



International Standard

ISO/IEC 15408-3

Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

Part 3: Security assurance components

*Sécurité de l'information, cybersécurité et protection de la vie
privée — Critères d'évaluation pour la sécurité des technologies
de l'information —*

Partie 3: Composants d'assurance de sécurité

**Fifth edition
2026-05**



Please share your feedback about
the standard. Scan the QR code
with your phone or click the link

[Customer Feedback Form](#)

ISO/IEC 15408-3:2026(en)



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2026

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

	Page
Foreword	x
Introduction	xi
1 Scope	1
2 Normative references	1
3 Terms and definitions	1
4 Overview	5
5 Assurance paradigm	5
5.1 General.....	5
5.2 CC approach.....	5
5.3 Assurance approach.....	5
5.3.1 General.....	5
5.3.2 Significance of vulnerabilities.....	5
5.3.3 Cause of vulnerabilities.....	6
5.3.4 CC assurance.....	6
5.3.5 Assurance through evaluation.....	6
5.4 CC evaluation assurance scale.....	7
6 Security assurance components	7
6.1 General.....	7
6.2 Assurance class structure.....	7
6.2.1 General.....	7
6.2.2 Class name.....	8
6.2.3 Class introduction.....	8
6.2.4 Class application notes.....	8
6.2.5 Assurance families.....	8
6.3 Assurance family structure.....	8
6.3.1 General.....	8
6.3.2 Family name.....	8
6.3.3 Family objectives.....	8
6.3.4 Component levelling.....	9
6.3.5 Family application notes.....	9
6.3.6 Assurance components.....	9
6.4 Assurance component structure.....	9
6.4.1 General.....	9
6.4.2 Component name.....	9
6.4.3 Component objectives.....	10
6.4.4 Component application notes.....	10
6.4.5 Component dependencies.....	10
6.4.6 Assurance elements.....	10
6.5 Assurance elements.....	11
6.6 Component taxonomy.....	11
7 Class APE Protection Profile (PP) evaluation	11
7.1 General.....	11
7.2 PP introduction (APE_INT).....	12
7.2.1 Objectives.....	12
7.2.2 PP introduction (APE_INT.1).....	13
7.3 Conformance claims (APE_CCL).....	13
7.3.1 Objectives.....	13
7.3.2 Conformance claims (APE_CCL.1).....	13
7.4 Security problem definition (APE_SPD).....	15
7.4.1 Objectives.....	15
7.4.2 Security problem definition (APE_SPD.1).....	15
7.5 Security objectives (APE_OBJ).....	15

ISO/IEC 15408-3:2026(en)

7.5.1	Objectives	15
7.5.2	Component levelling.....	15
7.5.3	Security objectives for the operational environment (APE_OBJ.1).....	15
7.5.4	Security objectives (APE_OBJ.2).....	16
7.6	Extended components definition (APE_ECD).....	17
7.6.1	Objectives	17
7.6.2	Extended components definition (APE_ECD.1).....	17
7.7	Security requirements (APE_REQ).....	18
7.7.1	Objectives	18
7.7.2	Component levelling.....	18
7.7.3	Direct rationale security requirements (APE_REQ.1).....	18
7.7.4	Derived security requirements (APE_REQ.2).....	19
8	Class ACE Protection Profile Configuration evaluation	20
8.1	General.....	20
8.2	PP-Module introduction (ACE_INT).....	22
8.2.1	Objectives	22
8.2.2	PP-Module introduction (ACE_INT.1).....	22
8.3	PP-Module conformance claims (ACE_CCL).....	22
8.3.1	Objectives	22
8.3.2	PP-Module conformance claims (ACE_CCL.1).....	23
8.4	PP-Module security problem definition (ACE_SPD).....	24
8.4.1	Objectives	24
8.4.2	PP-Module security problem definition (ACE_SPD.1).....	24
8.5	PP-Module security objectives (ACE_OBJ).....	24
8.5.1	Objectives	24
8.5.2	Component levelling.....	24
8.5.3	PP-Module security objectives for the operational environment (ACE_OBJ.1).....	24
8.5.4	PP-Module security objectives (ACE_OBJ.2).....	25
8.6	PP-Module extended components definition (ACE_ECD).....	26
8.6.1	Objectives	26
8.6.2	PP-Module extended components definition (ACE_ECD.1).....	26
8.7	PP-Module security requirements (ACE_REQ).....	27
8.7.1	Objectives	27
8.7.2	Component levelling.....	27
8.7.3	PP-Module direct rationale security requirements (ACE_REQ.1).....	27
8.7.4	PP-Module derived security requirements (ACE_REQ.2).....	28
8.8	PP-Module consistency (ACE_MCO).....	29
8.8.1	Objectives	29
8.8.2	PP-Module consistency (ACE_MCO.1).....	29
8.9	PP-Configuration consistency (ACE_CCO).....	30
8.9.1	Objectives	30
8.9.2	PP-Configuration consistency (ACE_CCO.1).....	30
9	Class ASE Security Target (ST) evaluation	32
9.1	General.....	32
9.2	ST introduction (ASE_INT).....	34
9.2.1	Objectives	34
9.2.2	ST introduction (ASE_INT.1).....	34
9.3	Conformance claims (ASE_CCL).....	34
9.3.1	Objectives	34
9.3.2	Conformance claims (ASE_CCL.1).....	35
9.4	Security problem definition (ASE_SPD).....	36
9.4.1	Objectives	36
9.4.2	Security problem definition (ASE_SPD.1).....	36
9.5	Security objectives (ASE_OBJ).....	36
9.5.1	Objectives	36
9.5.2	Component levelling.....	36
9.5.3	Security objectives for the operational environment (ASE_OBJ.1).....	36
9.5.4	Security objectives (ASE_OBJ.2).....	37

