
**Intelligent transport systems —
Communications access for land mobiles
(CALM) — Application management —
Part 2:
Conformance test**

*Systèmes intelligents de transport — Accès aux communications des
services mobiles terrestres (CALM) — Gestion d'application —*

Partie 2: Essai de conformité



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

STANDARDSISO.COM : Click to view the full PDF of ISO 24101-2:2010



COPYRIGHT PROTECTED DOCUMENT

© ISO 2010

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword	iv
Introduction.....	v
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 Abbreviated terms	2
5 General	3
6 Test system	4
7 Functionality of entities	5
7.1 Test applications	5
7.2 Test operator.....	5
7.3 Test installer	5
7.4 Verification means.....	6
7.5 Communication station.....	6
7.6 Connection between entities.....	6
8 Test cases	6
8.1 General	6
8.2 Basic tests.....	6
8.3 Advanced tests	7
9 Test conditions	7
9.1 Test applications	7
9.2 Common files	8
9.3 Security information for operator authentication	8
9.4 Manager certificate for access control.....	8
9.5 Combination example of test conditions	8
10 Test procedures	9
10.1 Basic tests.....	9
10.2 Advanced tests.....	9
10.3 Notation for test procedures	9
10.4 Commands in test procedures.....	9
11 Test results	10
12 Test documentation	10
12.1 Implementation conformance statement (ICS).....	10
12.2 Implementation eXtra information for testing (IXIT)	10
12.3 Test report.....	10
Annex A (normative) Basic tests.....	11
Annex B (normative) Advanced tests	30
Annex C (informative) Commands in test procedures.....	48
Annex D (normative) Implementation conformance statement proforma	50
Annex E (normative) Implementation eXtra information for testing proforma.....	53
Bibliography.....	56

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

ISO 24101-2 was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

ISO 24101 consists of the following parts, under the general title *Intelligent transport systems — Communications access for land mobiles (CALM) — Application management*:

- *Part 1: General requirements*
- *Part 2: Conformance test*

Introduction

This part of ISO 24101 is part of a family of International Standards for communications access for land mobiles (CALM) which determine a common architecture, network protocols and air interface definitions for wireless communications using cellular second generation, cellular third generation, 5 GHz, millimeter, and infrared communications. Other air interfaces can be added at a later date. Air interfaces included in the CALM International Standards provide facilities for broadcast, point-to-point, vehicle-to-vehicle, and vehicle-to-point communications in the intelligent transport systems (ITS) sector.

The application management (AM), as defined in ISO 24101-1:2008, is the function to install, uninstall and modify the ITS applications in a reliable and secure manner, and it addresses the following requirements:

- a) installation of applications on CALM equipment after the equipment has been deployed;
- b) updating of applications, including uninstalling, on on-board equipment (OBE) as well as wireless access equipment (WAE) after the equipment has been deployed;
- c) providing a standardized interface and functionality so that application developers and system operators can successfully perform the functions in a) and b).

The purpose of this part of ISO 24101 is to specify a standardized conformance test for the AM, which verifies that the function of the developed AM complies with the requirements specified in ISO 24101-1:2008. See Figure 1.

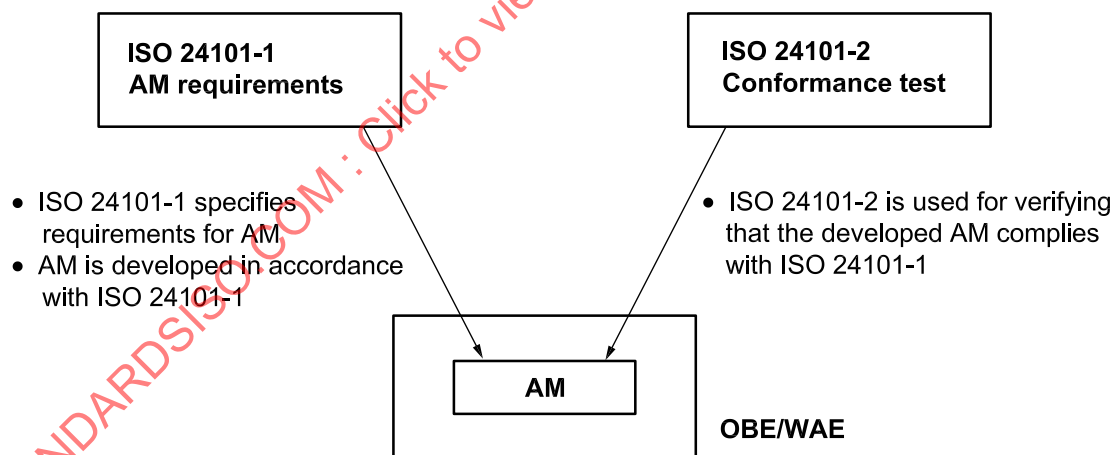


Figure 1

STANDARDSISO.COM : Click to view the full PDF of ISO 24101-2:2010

Intelligent transport systems — Communications access for land mobiles (CALM) — Application management —

Part 2: Conformance test

1 Scope

This part of ISO 24101 specifies the test system, test cases, test conditions, test procedures and test results for examining the function of the communications access for land mobiles (CALM) application management (AM).

NOTE Accreditation and certification are outside the scope of this part of ISO 24101.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 24101-1:2008, *Intelligent transport systems — Communications access for land mobiles (CALM) — Application management — Part 1: General requirements*

3 Terms and definitions

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

3.1 communication station

wireless station that communicates with the equipment under test (EUT) by using the communications access for land mobiles (CALM) medium for the application management (AM) conformance test

3.2 normal operation

coincidence of the test results and the predetermined states, which are defined before the test, when the test application is executed on the equipment under test (EUT)

3.3 test applications

applications that are used for installing, uninstalling or modifying an application and are pre-installed in the equipment under test (EUT) or the test installer for the application management (AM) conformance test

3.4 test installer

installer that is used for the application management (AM) conformance test

3.5

test operator

operator who operates the test installer for the application management (AM) conformance test

3.6

verification means

means that verifies the results of the application management (AM) conformance test and sets the equipment under test (EUT) to the state defined by the test procedures

4 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

AM	Application Management
AME	Application Management Entity
AMT	Application Management Table
API	Application Programming Interface
BER	Bit Error Rate
CALM	Communications Access for Land Mobiles
CPU	Central Processing Unit
ETSI	European Telecommunications Standards Institute
EUT	Equipment Under Test (ISO/IEC 9646-1:1994)
ICS	Implementation Conformance Statement (ISO/IEC 9646-1:1994)
ITS	Intelligent Transport Systems
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
IUT	Implementation Under Test (ISO/IEC 9646-1:1994)
IXIT	Implementation extra Information for Testing (ISO/IEC 9646-1:1994)
OBE	On-Board Equipment
OS	Operating System
PER	Packet Error Rate
RSSI	Received signal strength indication
TS	Test System
TTCN-3	Testing and Test Control Notation Version 3
URL	Uniform Resource Locator
VM	Virtual Machine
WAE	Wireless Access Equipment

5 General

The methodology and concepts of the AM conformance test are based on ISO/IEC 9646-1:1994.

An AM is developed firstly by the service provider in accordance with ISO 24101-1:2008. The developed AM is software and is installed in the equipment under test (EUT), i.e. on-board equipment (OBE) or wireless access equipment (WAE), prior to the AM conformance test.

Figure 2 shows the basic approach to the AM conformance test.

