INTERNATIONAL STANDARD

ISO/IEC 19637

First edition
2016-12-01

Information technology — Sensor network testing framework

Technologies de l'information — Cadre général pour les essais de réseaux de capteurs

Reference number
ISO/IEC 19637:2016(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*.

Introduction

Sensor network is widely used around the world in multiple fields such as industrial automation, environment monitoring, smart home, intelligent health-care and smart grid. The applications can involve different devices supplied by different manufacturers, e.g. sensors, actuators, controllers, routers and gateways, etc. Data can be collected and processed by use of different wired/wireless communication technologies. Thus, various test systems should be employed to satisfy some specific requirements. The operations of test systems is a challenge to users, if without a uniform test platform.

When designing and developing a sensor network test system, the characteristics regarding the following aspects should be considered:

a) Sensor network heterogeneity. It is necessary to verify the interoperability of sensor networks based on different protocols prior to system application;

b) Diversity of sensor network applications.

However, an international test standard for sensor networks which can provide guidance to design and develop a uniform platform integrating different tests for sensor networks is still unavailable.

Information technology — Sensor network testing framework

1 Scope

This document specifies:

— testing framework for conformance test for heterogeneous sensor networks,

— generic services between test manager (TMR) and test agent (TA) in the testing framework, and

— guidance for creating testing platform and enabling the test of different sensor network protocols.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1
analysis module
logical unit within a testing application process which is used to analyse the information from test agent and module-based test depending on a particular strategy

3.2
test agent
device designed for different sensor network protocols or various kinds of hardware that can communicate directly with the test manager and the systems under test

3.3
testing application process
software functional entity that performs the processing by combining test modules, analysis modules and report modules to fulfil test purposes

Note 1 to entry: It is an application platform that supervises various operational aspects of testing activities and entities, usually through interaction with test agents.

3.4
test module
logical unit within a testing application process that performs operations depending on a specified testing requirements

3.5
testing platform
testing entity that could integrate different test systems for different protocols and technologies

EXAMPLE A platform can provide IPv4 and IPv6 conformance testing systems.

3.6
test report
logical unit of software within a testing application process that produces documents at the end of test assessment

3.7
view object
logical element provided to support efficient access to data within a testing or analysis module

Note 1 to entry: It is visible on graphic user interfaces.

4 Abbreviated terms

TE	auxiliary testing equipment
DUT	device under test
ICS	implementation conformance statement
IUT	implementation under test
MIB	management information base
OD	object dictionary
SAPs	service access points
SNTF	sensor network testing framework
SUT	system under test
TA	test agent
TAP	testing application process
TDSs	testing data services
TM	test module
TMR	test manager
TMSs	testing management services
TP	test purpose
VO	view object

5 Overview of a sensor network testing framework (SNTF)

5.1 Testing requirements of sensor networks

Varied with applications, sensor networks are likely employ different communication technologies and protocols. To ensure their interconnection and operation, it is required to deploy a great number of test systems, which are designed to test individual technologies and protocols with regard to protocol conformance. Therefore, it is a challenging task to integrate the variety of the test systems for sensor networks.

It is difficult to manage different test systems, make them work together and execute test cross various technologies and protocols for specific user requirements. Various application systems need different communication interfaces and protocols, which can't interact directly with each other.

With changing testing requirements, test system for sensor networks should be scalable and adaptable on demand. For example, when new sensor networks are added into the application, the corresponding test systems should be integrated into the test platform with low costs.

Annex A describes an example of a testing platform for hybrid sensor networks based on IPv6.

5.2 Conceptual model of SNTF

Figure 1 shows the concept model of SNTF. The framework consists of three parts: test manager (TMR), test agent (TA) and system under test (SUT). As an actual management controller, the TMR conducts a test indirectly by controlling the TA. The test activities of the TMR should be converted into unified services and transferred to the TA. After having processed the testing services from TMR, TA conducts test interactions directly with SUT. Therefore, TA bridges TMR and SUT while TA needs to be equipped with the specific physical communication interface and protocol stack as in SUT.

The SUT is a system that can include only one device under test (DUT), or a DUT and some other devices used to activate the behaviours of the specific protocol residing in DUT, which is named auxiliary testing equipment (ATE). DUT needs to be verified for having certain required protocol implementations. SUT contains points of control and observation at the upper or lower service boundary of the protocol implementations that resides in DUT to execute tests. The protocol implementations during testing is called implementations under test (IUT). Before starting performing the conformance tests, IUT should be configured by instructions from TMR. In a complicated environment, ATE can be used to stimulate the DUTs to ensure TAs observe the expected responses from DUT if the IUT can't activate some particular behaviours of protocol by itself.

