International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

M.3706
(11/2013)

SERIES M: TELECOMMUNICATION MANAGEMENT, INCLUDING TMN AND NETWORK MAINTENANCE

Integrated services digital networks

Common management services – Test management – Protocol neutral requirements and analysis

Recommendation ITU-T M.3706

ITU-T M-SERIES RECOMMENDATIONS

TELECOMMUNICATION MANAGEMENT, INCLUDING TMN AND NETWORK MAINTENANCE

Introduction and general principles of maintenance and maintenance organization	M.10–M.299
International transmission systems	M.300–M.559
International telephone circuits	M.560–M.759
Common channel signalling systems	M.760–M.799
International telegraph systems and phototelegraph transmission	M.800–M.899
International leased group and supergroup links	M.900–M.999
International leased circuits	M.1000–M.1099
Mobile telecommunication systems and services	M.1100–M.1199
International public telephone network	M.1200–M.1299
International data transmission systems	M.1300–M.1399
Designations and information exchange	M.1400–M.1999
International transport network	M.2000–M.2999
Telecommunications management network	M.3000–M.3599
Integrated services digital networks	**M.3600–M.3999**
Common channel signalling systems	M.4000–M.4999

For further details, please refer to the list of ITU-T Recommendations.

Recommendation ITU-T M.3706

Common management services – Test management – Protocol neutral requirements and analysis

Summary

Recommendation ITU-T M.3706 provides the requirements and analysis for test management which is one of the common management services. Test management functions are used to assist in various high level management function sets such as fault management or performance management. The purpose of testing is to obtain information about the functionality and performance of the managed network subjected to the test.

The functional requirements for the test management interface include the management functions for test object controlling, test category, test object state monitoring, test result reporting, etc.

In the analysis part, the detailed information model supporting the above functions across the management interface is provided.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T M.3706	2013-11-13	2	11.1002/1000/12045

Keywords

Common management service, test management, test management IRP.

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

Table of Contents

Recommendation ITU-T M.3706

Common management services – Test management – Protocol neutral requirements and analysis

1 Scope

The purpose of this Recommendation is to define an interface through which a manager can perform test functions on an agent. This document specifies the overall requirements and the analysis for test management and defines the semantics of operations (and their parameters) visible via the management interface.

This Recommendation provides the protocol neutral model definition for notification. It defines, for the purpose of performing test functions on an agent, the information is observable and controlled by the management system's client and it also specifies the semantics of the interactions used to carry this information.

This Recommendation does not define the syntax or encoding of the operations and their parameters.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T M.3020]	Recommendation ITU-T M.3020 (2009), *Management interface specification methodology.*
[ITU-T M.3160]	Recommendation ITU-T M.3160 (2008), *Generic, protocol-neutral management information model*.
[ITU-T M.3702]	Recommendation ITU-T M.3702 (2010), *Common management services – notification management – Protocol neutral requirement and analysis*.
[ITU-T X.680]	Recommendation ITU-T X.680 (2008) | ISO/IEC 8824-1:2008, *Information technology – Abstract syntax Notation One (ASN.1): Specification of basic notation.*
[ITU-T X.737]	Recommendation ITU-T X.737 (1995) | ISO/IEC 10040:1998, *Information technology – Open Systems Interconnection – Systems Management: Confidence and diagnostic test categories.*
[ITU-T X.745]	Recommendation ITU-T X.745 (1993) | ISO/IEC 10164-12:1994, *Information technology – Open Systems Interconnection – Systems Management: Test management function.*

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following terms defined elsewhere:

3.1.1 agent [ITU-T M.3020]

3.1.2 information object class [ITU-T M.3020]

3.1.3 manager [ITU-T M.3020]

3.1.4 test category [ITU-T X.745]

3.1.5 test conductor [ITU-T X.745]

3.1.6 test object [ITU-T X.745]

3.1.7 managed object referring to test [ITU-T X.745]

3.2 Terms defined in this Recommendation

This Recommendation has no new terms or definitions.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

EMS	Element Management System
FS	Function Set
ID	Identifier
IOC	Information Object Class
IRP	Integration Reference Point
MORT	Managed Object Referring to Test
NE	Network Element
NMS	Network Management System
OS	Operating System
TO	Test Object

5 Conventions

This Recommendation follows the conventions defined in [ITU-T M.3020].

In this Recommendation, Test management IRP is an interface in an agent providing test management functions.

6 Requirements

6.1 Concepts and background

The test management function is used to assist in various high level management function sets such as fault management or performance management. The purpose of testing is to obtain information about the functionality and performance of the managed network subjected to the test.

Testing is an activity that involves the operator, the managing system (operating system (OS)) and the managed system (network element (NE)). Generally the operator requests the execution of tests from the OS and the managed NE autonomously executes the tests without any further support from the operator.

Based on the descriptions in [ITU-T X.737], there are eight test categories: Connection test, Connectivity test, Data integrity test, Loopback test, Protocol integrity test, Resource boundary test, Resource self-test and Test infrastructure test. This Recommendation reuses the above test categories and provides a test category extension mechanism.

In the case of intrusive tests it is required that the network resources to be tested are locked prior to test execution. During the test execution the telecommunication service provided by the network resources is interrupted. After completion of the test, depending on the test result, the network resources shall be set to the most appropriate state.

Test management capabilities should be provided over the NMS-EMS in order to allow the NMS operator to perform tests on network resources. This capability is especially important at times when only the NMS but not the EMS is attended, which might be the case at night or during weekends.

In the context of fault management the NMS operator may use the testing capabilities over the NMS-EMS interface for numerous purposes:

– When a fault has been detected and if the information provided in the alarm report is not sufficient to localise the faulty resource, tests can be executed in order to localise the fault.
– When a fault has been detected and if the information provided in the alarm report specifies the faulty resource, tests can be executed on that resource in order to determine the required repair action.
– During normal operation of the NE, tests can be executed for the purpose of discovering undetected faults.
– After a faulty resource has been repaired or replaced and before it is restored to service, tests can be executed on that resource in order to make sure that it is fault free.

However, regardless of the context where the testing is used, its target is always the same: verify if a system's physical or functional resource performs properly and, in case it happens to be faulty, provide all the information to help the operator to localise and correct the fault.

The test management IRP bases its design on work referenced in [ITU-T X.745].

6.2 Business level requirements

6.2.1 Requirements

The test management requirements can be grouped into one of the following categories:

– General, i.e., the test management requirements for general purposes, e.g., test identification, test result header format, etc.
– Test object management, e.g., initiate test object, terminate test object, subscribe test object, resume test object, query test object information.
– Test category, e.g., query or specify test types.
– Test result reporting, e.g., report test results.

6.2.1.1 General

The following requirements apply for general purposes.

REQ-TM-FUN-01	An agent shall provide a mechanism for managers to perform test functions on managed networks.
REQ-TM-FUN-02	An agent shall provide a mechanism to identify each test object initiated by a manager.

REQ-TM-FUN-03	It is required that all test results generated by agents support the same header format that contains enough information to identify the test category, the managed entities that are tested and the time stamp at which the test result is generated.

6.2.1.2 Test object management

The following requirements apply to the test management interface.

REQ-TM-FUN-04	A manager shall be able to initiate a test object to be performed in an agent.
REQ-TM-FUN-05	A manager shall be able to query the detailed information of a specified test object in an agent.
REQ-TM-FUN-06	A manager shall be able to suspend the execution of a specified test object in an agent.
REQ-TM-FUN-07	A manager shall be able to resume the execution of a suspended test object in agent.
REQ-TM-FUN-08	A manager shall be able to terminate a specified test object in an agent.
REQ-TM-FUN-09	The agent shall provide managers with the capability to monitor the states of a test object. The test state may assume one of the following values: not initialised, idle, initialising, testing, terminating or disabled. The actual test state value shall be derived from the actual operational state and procedural status according to the mapping table specified in [ITU-T X.745]. Any state change shall be reported to the manager using notifications.