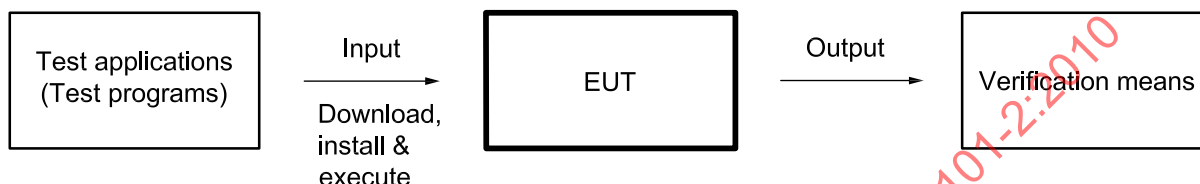


Figure 2 — Basic approach to conformance test

The basic approach is as follows:

- a) collect requirements and specifications for the EUT;
- b) prepare implementation conformance statement(s) (ICS) and implementation eXtra information for testing [IXIT(s)];
- c) prepare some test applications;
- d) prepare test suites, i.e. the set of basic tests and advanced tests, and test system including the verification means, etc.;
- e) install the AM in the EUT, and install the test applications in the EUT and the test installer;
- f) perform the conformance test;
 - 1) download and install the test application into the EUT for installation or modification,
 - 2) execute the test application,
 - 3) confirm the results of the execution by the verification means, and compare the results with the predetermined states, i.e. expected states;
 - i) when all results coincide with the predetermined states: verdict is set into pass,
 - ii) if there are one or more differences: verdict is set into fail;
 - 4) send the command of uninstallation to the EUT and confirm the results of the execution by reading the contents of the application management table (AMT) by the verification means, and
- g) analyse results and prepare the conformance test report.

Figure 3 shows the conformance test process.

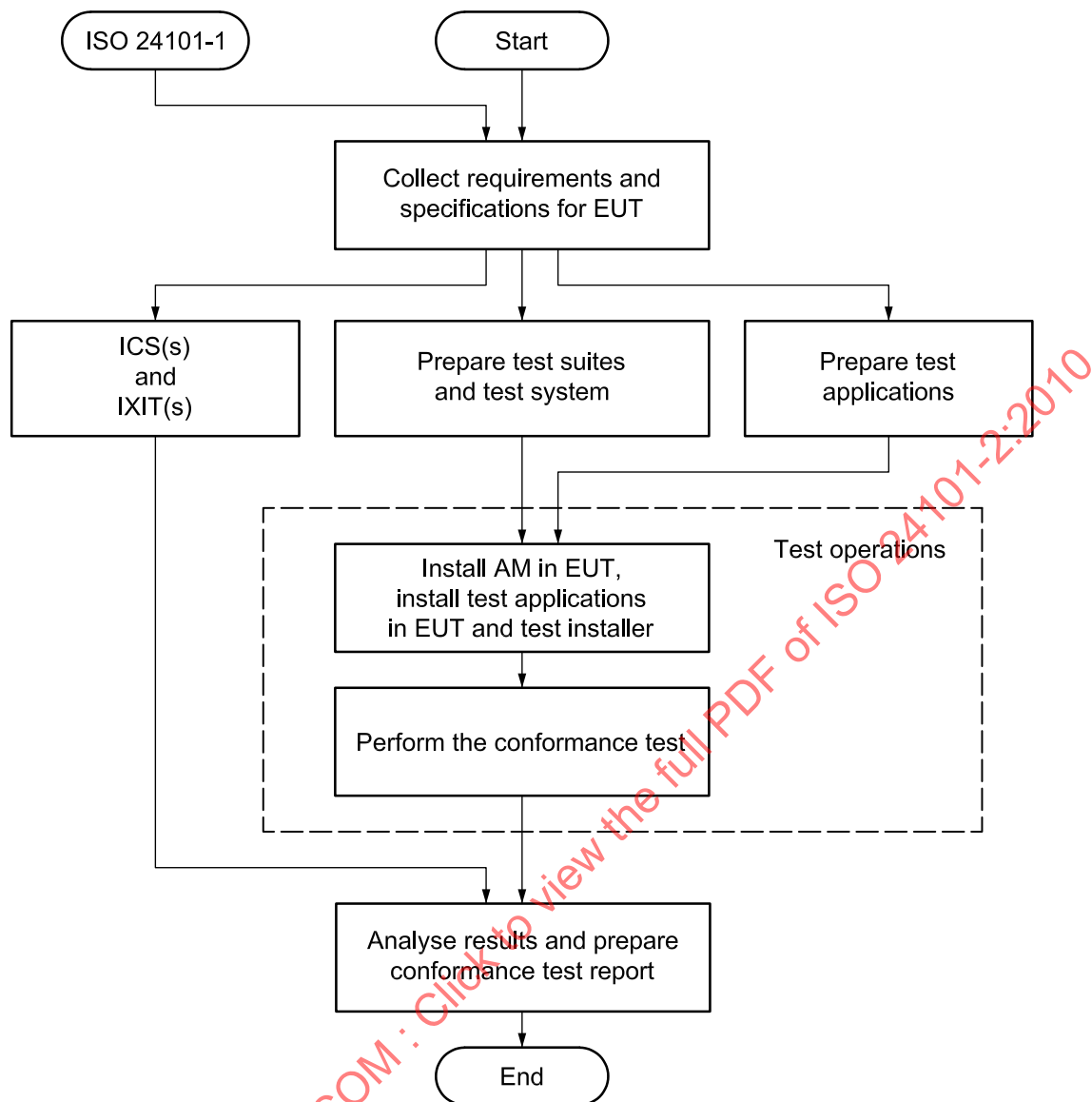


Figure 3 — Conformance test process

6 Test system

Figure 4 shows the test system for the AM conformance test.

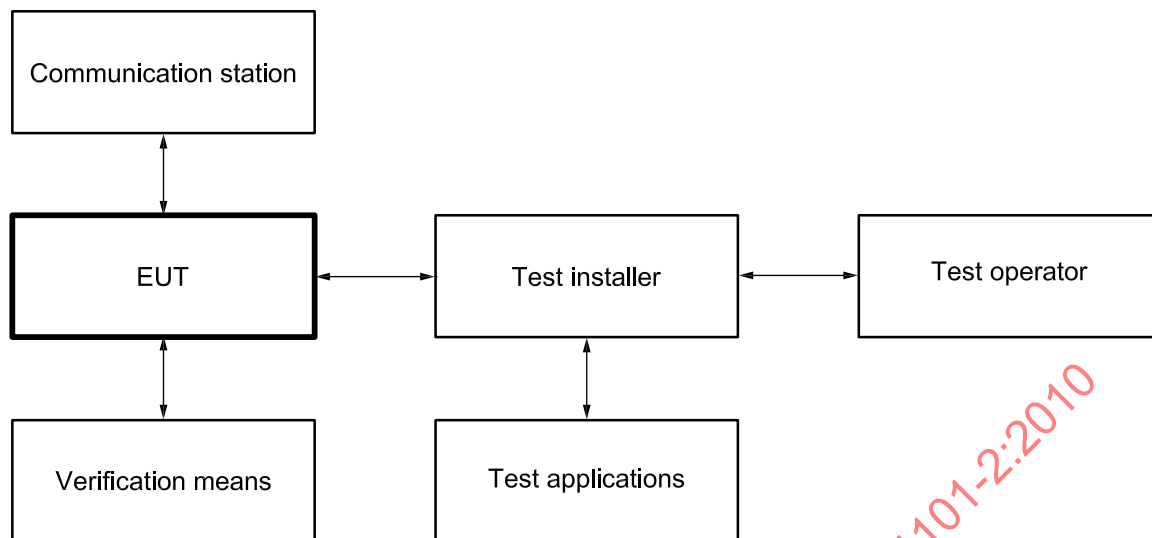


Figure 4 — Test system

7 Functionality of entities

7.1 Test applications

Test applications are pre-installed in the test installer for installation or modification of applications, and also in the EUT for uninstallation. See 9.1 and 9.5.

7.2 Test operator

The test operator installs, uninstalls or modifies the test applications via the test installer, and verifies the test results by the verification means.

7.3 Test installer

7.3.1 OBE/WAE initiated download case

The test installer can be a test server in case of the OBE/WAE initiated download. See ISO 24101-1:2008.

