558)

[2.2 Supported Business scenarios 16](#_Toc402264559)

[2.2.1 Resource Provisioning 16](#_Toc402264560)

[2.2.2 Resource Trouble Management 16](#_Toc402264561)

[2.2.3 Resource Performance Management 16](#_Toc402264562)

[2.3 Benefits 17](#_Toc402264563)

[3 Relationship to other TMF Groups 18](#_Toc402264564)

[3.1 Business Process Framework 18](#_Toc402264565)

[3.2 Information Framework (SID) 18](#_Toc402264566)

[3.3 Application Framework (TAM) 18](#_Toc402264567)

[3.4 Relationship to other TMF Groups 18](#_Toc402264568)

[4 Requirements 19](#_Toc402264569)

[4.1 Business Requirements 19](#_Toc402264570)

[4.2 Category I: Static and Structural Requirements 21](#_Toc402264571)

[4.2.1 Framework DDP – Layered Parameters 21](#_Toc402264572)

[4.2.2 Framework DDP – Test Parameters 21](#_Toc402264573)

[4.2.3 Framework DDP – Test Statistics 21](#_Toc402264574)

[4.2.4 Resource Test 22](#_Toc402264575)

[4.2.5 Resource Test Specification 26](#_Toc402264576)

[4.2.6 Resource Test Parameter 27](#_Toc402264577)

[4.2.7 Resource Test Parameter Value 29](#_Toc402264578)

[4.2.8 Resource Test Measure Definition 30](#_Toc402264579)

[4.2.9 Resource Test Measure Value 30](#_Toc402264580)

[4.2.10 Threshold Rule Set 31](#_Toc402264581)

[4.2.11 Threshold Rule 31](#_Toc402264582)

[4.3 Category II: Normal Sequences, Dynamic Requirements 32](#_Toc402264583)

[4.3.1 Resource Specification Management 32](#_Toc402264584)

[4.3.1.1 Creation of Resource Test Specification 32](#_Toc402264585)

[4.3.1.2 Modification of Resource Test Specification 33](#_Toc402264586)

[4.3.1.3 Retrieval of Resource Test Specification 34](#_Toc402264587)

[4.3.1.4 Deletion of Resource Test Specification 34](#_Toc402264588)

[4.3.2 On-Demand Resource Test Control 35](#_Toc402264589)

[4.3.2.1 Start Test 35](#_Toc402264590)

[4.3.2.2 Suspend Test 37](#_Toc402264591)

[4.3.2.3 Resume Test 37](#_Toc402264592)

[4.3.2.4 Stop Test 37](#_Toc402264593)

[4.3.2.5 Get Test Results 38](#_Toc402264594)

[4.3.2.6 Delete Test 38](#_Toc402264595)

[4.3.3 Proactive Resource Test Control 39](#_Toc402264596)

[4.3.3.1 Create and Activate Flow Domain Fragment with Test 39](#_Toc402264597)

[4.3.3.2 Create and Activate Subnetwork Connection with Test 39](#_Toc402264598)

[4.4 Category III: Abnormal or Exception Conditions, Dynamic Requirements 41](#_Toc402264599)

[4.5 Category IV: Expectations and Non-Functional Requirements 41](#_Toc402264600)

[4.6 Category V: System Administration Requirements 41](#_Toc402264601)

[5 Use Cases 42](#_Toc402264602)

[5.1 Resource Specification Management 42](#_Toc402264603)

[5.1.1 Resource Test Specification Use Cases 42](#_Toc402264604)

[5.1.1.1 The Requesting OS Creates a Resource Test Specification 42](#_Toc402264605)

[5.1.1.2 The Requesting OS Modifies a Resource Test Specification 43](#_Toc402264606)

[5.1.1.3 The Requesting OS Deletes a Resource Test Specification 44](#_Toc402264607)

[5.2 Resource Test Management 46](#_Toc402264608)

[5.2.1 Resource Test Pro-Active Use Cases 46](#_Toc402264609)

[5.2.1.1 The Requesting OS Starts a Controlled, Proactive Test 46](#_Toc402264610)

[5.2.1.2 The Requesting OS Stops a Controlled, Proactive Test 49](#_Toc402264611)

[5.2.2 Resource Test On-Demand Use Cases 51](#_Toc402264612)

[5.2.2.1 The Requesting OS Starts an Uncontrolled, On-Demand Test 51](#_Toc402264613)

[5.2.2.2 The Requesting OS Starts a Controlled, On-Demand Test 53](#_Toc402264614)

[5.2.2.3 The Requesting OS Suspends a Test 55](#_Toc402264615)

[5.2.2.4 The Requesting OS Resumes a Test 56](#_Toc402264616)

[5.2.2.5 The Requesting OS Stops a Controlled, On-Demand Test 57](#_Toc402264617)

[5.2.2.6 The Requesting OS Retrieves Results from a Controlled, On-Demand or Pro-Active Test 59](#_Toc402264618)

[6 Traceability Matrices 62](#_Toc402264619)

[6.1 Use Case Matrix 62](#_Toc402264620)

[7 Future Directions 63](#_Toc402264621)

[8 References 64](#_Toc402264622)

[8.1 References 64](#_Toc402264623)

[8.2 IPR Releases and Patent Disclosure 64](#_Toc402264624)

[9 Administrative Appendix 65](#_Toc402264625)

[9.1 About this document 65](#_Toc402264626)

[9.2 Use and Extension of a TM Forum Business Agreement 65](#_Toc402264627)

[9.3 Document History 65](#_Toc402264628)

[9.4 Company Contact Details 66](#_Toc402264629)

[9.5 Acknowledgments 66](#_Toc402264630)

[9.6 About TM Forum 66](#_Toc402264631)

List of Requirements

[R\_TMF518\_RTD\_BR\_0027](#R_TMF518_RTD_BR_0027) 19

[R\_TMF518\_RTD\_BR\_0028](#R_TMF518_RTD_BR_0028) 19

[R\_TMF518\_RTD\_BR\_0029](#R_TMF518_RTD_BR_0029) 19

[R\_TMF518\_RTD\_BR\_0031](#R_TMF518_RTD_BR_0031) 19

[R\_TMF518\_RTD\_BR\_0032](#R_TMF518_RTD_BR_0032) 20

[R\_TMF518\_RTD\_I\_0005](#R_TMF518_RTD_I_0005) 22

[R\_TMF518\_RTD\_I\_0006](#R_TMF518_RTD_I_0006) 22

[R\_TMF518\_RTD\_I\_0007](#R_TMF518_RTD_I_0007) 26

[R\_TMF518\_RTD\_I\_0008](#R_TMF518_RTD_I_0008) 26

[R\_TMF518\_RTD\_I\_0009](#R_TMF518_RTD_I_0009) 27

[R\_TMF518\_RTD\_I\_0010](#R_TMF518_RTD_I_0010) 29

[R\_TMF518\_RTD\_I\_0011](#R_TMF518_RTD_I_0011) 30

[R\_TMF518\_RTD\_I\_0012](#R_TMF518_RTD_I_0012) 30

[R\_TMF518\_RTD\_I\_0013](#R_TMF518_RTD_I_0013) 31

[R\_TMF518\_RTD\_I\_0014](#R_TMF518_RTD_I_0014) 31

[R\_TMF518\_RTD\_II\_0001](#R_TMF518_RTD_II_0001) 39

[R\_TMF518\_RTD\_II\_0002](#R_TMF518_RTD_II_0002) 39

[R\_TMF518\_RTD\_II\_0003](#R_TMF518_RTD_II_0003) 39

[R\_TMF518\_RTD\_II\_0004](#R_TMF518_RTD_II_0004) 40

[R\_TMF518\_RTD\_II\_0015](#R_TMF518_RTD_II_0015) 32

[R\_TMF518\_RTD\_II\_0016](#R_TMF518_RTD_II_0016) 32

[R\_TMF518\_RTD\_II\_0017](#R_TMF518_RTD_II_0017) 33

[R\_TMF518\_RTD\_II\_0018](#R_TMF518_RTD_II_0018) 34

[R\_TMF518\_RTD\_II\_0019](#R_TMF518_RTD_II_0019) 34

[R\_TMF518\_RTD\_II\_0020](#R_TMF518_RTD_II_0020) 35

[R\_TMF518\_RTD\_II\_0021](#R_TMF518_RTD_II_0021) 35

[R\_TMF518\_RTD\_II\_0022](#R_TMF518_RTD_II_0022) 37

[R\_TMF518\_RTD\_II\_0023](#R_TMF518_RTD_II_0023) 37

[R\_TMF518\_RTD\_II\_0024](#R_TMF518_RTD_II_0024) 37

[R\_TMF518\_RTD\_II\_0025](#R_TMF518_RTD_II_0025) 38

[R\_TMF518\_RTD\_II\_0026](#R_TMF518_RTD_II_0026) 38

List of Use Cases

[UC\_TMF518\_RTD\_0001](#UC_TMF518_RTD_0001) 42

[UC\_TMF518\_RTD\_0002](#UC_TMF518_RTD_0002) 43

[UC\_TMF518\_RTD\_0003](#UC_TMF518_RTD_0003) 44

[UC\_TMF518\_RTD\_0004](#UC_TMF518_RTD_0004) 46

[UC\_TMF518\_RTD\_0005](#UC_TMF518_RTD_0005) 55

[UC\_TMF518\_RTD\_0006](#UC_TMF518_RTD_0006) 56

[UC\_TMF518\_RTD\_0007](#UC_TMF518_RTD_0007) 49

[UC\_TMF518\_RTD\_0008](#UC_TMF518_RTD_0008) 51

[UC\_TMF518\_RTD\_0009](#UC_TMF518_RTD_0009) 53

[UC\_TMF518\_RTD\_0010](#UC_TMF518_RTD_0010) 57

[UC\_TMF518\_RTD\_0011](#UC_TMF518_RTD_0011) 59

Executive Summary

This document is a separate Business Agreement called Resource Test and Diagnostics (RTD) that is part of a separate Resource Test Management Document Delivery Package (DDP). It focuses exclusively on test and diagnostic capabilities beyond those already specified as part of Resource Trouble Management, more specifically with the maintenance operation commands.. The document is intentionally separated out as it is a new feature to MTOSI 4.0 and will also be used as part of the TIP resource trouble management interfaces.

The business agreement covers requirements and use cases for the following aspects of RTD: resource test configuration and execution, and associated retrieval of test statistics.

See Section 1 for additional introductory material related to this document.

# Introduction

## Overview

This feature supports resource test and diagnostic capabilities over a TIP interface in support of multiple network technologies. The tests are divided into two categories:

* **Pro-active/Controlled** Tests that are setup at service configuration and activation time, and associated resource provisioning, and are generally run continuously throughout the lifetime of the service, though they may be one-time benchmark tests, or other such tests. These tests may be suspended, resumed, or stopped. The test results may also be retrieved.
* **On-demand/Controlled** Tests that are requested at a specific point in time, but that may be scheduled to actually execute at a future point in time. These tests may also be suspended, resumed, or stopped. The test results may also be retrieved.
* **On-demand/Uncontrolled** Tests that are requested to execute immediately, and for which the request is blocking until the test completes and returns results in real time. These tests are *uncontrolled* in that they may not be suspended, resumed, or stopped – due to the fact that the test executes while the client OS waits. As the results are also returned as part of the response from the target OS, they are not retrievable at a later point in time.