6.2.1.3 Test category discovery

The following requirements apply for test category discovery.

REQ-TM-FUN-10	An agent shall provide managers with the capability to discover the test categories (types) supported by the agent. The possible test categories include but are not limited to: Connection test, Connectivity test, Data integrity test, Loopback test, Protocol integrity test, Resource boundary test, Resource self-test and Test infrastructure test. These test categories are defined in [ITU-T X.737]. When there are new test categories to be used in the test management interface, an agent should provide means for a manager to discover the new test categories.

6.2.1.4 Test result management

The following requirements apply for test result management.

REQ-TM-FUN-11	When predefined trigger events occur, the test result of the test object should be generated and stored in an agent. The test results shall be made available to the manager by one or more notifications. The notifications may contain either the test results themselves or the information of data files which store the test result in the case of a large amount of data. The triggering events depend on the test category and shall be defined in the behaviour of the test object.
REQ-TM-FUN-12	A manager shall be able to specify the destination to which test results should be forwarded by an agent.

REQ-TM-FUN-13	A manager shall be able to request a list of available files related to a specified test object.
REQ-TM-FUN-14	A manager shall be able to manage the transfer of date files contained the test result.

The above requirements are documented in the subsequent use cases.

6.2.2 Actor roles

The capabilities described in this document are available and relevant to all agents and managers.

6.2.3 Telecommunication resources

The test management functionality is applicable to all types of telecommunication resources.

6.2.4 High-level use case diagrams

The first overview use case diagram in Figure 6-1 shows the overall interaction of the notification management interface.

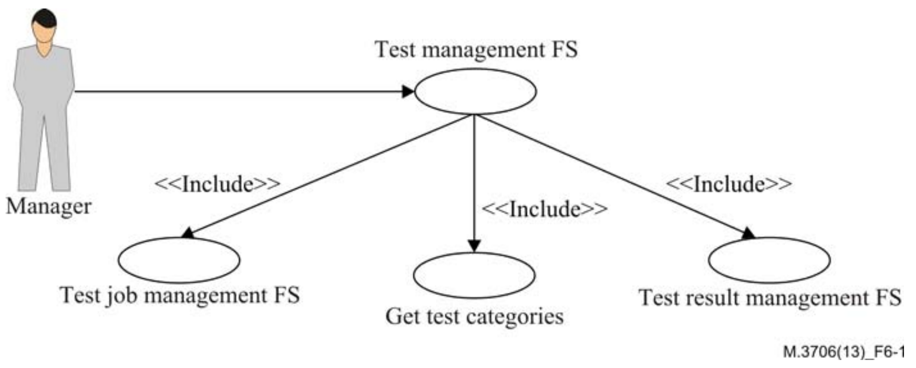


Figure 6-1 – Use case diagram of the test management function set – overview

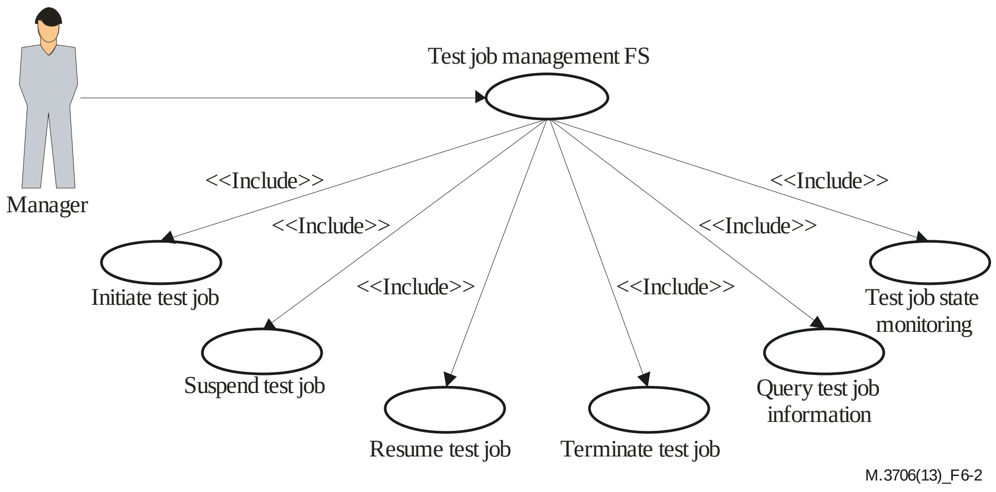


Figure 6-2 – Use case diagram of the test object management function set

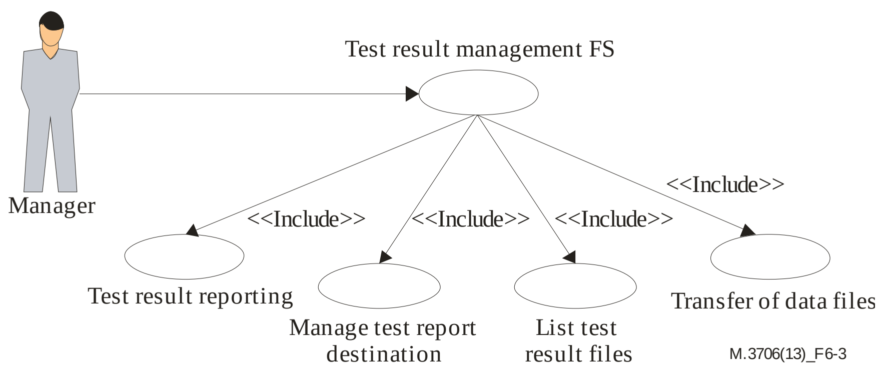


Figure 6-3 – Use case diagram of the test result management function set

6.3 Specification-level requirements

6.3.1 Requirements

There are no specification level requirements.

6.3.2 Actor roles

See clause 6.2.2.

6.3.3 Telecommunications resources

See clause 6.2.3.

6.3.4 Use cases

The general exceptions (e.g., communication error, processing error) related to all uses cases will not be described in the following uses cases and will only be handled in design phases.

6.3.4.1 Initiate test object

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to initiate a test object through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resources	All types of telecommunication resources.	
Assumptions	The communication between the agent and the manager is available.	
Pre-conditions	–	
Begins when	The manager sends a request to the agent to initiate a test object on one or more managed entity instances.	
Step 1	The manager sends a request to the agent to start a test object. The test request may contain: – the ID or criteria for the managed entities (network resources) to be tested, – the test category to be performed, – the test parameters to be performed on the managed entities (optional), – the start time of the test task (optional), – the stop time of the test task (optional).	