External human machine interfaces (e.g. keyboard and display) may be temporally connected to the EUT via its external interface (if necessary for the purpose of the AM conformance test) for providing the following operator's human machine interfaces:

- inputting commands to the EUT for installation, uninstallation or modification of an application;
- display of the command execution results.

EXAMPLE "Under command execution", "The command has successfully completed", "Operator authentication error" and "Manager certificate error" are examples of the display.

7.3.2 Installer initiated download case

In case of installer initiated download, the test installer has the functions specified in 7.3.1 inside.

7.4 Verification means

The verification means has the following functions:

- a) confirming the execution results of the installed or modified application in cooperation with the communication station (e.g. the communication results with the communication station is displayed on the verification means);
- b) reading and confirming the AMT information of the EUT;
- c) rewriting the AMT information for intentional error;
- d) setting the clock data of the EUT for the scheduled update (T-10 in Table 1);
- e) setting the test conditions in the EUT (optional, e.g. halt under program execution).

NOTE The verification means can use a debugger, an emulator or a personal computer, etc. For example, a debugger has useful functions such as the break point function and the contents rewriting function of a specified program address. Also, in a simple case, the verification means can be a test installer (e.g. use of the AMT reading function).

7.5 Communication station

The communication station is used for verifying that applications in the EUT operate normally after installation or modification of an application in cooperation with the verification means. The verification includes wireless communication with CALM media.

7.6 Connection between entities

The connection methods between entities are not specified in this part of ISO 24101.

8 Test cases

8.1 General

Test cases are subdivided into basic tests and advanced tests.

The basic tests verify the basic operations of the AM, and the advanced tests verify that the AM can detect abnormality when the test system is set intentionally in abnormal state.

8.2 Basic tests

Table 1 specifies the test cases of basic tests.

Table 1 — Basic tests

Test number	Test name
T-1	Installation of application
T-2	Modification of application
T-3	Uninstallation of application
T-4	Function of reading application
T-5	Installation of common file
T-6	Modification of common file
T-7	Uninstallation of common file
T-8	Function of reading common file
T-9	Function of reading AMT information
T-10	Function of the scheduled update

The function of each test name is given in ISO 24101-1:2008.

8.3 Advanced tests

Table 2 specifies the test cases of advanced tests.

Table 2 — Advanced tests

Test number	Test name
T-11	Operator authentication
T-12	Manager certificate
T-13	The same file name
T-14	Lack of sufficient available memory space
T-15	Prohibition of application management entity (AME) start while an application is running
T-16	Prohibition of application start while AME is running
T-17	Abnormal termination of download
T-18	Archival records
T-19	Restoration function

The function of each test name is given in ISO 24101-1:2008.

9 Test conditions

9.1 Test applications

Test applications shall have the following functions:

- communicate with the CALM compliant communication station [*CALM compliant*];
- meaning of normal operation is defined beforehand and the test applications are programmed according to the definitions [*Definitions in advance*];

- c) decision criteria of the test results are explicit [*Unambiguity*];
- d) test applications are programmed so that the test results can be confirmed by communication with the communication station and the verification means [*Testability*].

Test applications and verification means should co-operate with each other. Test applications should be programmed to output or display the communication result with the communication station and the state of the EUT to the verification means so that the test operator can confirm the test results by the verification means.

At least two applications with different manager certificates should be installed in the EUT for modification or uninstallation, and at least two applications should be installed in the test installer for installation or modification before the AM conformance test. See Table 3.

9.2 Common files

At least two common files should be installed in the EUT for modification or uninstallation, and at least two common files should be installed in the test installer for installation or modification before the AM conformance test. See Table 3.

9.3 Security information for operator authentication

Security information (e.g. public key) that allows the test installer to authenticate a test operator is stored in the test installer in advance. Mutual authentication of the test operator and test installer shall precede any exchange of application information. See ISO 24101-1:2008, 8.3.1.

The details of operator authentication are not specified in this part of ISO 24101.

9.4 Manager certificate for access control

Manager certificate is a manager's security credential containing information used to verify the identity of the source of the credential. For example a manager certificate is sent by the manager of an application to the OBE/WAE and is used by the AME to authenticate the manager. See ISO 24101-1:2008, 8.3.2.

At least six manager certificates may be set corresponding to the test applications and common files. See Table 3.

9.5 Combination example of test conditions

Table 3 shows the kind of test applications and common files that are required for the conformance test, and also shows the combination example of pre-installation place, manager certificate and test purpose for each kind.

Table 3 — Combination example of test conditions

Test application or common file	Pre-installation place	Manager certificate	Test purpose
testApplication-a	Installer	managerCertificate-a	Installation
testApplication-b1	EUT	managerCertificate-b	Modification
testApplication-b2	Installer	managerCertificate-b	Modification
testApplication-c	EUT	managerCertificate-c	Uninstallation
commonFile-a	Installer	managerCertificate-d	Installation
commonFile-b1	EUT	managerCertificate-e	Modification
commonFile-b2	Installer	managerCertificate-e	Modification
commonFile-c	EUT	managerCertificate-f	Uninstallation

NOTE 1 The file names are shown as testApplication-a, -b, -c, etc. The number following it shows a version number.

NOTE 2 The file names are representatively shown as testApplication-a, etc. The naming method of file names (i.e. identification of the application program) is specified in ISO 24101-1:2008, 8.1, although the file names of test applications can be named by the arbitrary method distinguished mutually if the test applications are test purpose only and they are not installed in the OBE/WAE products. This is applicable also to common files.

Figure 5 shows the test purposes and corresponding pre-installations for test applications (testApplications) and common files (commonFiles).

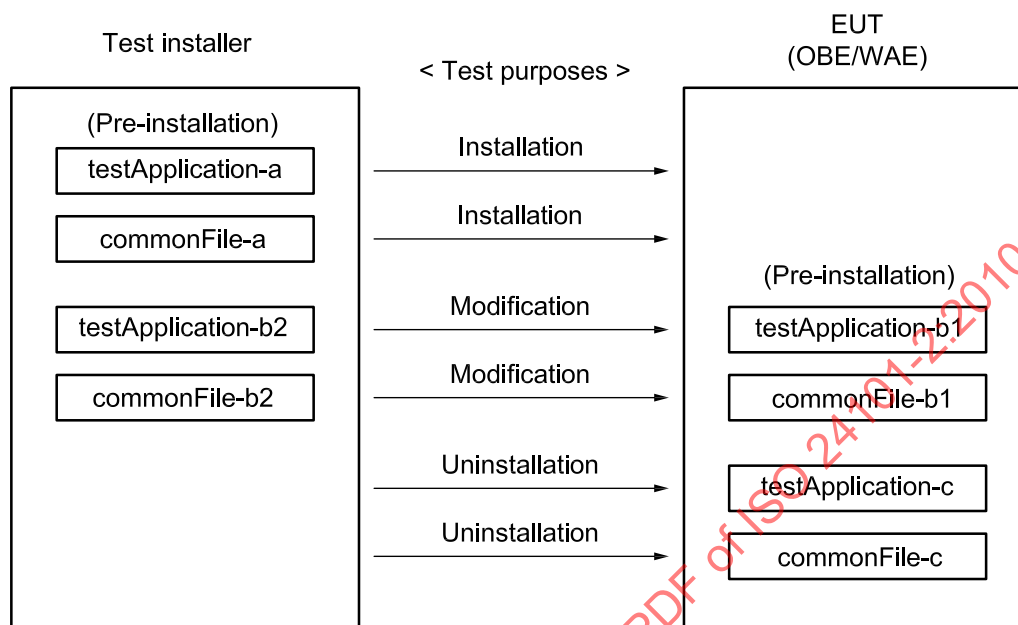


Figure 5 — Test purposes and pre-installations

10 Test procedures

10.1 Basic tests

Annex A specifies the test procedures for each test case of the basic tests.

10.2 Advanced tests

Annex B specifies the test procedures for each test case of the advanced tests.