The 802.1ag/ITU-T Y.1731 and G.8113/Y.1372 tests that are run require the use of Maintenance Entity Groups (MEGs), Maintenance Endpoints (MEPs), and Maintenance Intermediate Points (MIPs). These are not created as new objects over the interface, but rather as layered transmission parameters on associated FDFr/SNC and TP objects and are set via operations on these objects that allow for the manipulation of the transmission parameters. Particularly in the case of pro-active tests these parameters may be set at the moment the FDFr or SNC is created and activated.

A *resource test* object is introduced to essentially hold the configuration and results of any given instantiated controlled test. A *resource test specification* is also introduced, and contains the test configuration parameters and their associated values.

In the case of pro-active tests, a *resource test* object would be created automatically by the target OS at the moment the FDFr/SNC is created and activated over the interface. The FDFr/SNC would be associated to a resource test specification in order to pass in the associated test parameters, and/or they can be specified over the interface.

For on-demand controlled tests, the test object is created at the moment the test is started. The client OS may also refer to a resource test specification and/or pass the required test parameters and their values directly over the interface.

This feature provides the new objects and operations in support of the tests defined above. It also provides standard configuration parameters and test result data for each of the tests supported over the interface. Vendor extensions are available for the support of tests that are not defined as part of this feature.

## Interface Scope

This particular project addresses the following process areas for resource test and diagnostics, with specific test characteristics provided for GPON, Ethernet, and MPLS/TP. Note: the areas extend beyond the Resource and Trouble Management process areas as defined in the eTOM, but are in scope for feature consolidation purposes, rather than spreading the feature across the different DDPs.

* Resource Inventory

This involves the retrieval of test and diagnostic related entities such as IEEE 802.1ag MEG/MEP/MIPs. The IEEE 802.1ag entities are modeled not by explicit new classes and operations, rather by new additional attributes/parameters of existing classes/operations such as the FDFr/SNC and associated termination points.The resource specifications will be retrievable via the general “getInventory” operation. However, the test object instances will be retrievable as part of the “getTestResults” whereby the client OS may specify through the use of a filter to either retrieve the entire object, or just the results.

* Resource Provisioning

This task involves the provisioning of OAM entities such as MEGs, MEPs, MIPS, and the management of test configurations. As the MEG, MEP, and MIP are all defined as layered transmission parameters on the FDFr/SNC and TP objects respectively, they can be setup and modified on related provisioning operations on these objects. The configuration and execution of *pro-active tests* will be done through the configuration and activation of the FDFr or SNC that is associated to the service. They are usually performed as service turn-up time to verify proper functioning of the service.

* Resource Trouble Management

This task involves the execution of on-demand tests for diagnosing a particular problem. They are normally executed in response to a problem that has been either reported by the end customer, or as a result of network operations. Examples of these types of tests include Loopback, Linktrace, and other diagnostic tests.

* Resource Performance Management

This area involves the execution of *on demand* performance related tests in order to measure specific network key performance indicators (KPI). These tests provide the KPI’s without actually applying any algorithm to them (e.g. the measurement of frame delay, versus the calculation of frame delay variation). Such tests are usually run when service degradation is both detected and reported by the end customer, or as part of network operations. Examples of these tests include one way and two way frame loss, frame delay, throughput and other such diagnostic tests.

The feature focuses exclusively on the resource management domain, and does not extend into service management areas such as Service Configuration & Activation, Service Problem Management, and Service Quality Management. These areas are considered to be out-of-scope.

## Document Structure

The following sections are contained in this document:

* [Section 1](#_Introduction) is the document introduction
* [Section 2](#_Business_Problem_Description) defines the business problem description and supported scenarios
* [Section 3](#_Project_Scope) covers the project scope and the relationship to other TMF activities
* [Section 4](#_Business_Requirements) includes all the requirements by category
* [Section 5](#_Use_Cases) defines the use cases
* [Section 6](#_Traceability_Matrices) traceability matrices between use cases and requirements
* [Section 7](#_Future_Directions) outlines future directions
* [Section 8](#_References) lists references and any Intellectual Property Right (IPR) claims
* [Section 9](#_Administrative_Appendix) contacts, acknowledgements and other administrative items

## Terminology Used In This Document

### TMF Terminology

Refer to the [SD0-1](https://collab.tmforum.org/integration/viewvc/viewvc.cgi/mtosi_2.1/main/DDPs/Framework/SD/SD0-1_Dictionary.doc) supporting document for general terminology used by the TMF. Explicitly for this feature the following terms as defined in the table below are used.

### MEF Terminology

| Term | Definition | Reference |
| --- | --- | --- |
| Availability Performance | A measure of the percentage of time that a service is useable. | MEF 10.1 |
| Bandwidth Profile | A characterization of Service Frame arrival times and lengths at a reference point and a specification of the disposition of each Service Frame based on its level of compliance with the Bandwidth Profile | MEF 10.1 |
| Committed Information Rate (CIR) | CIR is a Bandwidth Profile parameter. It defines the average rate in bits/s of Service Frames up to which the network deliv-ers Service Frames and meets the performance objectives defined by the CoS Service Attribute | MEF 10.1 |
| Class of Service | A set of Service Frames that have a commitment from the Service Provider to receive a particular level of performance. | MEF 10.1 |
| Class of Service Identifier | An indicator for a particular CoS instance. Information derivable from a) the EVC to which the Service Frame is mapped, b) the combination of the EVC to which the Service Frame is mapped and a set of one or more than one CE-VLAN CoS values, c) the combination of the EVC to which the Service Frame is mapped and a set of one or more than one DSCP values, or d) the combination of the EVC to which the Service Frame is mapped and a set of one or more than one tunneled Layer 2 Control Protocols. | MEF 10.1 |
| Committed Burst Size (CBS) | CBS is a Bandwidth Profile parameter. It limits the maximum number of bytes available for a burst of Service Frames sent at the UNI speed to remain CIR-conformant. | MEF 10.1 |
| Committed Information Rate (CIR) | CIR is a Bandwidth Profile parameter. It defines the average rate in bits/s of Service Frames up to which the network delivers Service Frames and is committed to meeting the performance objectives defined by the CoS Service Attribute. | MEF 10.1 |
| Excess Burst Size (EBS) | EBS is a Bandwidth Profile parameter. It limits the maximum number of bytes available for a burst of Service Frames sent at the UNI speed to remain EIR-conformant. | MEF 10.1 |
| Excess Information Rate (EIR) | EIR is a Bandwidth Profile parameter. It defines the average rate in bits/s of Service Frames up to which the network may deliver Service Frames but without any performance objectives. | MEF 10.1 |
| E-LAN Service | An Ethernet service type that is based on a Multipoint-to-Multipoint EVC. | MEF 10.1 |
| E-Line Service | An Ethernet service type that is based on a Point-to-Point EVC. | MEF 10.1 |
| E-Tree Service | An Ethernet service type that is based on a Rooted-Multipoint EVC. | MEF 10.1 |
| External Network-to-Network Interface (ENNI) | A reference point representing the boundary between two Operator Metro Ethernet Networks that are operated as separate administrative domains | MEF 26 |
| Ethernet Virtual Connection (EVC) | An association of two or more UNIs that limits the exchange of Service Frames to UNIs in the Ethernet Virtual Connection. | MEF 10.1 |
| Frame Delay(FD) | The Frame Delay for a Service Frame is defined as the time elapsed from reception at the ingress UNI of the first bit of the ingress Service Frame until the transmission of the last bit of the Service Frame at the egress UNI. | MEF 10.1 |
| Frame Delay Variation (FDV) | Frame Delay Variation is the difference between the one-way delays of a pair of selected Service Frames. This definition is borrowed from RFC3393 where IP packet delay variation is defined. | MEF 10.1 |
| Frame Loss Ratio Performance | Frame Loss Ratio is a measure of the number of lost frames between the ingress UNI and the egress UNI. Frame Loss Ratio is expressed as a percentage. | MEF 10.1 |
| Maintenance Entity (ME) | ME represents an OAM entity that requires management. | MEF 17 |
| Maintenance Entity Group (MEG) | A ME Group (MEG) consists of the MEs that belong to the same service inside a common OAM domain. For a Point-to-Point EVC, a MEG contains a single ME. For a Multipoint-to-Multipoint EVC of n UNIs, a MEG contains n\*(n-1)/2 MEs. | MEF 17 |
| MEG Level | MEG Level is used to distinguish between OAM frames belonging to different nested MEs. MEs belonging to the same MEG share a common MEG Level. Eight MEG Levels have been identified for the purposes of Ethernet OAM [Y.17ethoam] [802.1ag]. | MEF 17 |
| MEG CoS | The MEG CoS represents one or more priorities associated with the OAM frames for a given ME. All MEs inside a MEG share a common CoS profile. Since an EVC can be associated with service frames with different CoS levels, an EVC ME can be associated with OAM frames with multiple priorities. | MEF 17 |
| Maintenance Endpoint (MEP) | A MEG End Point (MEP) is a provisioned OAM reference point which can initiate and terminate proactive OAM frames. A Point-to-Point EVC has two MEPs, one on each end point of the ME. A Multipoint-to-Multipoint EVC of n UNIs has n MEPs, one on each end point. | MEF 17 |
| Maintenance Intermediate Point (MIP) | MEG Intermediate Point (MIP) is a provisioned OAM reference point which is capable to react to diagnostic OAM frames initiated by MEPs. The number of MIPs in a Point-to-Point EVC or Multipoint-to-Multipoint EVC is dependent on the specific deployments. | MEF 17 |
| User Network Interface (UNI) | The physical demarcation point between the responsibility of the Service Provider and the responsibility of the Subscriber. | MEF 10.1 |