Use case stage	Evolution/Specification	<<Uses>> Related use
Step 2.1	If the test object is initiated successfully, the unique test object ID will be returned to the manager, and the agent will start the execution of the test object on the specified network resources according to the parameters of the request. Test result will be generated and are stored in files, and on each reporting interval, the test result information will be reported to the manager.	
Step 2.2	Otherwise, it will return error information to the manager.	
Ends when	Requested information or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	Invalid parameter	
Post-conditions	A test object is initiated on request, and it starts to perform test execution based on the specified request parameters. The agent may send an object creation notification to the manager.	
Traceability	REQ-TM-FUN-01, REQ-TM-FUN-02, REQ-TM-FUN-04	

6.3.4.2 Suspend test object

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to suspend a test object through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resources	All types of telecommunication resources.	
Assumptions	The communication between the agent and the manager is available.	
Pre-conditions	The specified test object exists in the agent and it is not suspended.	
Begins when	The manager sends a request to suspend a test object.	
Step 1	The manager sends a request to the agent to suspend a test measurement job. The request parameter is the identifier of the test object.	
Step 2.1	If the operation succeeds, the test object will no longer perform any test actions until resumed.	
Step 2.2	If the operation fails, it will return error information to the manager.	
Ends when	A success response or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	– unknown test object; – Test object already suspended.	
Post-conditions	The specified test object is suspended on request. The agent may send a state change notification to the manager.	
Traceability	REQ-TM-FUN-06	

6.3.4.3 Resume test object

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to resume a suspended test object through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available.	
Preconditions	The specified test object exists in the agent and it is suspended.	
Begins when	The manager sends a request to resume a test object.	
Step 1	The manager sends a request to the agent to resume a test object. The request parameter is the identifier of the test object.	
Step 2.1	If the operation succeeds, the test object will continue performing the test actions.	
Step 2.2	If the operation fails, it will return error information to the manager.	
Ends when	Requested information or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	– unknown test object; – test object not suspended.	
Post conditions	The specified test object is resumed on request, and it continues to perform the test actions. The agent may send a state change notification to the manager.	
Traceability	REQ-PM-FUN-07	

6.3.4.4 Terminate test object

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to terminate a test object through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available.	
Preconditions	The specified test object exists in the agent and it is suspended.	
Begins when	The manager sends a request to terminate a test object.	
Step 1	The manager sends a request to the agent to terminate a test object. The request parameter is the identifier of the test object.	
Step 2.1	If the operation succeeds, the specified test object will stop working and the related testing resources, including the test result, will be released, and the agent will return success information.	
Step 2.2	Otherwise, it will return error information to the manager.	
Ends when	Requested information or an exception is returned to the manager, or the operation is cancelled by the manager.	

Use case stage	Evolution/Specification	<<Uses>> Related use
Exceptions	– unknown test object; – test object not suspended.	
Post conditions	The specified test object is terminated on request. The agent may send an object deletion notification to the manager.	
Traceability	REQ-PM-FUN-09	

6.3.4.5 Query test object information

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to query the parameter values of a test object through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available.	
Preconditions	The specified test object exists in the agent.	
Begins when	The manager sends a request to query the information of a test object.	
Step 1	The manager sends a request to the agent to query the parameters of a test object. The request parameter is the identifier of the test object. The following parameters can be queried: – the ID or criteria for the managed entities (network resources) to be tested, – the test category to be performed, – the test parameters to be performed on the managed entities, – the start time of the test object, – the stop time of the test object, – the report interval of the test object, – the schedule of the test object, etc.	
Step 2.1	If the operation succeeds, the agent will return the attribute information of the test object.	
Step 2.2	If the operation fails, it will return error information to the manager.	
Ends when	Requested information or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	Unknown test object.	
Post conditions	The corresponding attribute information is returned by the agent as requested.	
Traceability	REQ-PM-FUN-09	

6.3.4.6 Test object state monitoring

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can monitor the state of a test object. The state change of a test object should be reported to the test manager.	
Actor and roles	The manager receives notifications from the agent.	

Use case stage	Evolution/Specification	<<Uses>> Related use
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available, the manager already subscribed notifications from the agent.	
Preconditions	–	
Begins when	The state of a test object is changed.	
Step 1	The agent sends a notification to the manager, indicating the state change of the test object. The notification includes the following parameters: – the identifier of the test object, – the new value of the test object state attribute, which can be one of the following: Not initialized, Idle, Initializing, Suspended, Testing, Terminating, Disabled. The meaning of the above states can be found in [ITU-T X.745], – the timestamp indicating when the state change occurred.	
Ends when	The notification is sent to the manager or an exception occurred.	
Exceptions	Unknown test object.	
Post conditions	The state change notification is received by the manager.	
Traceability	REQ-PM-FUN-09	

6.3.4.8 Query test categories

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to query the supported test categories for a managed entity (MORT).	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available.	
Preconditions	The specified test object exists in the agent and it is suspended.	
Begins when	The manager sends a request to delete a measurement job.	
Step 1	The manager sends a request to the agent to terminate a test object. The request parameter is the identifier of the test object.	
Step 2.1	If the operation succeeds, the specified test object will stop working and the related testing resources, including the test result, will be released, and the agent will return success information.	
Step 2.2	Otherwise, it will return error information to the manager.	
Ends when	Requested information or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	– unknown test object; – test object not suspended.	
Post conditions	The specified test object is terminated on request. The agent may send an object deletion notification to the manager.	
Traceability	REQ-PM-FUN-10	

6.3.4.9 Test result reporting

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	For controlled tests, the results may be emitted as notifications (unsolicited) from the test object to the manager.	
Actor and roles	The manager receives notifications from the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available, the manager already subscribed notifications from the agent.	
Preconditions	–	
Begins when	The result of a test object becomes ready.	
Step 1	The agent sends one or more notifications containing the test results to the manager. The notification includes the following parameters: – the test invocation identifier, – the test outcome, – any other information related to this TO.	
Ends when	The notification(s) is sent to the manager or an exception occurred.	
Exceptions	–	
Post conditions	The test result notification is received by the manager.	
Traceability	REQ-PM-FUN-11	

6.3.4.10 Manage test result destination

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to manage the destinations where the test results should be sent through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available.	
Preconditions	The specified test performer exists in the agent.	
Begins when	The manager sends a request to modify destinations of test results.	
Step 1	The manager sends a request to the agent to modify the destinations of test result to be sent. The request parameter is the following: – identifier of the test action performer, – the destination(s) to be operated, – the operation type, which can be: ADD, REMOVE, REPLACE.	
Step 2.1	If the operation succeeds, it will return successful information, the modified result destination(s) will be recorded in the agent, and the test result will be forwarded to the new destination when ready.	
Step 2.2	Otherwise, it will return error information to the manager.	
Ends when	Response or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	– unknown test performer; – invalid destinations.	

Use case stage	Evolution/Specification	<<Uses>> Related use
Post conditions	The new destination(s) are setup on request. The test performer will forward the test results to the new destinations.	
Traceability	REQ-TM-FUN-12	

6.3.4.11 Query test results

Use case stage	Evolution/Specification	<<Uses>> Related use
Goal	The manager can request the agent to query the test results of one or more test object(s) through the management interface.	
Actor and roles	The manager invokes an operation on the agent.	
Telecom resource	All types of telecommunication resources.	
Assumptions	The communication between the manager and the agent is available.	
Preconditions	The specified test objects exist in the agent.	
Begins when	The manager sends a request to query the test result of one or more test objects.	
Step 1	The manager sends a request to the agent to query the test results of test objects. The request parameter is the identifiers of the test object.	
Step 2.1	If the operation succeeds, the agent will return the test results information of the specified test objects. The output parameters contain a list of the following: – the identifier of a test object, – the test outcome of the test object, – other information related to the test object.	
Step 2.2	If the operation fails, it will return error information to the manager.	
Ends when	Requested information or an exception is returned to the manager, or the operation is cancelled by the manager.	
Exceptions	Unknown test objects.	
Post conditions	The corresponding test results information is returned by the agent as requested.	
Traceability	REQ-TM-FUN-13, REQ-TM-FUN-14	

7 Analysis

7.1 Concepts and background

The system contexts for test management service are shown in Figure 7-1.

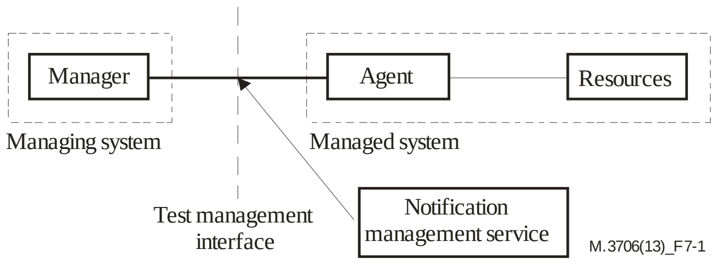


Figure 7-1 – System context for test management service

7.2 Information object classes

7.2.1 Information entities imported and local label

Label reference	Local label
[ITU-T M.3702], information object class, NotificationIRP	NotificationIRP
[ITU-T M.3160], information object class, Top	Top
[ITU-T M.3160], information object class, Network	Network

7.2.2 Class diagram

This clause introduces the set of IOCs that encapsulate information within the agent. The intent is to identify the information required for the test management agent implementation of its operations and notification emission. This clause provides the overview of all support object classes in UML. Subsequent clauses provide more detailed specification of various aspects of these support object classes.