ISO/IEC 15408-3:2026(en)

9.6	Extended components definition (ASE_ECD).....	38
9.6.1	Objectives.....	38
9.6.2	Extended components definition (ASE_ECD.1).....	38
9.7	Security requirements (ASE_REQ).....	39
9.7.1	Objectives.....	39
9.7.2	Component levelling.....	39
9.7.3	Direct rationale security requirements (ASE_REQ.1).....	39
9.7.4	Derived security requirements (ASE_REQ.2).....	40
9.8	TOE summary specification (ASE_TSS).....	41
9.8.1	Objectives.....	41
9.8.2	Component levelling.....	41
9.8.3	TOE summary specification (ASE_TSS.1).....	41
9.8.4	TOE summary specification with architectural design summary (ASE_TSS.2).....	42
9.9	Consistency of composite product Security Target (ASE_COMP).....	42
9.9.1	Objectives.....	42
9.9.2	Component levelling.....	43
9.9.3	Application notes.....	43
9.9.4	Consistency of Security Target (ST) (ASE_COMP.1).....	44
10	Class ADV Development.....	44
10.1	General.....	44
10.2	Security architecture (ADV_ARC).....	50
10.2.1	Objectives.....	50
10.2.2	Component levelling.....	50
10.2.3	Application notes.....	50
10.2.4	Security architecture description (ADV_ARC.1).....	51
10.3	Functional specification (ADV_FSP).....	52
10.3.1	Objectives.....	52
10.3.2	Component levelling.....	52
10.3.3	Application notes.....	52
10.3.4	Basic functional specification (ADV_FSP.1).....	55
10.3.5	Security-enforcing functional specification (ADV_FSP.2).....	55
10.3.6	Functional specification with complete summary (ADV_FSP.3).....	56
10.3.7	Complete functional specification (ADV_FSP.4).....	57
10.3.8	Complete semi-formal functional specification with additional error information (ADV_FSP.5).....	57
10.3.9	Complete semi-formal functional specification with additional formal specification (ADV_FSP.6).....	58
10.4	Implementation representation (ADV_IMP).....	59
10.4.1	Objectives.....	59
10.4.2	Component levelling.....	59
10.4.3	Application notes.....	59
10.4.4	Implementation representation of the TSF (ADV_IMP.1).....	60
10.4.5	Complete mapping of the implementation representation of the TSF (ADV_IMP.2).....	61
10.5	TSF internals (ADV_INT).....	62
10.5.1	Objectives.....	62
10.5.2	Component levelling.....	62
10.5.3	Application notes.....	62
10.5.4	Well-structured subset of TSF internals (ADV_INT.1).....	62
10.5.5	Well-structured internals (ADV_INT.2).....	63
10.5.6	Minimally complex internals (ADV_INT.3).....	64
10.6	Formal TSF model (ADV_SPM).....	65
10.6.1	Objectives.....	65
10.6.2	Component levelling.....	65
10.6.3	Application notes.....	65
10.6.4	Formal TSF model (ADV_SPM.1).....	66
10.7	TOE design (ADV_TDS).....	67
10.7.1	Objectives.....	67
10.7.2	Component levelling.....	67

ISO/IEC 15408-3:2026(en)