### IEEE Terminology

| Term | Definition | Reference |
| --- | --- | --- |
| Connectivity Fault Management (CFM) | CFM comprises capabilities for detecting, verifying, and isolating connectivity failures in Virtual Bridged Local Area Networks. | 802.1ag |
| Connectivity Check Message (CCM) | A multicast CFM PDU transmitted periodically by a MEP in order  to ensure continuity over the MA to which the transmitting MEP belongs. No reply is sent by any MP in  response to receiving a CCM. | 802.1ag |
| Domain Service Access Point (DoSAP) | A member of a set of SAPs at which a Maintenance Domain is  capable of offering connectivity to systems outside the Maintenance Domain. | 802.1ag |
| Down MEP | A MEP residing in a Bridge that receives CFM PDUs from, and transmits them towards, the direction of the LAN. | 802.1ag |
| Linktrace Message (LTM) | A CFM PDU initiated by a MEP to trace a path to a target MAC address,  forwarded from MIP to MIP, up to the point at which the LTM reaches its target, a MEP, or can no longer be forwarded. Each MP along the path to the target generates an LTR. | 802.1ag |
| Linktrace Reply (LTR) | A unicast CFM PDU sent by an MP to a MEP, in response to receiving an  LTM from that MEP. | 802.1ag |
| Loopback Message (LBM) | A unicast CFM PDU transmitted by a MEP, addressed to a specific MP,  in the expectation of receiving an LBR. | 802.1ag |
| Loopback Reply (LBR) | A unicast CFM PDU transmitted by an MP to a MEP, in response to an LBM  received from that MEP. | 802.1ag |
| Maintenance Association (MA) | A set of MEPs, each configured with the same MAID and MD Level, established to verify the integrity of a single service instance. An MA can also be thought of as a full mesh of Maintenance Entities among a set of MEPs so configured. | 802.1ag |
| Maintenance Domain (MD) | Maintenance Domains are management space on a network, typically owned and operated by a single entity. MDs are configured with *Names* and *Levels*, where the eight levels range from 0 to 7. A hierarchical relationship exists between domains based on levels. The larger the domain, the higher the level value. | 802.1ag |
| Maintenance Entity (ME) | A point-to-point relationship between two MEPs within a single MA. | 802.1ag |
| Maintenance Endpoint (MEP) | An actively managed CFM entity, associated with a specific DoSAP of a service instance, which can generate and receive CFM PDUs and track any responses.  It is an end point of a single MA and is an end point of a separate Maintenance Entity for each of the other MEPs in the same MA. | 802.1ag |
| Maintenance Domain Intermediate Point (MIP) | A CFM entity consisting of two MHFs. (see below) | 802.1ag |
| MIP Half Function (MHF) | A CFM entity, associated with a single Maintenance Domain, and thus  with a single MD Level and a set of VIDs, that can generate CFM PDUs, but only in response to received  CFM PDUs. | 802.1ag |
| Maintenance Domain Level (MD Level) | A small integer in a field in a CFM PDU that is used, along with the VID in the VLAN tag, to identify to which Maintenance Domain among those associated with the CFM frame’s VID, and thus towhich MA, a CFM PDU belongs. The MD Level determines the MPs that are a) interested in the contents ofa CFM PDU, and b) through which the frame carrying that CFM PDU is allowed to pass. | 802.1ag |
| Maintenance Association Identifier (MAID) | An identifier for a Maintenance Association, unique over the domain that CFM is to protect against the accidental concatenation of service instances. The MAIDhas two parts: the Maintenance Domain Name and the Short MA Name. | 802.1ag |
| Primary Vlan Identifier (Primary VID) | The VID, among a list of VIDs associated with a service instance, on which all CFM PDUs generated by MPs except for forwarded LTMs are to be transmitted. | 802.1ag |
| Service Access Point (SAP) | The point at which a service is offered. | 802.1ag |
| Up MEP | A MEP residing in a Bridge that transmits CFM PDUs towards, and receives them from, the direction of the Bridge Relay Entity. | 802.1ag |

### ITU Terminology

| Term | Definition | Reference |
| --- | --- | --- |
| Dual-Ended | A type of protocol messaging whereby an initiating MEP sends a request PDU to a receiving peer MEP, which does not send any response. In performance monitoring the receiving MEP performs measurement calculations. | Y.1731 |
| Far End | In single-ended or dual-ended messaging, the direction and information relating to a one-way measurement from an initiating MEP to a receiving or responding peer MEP. | Y.1731 |
| Initiating MEP | An initiating MEP initiates measurements by sending request PDUs and in the case of single-ended messaging by receiving response PDUs. | Y.1731 |
| Near End | In single-ended messaging, the direction and information relating to a one-way measurement from a responding peer MEP to an initiating MEP. | Y.1731 |
| One Way | A type of measurement of the performance of frames that is achieved in one direction from an initiating MEP to a peer MEP, or vice versa; i.e., a unidirectional measurement. | Y.1731 |
| Peer-MEP | The peer MEPs of a given MEP are all of the other MEPs in the same MEG that share an ME with the given MEP. | Y.1731 |
| Single-Ended | A type of protocol messaging whereby an initiating MEP sends a request PDU to a responding peer MEP, and the peer MEP responds by sending a response PDU which contains the original request data plus any additional data added by the responder. In performance monitoring, the initiating MEP performs performance measurement calculations. | Y.1731 |
| Two-Way | A type of protocol messaging whereby an initiating MEP sends a request PDU to a responding peer MEP, and the peer MEP responds by sending a response PDU which contains the original request data plus any additional data added by the responder. In performance monitoring, the initiating MEP performs performance measurement calculations. | Y.1731 |

# Business Problem Description, Project Scope

## Problem Statement

Assuring critical services is important to service providers in order maintain agreements for quality of service as defined in service level agreements (SLA). This involves the ability to check for service connectivity, verify key quality indicators such as frame loss, etc. and to perform diagnostic tests. The problem being addressed by this feature is the standard configuration across TMF interfaces of the test and diagnostic mechanisms required to do the following:

* Configure resource tests
* Manage the execution of resource tests
* Report resource test threshold crossings
* Retrieve resource test performance statistics

## Supported Business scenarios

### Resource Provisioning

These tests are setup and executed as part of testing the resources being provisioned as a result of an associated service configuration and activation. The tests ensure the viability of the supported service. They may be pro-active, in that they execute throughout the lifetime of the service and are continually detecting anomalies associated to the service; or, they may be on-demand tests, such as benchmark tests, that are run once to ensure the proper functioning of the service and that it meets service level criteria that are agreed upon between the provider and the end customer. At any time inventory requests may be made to determine which tests are operational and how they have been configured.

### Resource Trouble Management

These tests are setup and executed as part of diagnosing defects in resource connectivity or correct functioning of resources. They are exclusively on-demand tests that operate on resources that have already been provisioned and activated in support of a service, but are found to be faulty and causing problems for a service. The tests are normally triggered by a “trouble ticket” being reported by the end customer, but they may also be run by network operations in order to detect the trouble before the customer. At any time inventory requests may be made to determine which tests are operational and how they have been configured.

### Resource Performance Management

These tests are setup and executed as part of detecting resource performance problems that eventually result in service degradation. They are exclusively on-demand tests that operate on resources that have already been provisioned and activated in support of a service, but for which performance is found to be degrading, or potentially already degraded. The tests are normally triggered by a “trouble ticket” being reported by the end customer, but they may also be run by network operations to avoid performance related problems. At any time inventory requests may be made to determine which tests are operational and how they have been configured.

## Benefits

| **Key Element** | **Benefit** |
| --- | --- |
| Ability to consistently configure and execute tests across multiple resource management systems | * Reduction in OpEx * Faster time to revenue due to improved service turn up * Streamlined network tests across different management systems |
| Reduce complexity of managing tests across multiple technologies | * Streamlined network test processes across different network technologies * Normalized test execution |
| Manage end-to-end testing from a single point of control | * Faster detection of service level issues * More satisfied end customers implies higher revenue assurance |
| Standardized test control | * Improved processes * Lower OSS integration tax |
| Retrieve standardized performance statistics as a result of test execution across multiple technologies | * Improved service level performance * Quicker interpretation of results and application of any diagnostic formulas |

# Relationship to other TMF Groups

## Business Process Framework

The Resource Test and Diagnostics follows the eTOM process areas.

## Information Framework (SID)

A new Test ABE has been added to the SID, and a Resource Test ABE as a result of this work.

## Application Framework (TAM)

Not identified

## Relationship to other TMF Groups

There is a relationship to the TIP Test and Diagnostics project whereby service test and diagnostic requirements and use cases have been identified in a BA.

# Requirements

## Business Requirements

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_BR\_0027 | **Proactive / Controlled Tests**  The interface shall support the configuration and execution of proactive / controlled tests.  These are tests that are setup at service configuration and activation time, and associated resource provisioning, and are generally run continuously throughout the lifetime of the service, though they may be one-time benchmark tests, or other such tests. These tests may be suspended, resumed, or stopped. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_BR\_0028 | **On-demand / Uncontrolled Tests**  The interface shall support the configuration and execution of on-demand / uncontrolled tests.  These are tests that are requested to execute immediately, and for which the request is blocking until the test completes and returns results in real time. These tests are *uncontrolled* in that they may not be managed (suspend, resume, stop) – due to the fact that the test executes while the client OS waits. As the results are also returned as part of the response from the target OS, they are not retrievable at a later point in time. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_BR\_0030 | **On-demand / Controlled Tests**  The interface shall support the configuration and execution of on-demand / controlled tests.  These are tests that are requested to be executed immediately, or may be scheduled to actually execute at a future point in time. They are *non-blocking*. These tests may also be suspended, resumed, or stopped. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_BR\_0031 | **Retrieval of Test Configuration and Measures**  The interface shall support the retrieval of a test configuration and/or measures associated to execution of a specific proactive or on-demand, controlled test. A filter mechanism shall be supported to determine the information returned. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_BR\_0032 | **Supported Test Reporting Mechanisms**  The interface shall support the following 3 types of test reporting mechanisms for performance related results:   1. For pro-active measurements, collection of current and history data      1. For on-demand measurements, single instances of results      1. For on-demand measurements, single series measurements of results |
| Source |  |

## Category I: Static and Structural Requirements

### Framework DDP – Layered Parameters

Changes are introduced to the MTOSI layered parameters supporting document (SD1-16\_LayeredParameters.pdf in the Framework DDP) in support of this feature for GPON, ETHERNET, and MPLS/TP. They are noted under the category “Test & Diagnostics”.

These changes represent the following assumptions made about the proposed modeling for each of these technologies. MEPs and MIPs are not explicitly modeled over the MTOSI interface, but rather are defined as transmission parameters on TPs that are associated to a MEG/MA, which is modeled as an SNC/FDFr.

### Framework DDP – Test Parameters

A new supporting document has been added called “SDx-xx TestParameters.pdf” that contains *resource test parameters* and the possible values, known as *resource test parameter values*. These parameters define configuration input for given types of tests and constrain the values they may take when a test is configured.

They can be defined in a resource test specification that is then referenced by an entity under test to pre-define a set of fixed values to use as test characteristics; or, the values can be passed individually over the interface for the creation of a test.

### Framework DDP – Test Statistics

A new supporting document has been added called “SDx-xx TestStatistics.pdf” that contains *resource test measures* and associated threshold values. These parameters define the anticipated statistical results one can expect from the supported set of tests, and threshold values.

They can be defined in a resource test specification that is then referenced upon start of the test (to monitor threshold values), and then during the test result retrieval process to determine the set of results to be returned.