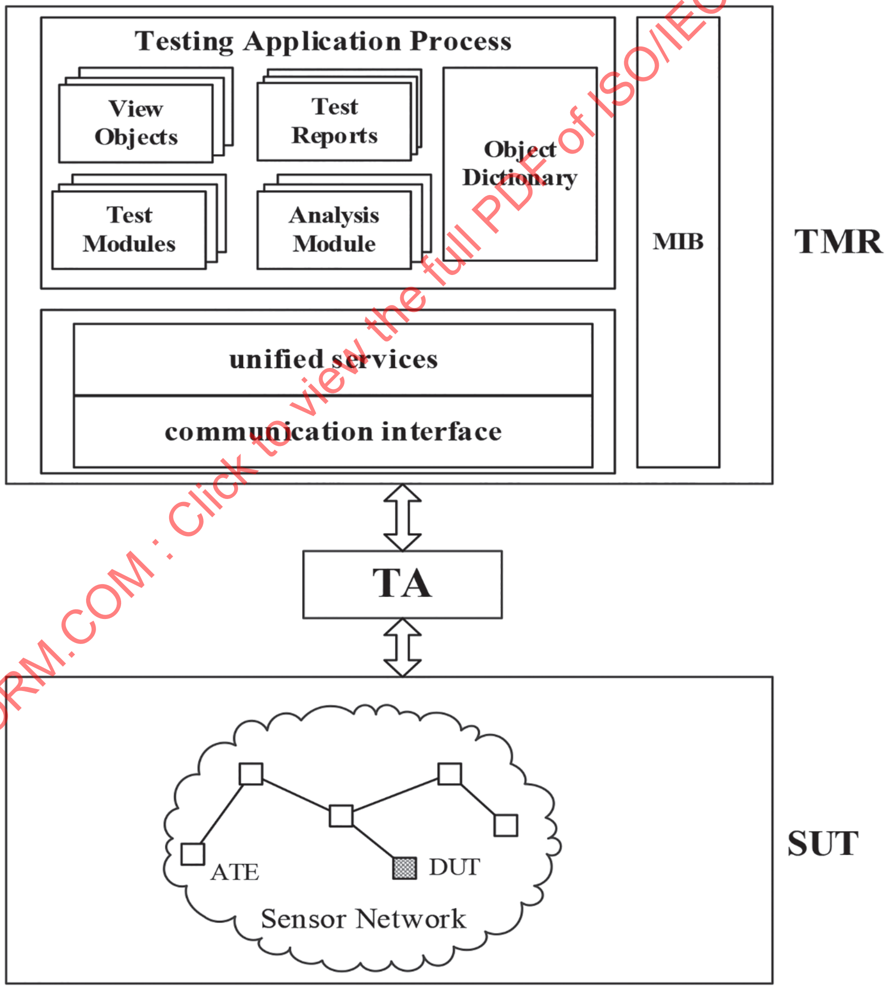

Figure 1 — Conceptual model of SNTF

5.3 Description of test manager (TMR)

The TMR can support several testing application processes (TAP). Each TAP has a specific test task to be performed. For example, it can create an application process for conformance testing of a particular protocol, and simultaneously another application process for a different protocol.

The testing application process is designed on a component basis. There are five types of components in a TAP:

— view object,

— test report,

— test module,

— analysis module, and

— object dictionary (OD).

Figure 2 shows the relationship of the five components.

The test module (TM) is the performer of test cases. Before executing a test, TM should be parameterized to configure the test type for a specific protocol. If the corresponding testing requirements and testing suits are inputted into a TM, it is instantiated. After the TM is started, the steps defined by test cases is executed.

The analysis module (AM) shall connect test modules to collect data to analysis test activities. It is needed to configure the corresponding report format in the AM before producing test reports. Moreover, AM can also collect the information of the operating conditions of the TM.

The view object (VO) has parameter sets from the TMs and AMs. VO can be selected for graphical user interfaces, which allows monitoring the real-time throughput during a certain period. VOs also can be grouped to monitor the parameters in the relevant TMs and AMs; its visual parameters can also be derived from different TMs or AMs.

Management information base (MIB) stores all object values of test module, view object and test report in TMR, including the management objects for TA. The values of the objects in testing application process can be indexed quickly from object dictionary (OD).

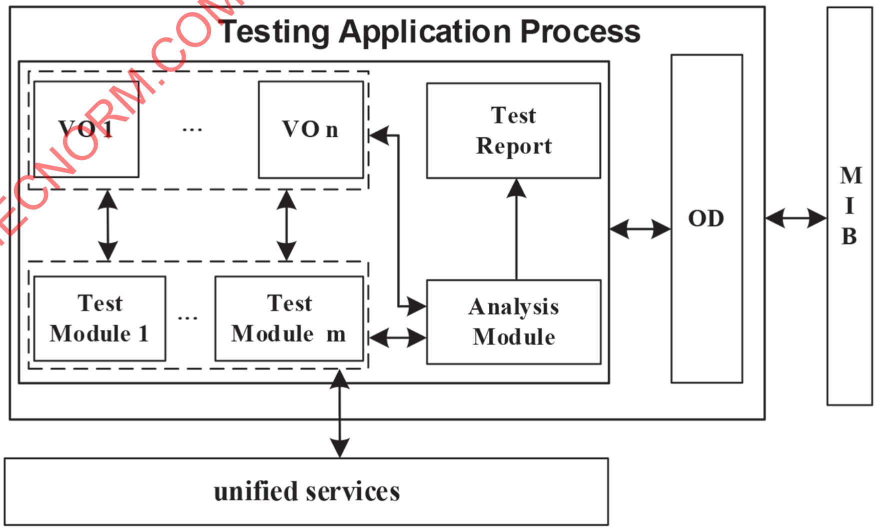

Figure 2 — Relationship of components in TAP

5.4 Description of test agent (TA)

A test agent (TA) is designed for individual protocol, and it can support multiple implementations of the same protocol. The TA handles any direct communications with the SUT with the same protocol through the test driver. Platform independence can be achieved by the test driver. The configuration parameters of test driver are defined in the TA description.