10.7.3	Application notes	67
10.7.4	Basic design (ADV_TDS.1)	68
10.7.5	Architectural design (ADV_TDS.2)	69
10.7.6	Basic modular design (ADV_TDS.3)	70
10.7.7	Semi-Formal modular design (ADV_TDS.4)	71
10.7.8	Complete semi-formal modular design (ADV_TDS.5)	71
10.7.9	Complete semi-formal modular design with formal high-level design presentation (ADV_TDS.6)	72
10.8	Composite design compliance (ADV_COMP)	73
10.8.1	Objectives	73
10.8.2	Component levelling	73
10.8.3	Application notes	73
10.8.4	Design compliance with the base component-related user guidance, ETR for composite evaluation and report of the base component evaluation authority (ADV_COMP.1)	74
11	Class AGD guidance documents	75
11.1	General	75
11.2	Operational user guidance (AGD_OPE)	75
11.2.1	Objectives	75
11.2.2	Component levelling	76
11.2.3	Application notes	76
11.2.4	Operational user guidance (AGD_OPE.1)	76
11.3	Preparative procedures (AGD_PRE)	77
11.3.1	Objectives	77
11.3.2	Component levelling	77
11.3.3	Application notes	77
11.3.4	Preparative procedures (AGD_PRE.1)	78
12	Class ALC life cycle support	78
12.1	General	78
12.2	CM capabilities (ALC_CMC)	80
12.2.1	Objectives	80
12.2.2	Component levelling	81
12.2.3	Application notes	81
12.2.4	Labelling of the TOE (ALC_CMC.1)	81
12.2.5	Use of the CM system (ALC_CMC.2)	82
12.2.6	Authorization controls (ALC_CMC.3)	83
12.2.7	Production support, acceptance procedures and automation (ALC_CMC.4)	84
12.2.8	Advanced support (ALC_CMC.5)	85
12.3	CM scope (ALC_CMS)	87
12.3.1	Objectives	87
12.3.2	Component levelling	87
12.3.3	Application notes	87
12.3.4	TOE CM coverage (ALC_CMS.1)	87
12.3.5	Parts of the TOE CM coverage (ALC_CMS.2)	88
12.3.6	Implementation representation CM coverage (ALC_CMS.3)	89
12.3.7	Problem tracking CM coverage (ALC_CMS.4)	89
12.3.8	Development tools CM coverage (ALC_CMS.5)	90
12.4	Delivery (ALC_DEL)	91
12.4.1	Objectives	91
12.4.2	Component levelling	91
12.4.3	Application notes	91
12.4.4	Delivery procedures (ALC_DEL.1)	92
12.5	Developer environment security (ALC_DVS)	92
12.5.1	Objectives	92
12.5.2	Component levelling	92
12.5.3	Application notes	93
12.5.4	Identification of security controls (ALC_DVS.1)	93
12.5.5	Sufficiency of security controls (ALC_DVS.2)	93

ISO/IEC 15408-3:2026(en)

12.6	Flaw remediation (ALC_FLR)	94
12.6.1	Objectives	94
12.6.2	Component levelling	94
12.6.3	Application notes	94
12.6.4	Basic flaw remediation (ALC_FLR.1)	94
12.6.5	Flaw reporting procedures (ALC_FLR.2)	95
12.6.6	Systematic flaw remediation (ALC_FLR.3)	96
12.7	Development life cycle definition (ALC_LCD)	97
12.7.1	Objectives	97
12.7.2	Component levelling	97
12.7.3	Application notes	97
12.7.4	Developer defined life cycle processes (ALC_LCD.1)	98
12.7.5	Measurable life cycle model (ALC_LCD.2)	98
12.8	TOE development artefacts (ALC_TDA)	99
12.8.1	Objectives	99
12.8.2	Component levelling	99
12.8.3	Application notes	99
12.8.4	Uniquely identifying implementation representation (ALC_TDA.1)	100
12.8.5	Matching CMS scope of implementation representation (ALC_TDA.2)	101
12.8.6	Regenerate TOE with well-defined development tools (ALC_TDA.3)	103
12.9	Tools and techniques (ALC_TAT)	105
12.9.1	Objectives	105
12.9.2	Component levelling	105
12.9.3	Application notes	105
12.9.4	Well-defined development tools (ALC_TAT.1)	106
12.9.5	Compliance with implementation standards (ALC_TAT.2)	106
12.9.6	Compliance with implementation standards - all parts (ALC_TAT.3)	107
12.10	Integration of composition parts and consistency check of delivery procedures (ALC_COMP)	108
12.10.1	Objectives	108
12.10.2	Component levelling	108
12.10.3	Application notes	108
12.10.4	Integration of the dependent component into the related base component and consistency check for delivery and acceptance procedures (ALC_COMP.1)	108
13	Class ATE Tests	109
13.1	General	109
13.2	Coverage (ATE_COV)	110
13.2.1	Objectives	110
13.2.2	Component levelling	111
13.2.3	Evidence of coverage (ATE_COV.1)	111
13.2.4	Analysis of coverage (ATE_COV.2)	111
13.2.5	Rigorous analysis of coverage (ATE_COV.3)	112
13.3	Depth (ATE_DPT)	113
13.3.1	Objectives	113
13.3.2	Component levelling	113
13.3.3	Application notes	113
13.3.4	Testing: basic design (ATE_DPT.1)	113
13.3.5	Testing: security enforcing modules (ATE_DPT.2)	114
13.3.6	Testing: modular design (ATE_DPT.3)	115
13.3.7	Testing: implementation representation (ATE_DPT.4)	116
13.4	Functional tests (ATE_FUN)	116
13.4.1	Objectives	116
13.4.2	Component levelling	117
13.4.3	Application notes	117
13.4.4	Functional testing (ATE_FUN.1)	117
13.4.5	Ordered functional testing (ATE_FUN.2)	118
13.5	Independent testing (ATE_IND)	118
13.5.1	Objectives	118