### Resource Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0005 | **Resource Test**  The ***Resource Test*** ***(RT)*** object shall represent the execution of a particular test against a given resource, and given set of resource test parameters, in accordance with a specified schedule, and a list of desired test measures. It shall be instantiated whenever a test is requested, either on-demand or pro-actively.  An ***RT*** is contained within an MD. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0006 | **Resource Test Attributes**  A *resource test*  object shall have, in addition to the attributes identified in R\_TMF518\_NRB\_I\_0001, the following attributes:   * **test mode**  identifies the mode in which the test is executed. The supported values include   + *Pro-active Controlled* Test was started proactively as part of turning up the service and provisioning the associated resources, and not explicitly via the “start test” operation   + *On-Demand Controlled* Test was started via a “start test” test and is non-blocking   Note: for uncontrolled, on-demand tests, no resource test object is instantiated, thus there is no corresponding test mode.   * **test type** identifies the type of test. If a reference is provided to a resource test specification (see below), this *resource test type* is validated against the type specified in the specification, if provided. The following test types are supported.   + *AIS* Alarm Indication Signal   + *CCM Continuity*  Connectivity Check Management Continuity (CCM)   + *CV Verification* Connectivity Verification (CV)   + *CFM Loopback*  Connectivity Fault Management loopback. (See 802.1ag) (CFM\_LB)   + *CFM Linktrace* Connectivity Fault Management linktrace (CFM\_LT)   + *EFM Loopback* Ethernet-in-the-First-Mile loopback (see 802.1ah) (EFM\_LB)   + *LCK* Lock Signal   + *SE Frame Loss* Single End Frame Loss (SE\_FL)   + *SES Frame Loss*  Single End Synthetic Frame Loss (SES\_FL)   + *OW Frame Delay*  One Way Frame Delay (OW\_FD)   + *OW Frame Delay Variation* Jitter (OW\_FDV)   + *TW Frame Delay* Two Way Frame Delay (TW\_FD)   + *TW Frame Delay Variation* Jitter (TW\_FDV)   + *DE Frame Loss* Dual End Frame Loss (DE\_FL)   + *IP Ping* (IP\_PING)   + *IP Traceroute* (IP\_TRT)   + *SELT* Single-end loop test   + *DELT* Dual end loop test   + *PCLT* Pots Circuit Line Test   + *PPPoE Dialup* Emulation PPP over Ethernet for GPON Subscriber Dialup Emulation (PPPoE)   + *CETI* Callee emulation test (incoming) for GPON POTs or xDSL Subscriber (CETI)   + *CETO* Caller emulation (outgoing) for GPON POTs or xDSL Subscriber (CETO)   + *RFC2544*  Benchmark Test (RFC2544) * **test admin state**  identifies the administrative state of the test   + *LOCKED* Test was originally set to locked when started   + *UNLOCKED*  When a test is originally started, and the test state not specified, it defaults to this value. * **test schedule (optional)** identifies when the test is executed and at what frequency   + *Start time type*  this parameter specifies the type of start type as immediate, fixed, or relative. The default is immediate   + *Start time*  this parameter specifies the scheduled start date/time to perform on-demand tests. It has no meaning for proactive tests and thus is not present. It is also not relevant for *start time type* = relative and thus is not present. The default value for this attribute is the current system date and time if *start time type* = fixed, but no start time is supplied.   + *Relative start time* this parameter specifies the relative start time, from the current system time, to perform on-demand tests. This attribute has no meaning for proactive tests and thus is not present. It is also not relevant when *start time type* = fixed and thus is not present. The default value for this attribute is zero, indicating the current system time, when *start time type* = relative, but no relative start time is provided.   + *Stop time type* this parameter specifies the type of start type as none, fixed, or relative. The default is none   + *Stop time* this parameter specifies the scheduled stop date/time to perform on-demand tests. This attribute has no meaning for proactive tests and is thus not present. It is also not relevant for *stop time type* = relative and thus is not present. The stop date/time value should be greater than or equal to the scheduled start date/time value. If stop time type = fixed, but no stop time is present, then this defaults to “forever”.   + *Relative stop time*  this parameters specifies the duration of the test. The duration time can be specified as forever (represented by a zero value) or as relative time (e.g., a given number of hours, minutes, and seconds from the start time). This attribute has no meaning for proactive tests and thus is not present. It is also not relevant when *stop time type* = fixed and thus is not present. The default value for this attribute is “forever”, indicating that the test will run until it is administratively stopped, or fails.   + *Recurring Frequency* this attribute specifies a configurable frequency per instance of a test. The frequency can be specified as none or in relative time (e.g., repeat the test every given number of hours, minutes, and seconds from the start time). If the *relative stop time* is forever, then the frequency should be none. If the *relative stop time* is relative time and the *frequency* is relative time, then the *frequency* time should be equal to or greater than the *relative stop time*. The default configured recurring frequency is none. * **test state** indicates the state of the test at any given moment in time. In general, it may have one of the following status values, but the exact interpretation depends on the test being executed.   + *Starting*  the test is in the process of being setup, or may be executing the first initial commands before returning successfully from the initial start command (i.e. the request has come in before the target OS has returned a response to the client OS that issued the *start).* Note: A response of “starting” does not indicate the next status will be “ongoing”. It may fail in the setup, or be stopped before entering into the ongoing state.   + *Ongoing*  this indicates the test has been successfully started and is still in the process of being executed.   + *Completed* the test has successfully completed.   + *Stopped* the test has been explicitly stopped. It may not have completed, but execution should come to a halt. Partial results may be available.   + *Failed* the test has failed   + *Unknown* indicates the status of the test cannot be determined. * **entity(s) under test**  identifies the entity, or entities under test and their associated parameters   + **resource test ref**  a reference to a resource   + ***transmission parameter list*** *(optional)* the transmission parameters relevant for defining a test   + ***test parameter list (optional)***  this attribute specifies a list of individual *resource test parameters*  and their associated *resource test parameter values*. The standard parameter names and valid values are those provided in SDx-xx TestParameters.pdf, and are passed in real-time over the interface upon starting the test. This may be an empty list.   + ***test specification (optional)***  a reference to the *resource test specification* used to configure the resource under test, if specified   + ***test measures list*** *-* a list of test measures and their associated threshold values, and associated test values when the test is executed. * **tp data list**  identifies transmission parameters for termination points involved in the tests. This is particularly used if the entity under test is an FDFr/SNC, and one wishes to specify transmission parameters (e.g. MEP/MIP) on the associated termination points. |
| Source |  |

### Resource Test Specification

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0007 | **Resource Test Specification**  The ***Resource Test Specification (RTS)*** shall represent a set of *resource test parameters* and their associated *resource test parameter values* that are used to configure a resource test. The specification may also be used as a means to validate test parameters passed over the interface directly when the test is instantiated.  An ***RTS*** is contained within an OS. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0008 | **Resource Test Specification Attributes**  An **RTS** object shall have, in addition to the attributes identified in R\_TMF518\_NRB\_I\_0001, the following attributes:   * **test type** identifies the type of test. See resource test object for definition of tests. * **version** identifies the version of the resource test specification (optional). Include the revision, number and name * **resource test parameter list** this attribute specifies a list of individual *resource test parameters* and their associated *resource test parameter values*. The standard parameter names and valid values are those provided in the document SDx-xx TestParameters.pdf.   When the resource under test references a *resource test specification* upon execution of a test, the contained parameters in this list are used to configure the actual test on the resource.  However, when executing a test, it may be possible to specify over the interface a resource test parameter value which overrides the corresponding value available in the associated *resource test specification*.  In this case the new proposed value applies only to the test instance created, and the *resource test parameter value* in the specification is not modified.   * **resource test measure list** specifies a list of measures that are to be reported by the test, and any associated threshold values against which the test is to report a consequence (i.e. send a notification, or other method) * **containedByTestSpecRef**   Provides the relationship to a resource test specification that will contains this test spec. For example, a parent test spec for “RFC2544” might contain this resource test specification that defines a “Frame Delay” test.   * **containTestSpecRefList**   Provides a list of resource test specifications that are contained in this specification. |
| Source |  |

### Resource Test Parameter

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0009 | **Resource Test Parameter**  The ***Resource Test Parameter (RTP)*** defines a parameter that will be used in configuring a test. It may be defined in a resource test specification, or passed directly over the interface.   * **name**  name of the parameter * **description**  description of the parameter. * **unique**  indication if value must be unique * **can be overridden**   An indicator that specifies that the ResourceTestParmValues associated with the ResourceTestParameter cannot be changed when instantiating a value. For example, a bandwidth of 64 MB cannot be changed.   * **minCardinality** Optional   The minimum number of instances a resource test parameter can take on. For example, 1-10 frame sizes in an RFC2544 test, where one is the value for the minCardinality.   * **maxCardinality** Optional   The maximum number of instances a resource test parameter can take on. For example, 1-10 frame sizes in an RFC2544 test, where ten is the value for the maxCardinality.   * **extensible**   An indicator that specifies that the values for the resource test parameter can be extended by adding new values when instantiating a resource test parameter over the interface. |
| Source |  |

### Resource Test Parameter Value

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0010 | **Resource Test Parameter Value**  The ***Resource Test Parameter Value (RTPV)*** defines a value to be applied to a parameter when configuring a test.   * **value type** the type of the value, i.e. string, integer.. * **value** the value to be applied to the resource test parameter * **default** a default value if no value is supplied * **unit of measure** the unit type associated to the parameter value * **value from** the lower bound of a value * **value to**  the upper bound of a value * **range Interval**  Indicates whether value “from” and “to” are inclusive, or not |
| Source |  |

### Resource Test Measure Definition

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0011 | **Resource Test Measure Definition**  The ***Resource Test Measure Definition (RTMD)*** defines a measure for which a result is obtained from the execution of the test.   * **name** name of the resource test measure * **description**  description of the parameter. * **unit of measure** specifies whether the value unit of measure, i.e. kbps, minutes, etc. * **capture frequency** the frequency with which the measure is taken * **capture period** the time period during which the measure is taken * **valueType** the type of value associated to the metric taken * **threshold rule set** a set of rules used to determine when a threshold has been passed |
| Source |  |

### Resource Test Measure Value

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0012 | **Resource Test Measure Value**  The ***Resource Test Measure Value (RTMV)*** defines a value resulting from execution of a test associated to a specific resource test measure for a given period of time.   * **name** name of the associated resource test measure definition * **value** the resulting test value. Note: this is not necessarily an integer, and be a “Boolean” indicating “true” or “false”, for example when measuring connectivity status. |
| Source |  |

### Threshold Rule Set

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0013 | **Threshold Rule Set**  The ***Threshold Rule Set*** is a set of rules that indicate under what condition(s) a threshold is considered to be exceeded.   * **name** name of the threshold rule set * **description** description of the rule set * **rules**  a set of rules associated to the rule set * **threshold type** the type of metric threshold |
| Source |  |