Before performing the corresponding test operations, TA shall establish communication connection with TM in TMR when receiving the start command from a TM. TA receives services from TM and translates them into the appropriate messages in conformance with the protocol of SUT. Structure model of TA is shown in Figure 3.

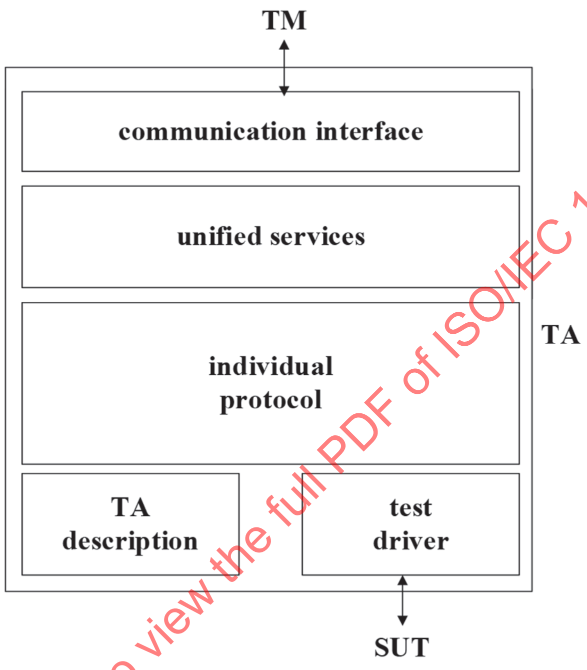

Figure 3 — Structure of TA

6 Testing services

6.1 General

Testing services are conceptually divided into two classes: testing data services (TDSs) and testing management services (TMSs) as shown in Figure 4. TMSs could be used to create application communication relationships or set the parameters of TA via Management Entry-Service Access Points (ME-SAPs). TDSs should be used to implement testing procedures between TMR and TA via Data Entry-Service Access Points (DE-SAPs). TMR can transfer testing data to TA and control TA testing behaviours.

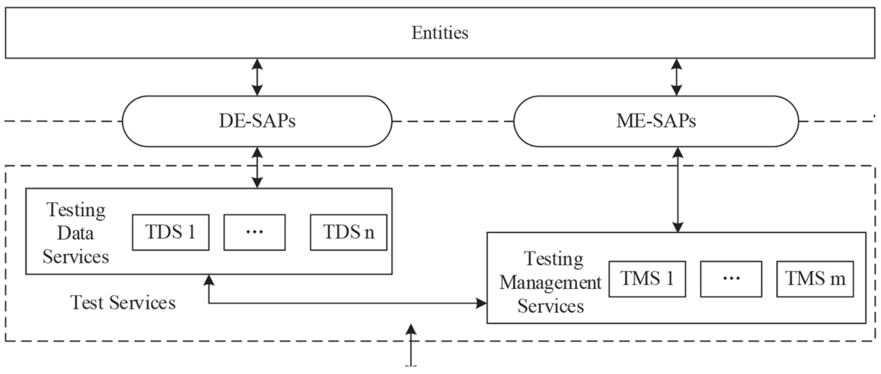

Figure 4 — Overview of test services for sensor networks

TDSs includes:

— EventReport service: This service is used to report one or more failures or exceptions from TAs. The content of the report can be generated dependent on changes of the test condition. EventReport service can be retried until an EventAck service for the EventReport received.

— EventAck service: EventAck is used to acknowledge an individual EventReport service has been received. An EventAck should result in the ceasing of the EventReport retry requests for the corresponding individual event.

— Read service: This service is used to read value of an object from TM in TMR or TA.

— Write service: This service is used to write value of an object from TM in TMR or TA.

— StartTest service: This service is used to start a test task between TM in TMA and TA. Test cases are executed after this service is called.

— StopTest service: This service is used to stop a test task between TM in TMR and TA. All test activities in test system is stopped after this service is called.

— DataUploading service: This service is used to transfer a block of data from TA to TM in TMA. It supports fragmentation to transmit a large amount of data. The destination device can reassemble the received messages.

— DataDownloading service: This service is the same way as the DataUploading service. The difference is that TM sends a block of data to TA.

— StartDownAndUploading service: This service is used to begin downloading or uploading of data.

— StopDownAndUploading service: This service is used to end downloading or uploading of data.

— ExecuteTesting service: This service is used to execute a test case.

TMSs includes:

— Associate service: This service provides mechanisms to create a logical connection between TA and TM in TMR. The connection is the prerequisite for the execution of the other services between TM and TAs.

— Abort service: This service permits to release a logic connection between TM and TAs. Communication activities including should be stopped until a new connection has been established.

— Sync service: The service enables setting of the time for a TA associated with a TM in TMR. This is useful for time synchronization in a test system with time constraints.

— AddressAllocation service: This service provides functions to allocate addresses for TAs. TM can allocate a unique identification for each TA through this service.