7.2.2.1 Attributes and relationships

Figure 7-2 shows the class containment relationships of the performance management related information object classes.

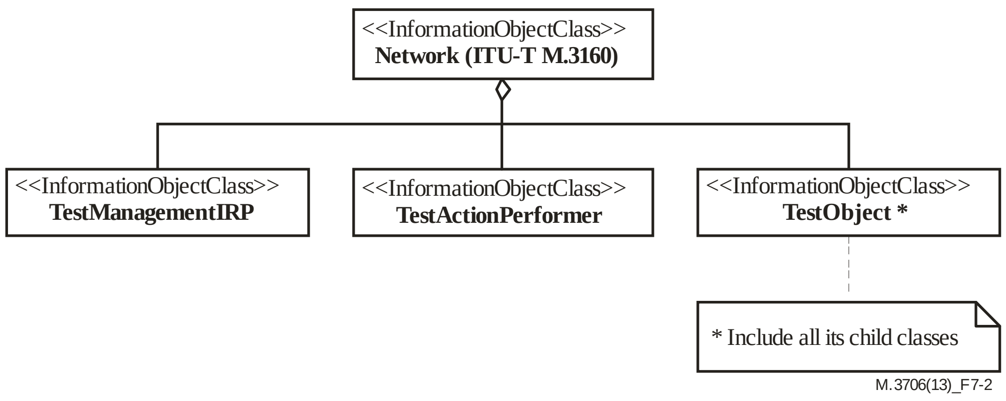
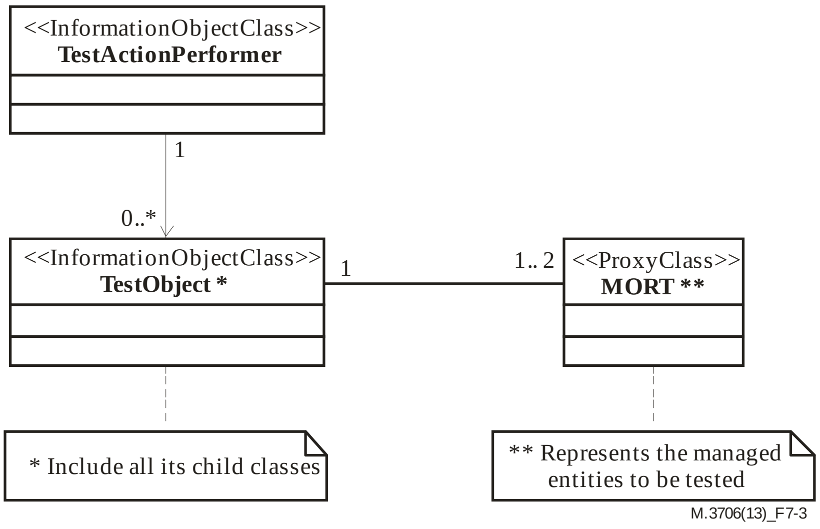

<<InformationObjectClass>>
Network (ITU-T M.3160)
<<InformationObjectClass>>
TestManagementIRP
<<InformationObjectClass>>
TestActionPerformer
<<InformationObjectClass>>
TestObject *
* Include all its child classes
M.3706(13)_F7-2

Figure 7-2 – Containment diagram of test management IOCs

Figure 7-3 shows the detailed class diagram of the information object class TestManagementIRP, TestActionPerformer, TestObject and its child classes, and the association relationships among them and the managed entities which are to be tested (MORTs).

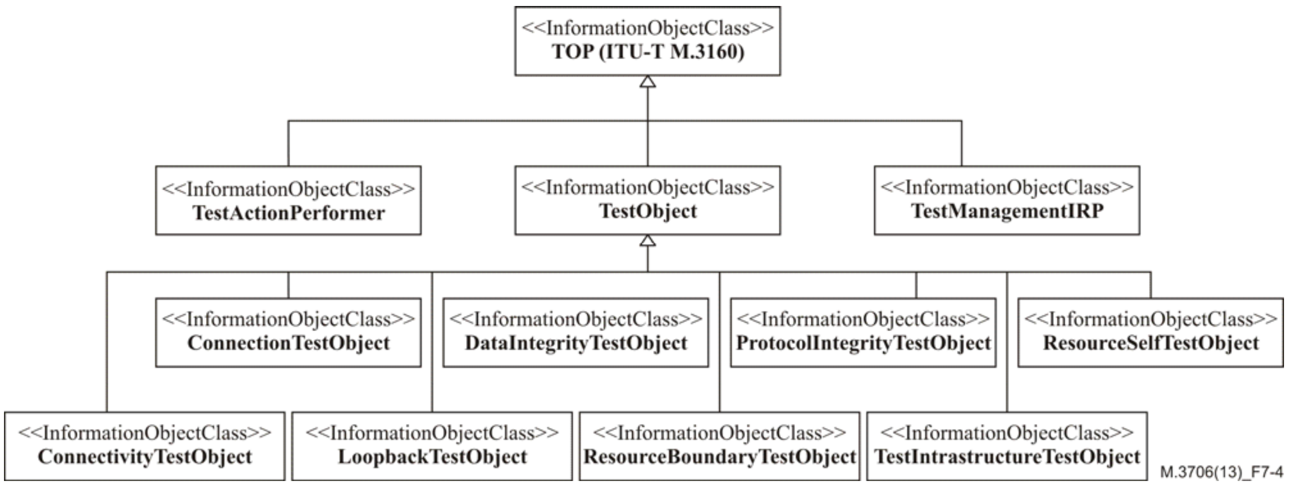


<<InformationObjectClass>>
TestActionPerformer
1
0..*
<<InformationObjectClass>>
TestObject *
1
1.. 2
<<ProxyClass>>
MORT **
* Include all its child classes
** Represents the managed entities to be tested
M.3706(13)_F7-3

Figure 7-3 – Test measurement information object classes

7.2.2.2 Inheritance

Figure 7-4 is the inheritance diagram of test management IOCs.



<<InformationObjectClass>>
TOP (ITU-T M.3160)
<<InformationObjectClass>>
TestActionPerformer
<<InformationObjectClass>>
TestObject
<<InformationObjectClass>>
TestManagementIRP
<<InformationObjectClass>>
ConnectionTestObject
<<InformationObjectClass>>
DataIntegrityTestObject
<<InformationObjectClass>>
ProtocolIntegrityTestObject
<<InformationObjectClass>>
ResourceSelfTestObject
<<InformationObjectClass>>
ConnectivityTestObject
<<InformationObjectClass>>
LoopbackTestObject
<<InformationObjectClass>>
ResourceBoundaryTestObject
<<InformationObjectClass>>
TestIntrastructureTestObject
M.3706(13)_F7-4

Figure 7-4 – Test management IOCs inheritance

7.2.3 Information object class definitions

Class name	Qualifier	Requirement IDs
TestManagementIRP	M	REQ-TM-FUN-01
TestActionPerformer	M	REQ-TM-FUN-01, REQ-TM-FUN-04, REQ-TM-FUN-05, REQ-TM-FUN-06, REQ-TM-FUN-07, REQ-TM-FUN-08, REQ-TM-FUN-09, REQ-TM-FUN-10
TestObject	M	REQ-TM-FUN-01, REQ-TM-FUN-02, REQ-TM-FUN-03, REQ-TM-FUN-11, REQ-TM-FUN-12, REQ-TM-FUN-13
MORT	O	REQ-TM-FUN-01

7.2.3.1 Information Object Class *TestManagementIRP*

7.2.3.1.1 Definition

The IOC *TestManagementIRP* together with the IOC *TestActionPerformer* represent the test management capabilities defined by this Recommendation. To conduct a test of network resources, this object may require capabilities of other objects such as *TestObject*. The IOC *TestManagementIRP* inherits from the IOC *TOP* from [ITU-T M.3160].