ISO/IEC 15408-3:2026(en)

13.5.2	Component levelling.....	119
13.5.3	Application notes.....	119
13.5.4	Independent testing - conformance (ATE_IND.1).....	119
13.5.5	Independent testing - sample (ATE_IND.2).....	120
13.5.6	Independent testing - complete (ATE_IND.3).....	121
13.6	Composite functional testing (ATE_COMP).....	122
13.6.1	Objectives.....	122
13.6.2	Component levelling.....	122
13.6.3	Application notes.....	122
13.6.4	Composite product functional testing (ATE_COMP.1).....	123
14	Class AVA Vulnerability assessment.....	123
14.1	General.....	123
14.2	Application notes.....	124
14.3	Vulnerability analysis (AVA_VAN).....	124
14.3.1	Objectives.....	124
14.3.2	Component levelling.....	125
14.3.3	Vulnerability survey (AVA_VAN.1).....	125
14.3.4	Vulnerability analysis (AVA_VAN.2).....	126
14.3.5	Focused vulnerability analysis (AVA_VAN.3).....	127
14.3.6	Methodical vulnerability analysis (AVA_VAN.4).....	128
14.3.7	Advanced methodical vulnerability analysis (AVA_VAN.5).....	129
14.4	Composite vulnerability assessment (AVA_COMP).....	130
14.4.1	Objectives.....	130
14.4.2	Component levelling.....	130
14.4.3	Application notes.....	130
14.4.4	Composite product vulnerability assessment (AVA_COMP.1).....	131
15	Class ACO Composition.....	131
15.1	General.....	131
15.2	Composition rationale (ACO_COR).....	135
15.2.1	Objectives.....	135
15.2.2	Component levelling.....	135
15.2.3	Composition rationale (ACO_COR.1).....	136
15.3	Development evidence (ACO_DEV).....	136
15.3.1	Objectives.....	136
15.3.2	Component levelling.....	136
15.3.3	Application notes.....	136
15.3.4	Functional description (ACO_DEV.1).....	137
15.3.5	Basic evidence of design (ACO_DEV.2).....	137
15.3.6	Detailed evidence of design (ACO_DEV.3).....	138
15.4	Reliance of dependent component (ACO_REL).....	139
15.4.1	Objectives.....	139
15.4.2	Component levelling.....	139
15.4.3	Application notes.....	139
15.4.4	Basic reliance information (ACO_REL.1).....	140
15.4.5	Reliance information (ACO_REL.2).....	140
15.5	Composed TOE testing (ACO_CTT).....	141
15.5.1	Objectives.....	141
15.5.2	Component levelling.....	141
15.5.3	Application notes.....	141
15.5.4	Interface testing (ACO_CTT.1).....	141
15.5.5	Rigorous interface testing (ACO_CTT.2).....	142
15.6	Composition vulnerability analysis (ACO_VUL).....	143
15.6.1	Objectives.....	143
15.6.2	Component levelling.....	143
15.6.3	Application notes.....	143
15.6.4	Composition vulnerability review (ACO_VUL.1).....	144
15.6.5	Composition vulnerability analysis (ACO_VUL.2).....	144
15.6.6	Enhanced-Basic composition vulnerability analysis (ACO_VUL.3).....	145

ISO/IEC 15408-3:2026(en)

Annex A (informative) Development (ADV)	146
Annex B (informative) Composition (ACO)	167
Bibliography	176

ISO/IEC 15408-3:2026(en)

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.

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 document 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 or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and <https://patents.iec.ch>. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

For an explanation of the voluntary nature of standards, 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 www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*, in collaboration with the European Committee for Standardization (CEN) Technical Committee CEN/CLC/JTC 13, *Cybersecurity and data protection*, in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement).

This fifth edition cancels and replaces the fourth edition (ISO/IEC 15408-3:2022), which has been technically revised.

The main changes are as follows:

- the terminology has been reviewed and updated;
- a missing developer action element has been added (ADV_SPM.1.7D).

A list of all parts in the ISO/IEC 15408 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

ISO/IEC 15408-3:2026(en)

Introduction

Security assurance components, as defined in this document, are the basis for the security assurance requirements expressed in a Security Assurance Package, Protection Profile (PP), a PP-Module, a PP-Configuration, or a Security Target (ST).

These requirements establish a standard way of expressing the assurance requirements for TOEs. This document catalogues the set of assurance components, families and classes. It also defines evaluation criteria for PPs, PP-Configurations, PP-Modules and STs.