### Threshold Rule

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_I\_0014 | **Threshold Rule**  The ***Threshold Rule*** indicates the condition under which a threshold is considered to have been crossed, and the consequence of this crossing, e.g. send a TCA, reconfigure a device, etc.   * **name** name of the threshold rule * **description** description of the threshold rule * **threshold rule condition** the definition of the trigger that results in a consequence being taken. Values are “raise” and “clear” * **threshold rule severity** the perceived severity associated to the condition * **conformance target lower** a lower bound threshold * **conformance comparator lower** operator is used to compare a value with the conformanceTargetLower, ie “=” , “>”, or “>=” * **conformance target upper** an upper bound threshold * **conformance comparator upper** operator is used to compare a value with the conformanceTargetUpper, ie “=” , “<”, or “=<” * **conformance period** An interval of time during which the conformanceTarget Lower and Upper must be measured * **action**  the action to be taken as a result of the threshold being passed |
| Source |  |

## Category II: Normal Sequences, Dynamic Requirements

### Resource Specification Management

#### Creation of Resource Test Specification

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0015 | **Create Resource Test Specification**  The interface shall allow the requesting OS to create a *Resource Test Specification (RTS).* An RTS describes a resource test in terms of resource test parameters for configuration of the test, and measures expected to be produced as a result of running a test. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0016 | **Resource Test Specification Attributes**  The Interface shall allow the requesting OS to specify the following parameters when it creates a resource test configuration specification:   1. Name   This parameter defines the identifier of the new resource test specification that will be used over the interface. The target OS has to make sure the name is unique within a given management domain. If a name is not provided, the target OS must ensure that a unique name is created.   1. User label (optional)   Refer to R\_TMF518\_FMW\_I\_0001 in the TMF518\_FMW BA.   1. User label uniqueness (optional)   This parameter shall indicate to the target OS that the value of the user label attribute must be unique amongst the FDFrs within the target OS.   1. Owner (optional)   Refer to R\_TMF518\_FMW\_I\_0001 in the TMF518\_FMW BA .   1. Test type   A type of test for which the resource test specification is to be applied. The resource test type in the RTS is validated against the resource test type specified at the time of execution of a test. Version (optional)   1. Version   Identifies the version of the resource test specification (optional). Include the revision, number and name   1. Test Parameter List   This attribute specifies a list of individual *resource test parameters* and their associated *resource test parameter values*. The standard parameter names and valid values are those provided in the document SDx-xx TestParameters.pdf.   1. Resource Test Measures   This attribute specifies a list of individual resource test measures that are expected to result from the execution of a test. It also defines a set of threshold rules that indicate under which condition(s) a threshold is considered to be passed, and the expected consequence(s).   1. Contained by Test Spec Ref   Provides the relationship to a resource test specification that will contains this test spec. For example, a parent test spec for “RFC2544” might contain this resource test specification that defines a “Frame Delay” test.   1. Contains Test Spec Ref List   Provides a list of resource test specifications that are contained in this specification.   1. Vendor Extensions   Provides the ability to extend the resource test specification object |
| Source |  |

#### Modification of Resource Test Specification

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0017 | **Modify Resource Test Specification**  The interface shall allow the requesting OS to modify a *Resource Test Specification (RTS).* The following attributes may be modified:   1. Version 2. Test parameter list 3. Test measure list 4. Contained by Test Spec Ref 5. Contains Test Spec Ref List 6. Vendor Extensions |
| Source |  |

#### Retrieval of Resource Test Specification

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0018 | **Retrieval of Resource Test Specification**  The interface shall allow the requesting OS to retrieve a *Resource Test Specification (RTS),* or a set of *RTS via* the generic get Inventory operation defined in TMF518\_MRI.  Note: There is no support for an individual operation to specifically retrieve RTS. |
| Source |  |

#### Deletion of Resource Test Specification

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0019 | **Delete Resource Test Specification**  The interface shall allow the requesting OS to delete a *Resource Test Specification (RTS).* |
| Source |  |

### On-Demand Resource Test Control

#### Start Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0020 | **Start Test**  The interface shall allow the requesting OS to start a test. The test represents the execution of a particular test against a given resource(s), and given set of configuration parameters, in accordance with a specified schedule, and a list of desired test measures.  The request may be on-demand and controlled, i.e. the test has been requested to start, but has not yet completed, or is to run continuously without a stop time, and hence the operation returns before the test completes.  The test may be on-demand and uncontrolled i.e. the test runs and returns the results as part of the response to the operation. The expected behavior is dependent on the type of test being executed and the associated implementation.  In the case of a controlled test, successful execution of this operation results in instantiation of a resource test object. |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0021 | **Start Test Parameters**  The Interface shall allow the requesting OS to specify the following parameters when it requests that a target OS start a *resource test:*   1. Name   This parameter defines the identifier of the new resource test which will be used over the interface. The target OS has to make sure the name is unique within a given management domain. If a name is not provided, the target OS must ensure that a unique name is created.   1. User label (optional)   Refer to R\_TMF518\_FMW\_I\_0001 in the TMF518\_FMW BA.   1. User label uniqueness (optional)   This parameter shall indicate to the target OS that the value of the user label attribute must be unique amongst the FDFrs within the target OS.   1. Owner (optional)   Refer to R\_TMF518\_FMW\_I\_0001 in the TMF518\_FMW BA .   1. Test mode   Indicates whether a test is PROACTIVE or ONDEMAND. For tests that are started via the “start test” operation, the test mode is ONDEMAND. For tests started alternatively via the creation and activation of a FDFr | SNC, the mode is PROACTIVE.   1. Test type   The type of test. If a reference is provided to a resource test configuration specification (see below), this resource test type is validated against the type specified in the specification. See section 4.2.4 for a description of test types.   1. Test admin state   The desired administrative state of the test. “LOCKED” implies the test is to be in a suspended state and will remain so until a resume operation is requested. “UNLOCKED” implies the test is available for execution.   1. Test ref   A reference to the entity under test   1. Test schedule   Identifies when the test should be executed and at what frequency. See section 4.2.4 for a description of a test schedule.   1. Configured by list   This attribute specifies a list of individual *resource test parameters* and their associated *resource test parameter values* that are passed directly over the interface. If there is an associated value in a specification that is referenced, and the parameter is marked as “can be overridden”, then the value passed in the list overrides the value in the specification.   1. Resource Test Specification   Indicates the name of a resource test specification (RTS) that is to be used to describe how the test should be configured, and what measures should be taken as a result of executing the test. The RTS may contain resource test parameters and their associated values and/or resource test measures.   1. Resource Test Measure List   This attribute specifies a list of individual resource test measures that are expected to result from the execution of a test. For each measure a set of threshold rules may be defined that indicate under which condition(s) a threshold is considered to be passed, and the expected consequence(s).   1. TP Data List to Modify   A list of TP’s, with associated layer rates and transmission parameters to be modified as part of the request.   1. Vendor Extensions   Refer to R\_TMF518\_FMW\_I\_0001 |
| Source |  |

#### Suspend Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0022 | **Suspend Test**  The interface shall allow the requesting OS to *suspend* a test,resulting in the administrative state transitioning to “LOCKED”. Behavior associated to the suspension of the test will depend on the test type. Any previous test results should still be available, even when the test is suspended. |
| Source |  |

#### Resume Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0023 | **Resume Test**  The interface shall allow the requesting OS to *resume* a test that has previously been suspended,resulting in the administrative state of the state transitioning to “UNLOCKED”.  The test should resume execution, and any results produced as part of the test (previous to suspension) should still be available, as well as any new results that are produced from resuming the test.  Resuming a test that already has an administrative state of UNLOCKED has no impact on the test. |
| Source |  |

#### Stop Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0024 | **Stop Test**  The interface shall allow the requesting OS to *stop* a test, providing the test *name* that was returned from the successful start of a test. This operation results in the target OS sending a request down to the network element(s) to halt the execution of an ongoing test. The test object remains until it is either deleted by the system or explicitly deleted via the "delete test" operation. When the test is stopped it transitions to a “STOPPED” state. Partial, or complete, results may be available. **Note**: the test cannot be restarted using the “startTest” operation as execution is considered to be terminated. See “suspend test” and “resume test” for temporary stopping of a test. |
| Source |  |

#### Get Test Results

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0025 | **Get Test Results**  The interface shall allow the requesting OS to retrieve a *resource test* providing the following as input:   1. Name   The name of the test as previously returned from either a *start test* operation or available in the FDFR/SNC parameters in the case of pro-active tests   1. Test measures list (optional)   A subset of the available resource test measures that are already available in the test object. Note: the requesting OS cannot specify the retrieval of measures (that may be valid for a given test) that weren’t defined as part of the test executed.   1. Start time (optional)   In the case of history data, specifies the start of the time window for collection. If no value is supplied, retrieval starts with the earliest time-stamped results.   1. End time(optional)   In the case of history data, specifies the end of the time window for collection. If no value is specified, current time is implied.  If the test has not completed, nor has it been stopped, then the results may be unpredictable. The *validity indicators* in the results provide an indication of the validity of the results that are returned. |
| Source |  |

#### Delete Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0026 | **Delete Test**  The interface shall allow the requesting OS to *delete* a test, providing the test *name* that was returned from the successful start of a test. A running test should be stopped before a test delete occurs, otherwise the most recent results may not be available. This action results in deletion of the resource test object. |
| Source |  |

### Proactive Resource Test Control

#### Create and Activate Flow Domain Fragment with Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0001 | **Create and Activate Flow Domain Fragment with Test**  The interface shall allow the requesting OS to create and activate a flow domain fragment with a specified test, or set of tests, that are to be executed at the time of flow domain fragment activation.  The general creation and activation of the flow domain fragment shall follow the requirement as specified in:  R\_TMF518\_RP\_II\_0060 |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0002 | The create and activate flow domain fragment with test shall take the following parameters, in addition to all the parameters specified in  R\_TMF518\_RP\_II\_0061  For the definitions of the various parameters, see “Start Test Parameters”   1. Resource Test List    1. Test name    2. Test type    3. Test mode    4. Configured by List    5. Resource Test Specification    6. Resource Test Measures |
| Source |  |

#### Create and Activate Subnetwork Connection with Test

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0003 | **Create and Activate Subnetwork Connection with Test**  The interface shall allow the requesting OS to create and activate a Subnetwork connection with a specified test, or set of tests, that are to be executed at the time of Subnetwork connection activation.  The general creation and activation of the Subnetwork connection shall follow the requirement as specified in:  R\_TMF518\_RP\_II\_0010 |
| Source |  |

|  |  |
| --- | --- |
| R\_TMF518\_RTD\_II\_0004 | The create and activate Subnetwork connection with test shall take the following parameters, in addition to all the parameters specified in  R\_TMF518\_RP\_II\_0011  For the definitions of the various parameters, see “Start Test Parameters”   1. Resource Test List    1. Test name    2. Test type    3. Test mode    4. Configured by List    5. Resource Test Specification   Resource Test Measures |
| Source |  |

## Category III: Abnormal or Exception Conditions, Dynamic Requirements

No requirements have been identified for this category.