7.2.3.1.1 Attributes

The IOC *TestManagementIRP* has no attributes specific to itself, only those inherited from the IOC *TOP*.

7.2.3.2 Information object class *TestActionPerformer*

7.2.3.2.1 Definition

The IOC *TestActionPerformer* provides the ability to receive and react upon test requests. This class must also be able to instantiate and delete tester objects or, in the case where the tester objects are permanently instantiated, to allocate and reserve them for their usage. This Recommendation does not require this IOC to be instantiated. It may be abstract and used for inheritance purposes only. In this way the ability to receive and react upon test requests may be included in any other IOC.

7.2.3.2.1 Attributes

Attribute name	Visibility	Support Qualifier	Read Qualifier	Write Qualifier
`supportedTOClasses`	+	M	M	–
`testActionPerformerId`	+	CM (Note)	M	–
`associatedTOList`	+	M	M	–
NOTE – This attribute is only mandatory in the case where the IOC TestActionPerformer is instantiated. In the case where this IOC is an abstract class and used for inheritance purposes only the attribute shall be omitted.				

7.2.3.3 Information object class *TestObject*

7.2.3.3.1 Definition

The IOC *TestObject* monitors and controls the testing of a *MORT* instance and reports the outcome of the test execution. Tester Objects (TOs) are instantiated by the IOC *TestActionPerformer* in response to a valid test initiation request (*initiateTests*). They are deleted after termination of the test. It is also possible that TOs are permanently instantiated. In this case they are allocated to a certain *TestActionPerformer* during the test execution. After termination of the test they are released.

The IOC *TestObject* defines a generic TO. It shall be used as an abstract class from which more specific tester objects shall be derived by specialisation for each test category. Test categories and the associated test category specific TOs are defined in [ITU-T X.737]. The generic TO defines attributes pertaining to a test and required for all test categories.

Each test invocation shall have only one associated TO.

Only test category specific TOs shall be instantiated.

For simplicity this Recommendation will often use only the term TO. In this case either the test category specific TO is referred to depending on which is actually instantiated.

7.2.3.3.2 Attributes

Attribute name	Visibility	Support Qualifier	Read Qualifier	Write Qualifier
`testOutcome`	+	M	M	–
`testState`	+	M	M	–
`testInvocationInitiator`	–	C	M	–
`additionalInformation`	–	O	–	–
`proposedRepairActions`	–	O	–	–
`fileReference`	–	CM (Note 1)	–	–
`fileExpiryDate`	–	CM (Note 1)	–	–
`testObjectId`	+	CM (Note 2)	M	–
`mortList`	+	M	M	–
NOTE 1 – In the case where the TO does support capturing test results in a file this parameter shall be present and contain information. In the case where the TO does not support capturing test results in a file this parameter shall contain no information or shall be absent. NOTE 2 – This attribute exists only when the IOC is instantiated, and it does not exist for an abstract Class.				

7.2.3.4 Information object class *ResourceSelfTestObject*

7.2.3.4.1 Definition

The IOC *ResourceSelfTestObject* is a specialised TO for the resource self-test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.4.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.5 Information object class ConnectivityTestObject

7.2.3.5.1 Definition

The IOC *ConnectivityTestObject* is a specialised TO for the connectivity test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.5.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.6 Information object class ConnectionTestObject

7.2.3.6.1 Definition

The IOC *Connection*T*estObject* is a specialised TO for the connection test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.6.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.7 Information object class LoopbackTestObject

7.2.3.7.1 Definition

The IOC *LoopbackTestObject* is a specialised TO for the loopback test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.7.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.8 Information object class DataIntegrityTestObject

7.2.3.8.1 Definition

The IOC DataIntegrityT*estObject* is a specialised TO for the data integrity test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.8.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.9 Information object class ResourceBoundaryTestObject

7.2.3.9.1 Definition

The IOC *ResourceBoundaryTestObject* is a specialised TO for the resource boundary test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.9.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.10 Information object class ProtocolIntegrityTestObject

7.2.3.10.1 Definition

The IOC *ProtocolIntegrityTestObject* is a specialised TO for the protocol integrity test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.10.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.11 Information object class TestInfrastructureTestObject

7.2.3.11.1 Definition

The IOC *TestInfrastructureTestObject* is a specialised TO for the test infrastructure test integrity test. It inherits from the abstract IOC *TestObject.* It specifies the triggering events for the emission of the test result notifications.

7.2.3.11.2 Attributes

This IOC has no attributes specific to itself, only those inherited from the generic IOC *TestObject*.

7.2.3.12 Proxy Class *MORT*

7.2.3.12.1 Definition

The ProxyClass *MORT* represents a network resource that is under test. Its class definition shall be one defined in the various Network Resource Model specifications.

7.2.3.12.2 Attributes

This IOC has no attributes.

7.2.4 Information relationships definition

7.2.4.1 Relationship between TestActionPerformer and TestObject

7.2.4.1.1 Definition

This relationship defines a binary association between the IOC *TestActionPerformer* and the IOC *TestObject*. The association is navigable from the *TestActionPerformer* to the *TestObject*.

7.2.4.1.2 Roles

Name	Definition
`associatedTOList`	This rolename provides a name allowing to navigate from an instance of *TestActionPerformer* to the associated instances of *TestObject*. If *tap* is an instance of *TestActionPerformert*, the expression *tap.associatedTOList* indicates the set of object instances of subclasses of *TestObject*.

7.2.4.2 Relationship between *TestObject* and *MORT*

7.2.4.2.1 Definition

This relationship defines a binary association between the IOC *TestObject* and the Proxy Class *MORT*.

The association is navigable from the *TestObject* to the *MORT*.

7.2.4.2.2 Roles

Name	Definition
`mortList`	This rolename provides a name allowing to navigate from an instance of subclasses of *TestObject* to the associated instances of *MORT*. If *to* is an instance of *TestObject*, the expression *to.mortList* indicates an object instance of *MORT*.

7.2.5 Information attributes definition

7.2.5.1 Definition and legal Values

Attribute name	Definition	Information type/Legal values
`testState`	This attribute reflects the actual test state ([ITU-T X.745]).	ENUM {notInitialized, idle, initializing, testing, terminating, disabled}
`testOutcome`	This attribute provides information about the test result, as perceived by the associated TO, in a standardised manner. The information in this parameter is only valid after termination of the test activity. This information shall be present in the last test result notification emitted by a TO prior to its deletion.	ENUM {pass, fail, inconclusive, timed-out, premature-termination} Pass indicates that the test exercise of the test invocation has executed correctly and has found no problem. Fail indicates that the test exercise of the test invocation has executed correctly and has found one or more problems. Inconclusive indicates that the TO has not determined if the execution is Pass or Fail. Timed-out indicates that the TO has terminated its execution because of the expiry of the timer (i.e., the current time – TestSession.sessionStartTime >= TestObject.timeOut). Premature termination indicates that the TO has (a) never started execution or (b) terminated its execution prematurely, either by *TestManagementIRP* and its associated objects internal problems or in response to a *terminateTests* operation.