10.3 Notation for test procedures

Test procedures are specified in TTCN-3 in Annex A and Annex B. See ITU-T Recommendations Z.161 through to Z.167 and ETSI ES 201 873-1 about TTCN-3.

In TTCN-3, a message is transferred between TS and EUT via a port, i.e. a message is sent to EUT via a port from TS, and TS receives the response message from EUT via the port. The response message is compared with the expected message and verdict is set according to the comparison result. On the other hand, a debugger may be used, for example as the verification means as shown in 7.4 in the implementation. In such a case, from a viewpoint of TTCN-3, it is considered that the connector which connects the debugger to EUT corresponds to the port and it is abstracted that a message is transferred via the port.

10.4 Commands in test procedures

Annex C specifies the commands that can be used for installation, uninstallation and modification in the test procedures of Annex A and Annex B.

11 Test results

Annex A also specifies the test results as the verdict in test procedures for each test case of the basic tests.

Annex B also specifies the test results as the verdict in test procedures for each test case of the advanced tests.

12 Test documentation

12.1 Implementation conformance statement (ICS)

The implementation conformance statement (ICS) is a statement made by the supplier that claims conformance to a certain specification. The ICS states which capabilities have been implemented in the specifications. The ICS also states possible limitations in the implementation of the specification.

This part of ISO 24101 describes testing of implementations according to ISO 24101-1:2008.

Annex D contains the ICS proforma that shall be used for the ICS.

12.2 Implementation eXtra information for testing (IXIT)

The implementation eXtra information for testing (IXIT) is a statement made by the supplier or an implementor of an implementation under test (IUT) which contains or references all of the information, in addition to that given in the ICS, related to the IUT and its testing environment. IXIT enables a test laboratory or a test organization to run an appropriate test suite against the IUT.

The IXIT shall also contain further information and procedures that are not specified in ISO 24101-1:2008 but are pre-requisites to perform the test.

Annex E contains the IXIT proforma that shall be used for the declaration of the IXIT.

12.3 Test report

Each test carried out shall be documented. For each test a pass or fail verdict shall be given. The results of all tests shall form the conformance test report.

Annex A (normative)

Basic tests

A.1 Installation of application

Table A.1 — Test number T-1

Test name
Installation of application
Purpose
Verifying that new application in EUT is installed by the command of installing an application from the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestSystem1 { type port TestInstaller message { out writeInstruction ; in resultWriteInstruction ; out readInstruction ; in resultReadInstruction ; out readAMTInstruction ; in resultReadAMTInstruction ; } type port ComStation message { out a_command ; in a_response ; } type port VerificationMeans message { out a_date ; in a_testresult ; out a_time ; out a_vcommand ; } type component TSSState1 { template charstring a_command := "Start testApplication-a" ; template charstring a_response := "Normal termination" ; template charstring a_testresult := "Normal" ; template charstring a_OBEorWAEid := "???" ; // OBE/WAE identification code, implementation matter timer t_installer ; timer t_com ; timer t_vmeans ; port TestInstaller pt_installer ; port ComStation pt_com ; port VerificationMeans pt_vmeans ; } } // </pre>

```

altstep alt_timeGuard ( inout timer p_t ) {                                     // Common time-out process
    [] p_t.timeout { setverdict ( fail ) }
}
//
import from writeInstruction language "ASN.1:2002" all ;                      // See ISO 24101-1:2008, Annex B.
import from resultWriteInstruction language "ASN.1:2002" all ;
import from readInstruction language "ASN.1:2002" all ;
import from resultReadInstruction language "ASN.1:2002" all ;
import from readAMTInstruction language "ASN.1:2002" all ;
import from resultReadAMTInstruction language "ASN.1:2002" all ;
}

module TestCase1 {
import from TestSystem1 all ;
modulepar { octetstring mp_applicationData_i1 } ;    // Contents of a new application file
modulepar { octetstring mp_managementData_i1 } ;    // Contents of AMT
    // These module parameters shall be provided prior to the execution of the test case.

testcase tc_t1 () runs on TSState1 {
    // Install the test application
    sourceCode      := "managerCertificate-a" ; // A component of writeInstruction. Proper certificate.
    destinationCode := a_OBEorWAEid ;
    controlId       := "installation" ;
    aPLIdentifier    := "testApplication-a" ;
    applicationData := mp_applicationData_i1 ;
    writeInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & applicationData ;
    writeInstruction := '01'O & writeInstruction ; // Message header addition
    pt_installer.send ( writeInstruction ) ;
    t_installer.start( 300.0 ) ; // The duration can be changed.

    //
    sourceCode      := a_OBEorWAEid ;
    destinationCode := "managerCertificate-a" ;
    resultCode       := "OK" ;
    resultWriteInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & resultCode ;
    resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition
    alt {
        [] pt_installer.receive( resultWriteInstruction ) {
            t_installer.stop ;
            setverdict( pass ) ;
        }
        [] pt_installer.receive {
            t_installer.stop ;
            setverdict( fail ) ; // Unexpected message received
        }
    }
    [] alt_timeGuard ( t_installer ) ;
}

// Check the communication with CALM medium
pt_com.send( a_command ) ;
t_com.start( 300.0 ) ;
t_vmeans.start( 300.0 ) ;
alt {

```

```

[] pt_com.receive( a_response ) {
    t_com.stop ;
    setverdict( pass ) ;
}

[] pt_com.receive {
    t_com.stop ;
    setverdict( fail ) ;    // Unexpected message received
}

[] alt_timeGuard ( t_com ) ;
}

// Check the test result
alt {
[] pt_vmeans.receive( a_testresult ) {
    t_vmeans.stop ;
    setverdict( pass ) ;
}

[] pt_vmeans.receive {
    t_vmeans.stop ;
    setverdict( fail ) ;    // Unexpected message received
}

[] alt_timeGuard ( t_vmeans ) ;
}

// Check the AMT contents
sourceCode      := "managerCertificate-a" ;    // A component of readAMTInstruction.
destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
pt_installer.send ( readAMTInstruction ) ;
t_installer.start( 300.0 ) ;

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-a" ;
managementData := mp_managementData_i1 ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ;
resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
alt {
[] pt_installer.receive( resultReadAMTInstruction ) {
    t_installer.stop ;
    setverdict( pass ) ;
}

[] pt_installer.receive {
    t_installer.stop ;
    setverdict( fail ) ;    // Unexpected message received
}

[] alt_timeGuard ( t_installer ) ;
}
}
}

```

NOTE The writeInstruction, etc., in the module test case 1 are used for the installer initiated download. Other commands can be used for the OBE/WAE initiated download.

A.2 Modification of application

Table A.2 — Test number T-2

Test name
Modification of application
Purpose
Verifying that the application in EUT is modified by the command of modifying application from the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase2 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_applicationData_m2 } ; // Contents of new application file modulepar { octetstring mp_managementData_m2 } ; // Contents of AMT after modification // These module parameters shall be provided prior to the execution of the test case. testcase tc_t2 () runs on TSSState1 { // See T-1 // Modify the test application sourceCode := "managerCertificate-b" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlId := "modification" ; aPLIdentifier := "testApplication-b2" ; applicationData := mp_applicationData_m2 ; writeInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-b" ; resultCode := "OK" ; resultWriteInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } [] alt_timeGuard (t_installer) ; } // Check the communication with CALM medium a_command := "Start testApplication-b2" ; pt_com.send(a_command) ; </pre>

```

t_com.start( 300.0 );
t_vmeans.start( 300.0 );
alt {
    [] pt_com.receive( a_response ) {
        t_com.stop ;
        setverdict( pass ) ;
    }

    [] pt_com.receive {
        t_com.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }

    [] alt_timeGuard ( t_com ) ;
}
// Check the test result
alt {
    [] pt_vmeans.receive( a_testresult ) {
        t_vmeans.stop ;
        setverdict( pass ) ;
    }

    [] pt_vmeans.receive {
        t_vmeans.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }

    [] alt_timeGuard ( t_vmeans ) ;
}
// Check the AMT contents
sourceCode      := "managerCertificate-b";    // A component of readAMTInstruction.
destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
pt_installer.send ( readAMTInstruction ) ;
t_installer.start( 300.0 ) ;

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-b" ;
managementData := mp_managementData_m2 ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ;
resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
alt {
    [] pt_installer.receive( resultReadAMTInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }

    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }

    [] alt_timeGuard ( t_installer ) ;
}
}
}