— DeviceStatus service: This service is used for a spontaneous transmission of device status.

6.2 Module interactions through unified services

Before testing the SUT, a TM in TMR should create a communication connection with TA and SUT. As shown in Figure 5, the TM sends an associate request message to open a connection for a specific protocol conformance testing. The messages between TM and TA should use the same protocol. Then, TA loads a specific individual protocol and test driver for a specific protocol used in SUT. TA acts as a protocol translator between TM and SUT. When TM intends to terminate the communication activities, it should initiate an Abort service request to TA. If it receives an Abort response from TA, the connection is released successfully.

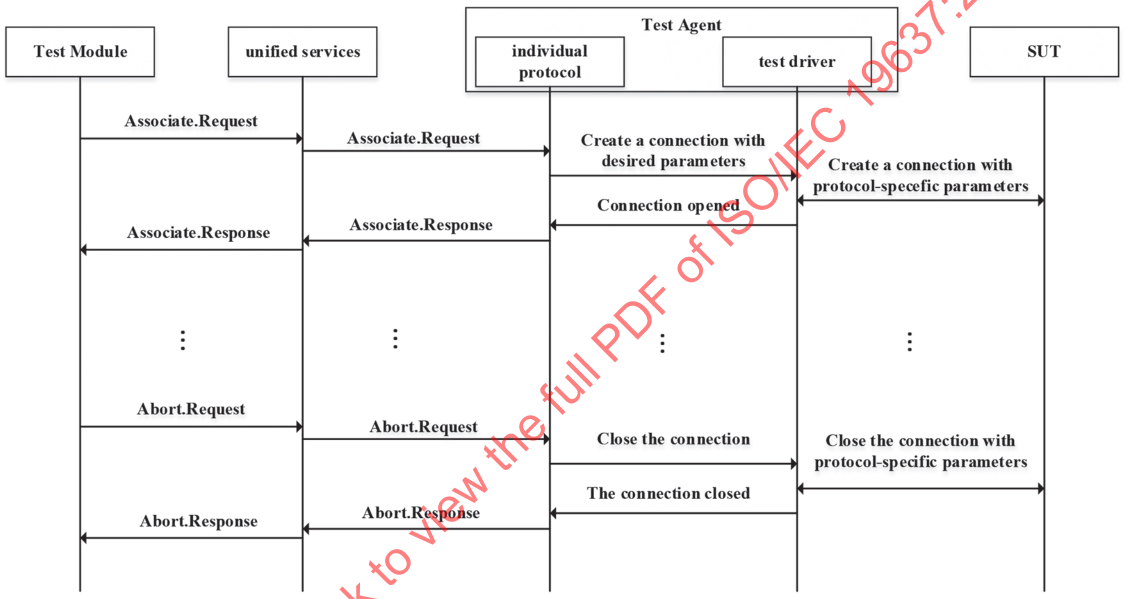

Figure 5 — Sequence diagram of creating and releasing a connection

StartTest service is used to launch a test task after a connection is opened. TA and SUT should allocate the testing resources and provide necessary testing configurations to prepare for testing activities. Test cases in a test suit shall be performed when TA receives ExceuteTesting request. The test results are transmitted to TM. If an analysis module (AM) is configured by the test application process to associate a TA, it should also receive the test results for analysis. When a test task is completed, TM sends a StopTest to end a task. A typed sequence diagram of the messages among modules is shown in Figure 6.