Attribute name	Definition	Information type/Legal values
`supportedTOClasses`	This attribute identifies the TO classes that are supported by a certain managed object instance whose class has inherited from *TestActionPerformer* or whose class is the *TestActionPerformer*.	SET OF TO String
`testActionPerformerId`	This attribute unambiguously identifies an instance of a *TestActionPerformer*.	String
`TestObjectId`	This attribute unambiguously identifies an instance of a *TestObject*.	String
`testInvocationInitiator`	It identifies the IRPManager.	String
`additionalInformation`	This attribute holds a set of additional information pertaining to the test.	String
`proposedRepairActions`	This attribute suggests one or more repair actions if the reason for a failure is known.	SET of String
`actualStartTime`	This attribute specifies the time at which the TO will enter or has entered the test state *testing*. Before the TO enters the testing state this is an estimated time. After entering the testing state this is the actual time. Note that this is not the time of the invocation of the operation *initiateTests*.	All values indicating a valid time.
`actualStopTime`	This attribute specifies the time at which the TO will leave or has left the test state *testing*. Before the TO leaves the testing state this is an estimated time. After leaving the testing state this is the actual time. Note that this is not the time of the invocation of the operation *terminateTests*.	All values indicating a valid time later than the *actualStartTime*.
`maxTestingPhaseDuration`	This attribute specifies the maximum amount of time that a TO may spend in the testing state.	All values indicating a valid amount of time.
`fileReference`	This attribute carries the reference to a file that contains the test result data set.	String
`fileExpiryDate`	This attribute carries the date and time after which the file, whose reference is carried by the *fileReference* attribute, may be removed.	GeneralizedTime All values indicating a valid time.
`testSourceAddress`	This attribute provides the source address of the test.	An IP address indicating source.
`testDestinationAddress`	This attribute provides the destination address of the test.	An IP address indicating destination. In a loopback test, it should be NULL.
`testLoopbackAddress`	This attribute provides the loopback address in a loopback test.	An IP address indicating loopback point in a loopback test. In connection test, it should be NULL.
`mortList`	This attribute provides DN List of the MORT object instances.	SET of DN
`associatedTOList`	This attribute provides DN List of the TO instances that are associated with this *TestActionPerformer.*	SET of DN

7.2.5.2 Constraints

None.

7.3 Interface definition

7.3.1 Class diagram representing interfaces

The following diagram depicts the interfaces of the test management IRP with their corresponding operations and notifications.

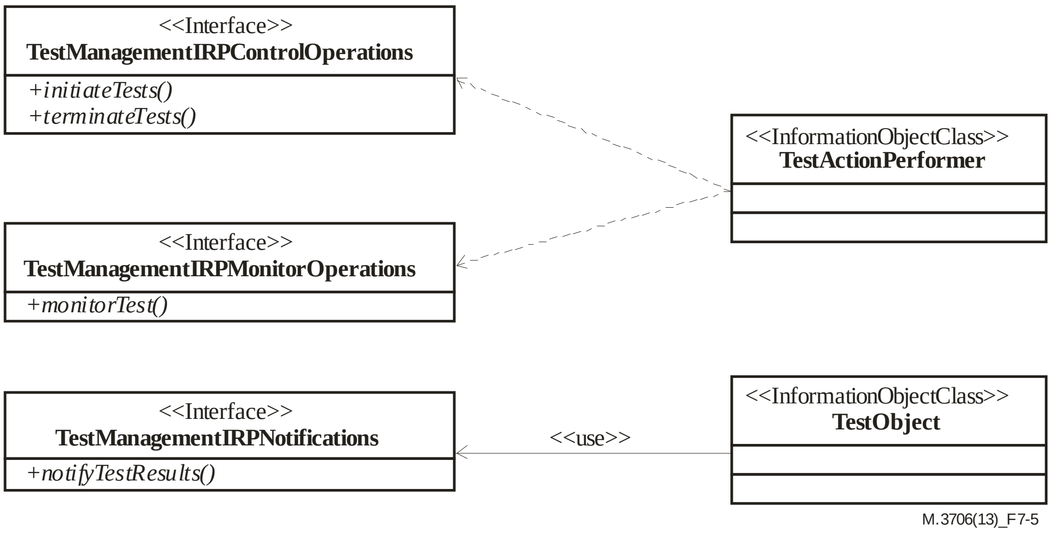


Figure 7-5 – Operations and notifications of TestManagementIRP

7.3.2 Generic rules

Rule 1: Each operation with at least one input parameter supports a pre-condition valid_input_parameter which indicates that all input parameters shall be valid with regard to their information type. Additionally, each such operation supports an exception operation_failed_invalid_input_parameter which is raised when pre-condition valid_input_parameter is false. The exception has the same entry and exit state.

Rule 2: Each operation with at least one optional input parameter supports a set of pre-conditions supported_optional_input_parameter_xxx where "xxx" is the name of the optional input parameter and the pre-condition indicates that the operation supports the named optional input parameter. Additionally, each such operation supports an exception operation_failed_unsupported_optional_input_parameter_xxx which is raised when (a) the pre-condition supported_optional_input_parameter_xxx is false and (b) the named optional input parameter is carrying information. The exception has the same entry and exit state.

Rule 3: Each operation shall support a generic exception operation_failed_internal_problem which is raised when an internal problem occurs and the operation cannot be completed. The exception has the same entry and exit state.

7.3.3 Interface testManagementIRPControlOperations

The interface *TestManagementIRPControlOperations* contains the operations *initiateTests*, *terminateTests*, *suspendTest*, and *resumeTest*. It must be implemented by every object with the ability to receive and react upon test requests, for example by every instance of *TestActionPerformer*.

7.3.3.1 Operation *initiateTests* (M)

7.3.3.1.1 Definition

The IRPManager uses this operation to request the IRPAgent to initiate controlled tests. A single test request may initiate multiple (one or more) tests.

For each test to be initiated the managed object representing the network resource to be tested and the tester object class must be specified.

The initiated tests are independent and not related to each other. This implies that independent test result notifications are sent for each of the tests initiated by s single *initiateTests* operation.

7.3.3.1.2 Input parameters

Parameter name	Qualifier	Information type/ Legal values	Comment
`testInvocationInitiator`	C	TestObject. testInvocationInitiator	This parameter identifies the IRPManager.

`testsToBeInitiated`	M	SET OF SET {
		maxTestingStateDuration (O)
		toBeTestedMORT (O)
		TestObjectClass (M)
		TestObjectName (O)
		TestObjectInitialAttributeList (O)
		}
			This sequence specifies the tests to be initiated.
			For each test the parameter *maxTestingStateDuration* specifies the timeout period of the tests to be initiated. One value shall indicate "No limit".
			toBeTestedMORT specifies the instance of the *MORT* to be tested. If the parameter is absent, the *MORT* is identical to the object instance to which the subject operation is directed to.
			The parameter *TestObjectClass* specifies the class of the associated tester object.
			Optionally, a name for the tester object instance may be specified in the parameter *TestObjectName*.
			The parameter *TestObjectInitialAttributeList* carries some or all the values of the attributes of the TO instance responsible for the test. The syntax and semantics of this attribute value is dependent on the specific TO class definition and is outside the scope of ITU.

7.3.3.1.3 Output parameters

Parameter name	Qualifier	Matching information	Comment

`response`	M	SEQUENCE OF CHOICE {
		testInitiated
		testNotInitiated
		}
		
		testInitiated = SEQUENCE {
		TestInvocation.testInvocationId (M)
		TestObjectName (O)
		}
		
		testNotInitiated =
		failureReason
			The number and the order, related to the tests to be initiated, of elements in this sequence and in the set of the input parameter *toBeInitiatedTests* shall be identical.
			For a successfully instantiated test the parameter *testInitiated* returns the test invocation identifier of the test. In the case where the tester object name has not been specified in the request it shall be returned in *TestObjectName*.
			For a failed test instantiation the parameter *testNotInvoked* returns the reason for which the instantiation of the test failed.
			Failure reasons are
			TO class does not exist
			MORT does not exist
			MORT is not available
			others.