## Category IV: Expectations and Non-Functional Requirements

No requirements have been identified for this category.

## Category V: System Administration Requirements

No requirements have been identified for this category.

# Use Cases

## Resource Specification Management

### Resource Test Specification Use Cases

#### The Requesting OS Creates a Resource Test Specification

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0001 |
| Use Case Name | The requesting OS creates a Resource Test Specification |
| Summary | This operation allows a requesting OS to create a Resource Test Specification.  The target OS validates the data provided by requesting OS, and creates the requested Resource Test Specification in accordance with the requesting OS parameter list. If the target OS cannot create the Resource Test Specification as specified, an appropriate exception is raised. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 target OS to create a Resource Test Specification |
| Description | * The requesting OS sends a request to target OS to create a Resource Test Specification. The requesting OS provides the Resource Test Specification name, test type, version, and test parameter list as a minimum. It may also specify a set of test measures, and associated thresholds. * The target OS validates the request: * If the syntax is in error, an Invalid Input exception is raised. * If uniqueness of the user label is required, the target OS checks the user label for uniqueness; i.e., if a Resource Test Specification object with the same user label exists already, then a User Label In Use exception is raised. * If the maximum number of Resource Test Specifications in the target OS has already been reached, a Capacity Exceeded exception is raised. * If the request is valid the target OS creates the Resource Test Specification. * The target OS replies with success indication.   The target OS sends a Resource Test Notification object creation notification to the notification service. |
| Ends When | In case of success:  The requesting OS receives an indication of the success of the request.  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The Resource Test Specification has been created by the target OS.  In case of failure:  Nothing has changed in the System, i.e., the Resource Test Specification has not been created, and any future reference to it in an operation shall result in an exception. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0015](#R_TMF518_RTD_II_0015) [R\_TMF518\_RTD\_II\_0016](#R_TMF518_RTD_II_0016) |

#### The Requesting OS Modifies a Resource Test Specification

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0002 |
| Use Case Name | The requesting OS modifies a Resource Test Specification |
| Summary | This operation allows a requesting OS to modify a Resource Test Specification.  The target OS validates the data provided by requesting OS, and modifies the requested Resource Test Specification in accordance with the requesting OS request parameter list. If the target OS cannot modify the Resource Test Specification as specified, an appropriate exception is raised. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 target OS to modify a Resource Test Specification |
| Description | * The requesting OS sends a request to target OS to modify a Resource Test Specification. The requesting OS provides one or more of the following parameters that can be modified: version, a list of resource test parameters, a list of test measures, and possibly an external reference. * The target OS validates the request: * If the syntax is in error, an Invalid Input exception is raised. * If the specified Resource Test Specification object does not exist, an Entity Not Found exception is raised. * If the request is valid the target OS modifies the Resource Test Specification with the new parameters passed. Note: This DOES NOT result in the modification of the tests that have the Resource Test Specification already associated to it. * The target OS replies with success indication.   The target OS sends a Resource Test Specification attribute value change notification to the notification service, indicating the attributes that have changed. |
| Ends When | In case of success:  The requesting OS receives an indication of the success of the request.  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The Resource Test Specification has been modified by the target OS. Only from the time of the modification do the new parameters take affect when the object is referenced.  In case of failure:  Nothing has changed in the System, i.e., the Resource Test Specification has not been modified. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0017](#R_TMF518_RTD_II_0017) |

#### The Requesting OS Deletes a Resource Test Specification

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0003 |
| Use Case Name | The requesting OS deletes a Resource Test Specification |
| Summary | This operation allows a requesting OS to delete a Resource Test Specification that already exists.  The target OS is required to validate that the Resource Test Specification is not associated to any test object before deleting it. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 target OS to delete a Resource Test Specification. |
| Description | * The requesting OS sends a request to target OS to delete a Resource Test Specification. The requesting OS provides the Resource Test Specification name. * The target OS validates the request: * If the syntax is in error, an Invalid Input exception is raised. * If the specified Resource Test Specification object is not known to the target OS, an Entity Not Found exception is raised. * If any Resource Test within the target OS has an association to the Resource Test Specification, an Object In Use exception is raised. * If the request is valid, the target OS deletes the Resource Test Specification. * The target OS replies with success indication.   The target OS sends a Resource Test Specification object deletion notification to the notification service. |
| Ends When | In case of success:  The requesting OS receives an indication of the success of the request.  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The Resource Test Specification has been deleted by the target OS.  In case of failure:  Nothing has changed in the System, i.e., the Resource Test Specification has not been deleted. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0019](#R_TMF518_RTD_II_0019) |

## Resource Test Management

### Resource Test Pro-Active Use Cases

#### The Requesting OS Starts a Controlled, Proactive Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0004 |
| Use Case Name | The requesting OS starts a controlled, pro-active test |
| Summary | The test is started at the moment a Flow Domain Fragment or SNC is created and activated “with test”. Thus, this use case applies to the following use cases in TMF518-RP (Resource Provisioning):   1. UC\_TMF518\_RP\_0003   The requesting OS creates and activates a Subnetwork Connection (SNC)   1. UC\_TMF518\_RP\_0031   The requesting OS creates and activates a Flow Domain Fragment (FDFr)  As part of the Flow Domain Fragment or SNC Create data, the client OS specifies any FDFr or SNC object test transmission parameters that are required (e.g. to create a MEG). On the TPDataListToModify, the client OS would specify any test related aEndPoint and zEndPoint transmission parameters, in particular the MEPs on CTPs.  Note: these parameters may be found in section “Test & Diagnostic Specific Parameters” in the document SD1-16 LayeredParameters.pdf found in the Framework DDP. The purpose of setting these parameters is to setup up the MEG/MEPs for tests that require these entities. Note: any of the TP related transmission parameters may be set in advance via the setTerminationPoint operation.  The client OS also specifies as part of the Flow Domain Fragment or SNC Create Data a resource test specification that contains the test parameters and test measure definitions, amongst other things. There may be multiple tests to execute, for example CCM, Frame Delay, and Frame Loss, in which case a list of resource test specifications is provided – one for each test.  The target OS proceeds in accordance with the general use cases specified above, including validation of the transmission parameters, and contents of the resource test specification(s) that may have been defined in support of pro-active tests.  If all validation is successful, the target OS starts execution of the test(s) and creates a resource test object for each test. It returns the names of the tests in the Flow Domain Fragment or Subnetwork Connection transmission parameters, if they were not specified on input. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 create and activate a Flow Domain Fragment or Subnetwork Connection |
| Description | 1. The requesting OS sends one of the following requests to the target OS: 2. createAndActivateFlowDomainFragmentWithTest 3. createAndActivateSubnetworkConnectionWithTest 4. The target OS validates the request in accordance with the use cases defined above. For the test related parameters, the following validation is performed: 5. If the syntax is in error, an Invalid Input exception is raised. 6. If the name of a Resource Test Specification supplied on the request is not known to the target OS, an Entity Not Found exception is raised. 7. If the requesting OS has provided a Resource Test name(s) as part of the transmission parameters, and the target OS does not support requesting OS supplied names, an Unable To Comply exception is raised. 8. If the test type is not recognized or not supported, an Unable to Comply exception is raised. 9. If the Resource Test Specification provided contains parameters that are not recognized or not supported by the target OS, an Unable To Comply exception is raised. 10. The target OS continues the creation and activation of the FDFr or the SNC in accordance with the associated use cases. 11. If the requesting OS provides transmission parameters for the involved FDFR | SNC | TPs, it is up to the target OS to provision the parameters on these TPs as part of requesting execution of the test. 12. As part of the creation and activation process, the target OS creates a resource test object, and fills in the structure based upon data passed on the create and activate operation. 13. The target OS examines the test admin state. If it is “LOCKED”, then the pro-active test is not started. If it is “UNLOCKED”, the target OS initiates the start of the resource test, configuring the resource under test with the resource test parameters and associated values specified in the resource test specification. 14. The test begins to execute if the admin state is UNLOCKED. 15. If the test fails, then the target OS will put the resource test status = “failed” in the resource test object, but the creation and activation request returns success, with the name(s) of the test(s) in the transmission parameter list for the FDFr | SNC. 16. If the test failed, but the FDFr or SNC was successfully created and activated, the client OS has the option of fixing the problem and restarting the test now via the *start test* operation |
| Ends When | In case of success:  The target OS returns the results of the create and activate FDFr or SNC, including a list of test name(s) associated with the entity created.  A resource test object creation notification is sent to the notification service, as well as the FDFr | SNC object creation notification.  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The resource test object has been created. Depending on the admin state, the test has been started, or not  Transmission parameters that have been successfully set on FDFR | SNC | TPs remain set and are available for future tests. This implies they need not be set a second time.  In case of failure:  The resources that may have been under test are free for future tests.  Transmission parameters that were not successfully set remain either unset, or with values that were previously set. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0001](#R_TMF518_RTD_II_0001) [R\_TMF518\_RTD\_II\_0002](#R_TMF518_RTD_II_0002)  [R\_TMF518\_RTD\_II\_0003](#R_TMF518_RTD_II_0003) [R\_TMF518\_RTD\_II\_0004](#R_TMF518_RTD_II_0004) |

#### The Requesting OS Stops a Controlled, Proactive Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0007 |
| Use Case Name | The requesting OS stops a controlled, pro-active test |
| Summary | There are two ways to stop a controlled, pro-active test:   1. Execute the stop test operation as defined in the associated use case, providing the name of the test. In this case the test is stopped, but the FDFR | SNC continue to function as provisioned. 2. Deactivate and delete the FDFr | SNC. In this case the test is stopped, the associated resource test object is deleted, and if successful, the FDFR | SNC is deactivated and deleted. Thus, this option builds upon the following use cases as defined in RTM518\_RP (Resource Provisioning) 3. UC\_TMF518\_RP\_0009   The requesting OS deactivates and deletes a Subnetwork Connection (SNC)   1. UC\_TMF518\_RP\_0033   The requesting OS deactivates and deletes a Flow Domain Fragment  The client OS does not need to pass in the list of associated test names, as the target OS maintains the relationship between the pro-active tests and the FDFr | SNC. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 deactivate and delete an FDFr or an SNC |
| Description | 1. The requesting OS sends a request to deactivate and delete the FDFr or the SNC to the target OS. 2. The target OS validates the request in accordance with the above specified UC in RTM518\_RP 3. The target OS then requests that the pro-active tests be stopped before deactivating and deleting the FDFr | SNC. When successful, the test transitions to a stopped state. 4. It deletes any MEG/MEP objects defined as part of the test. 5. It deletes the resource test object   The target OS proceeds with the deactivation and deletion of the FDFr | SNC in accordance with the UCs defined above. |
| Ends When | In case of success:  The target OS deactivates and deletes the FDFr | SNC.  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The test has stopped the test  A resource test state change notification will be sent every time the test transitions through a new state, in particular when it arrives at the stopped state.  When the resource test object is deleted an object deletion notification is sent.  In case of failure:  The test remains in the state it was in prior to execution of the operation, and the resource test object is not deleted. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0001](#R_TMF518_RTD_II_0001) [R\_TMF518\_RTD\_II\_0002](#R_TMF518_RTD_II_0002)  [R\_TMF518\_RTD\_II\_0003](#R_TMF518_RTD_II_0003) [R\_TMF518\_RTD\_II\_0004](#R_TMF518_RTD_II_0004) |