```

A.3 Uninstallation of application

Table A.3 — Test number T-3

Test name
Uninstallation of application
Purpose
Verifying that the application in EUT is uninstalled by the command of uninstalling application from the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase3 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_managementData_d1 } ; // Contents of AMT after uninstallation // The module parameter shall be provided prior to the execution of the test case. testcase tc_t3 () runs on TSSState1 { // See T-1 // Delete the test application sourceCode := "managerCertificate-c" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlIdentifier := "uninstallation" ; aPLIdentifier := "testApplication-c" ; writeInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-c" ; resultCode := "OK" ; resultWriteInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } [] alt_timeGuard (t_installer) ; } // Check the AMT contents </pre>

```

sourceCode      := "managerCertificate-c";    // A component of readAMTInstruction.
destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
    pt_installer.send ( readAMTInstruction ) ;
        t_installer.start( 300.0 ) ;

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-c" ;
managementData := mp_managementData_d1 ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ;
resultReadAMTInstruction := '06' O & resultReadAMTInstruction ; // Message header addition
alt {
    [] pt_installer.receive( resultReadAMTInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
}
}

```

A.4 Function of reading application

Table A.4 — Test number T-4

Test name
Function of reading application
Purpose
Verifying that the resident application in EUT is read by the command of reading application from the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase4 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_applicationData_m1 } ; // Contents of resident application file // The module parameter shall be provided prior to the execution of the test case. testcase tc_t4 () runs on TSState1 { // See T-1 // Read out the test application sourceCode := "managerCertificate-b" ; // A component of readInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controllIdentifier := "readout" ; aPLIdentifier := "testApplication-b1" ; readInstruction := sourceCode & destinationCode & controllIdentifier & aPLIdentifier ; readInstruction := '03'O & readInstruction ; // Message header addition pt_installer.send (readInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-b" ; applicationData := mp_applicationData_m1 ; resultReadInstruction := sourceCode & destinationCode & controllIdentifier & aPLIdentifier ; resultReadInstruction := resultReadInstruction & applicationData ; resultReadInstruction := '04'O & resultReadInstruction ; // Message header addition alt { [] pt_installer.receive(resultReadInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } } alt_timeGuard (t_installer) ; } } </pre>

A.5 Installation of common file

Table A.5 — Test number T-5

Test name
Installation of common file
Purpose
Verifying that new common file is installed in EUT by the command of installing common file from the test installer
Test conditions
See Clause 9. T-1 shall be executed before this test.
Test procedures
<pre> module TestCase5 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_commonFileData_i2 } ; // Contents of a new common file modulepar { octetstring mp_managementData_i2 } ; // Contents of AMT // These module parameters shall be provided prior to the execution of the test case. testcase tc_t5 () runs on TSSState1 { // See T-1 // Install the common file sourceCode := "managerCertificate-d" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlId := "installation" ; aPLIdentifier := "commonFile-a" ; applicationData := mp_commonFileData_i2 ; writeInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-d" ; resultCode := "OK" ; resultWriteInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } } [] alt_timeGuard (t_installer) ; </pre>

```

    }
    // Check the communication with CALM medium
    a_command := "Start testApplication-a with access to commonFile-a" ;
    pt_com.send( a_command ) ;    // Execution of the test application
    t_com.start( 300.0 ) ;
    alt {
        [] pt_com.receive( a_response ) {
            t_com.stop ;
            setverdict( pass ) ;
        }
        [] pt_com.receive {
            t_com.stop ;
            setverdict( fail ) ;    // Unexpected message received
        }
        [] alt_timeGuard ( t_com ) ;
    }
    // Check the AMT contents
    sourceCode    := "managerCertificate-d" ;    // A component of readAMTInstruction.
    destinationCode := a_OBEorWAEid ;
    controlIdIdentifier := "AMT" ;
    readAMTInstruction := sourceCode & destinationCode & controlIdIdentifier ;
    readAMTInstruction := '05'O & readAMTInstruction ;    // Message header addition
    pt_installer.send ( readAMTInstruction ) ;
    t_installer.start( 300.0 ) ;

    //
    sourceCode    := a_OBEorWAEid ;
    destinationCode := "managerCertificate-d" ;
    managementData := mp_managementData_i2 ;
    resultReadAMTInstruction := sourceCode & destinationCode & controlIdIdentifier & managementData ;
    resultReadAMTInstruction := '06'O & resultReadAMTInstruction ;    // Message header addition
    alt {
        [] pt_installer.receive( resultReadAMTInstruction ) {
            t_installer.stop ;
            setverdict( pass ) ;
        }
        [] pt_installer.receive {
            t_installer.stop ;
            setverdict( fail ) ;    // Unexpected message received
        }
        [] alt_timeGuard ( t_installer ) ;
    }
}
}

```

A.6 Modification of common file

Table A.6 — Test number T-6

Test name Modification of common file
Purpose Verifying that the common file in EUT is modified by the command of modifying common file from the test installer
Test conditions See Clause 9. T-2 shall be executed before this test.
Test procedures <pre> module TestCase6 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_commonFileData_m2 } ; // Contents of new common file modulepar { octetstring mp_managementData_m4 } ; // Contents of AMT after modification // These module parameters shall be provided prior to the execution of the test case. testcase tc_t6 () runs on TSSState1 { // See T-1 // Modify the common file sourceCode := "managerCertificate-e" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlIdentifier := "modification" ; aPLIdentifier := "commonFile-b2" ; applicationData := mp_commonFileData_m2 ; writeInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-e" ; resultCode := "OK" ; resultWriteInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } } alt_timeGuard (t_installer) ; </pre>

```

    }
// Check the communication with CALM medium
a_command := "Start testApplication-b2 with access to commonFile-b2" ;
pt_com.send( a_command ) ;    // Execution of the test application
t_com.start( 300.0 ) ;
alt {
    [] pt_com.receive( a_response ) {
        t_com.stop ;
        setverdict( pass ) ;
    }
    [] pt_com.receive {
        t_com.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_com ) ;
}
// Check the AMT contents
sourceCode    := "managerCertificate-e" ;    // A component of readAMTInstruction.
destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
pt_installer.send ( readAMTInstruction ) ;
t_installer.start( 300.0 ) ;

//
sourceCode    := a_OBEorWAEid ;
destinationCode := "managerCertificate-e"
managementData := mp_managementData_m4 ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ;
resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
alt {
    [] pt_installer.receive( resultReadAMTInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
}
}

```

A.7 Uninstallation of common file

Table A.7 — Test number T-7

Test name
Uninstallation of common file
Purpose
Verifying that the common file in EUT is uninstalled by the command of uninstalling common file from the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase7 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_managementData_d2 } ; // Contents of AMT after uninstallation // The module parameter shall be provided prior to the execution of the test case. testcase tc_t7 () runs on TSState1 { // See T-1 // Delete the common ffile sourceCode := "managerCertificate-f" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlId := "uninstallation" ; aPLIdentifier := "commonFile-c" ; writeInstruction := sourceCode & destinationCode & controlId & aPLIdentifier ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-f" ; resultCode := "OK" ; resultWriteInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } [] alt_timeGuard (t_installer) ; } } // Check the AMT contents </pre>

```

sourceCode      := "managerCertificate-f" ;    // A component of readAMTInstruction.
destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
pt_installer.send ( readAMTInstruction ) ;
t_installer.start( 300.0 ) ;

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-f" ;
managementData := mp_managementData_d2 ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ;
resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
alt {
    [] pt_installer.receive( resultReadAMTInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
}
}