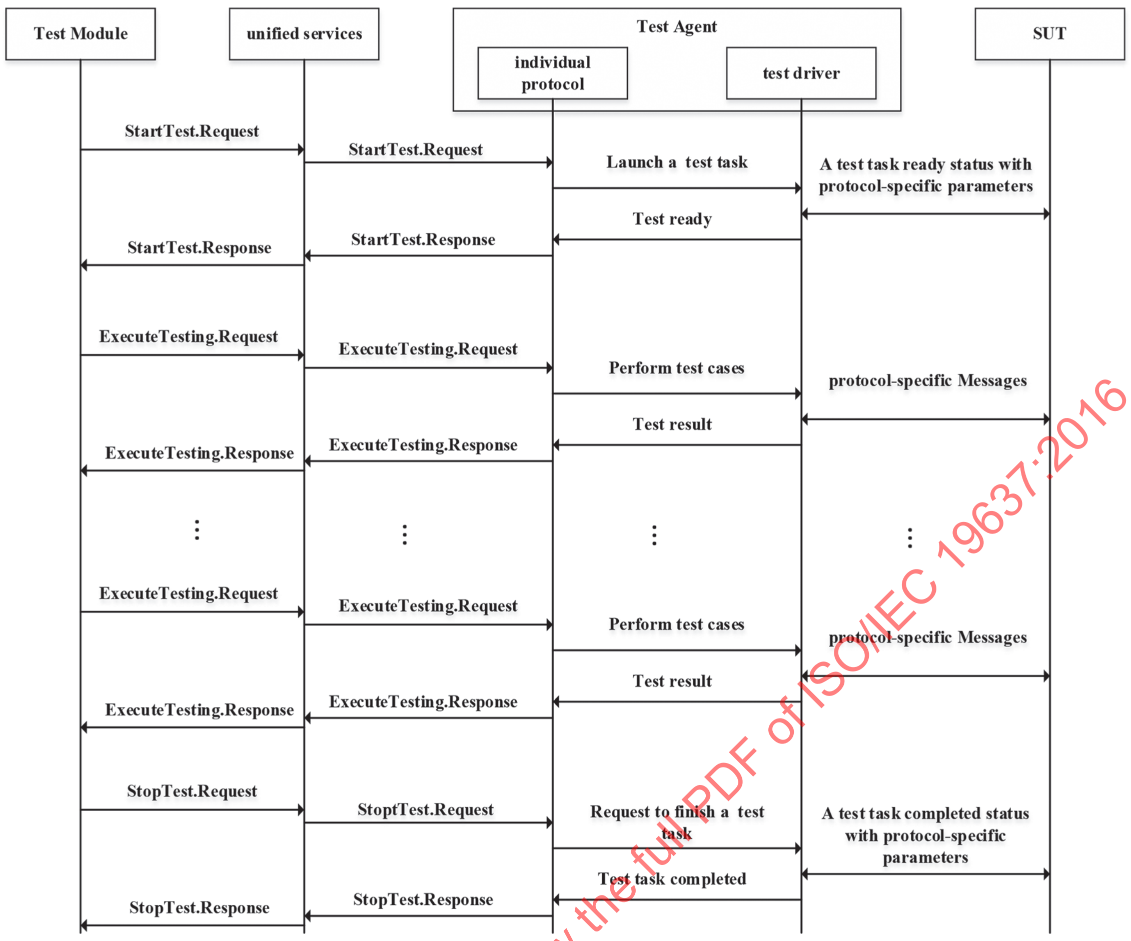

Figure 6 — Sequence diagram of performing test cases

6.3 Test data services (TDSs)

6.3.1 General

This subclause specifies test data services (TDSs) in a sensor networks testing platform. Service primitives and parameters of primitives are defined for each test data service. The names of service access points (SAPs) through which specific service is provided are given in Table 1.

Table 1 — Data services and the names of SAPs

Service name	SAP name
EventReport	EventReport-SAP
EventAck	EventAck-SAP
Read	Read-SAP
Write	Write-SAP
StartTest	Start-SAP
StopTest	Stop-SAP
StartDownAndUploading	StartDownAndUploading-SAP
StopDownAndUploading	StopDownAndUploading-SAP

Table 1 *(continued)*

Service name	SAP name
DataUploading	DataUpLoading-SAP
DataDownloading	DataDownLoading-SAP
ExecuteTesting	ExecuteTesting-SAP

6.3.2 EventReport service

EventReport service is provided through EventReport-SAP. The EventReport-SAP is a logical interface in the application that is issuing an EventReport. EventReport is an unconfirmed service. Table 2 lists the primitives supported by the EventReport-SAP. Table 3 outlines the primitive parameters.

Table 2 — EventReport primitive summary

Name	request	indication	response	confirm
EventReport	6.3.2.1	6.3.2.2		

Table 3 — EventReport primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
Mode	Communication mode. 0: broadcast; 1:client/server
ConnectID	The unique identifier of the connection established.
Priority	It defines the message priority. Possible values: high, medium or low.
EventID	Unique identifier of the individual event.
EventType	It indicates if this is a communication failure, testing process failure, device failure, module failure or a state change.
Timestamp	It specifies the time at which the event was detected.
AssociatedObjectID	Unique identifier of an object that generated an event.
Length	The number of bytes of value.
Value	The content of event

6.3.2.1 EventReport.request

This primitive requests the process of event report from the application layer. If the value of the parameter Mode is broadcast, ConnectID should be set to 0; otherwise, it is the value of the identification of communication connection established by source.

The parameters of this primitive are:

EventReport.request{

SourceAddress,

DestinationAddress,

Mode,

ConnectID

Priority,

EventID,

EventType,

Timestamp,

AssociatedObjectID,

Length.

Value

}

6.3.2.2 EventReport.indication

This primitive indicates the event report was received at the destination in the application process. Upon receipt of this primitive, the receiver should issue EventAck request to the source. The parameters of this primitive are:

EventReport.indication {

SourceAddress,

Mode,

ConnectID

EventID,

EventType,

Timestamp,

AssociatedObjectID,

Length.

Value

}

6.3.3 EventAck service

EventAck service is used to acknowledge an individual event. Table 4 lists the primitives supported by the EventAck-SAP. Table 5 describes the primitive parameters.

Table 4 — EventAck primitive summary

Name	request	indication	response	confirm
EventAck	6.3.3.1	6.3.3.2		

Table 5 — EventAck primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
Mode	Communication mode.
ConnectID	The unique identifier of the connection established.

Table 5 *(continued)*

Parameter Name	Description
Priority	It defines the message priority. Possible values: high, medium or low.
EventID	Unique identifier of the individual event.
AssociatedObjectID	Unique identifier of an object that generated an event.
Length	The number of bytes of result.
Result	The result after processing the event Report service.