### Resource Test On-Demand Use Cases

#### The Requesting OS Starts an Uncontrolled, On-Demand Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0008 |
| Use Case Name | The requesting OS starts an uncontrolled, on-demand test |
| Summary | This operation provides a way to start an uncontrolled, on-demand resource test. After successful verification of the parameters by the target OS, a request is sent to the appropriate resource, or set of resources, to begin execution of the test using the test configuration parameter list provided as input.  The client OS will wait until the test has finished execution to receive the results of the test, if available. Should the test fail, the operation will fail with an applicable exception.  Note that in this case, as the test is uncontrolled, that no resource test object is created, and thus no test name is returned to the client OS. This implies that no future requests can be made against the test. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 start a test to the client OS. |
| Description | 1. The requesting OS sends a request to start a test to the target OS. 2. The target OS validates the request: 3. If the syntax is in error, an Invalid Input exception is raised. 4. If the name of the resource(s) specified as under test is not known to the target OS, an Entity Not Found exception is raised. 5. If the name of a Resource Test Specification is supplied on the request is not known to the target OS, an Entity Not Found exception is raised. 6. If the requesting OS has provided a Resource Test name and the target OS does not support requesting OS supplied names, an Unable To Comply exception is raised. 7. If any of the mandatory input parameters cannot be satisfied, an Unable To Comply exception is raised. 8. If the test type is not recognized or not supported, an Unable to Comply exception is raised. 9. If the Resource Test Specification provided contain parameters that are not recognized by the target OS, an Unable To Comply exception is raised. 10. If the requesting OS provides transmission parameters for the involved FDFR | SNC | TPs, it is up to the target OS to provision the parameters on these TPs as part of requesting execution of the test. These are best effort, and if the set fails, default values on the resources go into effect, if they exist. If not, and execution of the test cannot proceed, an Unable to Comply exception is raised. 11. The target OS initiates the start of the resource test, configuring the resource(s) under test with the resource test parameters and associated values, and/or the values specified in a resource test specification. If a value is directly specified, but is also found in a resource test specification, the supplied value takes precedence over the one found in the specification. 12. The test begins to execute 13. If the test fails, then the target OS will reply with a failure indication and error reason, if available. 14. Otherwise, the target OS replies with a success indication and the results of the test.   As there is no resource test object created, NO object creation notification is sent. |
| Ends When | In case of success:  The target OS returns the results of the uncontrolled, on-demand test.  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The test has been executed  Transmission parameters that have been successfully set on FDFR | SNC | TPs remain set and are available for future tests. This implies they need not be set a second time.  In case of failure:  The resources that may have been under test are free for future tests.  Transmission parameters that were not successfully set remain either unset, or with values that were previously set. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0020](#R_TMF518_RTD_II_0020) [R\_TMF518\_RTD\_II\_0021](#R_TMF518_RTD_II_0021)  [R\_TMF518\_RTD\_BR\_0028](#R_TMF518_RTD_BR_0028) |

#### The Requesting OS Starts a Controlled, On-Demand Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0009 |
| Use Case Name | The requesting OS starts a controlled, on-demand test |
| Summary | This operation provides a way to start a controlled, on-demand resource test. After successful verification of the test parameters by the target OS, a determination is made whether MEG/MEP/MIP parameters are required and have been set up when the FDFr /SNC was first created and activated. Once validated, a request is sent to the appropriate resource, or set of resources, to begin execution of the test using the test configuration parameter list provided as input.  When the target OS receives notice from the resource under test, acknowledging the start of the test, a resource test object is created. If the test cannot be started, the request is considered to have failed, and no resource test object is created.  The target OS then returns to the client OS either a success or failure. In the case of success, the target OS returns to the client OS the name of the test as a “handle” for future control of the test. The name may be that which was provided by the client OS when the test was started, or a name provided by the target OS if no name was provided as input. In the case of failure, an exception is returned to the client OS.  If successful, a resource test object creation notification is sent to the notification service. Resource state change notifications will also be sent as the test advances through its’ various state transitions. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 start a controlled test to the client OS. |
| Description | 1. The requesting OS sends a request to start a test to the target OS. 2. The target OS validates the request: 3. If the syntax is in error, an Invalid Input exception is raised. 4. If the name of the resource(s) specified as under test is not known to the target OS, an Entity Not Found exception is raised. 5. If the name of a Resource Test Specification specified on the request is not known to the target OS, an Entity Not Found exception is raised. 6. If a Resource Test with the same properties as specified in the requesting OS request already exists, the target OS may reuse that Resource Test is the test status is “stopped”. However, any existing results present in the test may be cleared. The behavior is test dependent. 7. If the requesting OS has provided a Resource Test name and the target OS does not support requesting OS supplied names, an Unable To Comply exception is raised. 8. If uniqueness of the user label is required, the target OS checks the user label for uniqueness; i.e., if a Resource Test object with the same user label exists already, then a User Label In Use exception is raised. 9. If any of the mandatory input parameters cannot be satisfied, an Unable To Comply exception is raised. 10. If the test type is not recognized or not supported, an Unable to Comply exception is raised. 11. If the Resource Test Specification provided contains parameters that are not recognized by the target OS, an Unable To Comply exception is raised. 12. If the test requires that a MEG/MEP/MIP parameters be present, it validates them. If not valid, an Invalid Input exception is raised. 13. If the requesting OS provides transmission parameters for the involved FDFR | SNC | TPs, it is up to the target OS to provision the parameters on these TPs as part of requesting execution of the test. These are best effort, and if the set fails, default values on the resources go into effect, if they exist. If not, and execution of the test cannot proceed, an Unable to Comply exception is raised. 14. The target OS initiates the start of the resource test, configuring the resource under test with the resource test parameters and associated values, and/or the values specified in a resource test specification. If a value is directly specified, but is also found in a resource test specification, the supplied value takes precedence over the one found in the specification. 15. The target OS notified from the resource under test that the test has started. As a result, the target OS creates a resource test object and marks the status of the test as “starting”. If the start of the test is not successful, no resource object is created. 16. The target OS returns to the client OS with either “success” or “failure” as determined in the previous step. The name of the test returned is either that provided by the client OS, or a name generated by the target OS if no name was provided. This name is used in all future requests on the test. 17. A resource test object creation notification is sent to the notification service if the test has successfully started. |
| Ends When | In case of success:  The target OS returns the name of the test that was successfully started  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The test has started execution  Transmission parameters that have been successfully set on FDFR | SNC | TPs remain set and are available for future tests. This implies they need not be set a second time.  A resource test state change notification will be sent every time the test transitions through a new state.  In case of failure:  The resources that may have been under test are free for future tests.  Transmission parameters that were not successfully set remain either unset, or with values that were previously set. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0020](#R_TMF518_RTD_II_0020) [R\_TMF518\_RTD\_II\_0021](#R_TMF518_RTD_II_0021)  [R\_TMF518\_RTD\_BR\_0030](#R_TMF518_RTD_BR_0030) |

#### The Requesting OS Suspends a Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0005 |
| Use Case Name | The requesting OS suspends a test |
| Summary | The requesting OS sends a request to suspend a test, passing in the test name of a test to suspend. The request is validated. The target OS either attempts to suspend the test, or does nothing, depending on the state of the test. In all cases, other than failure, the test administrative state is set to “LOCKED”. |
| Actor(s) | Target OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 suspend a test to the target OS. |
| Description | 1. The target OS validates the name of the requests. If the test is not found, an ENTITY\_NOT\_FOUND 2. The target OS determines based on the test type, whether a test can be suspended or not. If not, an UNABLE\_TO\_COMPLY exception is sent. 3. The target OS determines the test administrative state. If it is already “LOCKED”, it does nothing and exits successfully. 4. The target OS then examines the state of the test object. Depending on the test state, the target OS takes action as follows    * State= Starting | Ongoing – Temporarily stops execution of the test    * State = Completed – Nothing    * State= Stopped – Nothing    * State=Failed – Nothing    * State=Unknown – Attempts to stop the test 5. The target OS sets the test administrative state to “LOCKED”, if successful. |
| Ends When | In case of success:   * The test has stopped the test * A resource test state change notification will be sent every time the test transitions through a new state * A resource test administrative state change will be sent when the state transitions to “LOCKED”.   In case of failure:  The test remains in the state it was in prior to execution of the operation, and the resource test administrative state remains as it was. |
| Post-Conditions | In case of success: The test is temporarily halted  In case of failure: The test remains in its’ previous state |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0022](#R_TMF518_RTD_II_0022) |

#### The Requesting OS Resumes a Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0006 |
| Use Case Name | The requesting OS resumes a test |
| Summary | The requesting OS sends a request to resume a test, passing in the test name of a test to resume. The request is validated. The target OS either attempts to resume the test, or does nothing, depending on the state of the test. In all cases, other than failure, the test administrative state is set to “UNLOCKED”. |
| Actor(s) | Target OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 resume a test to the target OS. |
| Description | 1. The target OS validates the name of the test request. If the test is not found, an ENTITY\_NOT\_FOUND 2. The target OS determines the test administrative state. If it is already “UNLOCKED”, it does nothing and exits successfully. Otherwise, the test is “LOCKED” from a previous “suspend” action. 3. The target OS then attempts to “restart” the test. If not successful, the target OS returns with an exception “UNABLE\_TO\_COMPLY” 4. The target OS sets the test administrative state to “LOCKED”, if successful. |
| Ends When | In case of success:   * The test has restarted * A resource test state change notification will be sent every time the test transitions through a new state * A resource test administrative state change will be sent when the state transitions to “UNLOCKED”.   In case of failure:  The test remains in the state it was in prior to execution of the operation, and the resource test administrative state remains as it was. |
| Post-Conditions | In case of success: The test is restarted  In case of failure: The test remains in its’ previous state |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0023](#R_TMF518_RTD_II_0023) |