The audience for this document includes consumers, developers, technical working groups and evaluators of secure IT products. ISO/IEC 15408-1 provides additional information on the target audience of the ISO/IEC 15408 series, and on the use of the ISO/IEC 15408 series by the groups that comprise the target audience. These groups may use this document as follows:

- consumers, who use this document when selecting components to express assurance requirements to satisfy the security objectives expressed in a PP or ST, determining required levels of security assurance of the TOE;
- developers, who respond to actual or perceived consumer security requirements in constructing a TOE, reference this document when interpreting statements of assurance requirements and determining assurance approaches of TOEs;
- evaluators, who use the assurance requirements defined in this document as a mandatory statement of evaluation criteria when determining the assurance of TOEs and when evaluating PPs and STs.

This document uses bold type to highlight hierarchical relationships between requirements. This convention calls for the use of bold type for all new requirements.

For security assurance requirements, special verbs relating to mandatory evaluation activities are presented in bold italic type.

Several governmental organizations have contributed to the development of this version of the Common Criteria for Information Technology Security Evaluations. As the joint holders of the copyright in the Common Criteria for Information Technology Security Evaluations (called CC), they hereby grant non-exclusive license to ISO/IEC to use CC in the continued development/maintenance of the ISO/IEC 15408 series of standards. However, these governmental organizations retain the right to use, copy, distribute, translate, or modify CC as they see fit. More information on these agencies can be found at <https://commoncriteriaportal.org/cc/copyright/index.cfm>.

Information security, cybersecurity and privacy protection — Evaluation criteria for IT security —

Part 3: Security assurance components

1 Scope

This document specifies the security assurance requirements of the ISO/IEC 15408 series. It includes the individual assurance components from which the evaluation assurance levels and other packages contained in ISO/IEC 15408-5 are composed, and the criteria for evaluation of Protection Profiles (PPs), PP-Configurations, PP-Modules and Security Targets (STs).

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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/IEC 15408-1:2026, *Information security — Evaluation criteria for IT security — Part 1: Introduction and general model*

ISO/IEC 15408-2:2026, *Information security — Evaluation criteria for IT security — Part 2: Security functional components*

ISO/IEC 15408-4, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 4: Framework for the specification of evaluation methods and activities*

ISO/IEC 15408-5:2026, *Information security — Evaluation criteria for IT security — Part 5: Pre-defined packages of security requirements*

ISO/IEC 18045:2026, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Requirements and methodology for IT security evaluation*

ISO/IEC/IEEE 24765:2017, *Systems and software engineering — Vocabulary*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15408-1, ISO/IEC 15408-2, ISO/IEC 18045, ISO/IEC/IEEE 24765 and the following apply.

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

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>

ISO/IEC 15408-3:2026(en)**3.1
acceptance procedure**

procedure followed in order to accept one or more newly created or modified *configuration item(s)* (3.4) as part of the target of evaluation (TOE), or to move them to the next step of the life cycle

Note 1 to entry: These procedures identify the roles or individuals responsible for the acceptance and the criteria to be applied in order to decide on the acceptance.

Note 2 to entry: There are several types of acceptance situations some of which can overlap:

- acceptance of an item into the configuration management system for the first time, in particular as part of an integration process;
- progression of configuration items to the next life cycle phase at each stage of the construction of the TOE.

**3.2
action**

element that specifies an activity undertaken by an evaluator or a developer

Note 1 to entry: These actions are either explicitly stated as evaluator actions or implicitly derived from developer actions (implied evaluator actions) within assurance components. For further information, see 6.5.

**3.3
actual test result**

result of a test which can be different from the expected test result if the actual test result is not equal to the expected test result

**3.4
configuration item**

item or aggregation of hardware, software, or both that is designated for configuration management and treated as a single entity in the configuration management process, during the target of evaluation (TOE) *development* (3.12)

Note 1 to entry: These can be either parts of the TOE or objects related to the development of the TOE, e.g. evaluation documents or development tools. Configuration management items can be stored in the configuration management system directly (e.g. files) or by reference (e.g. hardware parts) together with their version.

**3.5
configuration list**

document listing all *configuration items* (3.4) for a specific product together with the exact version of each configuration management item relevant for a specific version of the complete product

Note 1 to entry: This list allows distinguishing the items belonging to the evaluated version of the product from other versions of these items belonging to other versions of the product. The final configuration management list is a specific document for a specific version of a specific product. The list can be an electronic document inside of a *configuration management tool* (3.10). In this case, it can be seen as a specific view into the system or a part of the system rather than an output of the system. However, for the practical use in an evaluation the configuration list will probably be delivered as a part of the evaluation documentation. The configuration list defines the items that are under the configuration management requirements of ALC_CMC (CM capabilities) (12.2).