6.3.3.1 EventAck.request

If a duplicate EventReport has been received, a duplicate EventAck should been sent. The parameter ConnectID, EventID and AssociatedObjectID should be the same as the received EventReport service. The parameters of this primitive are:

EventAck.request {

SourceAddress,

DestinationAddress,

Mode,

ConnectID

Priority,

EventID,

AssociatedObjectID,

Length,

Result

}

6.3.3.2 EventAck.indication

When the EventAck has been received, the status of event report is be cleared. The parameters of this primitive are:

EventReport.indication{

EventID,

AssociatedObjectID,

Length.

Result

}

6.3.4 Read service

Read service supports client/server communication mode. It is usually used to get configuration of Test Agent. Table 6 lists the primitives supported by the READ-SAP. Table 7 outlines the primitive parameters.

Table 6 — Read primitive summary

Name	request	indication	response	confirm
Read	6.3.4.1	6.3.4.2	6.3.4.3	6.3.4.4

Table 7 — Read primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
Mode	Communication mode.
ConnectID	The unique identifier of the connection established.
ApplicationID	The unique identifier of an individual destination application.
ModuleID	The unique identifier of an individual destination module.
ObjectID	The unique identifier of an individual object.
Length	Length of bytes of Data.
Data	This parameter presents the values that the sender desires to read.
Status	This parameter specifies the access attribute.

6.3.4.1 Read.request

The parameters of this primitive are:

Read.request {

SourceAddress,

DestinationAddress,

Mode,

ConnectID

ApplicationID,

ModuleID,

ObjectID,

Length

}

6.3.4.2 Read.indication

The parameters of this primitive are:

Read.indication {

SourceAddress,

ConnectID

ApplicationID,

ModuleID,

ObjectID,

Length

}

6.3.4.3 Read.response

The parameters of this primitive are:

Read.response{

SourceAddress,

DestinationAddress,

Mode,

ConnectID

ApplicationID,

ModuleID,

ObjectID,

Length,

Data

}

6.3.4.4 Read.confirm

The parameters of this primitive are:

Read.confirm {

ConnectID

ApplicationID,

ModuleID,

ObjectID,

Data

}

6.3.5 Write service

Write service supports client/server communication mode. It is usually used to configure the parameters of Test Agent. Table 8 lists the primitives supported by the Write-SAP. Table 9 outlines the primitive parameters.

Table 8 — Write primitive summary

Name	request	indication	response	confirm
Write	6.3.5.1	6.3.5.2	6.3.5.3	6.3.5.4

Table 9 — Write primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
Mode	Communication mode.
ConnectID	The unique identifier of the connection established.
ApplicationID	The unique identifier of an individual destination application.
ModuleID	The unique identifier of an individual destination module.
ObjectID	The unique identifier of an individual destination object.
Length	Length of bytes of Data.
Data	This parameter is the values that the sender desires to write.
Status	The reason of the response.

6.3.5.1 Write.request

The parameters of this primitive are:

Write.request {

SourceAddress,

DestinationAddress,

Mode,

ConnectID

ApplicationID,

ModuleID,

ObjectID,

Length,

Data

}

6.3.5.2 Write.indication

The parameters of this primitive are:

Write.indication {

SourceAddress,

DestinationAddress,

Mode,

ConnectID

ApplicationID,

ModuleID,

ObjectID,

Length,

Data

}

6.3.5.3 Write.response

The parameters of this primitive are:

Write.response {

SourceAddress,

DestinationAddress,

Mode,

ConnectID

ApplicationID,

ModuleID,

Status

}

6.3.5.4 Write.confirm

The parameters of this primitive are:

Write.confirm {

SourceAddress,

ConnectID

ApplicationID,

ModuleID,

Status

}

6.3.6 StartTest service

This StartTest is a confirmed service. TM can use it to start a test task. Table 10 lists the primitives supported by the StartTest-SAP. Table 11 outlines the primitive parameters.

Table 10 — StartTest primitive summary

Name	request	indication	response	confirm
StartTest	6.3.6.1	6.3.6.2	6.3.6.3	6.3.6.4

Table 11 — StartTest primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
status	The reason of the response.

6.3.6.1 StartTest.request

The parameters of this primitive are:

StartTest.request{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication

}

6.3.6.2 StartTest.indication

The parameters of this primitive are:

StartTest.indication{

DestinationAddress,

ConnectID,

SourceApplication

}

6.3.6.3 StartTest.response

The parameters of this primitive are:

StartTest.response{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication,

Status

}

6.3.6.4 StartTest.confirm

The parameters of this primitive are:

StartTest.confirm{

Status

}

6.3.7 StopTest service

Table 12 lists the primitives supported by the Stop-SAP. Table 13 outlines the primitive parameters.

Table 12 — StopTest primitive summary

Name	Request	Indication	Response	Confirm
StopTest	6.3.7.1	6.3.7.2	6.3.7.3	6.3.7.4

Table 13 — StopTest primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
Status	The reason of the response.

6.3.7.1 StopTest.request

The parameters of this primitive are:

StopTest.request{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication

}

6.3.7.2 StopTest.indication

The parameters of this primitive are:

StopTest.indication{

SourceAddress,

ConnectID,

SourceApplication,

DestinationApplication

}

6.3.7.3 StopTest.response

The parameters of this primitive are:

StopTest.response{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication,

Status

}

6.3.7.4 StopTest.confirm

The parameters of this primitive are:

StopTest.confirm{

Status

}

6.3.8 StartDownAndUploading service

StartDownAndUploading is a confirmed service. Table 14 lists the primitives supported by the StartDownAndUploading-SAP. Table 15 outlines the primitive parameters.