#### The Requesting OS Stops a Controlled, On-Demand Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0010 |
| Use Case Name | The requesting OS stops a controlled, on-demand test |
| Summary | This operation provides a way to stop a controlled, on-demand resource test. After the target OS validates the name of the test and that there is a corresponding resource test object, the target OS examines the test status, and depending on the status may try to halt execution of the test (or do nothing in some cases).  If the target OS successfully stops the test, the test status transitions to “stopped”, and returns “success” to the client OS. If not, the target OS returns an exception. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 stop a controlled test to the client OS. |
| Description | 1. The requesting OS sends a request to stop a test to the target OS. 2. The target OS validates the request: 3. If the syntax is in error, an Invalid Input exception is raised. 4. If the name of the resource(s) specified as under test is not known to the target OS, an Entity Not Found exception is raised. 5. The target OS then examines the test status available in the resource test object, and does the following based on this status: (the exact behavior will depend on the type of test) 6. Status = starting    1. The target OS attempts to stop the test. It may be successful, but it may have to wait until the test transitions to ongoing and either completes, or completes a first cycle. If results are available, they are stored in the test object. 7. Status = ongoing    1. The target OS attempts to stop the test. It may be successful, but it might have to wait for a test cycle to complete if a periodicity has been specified. It updates the results in the test object, whether partial or complete. 8. Status = successful    1. The target OS does nothing, as the test has already completed. 9. Status = stopped    1. The target OS does nothing, as the test has already been stopped. 10. Status = failed     1. The target OS does nothing, as the test failed 11. Status = unknown     1. As the target OS doesn’t know what has occurred, it attempts to stop the test. If any results are available, they are stored in the resource test results list. 12. If in any of the cases above the target OS fails to stop the test (meaning it receives an error back from the resource under test), it returns to the client OS with an Unable To Comply exception.   If successful, the target OS changes the status of the resource test to “stopped” and returns to the client OS with “success”. Note: the test cannot be executed again unless it is explicitly “started” with the *start test* operation. |
| Ends When | In case of success:  The target OS returns the name of the test that was successfully stopped  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The test has stopped the test  A resource test state change notification will be sent every time the test transitions through a new state, in particular when it arrives at the stopped state. Note: it may transition for example from starting to ongoing, and then to stopped.  In case of failure:  The test remains in the state it was in prior to execution of the operation. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0024](#R_TMF518_RTD_II_0024) |

#### The Requesting OS Retrieves Results from a Controlled, On-Demand or Pro-Active Test

|  |  |
| --- | --- |
| Use Case Id | UC\_TMF518\_RTD\_0011 |
| Use Case Name | The requesting OS retrieves results from a controlled, on-demand or pro-active test |
| Summary | This operation provides a way to retrieve results of a controlled, on-demand or pro-active resource test. The target OS validates the name of the test and that there is a corresponding resource test object, then determines which resource test measures are being requested, or whether all are to be returned.  The target OS then determines whether current or history data is being requested; it also verifies the scope of the operation to determine whether to return the resource test configuration as well as the results.  The target OS then returns the requested data. If unable to do so, an exception is returned. |
| Actor(s) | The requesting OS |
| Pre-Conditions | The requesting OS and target OS have successfully 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 retrieve test results to the client OS. |
| Description | 1. The requesting OS sends a request to retrieve test results to the target OS. 2. The target OS validates the request: 3. If the syntax is in error, an Invalid Input exception is raised. 4. If the name of the resource(s) specified as under test is not known to the target OS, an Entity Not Found exception is raised. 5. If the requested test results are invalid, or not applicable to the associated test, an Unable To Comply exception is raised. 6. The target OS will return an Unable to Comply exception if any of the following conditions are met:    1. A start time and stop time are specified, and the stop time is earlier than the start time    2. The start time is later than the latest time-stamped result    3. No start time is present, and the stop time is earlier than any time-stamped result 7. The target OS determines if the results are available on a single resource entity under test (e.g. a TP), or whether they are available on multiple resources (e.g. an FDFr and associated endpoints) 8. The target OS then determines the scope of the request and whether it is “all” (return configuration and results) or just “results” (return only the results) 9. For pro-active tests for which a granularity is specified, the target OS determines which granularity to apply to the measurement. It also examines the start / stop time to determine if history measurements are being requested. 10. For on-demand tests, the target OS applies the “window” (start time/stop) time of the data requested. If the test has not completed, partial results will be returned. 11. For on-demand tests, if the start time and stop time are the same, or absent, a single instance of results is returned. 12. It then examines the resource test object test results to determine if there is any data available that meets the request and/or requests the result data from the resource under test. If so, the target OS selects the data to be returned. If not, “empty” test results are returned.   The target OS returns the requested resource test results. |
| Ends When | In case of success:  The target OS returns the requested resource test results  In case of failure:  The requesting OS receives an exception as an indication of the failure of the request. |
| Post-Conditions | In case of success:  The test results have been returned.  In case of failure:  The resource test object remains unchanged from the time prior to execution of the request. |
| Exceptions |  |
| Traceability | [R\_TMF518\_RTD\_II\_0025](#R_TMF518_RTD_II_0025) [R\_TMF518\_RTD\_BR\_0031](#R_TMF518_RTD_BR_0031)  [R\_TMF518\_RTD\_BR\_0032](#R_TMF518_RTD_BR_0032) |

# Traceability Matrices

## Use Case Matrix

| **Use Case Id** | **Use Case Name** | **Requirements** |
| --- | --- | --- |
| [UC\_TMF518\_RTD\_0001](#UC_TMF518_RTD_0001) | The requesting OS creates a Resource Test Specification | [R\_TMF518\_RTD\_II\_0015](#R_TMF518_RTD_II_0015) [R\_TMF518\_RTD\_II\_0016](#R_TMF518_RTD_II_0016) |
| [UC\_TMF518\_RTD\_0002](#UC_TMF518_RTD_0002) | The requesting OS modifies a Resource Test Specification | [R\_TMF518\_RTD\_II\_0017](#R_TMF518_RTD_II_0017) |
| [UC\_TMF518\_RTD\_0003](#UC_TMF518_RTD_0003) | The requesting OS deletes a Resource Test Specification | [R\_TMF518\_RTD\_II\_0019](#R_TMF518_RTD_II_0019) |
| [UC\_TMF518\_RTD\_0004](#UC_TMF518_RTD_0004) | The requesting OS starts a controlled, pro-active test | [R\_TMF518\_RTD\_II\_0001](#R_TMF518_RTD_II_0001) [R\_TMF518\_RTD\_II\_0002](#R_TMF518_RTD_II_0002)  [R\_TMF518\_RTD\_II\_0003](#R_TMF518_RTD_II_0003) [R\_TMF518\_RTD\_II\_0004](#R_TMF518_RTD_II_0004) |
| [UC\_TMF518\_RTD\_0005](#UC_TMF518_RTD_0005) | The requesting OS suspends a test | [R\_TMF518\_RTD\_II\_0022](#R_TMF518_RTD_II_0022) |
| [UC\_TMF518\_RTD\_0006](#UC_TMF518_RTD_0006) | The requesting OS resumes a test | [R\_TMF518\_RTD\_II\_0023](#R_TMF518_RTD_II_0023) |
| [UC\_TMF518\_RTD\_0007](#UC_TMF518_RTD_0007) | The requesting OS stops a controlled, pro-active test | [R\_TMF518\_RTD\_II\_0001](#R_TMF518_RTD_II_0001) [R\_TMF518\_RTD\_II\_0002](#R_TMF518_RTD_II_0002)  [R\_TMF518\_RTD\_II\_0003](#R_TMF518_RTD_II_0003) [R\_TMF518\_RTD\_II\_0004](#R_TMF518_RTD_II_0004) |
| [UC\_TMF518\_RTD\_0008](#UC_TMF518_RTD_0008) | The requesting OS starts an uncontrolled, on-demand test | [R\_TMF518\_RTD\_II\_0020](#R_TMF518_RTD_II_0020) [R\_TMF518\_RTD\_II\_0021](#R_TMF518_RTD_II_0021)  [R\_TMF518\_RTD\_BR\_0028](#R_TMF518_RTD_BR_0028) |
| [UC\_TMF518\_RTD\_0009](#UC_TMF518_RTD_0009) | The requesting OS starts a controlled, on-demand test | [R\_TMF518\_RTD\_II\_0020](#R_TMF518_RTD_II_0020) [R\_TMF518\_RTD\_II\_0021](#R_TMF518_RTD_II_0021)  [R\_TMF518\_RTD\_BR\_0030](#R_TMF518_RTD_BR_0030) |
| [UC\_TMF518\_RTD\_0010](#UC_TMF518_RTD_0010) | The requesting OS stops a controlled, on-demand test | [R\_TMF518\_RTD\_II\_0024](#R_TMF518_RTD_II_0024) |
| [UC\_TMF518\_RTD\_0011](#UC_TMF518_RTD_0011) | The requesting OS retrieves results from a controlled, on-demand or pro-active test | [R\_TMF518\_RTD\_II\_0025](#R_TMF518_RTD_II_0025) [R\_TMF518\_RTD\_BR\_0031](#R_TMF518_RTD_BR_0031)  [R\_TMF518\_RTD\_BR\_0032](#R_TMF518_RTD_BR_0032) |

# Future Directions

The test and diagnostic work is being continued as part of the “TIP Test and Diagnostics” project.

# References

## References

## IPR Releases and Patent Disclosure

This document may involve a claim of patent rights by one or more of the contributors to this document, pursuant to the Agreement on Intellectual Rights between the TM Forum and its members. Interested parties should contact the TM Forum office to obtain notice of current patent rights claims subject to this document.

# Administrative Appendix

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

## About this document

This document has been generated from the mTOP\_BA.dot Word template,   
which itself is based on Version 6.0 of the TMF 402, BA Template.

## Use and Extension of a TM Forum Business Agreement

This document defines the business problem and requirement model for <<problem area>>. 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 <<(TMF <<number>> if available)>> when the Information Agreement is available. Reviewing the two documents together helps in gaining a full understanding of how the technology neutral information model solution is defined for this requirement model. An initial Business Agreement may only deal with a subset of the requirements. It is acceptable for subsequent issues of the document to add additional requirements not addressed by earlier releases of the BA. Business Agreements are the basis for requirement traceability for information models.

It is expected that this document will be used:

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

## Document History

| Version Number | Date Modified | Modified by: | Description of changes |
| --- | --- | --- | --- |
| 1.0 | October 2014 | Jessie Jewitt | Initial document |

## Company Contact Details

|  |  |
| --- | --- |
| **Company** | **Team Member Representative** |
| Ciena | *Name: Jessie Jewitt*  *Title*  *Email: jjewitt@ciena.com*  *Phone: 650 714-2593*  *Fax* |

## Acknowledgments

This document was prepared by the members of the TM Forum Resource and Service Assurance team

* Jessie Jewitt, Ciena **Editor**
* Andrea Mazzini, Alcatel-Lucent

## About TM Forum

TM Forum is an international consortium of communications service providers and their suppliers. Its mission is to help service providers and network operators automate their business processes in a cost- and time-effective way. Specifically, the work of the TM Forum includes:

* Establishing operational guidance on the shape of business processes.
* Agreeing on information that needs to flow from one process activity to another.
* Identifying a realistic systems environment to support the interconnection of operational support systems.
* Enabling the development of a market and real products for integrating and automating telecom operations processes.

The members of TM Forum include service providers, network operators and suppliers of equipment and software to the communications industry. With that combination of buyers and suppliers of operational support systems, TM Forum is able to achieve results in a pragmatic way that leads to product offerings (from member companies) as well as paper specifications.