```

A.8 Function of reading common file

Table A.8 — Test number T-8

Test name
Function of reading common file
Purpose
Verifying that the resident common file in EUT is read by the command of reading common file from the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase8 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_commonFileData_m1 } ; // Contents of resident common file // The module parameter shall be provided prior to the execution of the test case. testcase tc_t8 () runs on TSSState1 { // See T-1 // Read out the common file sourceCode := "managerCertificate-e" ; // A component of readInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlIdentifier := "readout" ; aPLIdentifier := "commonFile-b1" ; readInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier ; readInstruction := '03'O & readInstruction ; // Message header addition pt_installer.send (readInstruction) t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-e" ; applicationData := mp_commonFileData_m1 ; resultReadInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier ; resultReadInstruction := resultReadInstruction & applicationData ; resultReadInstruction := '04'O & resultReadInstruction ; // Message header addition alt { [] pt_installer.receive(resultReadInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } } alt_timeGuard (t_installer) ; } } </pre>

A.9 Function of reading AMT information

Table A.9 — Test number T-9

Test name Function of reading AMT information
Purpose Verifying that the AMT information in EUT is read by the command of reading AMT information from the test installer
Test conditions See Clause 9.
Test procedures <pre> module TestCase9 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_managementData_m1 } ; // Contents of AMT // The module parameter shall be provided prior to the execution of the test case. testcase tc_t9 () runs on TSSState1 { // See T-1 // Read out the AMT contents sourceCode := "managerCertificate-a" ; // A component of readAMTInstruction. destinationCode := a_OBEorWAEid ; controlIdentifier := "AMT" ; readAMTInstruction := sourceCode & destinationCode & controlIdentifier ; readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition pt_installer.send (readAMTInstruction) ; t_installer.start(300.0) ; // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-a" ; managementData := mp_managementData_m1 ; resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ; resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition alt { [] pt_installer.receive(resultReadAMTInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } [] alt_timeGuard (t_installer) ; } } } </pre>

A.10 Function of the scheduled update

Table A.10 — Test number T-10

Test name
Function of the scheduled update
Purpose
Verifying that the application program in EUT is changed from the old one to new one after the scheduled time and date
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase10 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_applicationData_i3 } ; // Contents of a new application file modulepar { octetstring mp_managementData_i3 } ; // Contents of AMT // These module parameters shall be provided prior to the execution of the test case. template charstring a_date := "yyyymmdd" ; // Date after the scheduled time & date template charstring a_time := "hhmmss" ; // Waiting function function f_wait (in float p_duration) { timer t ; t.start (p_duration) ; alt { [] t.timeout {} ; [else] { repeat } } return ; } testcase tc_t10 () runs on TSState1 { // See T-1 // Install the test application sourceCode := "managerCertificate-a" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlIdentifier := "installation" ; aPLIdentifier := "testApplication-a1" ; applicationData := mp_applicationData_i3 ; writeInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-a" ; </pre>

```

resultCode      := "OK" ;
resultWriteInstruction := sourceCode & destinationCode & controlIdIdentifier & aPLIdentifier & resultCode ;
resultWriteInstruction := ' 02'O & resultWriteInstruction ;    // Message header addition
alt {
    [] pt_installer.receive( resultWriteInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
// Change the clock data
pt_vmeans.send( a_date & a_time ) ; // Setting the clock data
f_wait ( 120.0 ) ;    // New application program is activated within waiting duration
// Check the communication with CALM medium
a_command := "Start testApplication-a1" ;
pt_com.send( a_command ) ;
t_com.start( 300.0 ) ;
t_vmeans.start( 300.0 ) ;
alt {
    [] pt_com.receive( a_response ) {
        t_com.stop ;
        setverdict( pass ) ;
    }
    [] pt_com.receive {
        t_com.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_com ) ;
}
// Check the test result
alt {
    [] pt_vmeans.receive( a_testresult ) {
        t_vmeans.stop ;
        setverdict( pass ) ;
    }
    [] pt_vmeans.receive {
        t_vmeans.stop ;
        setverdict( fail ) ;    // Unexpected message received
    }
    [] alt_timeGuard ( t_vmeans ) ;
}
// Check the AMT contents
sourceCode      := "managerCertificate-a" ;    // A component of readAMTInstruction.

```

```

destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
pt_installer.send ( readAMTInstruction ) ;
t_installer.start( 300.0 ) ;

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-a" ;
managementData := mp_managementData_i3 ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdentifier & managementData ;
resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
alt {
    [] pt_installer.receive( resultReadAMTInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ; // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
}
}

```

NOTE The new application program shall contain the scheduled time and date information. The application program will be activated after the scheduled time and date and until then it remains inactive.

Annex B (normative)

Advanced tests

B.1 Operator authentication

Table B.1 — Test number T-11

Test name
Operator authentication
Purpose
Verifying that the test installer does not authenticate the test operator when the test operator inputs wrong security information for operator authentication to the test installer
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase11 { type port Operator message { out a_command ; in a_response ; } type component TSState2 { template charstring a_command := "?????" ; // Wrong security information for operator authentication template charstring a_response := "Operator authentication error" ; timer t_operator ; port Operator pt_operator ; } testcase tc_t11 () runs on TSState2 { pt_operator.send (a_command) ; t_operator.start(300.0) ; alt { [] pt_operator.receive(a_response) ; t_operator.stop ; setverdict(pass) ; } [] pt_operator.receive { t_operator.stop ; setverdict(fail) ; } [] t_operator.timeout { setverdict(fail) ; } } } } </pre>
NOTE In this test case, the test installer corresponds to EUT and the test operator corresponds to the test system, respectively.

B.2 Manager certificate

Table B.2 — Test number T-12

Test name Manager certificate
Purpose Verifying that the access is denied when the test operator sends the command of installing, uninstalling or modifying application by wrong manager certificate
Test conditions See Clause 9.
Test procedures <pre> module TestCase12 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_applicationData_i1 } ; // Contents of a new application file modulepar { octetstring mp_managementData_m1 } ; // Contents of AMT // These module parameters shall be provided prior to the execution of the test case. testcase tc_t12 () runs on TSSState1 { // See T-1 // Install the test application sourceCode := "managerCertificate-b" ; // Wrong manager certificate destinationCode := a_OBEorWAEid ; controlId := "installation" ; // Installation can be replaced by modification or uninstallation aPLIdentifier := "testApplication-a" applicationData := mp_applicationData_i1 ; writeInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-b" ; resultCode := "Manager certificate error" ; resultWriteInstruction := sourceCode & destinationCode & controlId & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } } [] alt_timeGuard (t_installer) ; </pre>

```

    }
    // Check the AMT contents
    sourceCode      := "managerCertificate-a" ;    // A component of readAMTInstruction.
    destinationCode := a_OBEorWAEid ;
    controlIdIdentifier := "AMT" ;
    readAMTInstruction := sourceCode & destinationCode & controlIdIdentifier ;
    readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
    pt_installer.send ( readAMTInstruction ) ;
    t_installer.start( 300.0 ) ;

    //
    sourceCode      := a_OBEorWAEid ;
    destinationCode := "managerCertificate-a" ;
    managementData := mp_managementData_m1 ;
    resultReadAMTInstruction := sourceCode & destinationCode & controlIdIdentifier & managementData ;
    resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
    alt {
        [] pt_installer.receive( resultReadAMTInstruction ) {
            t_installer.stop ;
            setverdict( pass ) ;
        }
        [] pt_installer.receive {
            t_installer.stop ;
            setverdict( fail ) ;    // Unexpected message received
        }
        [] alt_timeGuard ( t_installer ) ;
    }
}
}