**3.6
configuration management
CM**

discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a *configuration items* (3.4), control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements

[SOURCE: ISO/IEC/IEEE 24765:2017, 3.779, modified — the second and third definitions have been removed.]

ISO/IEC 15408-3:2026(en)**3.7
configuration management documentation
CM documentation**

documentation including *configuration lists* (3.5) and *configuration management plans* (3.8)

**3.8
configuration management plan
CM plan**

description of how the *configuration management system* (3.9) is used for the target of evaluation (TOE)

Note 1 to entry: The objective of issuing a configuration management plan is that staff members can see clearly what they have to do. From the point of view of the overall configuration management system, this can be seen as an output document (because it can be produced as part of the application of the configuration management system). From the point of view of the concrete project, it is a usage document because members of the project team use it in order to understand the steps that they have to perform during the project. The configuration management plan defines the usage of the system for the specific product; the same system can be used to a different extent for other products. The configuration management plan defines and describes the output of the configuration management system of a company which is used during the TOE development (3.12).

Note 2 to entry: The structure and content of a configuration management plan are presented in ISO 10007:2017, Annex A.

**3.9
configuration management system
CM system**

set of procedures and tools (including their documentation) used by a developer to develop and maintain configurations of their products during their life cycles

Note 1 to entry: Configuration management systems can have varying degrees of rigour and function. At higher levels, configuration management systems can be automated, with flaw remediation, change controls, and other tracking mechanisms.

**3.10
configuration management tool**

manually operated or automated tool realizing or supporting a *configuration management system* (3.10)

EXAMPLE Tools for the version management of the parts of the target of evaluation (TOE).

**3.11
delivery**

transmission of the finished target of evaluation (TOE) from the *production* (3.21) environment into the hands of the customer

Note 1 to entry: This product life cycle phase can include packaging and storage at the *development* (3.12) site, but does not include transportation of the unfinished TOE or parts of the TOE between different developers or different development sites.

**3.12
development**

product life cycle phase which is concerned with generating the implementation representation of the target of evaluation (TOE)

Note 1 to entry: Throughout the class ALC (life cycle support) (Clause 12) requirements, development, and related terms (developer, develop) are meant in the more general sense to comprise *development* (3.12) and *production* (3.21).

**3.13
evaluation deliverable**

resource required from the sponsor or developer by the evaluator or evaluation authority to perform one or more evaluation or evaluation oversight activities

ISO/IEC 15408-3:2026(en)**3.14****expected test result**

predefined correct system behavior or output under specified test conditions that serves as the basis for determining whether a test passes

3.15**exploitable vulnerability**

weakness in the target of evaluation (TOE) that can be used to violate the security functional requirements (SFRs) in the operational environment for the TOE

3.16**installation**

procedure performed by a human user embedding the target of evaluation (TOE) in its operational environment and putting it into an operational state

Note 1 to entry: This operation is performed normally only once, after receipt and acceptance of the TOE. The TOE is expected to be progressed to a configuration allowed by the Security Target (ST). If similar processes shall be performed by the developer, they are denoted as "generation" throughout class ALC (life cycle support) ([Clause 12](#)). If the TOE requires an initial start-up that does not need to be repeated regularly, this process would be classified as installation.

3.17**life cycle model**

framework containing the processes, activities, and tasks involved in the *development* ([3.12](#)), *operation* ([3.18](#)), and maintenance of a product, spanning the life of the system from the definition of its requirements to the termination of its use

[SOURCE: ISO/IEC/IEEE 24765:2017 3.2219, modified — the first and third definitions have been removed.]

3.18**operation**

<TOE life cycle> usage phase of the target of evaluation (TOE) including normal usage, administration, and maintenance of the TOE after *delivery* ([3.11](#)) and *preparation* ([3.20](#))

3.19**potential vulnerability**

suspected, but not confirmed, weakness

Note 1 to entry: Suspicion is by virtue of a postulated attack path to violate the security functional requirements (SFRs).

3.20**preparation**

activity in the life cycle phase of a product, comprising the customer's acceptance of the delivered target of evaluation (TOE) and its *installation* ([3.16](#))

Note 1 to entry: Preparation can include such things as booting, initialization, start-up and progressing the TOE to a state ready for operation.

3.21**production**

life cycle phase which consists of transforming the implementation representation into the implementation of the TOE, i.e. into a state acceptable for *delivery* ([3.12](#)) to the customer

Note 1 to entry: This phase can comprise manufacturing, integration, generation, internal transport, storage and labelling of the TOE.

3.22**residual vulnerability**

weakness that cannot be exploited in the operational environment for the target of evaluation (TOE), but that can be used to violate the security functional requirements (SFRs) by an attacker with greater attack potential than is anticipated in the operational environment for the TOE

ISO/IEC 15408-3:2026(en)

3.23

vulnerability

weakness in the TOE that can be used to violate the security functional requirements (SFRs) in some environment

4 Overview

[Clause 5](#) describes the paradigm used in the security assurance requirements of this document.