7.3.3.1.4 Pre-condition

The precondition must hold true before the operation is invoked. The pre-condition depends on the test category.

For at least one of the specified tests to be instantiated the following must hold true:

theIndicatedMORTIsExisting **AND** theIndicatedMORTIsAvailable **AND** theIndicatedTOClassIsExisting

Assertion name	Definition
`theIndicatedMORTIsExisting`	The MORT indicated by the subject operation for this test exists.
`theIndicatedMORTIsAvailable`	The MORT indicated by the subject operation for this test is available.
`theIndicatedTOClassIsExisting`	The TO class indicated by the subject operation for this test exists.

7.3.3.1.5 Post-condition

The post-condition must hold true after the completion of the operation:

allIndicatedTOsInstantiated **OR** notAllTestsInitiated **OR** noTestInitiated

Assertion name	Definition
`allTestsInitiated`	All tests indicated by the subject operation were initiated successfully.
`notAllTestsInitiated`	Not all but at least one test indicated by the subject operation was initiated successfully.
`noTestInitiated`	No test indicated by the subject operation was initiated successfully.

7.3.3.1.6 Exceptions

Exception name	Definition
`operationFailedEntirely`	**Condition:** noTestInitiated = TRUE **Returned information:** The response parameter is returned **Exit state:** Entry state
`operationFailedPartly`	**Condition:** notAllTestsInitiated = TRUE **Returned information:** The response parameter is returned **Exit state:** Entry state

7.3.3.2 Operation *terminateTests* (M)

7.3.3.2.1 Definition

The IRPManager uses this operation to request the IRPAgent to terminate tests during their life time. A single *terminateTests* operation may terminate multiple (one or more) tests.

The tests to be terminated are identified by their test invocation identifiers. The IRPManager terminating a test may be different from the IRPManager that initiated the test. The *terminateTests* operation must be invoked on the object which received the corresponding *initiateTests* operation.

7.3.3.2.2 Input parameters

Parameter name	Qualifier	Information type	Comment
`testsToBeTerminated`	M	SET OF TestInvocation.testInvocationId	This parameter specifies the tests that shall be terminated.

7.3.3.2.3 Output parameters

Parameter name	Qualifier	Matching information	Comment

`response`	M	SEQUENCE OF CHOICE {
		testTerminated
		testNotTerminated
		}
		testTerminated =
		TestInvocation.testInvocationId
		testNotTerminated =
		SEQUENCE {
		TestInvocation.testInvocationId,
		failureReason
		}
			The number and the order, related to the test invocation identifier, of elements in this sequence and in the set of the input parameter *toBeTerminatedTests* shall be identical.
			It specifies the test invocation ids of the tests, that were successfully terminated, and the ids of the tests, that failed to be terminated successfully together with the reason for the failure.
			Failure reasons are
			test invocation id does not exist,
			others.

7.3.3.2.4 Pre-condition

The precondition must hold true before the operation is invoked.

allIndicatedTestInvocationIdsAreExisting **OR** notAllIndicatedTestInvocationIdsAreExisting

Assertion name	Definition
`allIndicatedTestInvocationIdsAreExisting`	All test invocation identifiers specified by the subject operation exist.
`notAllIndicatedTestInvocationIdsAreExisting`	Not all but at least one test invocation identifier specified by the subject operation exists.

7.3.3.2.5 Post-condition

The post-condition must hold true after the completion of the operation.

allIndicatedTestsTerminated **OR** notAllIndicatedTestsTerminated **OR** noIndicatedTestTerminated

Assertion name	Definition
`allIndicatedTestsTerminated`	All tests indicated in the subject operation were terminated successfully.
`notAllIndicatedTestsTerminated`	Not all but at least one test indicated in the subject operation aaws terminated successfully.
`noIndicatedTestTerminated`	No test indicated in the subject operation aaws terminated successfully.

7.3.3.2.6 Exceptions

Exception name	Definition
`operationFailedEntirely`	**Condition:** noIndicatedTestTerminated = TRUE **Returned information:** The response parameter is returned **Exit state:** Entry state
`operationFailedPartly`	**Condition:** notAllIndicatedTestInvocationIdsAreExisting = TRUE **OR** notAllIndicatedTestsTerminated = TRUE **Returned information:** The response parameter is returned **Exit state:** Entry state

7.3.3.2 Operation *suspendTest* (M)

7.3.3.2.1 Definition

The IRPManager uses this operation to request the IRPAgent to suspend a test represented by an instance of the derived class of abstract IOC TestObject.

The test to be suspended is identified by the DN of the test object. The IRPManager suspending a test may be different from the IRPManager that initiated the test. The *suspendTest* operation must be invoked on the object which received the corresponding *initiateTests* operation.

7.3.3.2.2 Input parameters

Parameter name	Qualifier	Information Type	Comment
`testObjectToBeSuspendeded`	M	Name (DN of a TestObject instance)	This parameter specifies the Test Object that shall be suspended.

7.3.3.2.3 Output parameters

Parameter name	Qualifier	Matching Information	Comment
`successIndicator`	M	boolean	The successIndicator indicates whether the suspendTest operation performs successfully or failed.

7.3.3.2.4 Pre-condition

The precondition must hold true before the operation is invoked.

Assertion name	Definition
`specifiedTestObjectExists`	The test object DN specified by the subject operation exists.
`spefifiedTestNotSuspended`	The test object DN specified by the subject operation is not suspended.

7.3.3.2.5 Post-condition

The post-condition must hold true after the completion of the operation.

Assertion name	Definition
`specifiedTestSuspended`	The test indicated in the subject operation was suspended successfully.

7.3.3.2.6 Exceptions

Exception name	Definition
`UnknownTestObject`	**Condition:** `specifiedTestObjectExists` = FALSE **Returned information:** The response parameter is returned **Exit state:** Entry state
`SpecifiedTestObjectAlreadySuspended`	**Condition:** `specifiedTestObjectSuspended` = FALSE **Returned information:** The response parameter is returned **Exit state:** Entry state

7.3.3.3 Operation resume*Test* (M)

7.3.3.3.1 Definition

The IRPManager uses this operation to request the IRPAgent to resume a suspended test represented by an instance of the derived class of abstract IOC TestObject.

The test to be terminated is identified by their test invocation identifiers. The IRPManager resuming a test may be different from the IRPManager that initiated the test. The *resumeTest* operation should be invoked on the object which is already suspended.

7.3.3.3.2 Input parameters

Parameter name	Qualifier	Information type	Comment
`testObjectToBeResumed`	M	Name (DN of a TestObject instance)	This parameter specifies the test object that shall be resumed.

7.3.3.3.3 Output parameters

Parameter name	Qualifier	Matching information	Comment
`successIndicator`	M	Boolean	The successIndicator indicates whether the resumeTest operation performs successfully or failed.

7.3.3.3.4 Pre-condition

The precondition must hold true before the operation is invoked.

Assertion name	Definition
`specifiedTestObjectExists`	The test object DN specified by the subject operation exists.
`specifiedTestObjectSuspended`	The test object DN specified by the subject operation is suspended.

7.3.3.3.5 Post-condition

The post-condition must hold true after the completion of the operation.

Assertion name	Definition
`specifiedTestResumed`	The test indicated in the subject operation was resumed successfully.

7.3.3.3.6 Exceptions

Exception name	Definition
`UnknownTestObject`	**Condition:** `specifiedTestObjectExists` = FALSE **Returned information:** The response parameter is returned **Exit state:** Entry state
`SpecifiedTestObjectAlreadyResumed`	**Condition:** `specifiedTestObjectSuspended` = FALSE **Returned information:** The response parameter is returned **Exit state:** Entry state

7.3.4 Interface TestManagementIRPMonitorOperations

The interface *TestManagementIRPMonitorOperations* contains the operation *monitorTest*. It has a realisation relationship with the IOC *TestActionPerformer*.