```

B.3 The same file name

Table B.3 — Test number T-13

Test name The same file name
Purpose Verifying that the installation of application is not executed when there is already the same file name in the AMT
Test conditions See Clause 9.
Test procedures <pre> module TestCase13 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_applicationData_m1 } ; // Contents of a new application file modulepar { octetstring mp_managementData_m1 } ; // Contents of AMT // These module parameters shall be provided prior to the execution of the test case. testcase tc_t13 () runs on TSSState1 { // See T-1 // Install the test application sourceCode := "managerCertificate-b" ; // Proper manager certificate destinationCode := a_OBEorWAEid ; controlIdler := "installation" ; aPLIdentifier := "testApplication-b1" ; applicationData := mp_applicationData_m1 ; writeInstruction := sourceCode & destinationCode & controlIdler & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. // sourceCode := a_OBEorWAEid ; destinationCode := "managerCertificate-b" ; resultCode := "The same file name error" ; resultWriteInstruction := sourceCode & destinationCode & controlIdler & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition alt { [] pt_installer.receive(resultWriteInstruction) { t_installer.stop ; setverdict(pass) ; } [] pt_installer.receive { t_installer.stop ; setverdict(fail) ; // Unexpected message received } } [] alt_timeGuard (t_installer) ; </pre>

```

    }
    // Check the AMT contents
    sourceCode      := "managerCertificate-b" ;    // A component of readAMTInstruction.
    destinationCode := a_OBEorWAEid ;
    controlIdIdentifier := "AMT" ;
    readAMTInstruction := sourceCode & destinationCode & controlIdIdentifier ;
    readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
    pt_installer.send ( readAMTInstruction ) ;
    t_installer.start( 300.0 ) ;

    //
    sourceCode      := a_OBEorWAEid ;
    destinationCode := "managerCertificate-b" ;
    managementData := mp_managementData_m1 ;
    resultReadAMTInstruction := sourceCode & destinationCode & controlIdIdentifier & managementData ;
    resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
    alt {
        [] pt_installer.receive( resultReadAMTInstruction ) {
            t_installer.stop ;
            setverdict( pass ) ;
        }
        [] pt_installer.receive {
            t_installer.stop ;
            setverdict( fail ) ;    // Unexpected message received
        }
        [] alt_timeGuard ( t_installer ) ;
    }
}
}

```

B.4 Lack of sufficient available memory space

Table B.4 — Test number T-14

Test name
Lack of sufficient available memory space
Purpose
Verifying that the installation and modification of application are not executed when there is not sufficient available memory space in the EUT
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase14 { import from TestSystem1 all ; // See T-1 modulepar { octetstring mp_applicationData_i1 } ; // Contents of a new application file modulepar { octetstring mp_managementData_i } ; // Contents of AMT // These module parameters shall be provided prior to the execution of the test case. testcase tc_t14 () runs on TSState1 { // See T-1 // Rewrite the data on available memory space in AMT a_vcommand := "Set available memory space to zero" ; a_testresult := "Executed" ; pt_vmeans.send(a_vcommand) ; // Rewrite the available memory space data t_vmeans.start(60.0) ; alt { [] pt_vmeans.receive(a_testresult) { t_vmeans.stop ; setverdict(pass) ; } [] pt_vmeans.receive { t_vmeans.stop ; setverdict(fail) ; // Unexpected message received } [] alt_timeGuard (t_vmeans) ; } // Check the available memory space error sourceCode := "managerCertificate-a" ; // A component of writeInstruction. Proper certificate. destinationCode := a_OBEorWAEid ; controlIdentifier := "installation" ; aPLIdentifier := "testApplication-a" ; applicationData := mp_applicationData_i1 ; writeInstruction := sourceCode & destinationCode & controlIdentifier & aPLIdentifier & applicationData ; writeInstruction := '01'O & writeInstruction ; // Message header addition pt_installer.send (writeInstruction) ; t_installer.start(300.0) ; // The duration can be changed. </pre>

```

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-a" ;
resultCode      := "Lack of sufficient available memory space" ;
resultWriteInstruction := sourceCode & destinationCode & controlIdIdentifier & aPLIdentifier & resultCode ;
resultWriteInstruction := '02'O & resultWriteInstruction ; // message header addition
alt {
    [] pt_installer.receive( resultWriteInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ; // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
// Check the AMT contents
sourceCode      := "managerCertificate-a" ; // A component of readAMTInstruction.
destinationCode := a_OBEorWAEid ;
controlIdentifier := "AMT" ;
readAMTInstruction := sourceCode & destinationCode & controlIdIdentifier ;
readAMTInstruction := '05'O & readAMTInstruction ; // Message header addition
pt_installer.send ( readAMTInstruction ) ;
t_installer.start( 300.0 ) ;

//
sourceCode      := a_OBEorWAEid ;
destinationCode := "managerCertificate-a" ;
managementData := mp_managementData_i ;
resultReadAMTInstruction := sourceCode & destinationCode & controlIdIdentifier & managementData ;
resultReadAMTInstruction := '06'O & resultReadAMTInstruction ; // Message header addition
alt {
    [] pt_installer.receive( resultReadAMTInstruction ) {
        t_installer.stop ;
        setverdict( pass ) ;
    }
    [] pt_installer.receive {
        t_installer.stop ;
        setverdict( fail ) ; // Unexpected message received
    }
    [] alt_timeGuard ( t_installer ) ;
}
}
}

```

B.5 Prohibition of AME start while an application is running

Table B.5 — Test number T-15

Test name Prohibition of AME start while an application is running
Purpose Verifying that the installation, uninstallation and modification of application are not executed while an application is running in the EUT
Test conditions See Clause 9.
Test procedures <pre> module TestSystem2 { type port TestPort message { inout TestMessage } type component ConcurrentTester { port TestPort pt_installer ; port TestPort pt_com ; port TestPort pt_vmeans ; } type component EmptyComponent {} type component TestEntity { port TestPort pt ; } } module TestCase15 { import from TestSystem2 all ; // See above in this test procedure import from functions all ; // See below in this test procedure testcase tc_15() runs on EmptyComponent system ConcurrentTester { var TestEntity v_installer := TestEntity.create ; // Test component creation as initialization var TestEntity v_com := TestEntity.create ; var TestEntity v_vmeans := TestEntity.create ; modulepar { octetstring mp_managementData_i } ; // Contents of AMT // The module parameter shall be provided prior to the execution of the test case. template charstring a_command := "Send back AMT contents" ; template octetstring a_response := mp_managementData_i ; template charstring a_rsp := "Application is running" ; var integer v_para ; timer t_guard ; map(v_installer:pt, system:pt_installer) ; </pre>

```

map( v_com:pt,    system:pt_com );
map( v_vmeans:pt, system:pt_vmeans );
//
t_guard.start( 300.0 );
v_com.start ( f_com( ) );           // Test application starts
    if ( v_com.running ) {
        v_installer.start ( f_installer() );    // Installation starts
    }
    alt {
        [] pt_installer.receive( a_rsp ) {
            setverdict( pass );
        }
        [] pt_installer.receive {
            t_guard.stop ;
            setverdict( fail ) ;
        }
        [] t_guard.timeout {
            setverdict( fail ) ;
        }
    }
//
v_means.start( f_vmesans( a_command, a_response, v-param ) ); // AMT contents
alt {
    [] if ( v_param == 1 ) {
        t_guard.stop ;
    }
    [] t_guard.timeout {
        setverdict( fail ) ;
    }
}
unmap( v_installer:pt, system:pt_installer );
unmap( v_com:pt,    system:pt_com );
unmap( v_vmeans:pt, system:pt_vmeans );
}

module functions {
//
function f_installer () runs on TestEntity {           // Installation
import from writeInstruction language "ASN.1:2002" all ;
modulepar { octetstring mp_applicationData_i1 } ;      // Contents of application file
    // The module parameter shall be provided prior to the execution of the test case.
//
sourceCode      := "managerCertificate-a" ;
destinationCode := "OBEorWAEid" ;    // OBE/WAE identification code, implementation matter
controlIdentifier := "installation" ;
aPLIdentifier    := "testApplication-a" ;
applicationData := mp_applicationData_i1 ;