Table 14 — StartDownAndUploading primitive summary

Name	Request	Indication	Response	Confirm
StartDownAndUploading	6.3.8.1	6.3.8.2	6.3.8.3	6.3.8.4

Table 15 — StartDownAndUploading-SAP primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
AreaID	The unique identifier of area that is part of memory.
MaxSize	Maximum size available for download or upload as a whole.
Length	The number of bytes of data that is sent.
Status	The reason of the response.

6.3.8.1 StartDownAndUploading.request

The parameters of this primitive are:

StartDownAndUploading.request{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication,

AreaID,

MaxSize,

Length

}

6.3.8.2 StartDownAndUploading.indication

The parameters of this primitive are:

StartDownAndUploading.indication{

SourceAddress,

ConnectID,

SourceApplication,

AreaID

MaxSize

Length

}

6.3.8.3 StartDownAndUploading.response

The parameters of this primitive are:

StartDownAndUploading.response{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication,

AreaID,

Status

}

6.3.8.4 StartDownAndUploading.confirm

The parameters of this primitive are:

StartDownAndUploading.confirm{

Status

}

6.3.9 StopDownAndUploading service

StopDownAndUploading is a confirmed service. Table 16 lists the primitives supported by the StopDownAndUploading-SAP. Table 17 outlines the primitive parameters.

Table 16 — StopDownAndUploading primitive summary

Name	Request	Indication	Response	Confirm
StartDownAndUploading	6.3.9.1	6.3.9.2	6.3.9.3	6.3.9.4

Table 17 — StopDownAndUploading primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
AreaID	The unique identifier of area that is part of memory.
Status	The reason of the response.

6.3.9.1 StopDownAndUploading.request

The parameters of this primitive are:

StartDownAndUploading.request{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication,

AreaID

}

6.3.9.2 StopDownAndUploading.indication

The parameters of this primitive are:

StartDownAndUploading.indication{

SourceAddress,

ConnectID,

SourceApplication,

AreaID

}

6.3.9.3 StopDownAndUploading.response

The parameters of this primitive are:

StartDownAndUploading.response{

SourceAddress,

DestinationAddress,

ConnectID,

SourceApplication,

DestinationApplication,

AreaID,

Status

}

6.3.9.4 StopDownAndUploading.confirm

The parameters of this primitive are:

StartDownAndUploading.confirm{

Status

}

6.3.10 DataUploading service

Table 18 lists the primitives supported by the DataUploading-SAP. Table 19 outlines the primitive parameters.

Table 18 — DataUploading primitive summary

Name	Request	Indication	Response	Confirm
Data uploading	6.3.10.1	6.3.10.2	6.3.10.3	6.3.10.4

Table 19 — DataUploading primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
AreaID	The unique identifier of area that is part of memory.

Table 19 *(continued)*

Parameter Name	Description
SquenceID	The unique number that is determined by the generator of the request.
IsFinshed	If the request is the last message for upload, the parameter is set to 1, otherwise, it should be set to 0.
currentLength	The number of bytes of data in the request.
data	This parameter is the values that the sender desires to upload.
status	The reason of the response.

6.3.10.1 DataUploading.request

The parameters of this primitive are:

DataUploading.indication {

SourceAddress,

DestinationAddress,

SourceApplication,

DestinationApplication,

ConnectID,

AreaID,

SquenceID,

IsFinshed,

currentLength,

data

}

6.3.10.2 DataUploading.indication

The parameters of this primitive are:

DataUploading.indication {

SourceAddress,

SourceApplication,

ConnectID,

AreaID,

SquenceID,

IsFinshed,

currentLength,

data

}

6.3.10.3 DataUploading.response

The parameters of this primitive are:

DataUploading.response {

ConnectID,

AreaID,

status

}

6.3.10.4 DataUploading.confirm

The parameters of this primitive are:

DataUploading.confirm {

status

}

6.3.11 DataDownloading service

DataDownloadingis a confirmed service. Table 20 lists the primitives supported by the DataDownloading-SAP. Table 21 outlines primitive parameters.

Table 20 — DataDownloading primitive summary

Name	Request	Indication	Response	Confirm
Data downloading	6.3.11.1	6.3.11.2	6.3.11.3	6.3.11.4

Table 21 — DataDownloading primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
AreaID	The unique identifier of area that is part of memory.
SquenceID	The unique number that is determined by the generator of the request.
IsFinshed	If the request is the last message for upload, the parameter is set to 1, otherwise, it should be set to 0.
currentLength	The number of bytes of data in the request.
data	This parameter is the values that the sender desires to upload.
status	The reason of the response.