7.3.4.1 Operation *monitorTest* (M)

7.3.4.1.1 Definition

IRPManager shall be able to retrieve information about the test as observed by the TO during the test execution by reading the relevant attributes of the TO associated with the test. Also after the test execution the manager shall be able to read these attributes as long as the TO exists. Attributes conveying information about the test execution are *testState* and *testOutcome*. Depending on the specific test category specific TO other attributes may also contain information about the test execution. In this case the subject operation may also allow the values of these attributes to be read.

7.3.4.1.2 Input parameters

Parameter name	Qualifier	Information type	Comment
`testObject`	M	Name	This parameter specifies the instance of the tester object, whose attribute values of *testState*, *testOutcome* and other attributes shall be retrieved.

7.3.4.1.3 Output parameters

Parameter name	Qualifier	Matching information	Comment

`monitoredAttributeValues`	M	SET {
		TestObject.testState (M)
		TestObject.testOutcome (M)
		other attributes (O)
		}
			This parameter shall be returned if all attributes were read successfully and may be returned, if at least one attribute was read successfully.
			The values to be returned are those prevalent at the time of the reception of the subject operation.

`error`	M	failureReason	This parameter shall be returned if the specified tester object instance does not exist or, in the case where the tester object instance exists, at least one attribute could not be read, i.e., if operationFailedEntirely = TRUE OR operationFailedPartly = TRUE The parameter returns the failure reason.

7.3.4.1.4 Pre-condition

The precondition must hold true before the operation is invoked.

indicatedTOInstanceIsExisting

Assertion name	Definition
`toBeMonitoredTOIsExisting`	The TO instance indicated by the subject operation exists.

7.3.4.1.5 Post-condition

The post-condition must hold true after the completion of the operation.

allAttributeValuesRead **OR** notAllAttributeValuesRead **OR** noAttributeValueRead

Assertion name	Definition
`allAttributeValuesRead`	All attributes of the TO indicated by the subject operation were read successfully.
`notAllAttributeValuesRead`	Not all but at least one attribute of the TO indicated by the subject operation were read successfully.
`noAttributeValueRead`	No attribute of the TO indicated by the subject operation was read successfully.

7.3.4.1.6 Exception

Exception name	Definition
`operationFailedEntirely`	**Condition:** toBeMonitoredTOIsExisting = FALSE **OR** noAttributeValueRead = TRUE **Returned information:** The error parameter returns the object identifier of the TO that does not exist or the reasons for which the attributes could not be read **Exit state:** Entry state
`operationFailedPartly`	**Condition:** toBeMonitoredTOIsExisting = TRUE **AND** notAllAttributeValuesRead = TRUE **Returned information:** The error parameter returns the reason for which an attribute could not be read. The attribute that could be read my be returned in the parameter *error* or the parameter *attributeList* **Exit state:** Entry state

7.3.5 Interface TestManagementIRPNotifications

7.3.5.1 Notification *notifyTestResults* (M)

7.3.5.1.1 Definition

Test results are made available to the IRPManager by one or more notifications *notifyTestResults* emitted by the TO that is related to the test invocation.

Depending on the nature of the test and the specification of the TO behaviour, the TO may need to convey to the IRPManager a test result data set. There are two ways to convey this kind of information. One way is to use the parameter *additionalInformation* of the notification. In this case, the *fileReference* and *fileExpiryDate* shall contain no information or be absent. The other way is to use a file to capture the test result data set. In this case, the *additionalInformation* parameter may contain no information or be absent and the *fileReference* and *fileExpiryDate* shall be present. The file that captures the test result data set shall contain specific attributes and other attributes such as *TestObjectClass*, *testOutcome*, etc.

The use of the *additionalInformation* parameter or a file to capture the test result data set is specified by the class specification of the TO.

In case the TO uses *additionalInformation* (and not a file) to capture the test result data set, that TO may emit this notification to transfer intermediate (non-final) test results. In this kind of notifications, the *testOutcome* parameter shall be absent. The TO should emit at least one more notification regarding the subject test invocation in the future. The last notification pertaining to a particular test invocation shall be indicated by including the *testOutcome* parameter in the notification.

In the case the TO uses a file to capture the test result data set, that TO shall not issue any notifications to transfer intermediate test results. The TO may capture the non-final test results in the file used to capture the final test result data set.

The events triggering the emission of test result notifications depend on the specific test. They shall be specified by the TO that is actually instantiated, i.e., either by the test category specific TO. Some generic triggering events are included in this specification. It is expected that vendors specify more triggering events.

Parameter name	Qualifier	Matching information	Comment
`objectClass`	M, Y	TestObject.objectClass	Notification header – see [1]. It shall carry the TO class name.
`objectInstance`	M, Y	TestObject.objectInstance	Notification header – see [1]. It shall carry the DN of the TO.
`notificationId`	O, N	–	Notification header – see [1].
`eventTime`	M, Y	–	Notification header – see [1].
`systemDN`	C, Y	–	Notification header – see [1].
`notificationType`	M, Y	"notifyTestResults"	Notification header – see [1].
`testInvocationInitiator`	C, Y	TestObject.testInvocationInitiator	
`testOutcome`	O, N (Note)	TestObject.testOutcome	It shall be included only in the last notification emitted by a TO. In this way the TO indicates that it is sending no more notifications.
`mORT`	O, N	TestObject.theMORT	It identifies the object instance of the MORT that was subject to the test.
`proposedRepairActions`	O, N	TestObject.proposedRepairActions	
`additionalInformation`	O, N (Note)	TestObject.additionalInformation	It allows the inclusion of any additional information in the notification. As such, it may carry a test result data set. The exact semantics of this parameter is outside the scope of this specification. This parameter may contain no information or be absent, if the test results are captured in a file. It may be present if the test results are not captured in a file.
`fileReference`	M, N (Note)	TestObject.fileReference	It shall contain no information or be absent if there is no test result captured in a file. It shall contain information if the test results are captured in a file.
`fileExpiryDate`	M, N (Note)	TestObject.fileExpiryDate	It shall contain no information or be absent if fileReference carries no information or absent. Otherwise, it shall contain a valid future date and time.
NOTE – As for the correct interpretation of this qualifier refer to the comment column.			

7.3.5.1.2 Triggering Events for the test

For the test the events triggering the emission of test result notifications are:

- Termination of the test execution.

The test may be terminated explicitly by a test termination request. The events triggering an implicit termination are:

- Fulfilment of the conditions for a successful termination of the test.
- Fulfilment of the conditions for a premature termination of the test.
- Occurrence of an error situation.

7.3.5.1.2.1 From-State

testTerminateRequestReceived OR testCompleted OR prematureTermination OR testTimedOut OR errorSituationOccured.

Assertion name	Definition
`testTerminateRequestReceived`	The object with the ability to receive and react upon test requests has received a test termination request (Note).
`testCompleted`	The predefined conditions for a successful completion of the test are fulfilled (Note).
`prematureTermination`	The predefined conditions for a premature termination of the test are fulfilled (Note).
`errorSituationOccured`	An error situation has occurred during the test execution and the tester object has aborted the test invocation (Note).
NOTE – The conditions to satisfy this trigger are related to the specific TO definition and therefore their specifications are outside the scope of this Recommendation.	

7.3.5.1.2.2 To-State

testTerminated

Assertion name	Definition
`testTerminated`	The test has been terminated successfully.

Bibliography

[b-3GPP TS 32.322] 3GPP TS 32.322 V11.0.0 (2012), *Telecommunication management; Test Management Integration Reference Point (IRP); Information Service (IS) (Release 11)*.

Printed in Switzerland
Geneva, 2014