[Clause 6](#) describes the presentation structure of the assurance classes, families, components.

[Clause 7](#) to [Clause 15](#) provide the detailed definitions of the assurance classes.

[Annex A](#) provides further explanations and examples of the concepts behind the development class.

[Annex B](#) provides an explanation of the concepts behind composed target of evaluation (TOE) evaluations and the composition class.

5 Assurance paradigm

5.1 General

The purpose of this clause is to document the underlying approach to assurance of the ISO/IEC 15408 series. An understanding of this clause allows the reader to understand the rationale behind the assurance requirements of this document.

5.2 CC approach

The approach of the ISO/IEC 15408 series is that the threats to security and organisational security policy commitments should be clearly articulated and the proposed security controls be demonstrably sufficient for their intended purpose.

Furthermore, measures should be adopted that reduce the likelihood of vulnerabilities, the ability to exercise (i.e. intentionally exploit or unintentionally trigger) a vulnerability, and the extent of the damage that can occur from a vulnerability being exercised. Additionally, measures should be adopted that facilitate the subsequent identification of vulnerabilities and the elimination, mitigation and notification that a vulnerability has been exploited or triggered.

5.3 Assurance approach

5.3.1 General

The approach of the ISO/IEC 15408 series is to provide assurance based upon an evaluation of the IT product that must be trusted. Evaluation has been the traditional means of providing assurance and is the basis for prior evaluation criteria documents. In aligning the existing approaches, the ISO/IEC 15408 series adopts the same approach. The ISO/IEC 15408 series proposes measuring the validity of the documentation and the resulting IT product by expert evaluators with increasing emphasis on scope, depth and rigour.

The ISO/IEC 15408 series does not exclude, nor does it comment upon, the relative merits of other means of gaining assurance. Research continues with respect to alternative ways of gaining assurance. As mature alternative approaches emerge from these research activities, they can be considered for inclusion in the ISO/IEC 15408 series, which is structured so as to allow their future introduction.

5.3.2 Significance of vulnerabilities

It is assumed that there are threat agents that actively seek to exploit opportunities to violate security policies both for illicit gains and for well-intentioned, but nonetheless insecure actions. Threat agents can also accidentally trigger security vulnerabilities, causing harm to the organization. Due to the need to

ISO/IEC 15408-3:2026(en)

process sensitive information and the lack of availability of sufficiently trusted products, there is significant risk due to failures of IT. It is, therefore, likely that IT security breaches can lead to significant loss.

IT security breaches arise through the intentional exploitation or the unintentional triggering of vulnerabilities in the application of IT within business concerns.

Steps should be taken to prevent vulnerabilities arising in IT products. To the extent feasible, vulnerabilities should be:

- eliminated: active steps should be taken to expose, and remove or neutralize, all exercisable vulnerabilities;
- minimized: active steps should be taken to reduce, to an acceptable residual level, the potential impact of any exercise of a vulnerability;
- monitored: active steps should be taken to ensure that any attempt to exercise a residual vulnerability will be detected so that steps can be taken to limit the damage.

5.3.3 Cause of vulnerabilities

Vulnerabilities can arise through failures in:

- requirements: an IT product may possess all the functions and features required of it and still contain vulnerabilities that render it unsuitable or ineffective with respect to security;
- design: an IT product has been poorly designed. Building a secure product, system, or application requires not only the implementation of functional requirements but also an architecture that allows for the effective enforcement of specific security properties the product, system, or application is supposed to enforce. The ability to withstand attacks the product, system, or application can be faced in its intended operational environment is highly dependent on an architecture that prohibits those attacks. If such attacks cannot be prohibited, the environment should allow for detection of such attacks and limitation of the damage caused by such an attack;
- development: an IT product does not meet its specifications and vulnerabilities have been introduced as a result of poor development standards or incorrect design choices;
- delivery, installation and configuration: an IT product has vulnerabilities introduced during the delivery, installation and configuration of the product;
- operation: an IT product has been constructed correctly to a correct specification, but vulnerabilities have been introduced as a result of inadequate controls upon the operation;
- maintenance: an IT product is maintained in such a way that new vulnerabilities are introduced.

5.3.4 CC assurance

Assurance can be derived from reference to sources such as unsubstantiated assertions, prior relevant experience, or specific experience. However, the ISO/IEC 15408 series provides assurance through active investigation or a specification-based approach. Active investigation is an evaluation of the IT product in order to determine its security properties.