```

```

writeInstruction := sourceCode & destinationCode & controlIdIdentifier & aPLIdentifier & applicationData ;
    writeInstruction := '01'O & writeInstruction ; // Message header addition
    pt_installer.send( writeInstruction ) ;
return ;
}

//
function f_com () runs on TestEntity { // Communication
    template charstring a_comcommand := "Start testApplication-b1" ;
    pt_com.send( a_comcommand ) ;
    return ;
}

//
function f_vmeans ( in charstring p_command, // AMT contents
                    in charstring p_response,
                    out integer p_para ) runs on TestEntity {
    p_para == 0 ;
    pt_vmeans.send( p_command ) ;
    alt {
        [] pt_vmeans.receive( p_response ) {
            setverdict( pass ) ;
            p_para ==1 ;
        }
        [] pt_vmeans.receive {
            setverdict( fail ) ;
        }
    }
    return ;
}
}

```

B.6 Prohibition of application start while AME is running

Table B.6 — Test number T-16

Test name Prohibition of application start while AME is running
Purpose Verifying that an application is not started in the EUT while the command of installing, uninstalling or modifying application is in execution in the EUT
Test conditions See Clause 9.
Test procedures <pre> module TestCase16 { import from TestSystem2 all ; // See T-15 import from functions all ; // See T-15 testcase tc_16 () runs on EmptyComponent system ConcurrentTester { var TestEntity v_installer := TestEntity.create ; // Test component creation as initialization var TestEntity v_com := TestEntity.create ; var TestEntity v_vmeans := TestEntity.create ; modulepar { octetstring mp_managementData_i1 } ; // Contents of AMT // The module parameter shall be provided prior to the execution of the test case. template charstring a_command := "Send back AMT contents" ; template octetstring a_response := mp_managementData_i1 ; template charstring a_rsp := "AME is running" ; var integer v_para ; timer t_guard ; map(v_installer:pt, system:pt_installer) ; map(v_com:pt, system:pt_com) ; map(v_vmeans:pt, system:pt_vmeans) ; t_guard.start(300.0) ; // v_installer.start (f_installer()) ; // Installation starts if (v_installer.running) { v_com.start (f_com()) ; // Test application starts while installation } alt { [] pt_com.receive(a_rsp) { setverdict(pass) ; } [] pt_installer.receive { t_guard.stop ; setverdict(fail) ; } } } </pre>

```

        [] t_guard.timeout {
            setverdict( fail );
        }
    }

//
v_means.start( f_vmesans( a_command, a_response, v_para )); // AMT contents
alt {
    [] if ( v_para ==1 ) {
        t_guard.stop ;
    }
    [] t_guard.timeout {
        setverdict( fail );
    }
}
unmap( v_installer:pt, system:pt_installer );
unmap( v_com:pt,    system:pt_com );
unmap( v_vmeans:pt, system:pt_vmeans );
}

```

B.7 Abnormal termination of download**Table B.7 — Test number T-17**

Test name
Abnormal termination of download
Purpose
Verifying operation of the EUT when the download is abnormally terminated
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase17 { import from TestSystem2 all ; // See T-15 import from functions all ; // See T-15 import from resultWriteInstruction language "ASN.1:2002" all ; testcase tc_17 () runs on EmptyComponent system ConcurrentTester { var TestEntity v_installer := TestEntity.create ; // Test component creation as initialization var TestEntity v_com := TestEntity.create ; var TestEntity v_vmeans := TestEntity.create ; modulepar { octetstring mp_managementData_i } ; template charstring a_command := "Send back AMT contents" ; template octetstring a_response := mp_managementData_i ; var integer v_para ; timer t_guard ; map(v_installer:pt, system:pt_installer) ; map(v_com:pt, system:pt_com) ; map(v_vmeans:pt, system:pt_vmeans) ; // t_guard.start(300.0) ; // Installation starts v_installer.start(f_installer()) ; if (v_installer.running) { v_installer.stop ; // Intentional and abnormal termination } // Confirm the result of writeInstruction sourceCode := "OBEorWAEid" ; destinationCode := "managerCertificate-a" ; controlIdIdentifier := "installation" ; aPLIdentifier := "testApplication-a" ; resultCode := "Abnormal termination of download" ; resultWriteInstruction := sourceCode & destinationCode & controlIdIdentifier & aPLIdentifier & resultCode ; resultWriteInstruction := '02'O & resultWriteInstruction ; // Message header addition </pre>

```

alt {
    [] pt_installer.receive( resultWriteInstruction ) {
        setverdict( pass );
    }

    [] pt_installer.receive {
        t_guard.stop ;
        setverdict( fail );
    }

    [] t_guard.timeout {
        setverdict( fail );
    }

}

// AMT contents
v_means.start( f_vmesans( a_command, a_response, v_para )) ;
alt {
    [] if ( v_para ==1 ) {
        t_guard.stop ;
    }

    [] t_guard.timeout {
        setverdict( fail );
    }

}

unmap( v_installer:pt, system:pt_installer );
unmap( v_com:pt,    system:pt_com );
unmap( v_vmeans:pt, system:pt_vmeans );
}

```

B.8 Archival records

Table B.8 — Test number T-18

Test name
Archival records
Purpose
Verifying that archival records are saved in the test installer
Test conditions
See Clause 9. T-2 shall be executed before this test.
Test procedures
<pre> module TestCase18 { type port Operator message { out a_command ; in a_response ; } type component TSState3 { modulepar { octetstring mp_archivalRecords } ; // Archival records with time stamp // The module parameter shall be provided prior to the execution of the test case. template charstring a_command := "Send back archival records" ; template octetstring a_response := mp_archivalRecords ; timer t_operator ; port Operator pt_operator ; } testcase tc_t18 () runs on TSState3 { pt_operator.send (a_command) ; t_operator.start(300.0) ; alt { [] pt_operator.receive(a_response) ; t_operator.stop ; setverdict(pass) ; } [] pt_operator.receive { t_operator.stop ; setverdict(fail) ; } [] t_operator.timeout { setverdict(fail) ; } } } } </pre>
NOTE 1 The record of the operator authentication, all operations requested and the results of operations are recorded in the test installer with time stamps.
NOTE 2 In this test case, the test installer corresponds to EUT and the test operator corresponds to the test system, respectively.

B.9 Restoration function

Table B.9 — Test number T-19

Test name
Restoration function
Purpose
Verifying the restoration function when modification of an application or a common file has failed
Test conditions
See Clause 9.
Test procedures
<pre> module TestCase19 { import from TestSystem2 all ; // See T-15 import from functions2 all ; testcase tc_19 () runs on EmptyComponent system ConcurrentTester { var TestEntity v_installer := TestEntity.create ; // Test component creation as initialization var TestEntity v_com := TestEntity.create ; var TestEntity v_vmeans := TestEntity.create ; modulepar { octetstring mp_managementData_m1 } ; // Old data // The module parameter shall be provided prior to the execution of the test case. template charstring a_send1 := "Break at the specified step" ; // A debugger may be used in implementation template charstring a_send2 := "Rewrite the downloaded data" ; template charstring a_send3 := "Continue the T-2 operation" ; template charstring a_command := "Send back AMT contents" ; template octetstring a_response := mp_managementData_m1 ; var charstring v_aplData ; var integer v_para ; timer t_guard ; map(v_installer:pt, system:pt_installer) ; map(v_com:pt, system:pt_com) ; map(v_vmeans:pt, system:pt_vmeans) ; t_guard.start(600.0) ; // Save resident application and modify to new application v_installer.start (f_installer1(v_aplData)) ; v_installer.done ; // Wait until the component has terminated // pt_vmeans.send(a_send1) ; // Halt under program execution // pt_vmeans.send(a_send2) ; // pt_vmeans.send(a_send3) ; // v_installer.start(f_installer2(v_aplData)) ; v_installer.done ; </pre>