6.3.11.1 DataDownloading.request

The parameters of this primitive are:

DataDownloading.request {

SourceAddress,

DestinationAddress,

SourceApplication,

DestinationApplication,

ConnectID,

AreaID,

SquenceID,

IsFinshed,

currentLength,

Data

}

6.3.11.2 DataDownloading.indication

The parameters of this primitive are:

DataDownloading.indication {

SourceAddress,

SourceApplication,

ConnectID,

AreaID,

SquenceID,

IsFinshed,

currentLength,

Data

}

6.3.11.3 DataDownloading.response

The parameters of this primitive are:

DataDownloading.response {

ConnectID,

AreaID,

Status

}

6.3.11.4 DataDownloading.confirm

The parameters of this primitive are:

DataDownloading.confirm {

Status

}

6.3.12 ExecuteTesting service

ExecuteTesting is a confirmed service. Table 22 lists the primitives supported by the ExecuteTesting-SAP. Table 23 outlines primitive parameters.

Table 22 — ExecuteTesting primitive summary

Name	Request	Indication	Response	Confirm
ExecuteTesting	6.3.12.1	6.3.12.2	6.3.12.3	6.3.12.4

Table 23 — ExecuteTesting primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
ConnectID	The unique identifier of the connection established.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
ModuleID	The unique identifier of the module that desire to join the test system.
TestType	This parameter indicates if this is a test system with ATE or a test system without ATE.
TestClass	This parameter indicates if this is a conformance testing system, performance testing system, interoperability testing or function testing system.
TestSuitID	The unique identifier of an individual test suit.
TestCaseID	The unique identifier of an individual test case.
Length	The number of bytes of the information from a test case.
TestData	The information from a test case.
status	The reason of the response.

6.3.12.1 ExecuteTesting.request

The parameters of this primitive are:

ExecuteTesting.request {

SourceAddress,

DestinationAddress,

SourceApplication,

DestinationApplication,

ConnectID,

ModuleID,

TestType,

TestClass,

TestSuitID,

TestCaseID,

Length,

TestData

}

6.3.12.2 ExecuteTesting.indication

The parameters of this primitive are:

ExecuteTesting.indication {

SourceAddress,

SourceApplication,

DestinationApplication,

ConnectID,

ModuleID,

TestType,

TestClass,

TestSuitID,

TestCaseID,

Length,

TestData

}

6.3.12.3 ExecuteTesting.response

The parameters of this primitive are:

ExecuteTesting.response {

SourceAddress,

DestinationAddress,

SourceApplication,

DestinationApplication,

ConnectID,

ModuleID,

TestType,

TestClass,

TestSuitID,

TestCaseID,

Length,

TestData,

Status

}

6.3.12.4 ExecuteTesting.confirm

The parameters of this primitive are:

ExecuteTesting.confirm {

ModuleID,

TestSuitID,

TestCaseID,

Length,

TestData,

Status

}

6.4 Testing management services

6.4.1 Overview

This clause specifies the TMS in sensor networks testing platform. Service primitives and the parameters of primitives are defined for each service. In Table 24, the names of service access points (SAPs) through which specific service is provided are listed.

Table 24 — Management services and the names of SAPs

Service name	SAP name
Associate service	Associate-SAP
Abort service	Abort-SAP
Sync Service	Sync-SAP
AddressAllocation service	AddressAllocation-SAP
DeviceStatus service	DeviceStatus-SAP

6.4.2 Associate service

This service is used to create a new connection between the sender and receiver. It is a confirmed service. Table 25 lists the primitives supported by the Associate-SAP. Table 26 outlines primitive parameters.

Table 25 — Associate primitive summary

Name	Request	Indication	Response	Confirm
Associate	6.4.2.1	6.4.2.2	6.4.2.3	6.4.2.4

Table 26 — Associate primitive parameters

Parameter Name	Description
SourceAddress	The address from which the service request is sent.
DestinationAddress	The address to which the service request is to be sent.
SourcePort	The port from which the service request is sent. The response uses it when returning a message.
DestinationPort	The port to which the service request is to be sent. The response receives the request at this port.
SourceApplication	The unique identifier of the source application.
DestinationApplication	The unique identifier of the destination application.
ConnectID	The unique identifier of the connection established.
CommunicationMode	Communication mode.
Status	The reason of the response.
Detail	More information that specifies the reason of the response.

6.4.2.1 Associate.request

This primitive requests the process of event subscription from the application layer. The parameters of this primitive are:

Associate.request{

SourceAddress,

DestinationAddress,

SourcePort,

DestinationPort,

SourceApplication,

DestinationApplication,

ConnectID,

CommunicationMode

}

6.4.2.2 Associate.indication

The parameters of this primitive are:

Associate.indication{

SourceAddress,

SourcePort,

SourceApplication,

DestinationApplication,

ConnectID,

CommunicationMode

}

6.4.2.3 Associate.response

The parameters of this primitive are:

Associate.response{

SourceAddress,

DestinationAddress,

SourcePort,

DestinationPort,

SourceApplication,

DestinationApplication,

ConnectID,

CommunicationMode,

Status,

Detail

}

6.4.2.4 Associate.confirm

The parameters of this primitive are:

Associate.confirm{

SourceApplication,

DestinationApplication,

ConnectID,

CommunicationMode,

Status,

Detail

}

6.4.3 Abort service

Table 27 lists the primitives supported by the Abort-SAP. Table 28 outlines the primitive parameters.

Table 27 — Abort primitive summary

Name	Request	Indication	Response	Confirm
Abort	6.4.3.1	6.4.3.2		