5.3.5 Assurance through evaluation

Evaluation has been the traditional means of gaining assurance and is the basis of the ISO/IEC 15408 series approach. Evaluation techniques can include, but are not limited to:

- analysis and checking of process(es) and procedure(s);
- checking that process(es) and procedure(s) are being applied;
- analysis of the correspondence between TOE design representations;
- analysis of the TOE design representation against the requirements;

ISO/IEC 15408-3:2026(en)

- verification of proofs;
- analysis of guidance documents;
- analysis of functional tests developed and the results provided;
- independent functional testing;
- analysis for vulnerabilities (including flaw hypotheses);
- penetration testing;
- analysis of the delivery process;
- analysis of the maintenance process.

5.4 CC evaluation assurance scale

The approach of the ISO/IEC 15408 series asserts that greater assurance results from the application of greater evaluation effort, and that the goal is to apply the minimum effort required to provide the necessary assurance. The increasing level of effort is based upon:

- scope: the effort is greater because a larger portion of the IT product is included;
- depth: the effort is greater because it is deployed to a finer level of design and implementation detail;
- rigour: the effort is greater because it is applied in a more structured, formal manner.

6 Security assurance components

6.1 General

This clause describes the constructs used in representing the assurance classes, families, and components that define the security assurance requirements (SARs) defined in this document.

The most abstract collection of SARs is referred to as a class. Each class contains assurance families, which then contain assurance components, which in turn contain assurance elements. Classes and families are used to provide a taxonomy for classifying SARs, while components are used to specify SARs in a PP, PP-Module, assurance package or ST.

6.2 Assurance class structure

6.2.1 General

[Figure 1](#) illustrates the assurance class structure in diagrammatic form. Each assurance class includes a class name, class introduction, an optional application notes, and one or more assurance families.

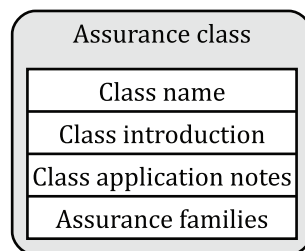


Figure 1 — Assurance class structure

6.2.2 Class name

Each assurance class is assigned a unique name. The name indicates the topics covered by the assurance class.

A unique short form of the assurance class name is also provided. This is the primary means for referencing the assurance class. The convention adopted is an "A" followed by two letters related to the class name.

6.2.3 Class introduction

Each assurance class has an introductory subclause that describes the composition of the class and contains supportive text covering the intent of the class.

The class introduction includes a class decomposition diagram, showing the families and components of the class. It also provides a dependency table that shows the hierarchical, direct, indirect, and optional dependencies among assurance components, as explained in [6.4.5](#).

6.2.4 Class application notes

The optional application notes contain additional information that supports the interpretation and understanding of the concepts, structure and usage of the assurance class.

6.2.5 Assurance families

Each assurance class contains at least one assurance family. The structure of the assurance families is described in [6.3](#).

6.3 Assurance family structure

6.3.1 General

[Figure 2](#) illustrates the assurance family structure in diagrammatic form.

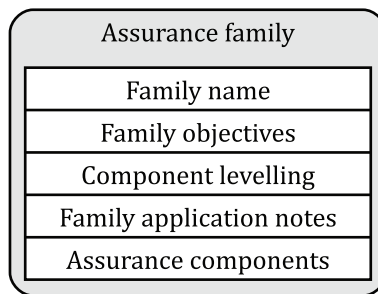


Figure 2 — Assurance family structure

6.3.2 Family name

Every assurance family is assigned a unique name. The name provides descriptive information about the topics covered by the assurance family. Each assurance family is placed within the assurance class that contains other families with the same intent.

A unique short form of the assurance family name is also provided. This is the primary means used to reference the assurance family. The convention adopted is that the short form of the class name is used, followed by an underscore, and then three letters related to the family name.

6.3.3 Family objectives

The objectives subclause of the assurance family presents the intent of the assurance family.

ISO/IEC 15408-3:2026(en)

This subclause describes the objectives, particularly those related to the ISO/IEC 15408 series assurance paradigm, that the family is intended to address. The description for the assurance family is kept at a general level. Any specific details required for objectives are incorporated in the particular assurance component.

6.3.4 Component levelling

Each assurance family contains one or more assurance components. This subclause of the assurance family describes the components available and explains the distinctions between them. Its main purpose is to differentiate between the assurance components once it has been determined that the assurance family is a necessary or useful part of the SARs for a PP/ST.

Assurance families containing more than one component are levelled and rationale is provided as to how the components are levelled. This rationale is in terms of scope, depth, rigour or a combination of them.

6.3.5 Family application notes

The application notes subclause of the assurance family, if present, contains additional information for the assurance family. This information should be of particular interest to users of the assurance family (e.g. PP and ST authors, designers of TOEs, evaluators). The presentation is informal and covers, for example, warnings about limitations of use and areas where specific attention may be required.

6.3.6 Assurance components

Each assurance family has at least one assurance component. The structure of the assurance components is provided in [6.4](#).

6.4 Assurance component structure

6.4.1 General

[Figure 3](#) illustrates the assurance component structure.

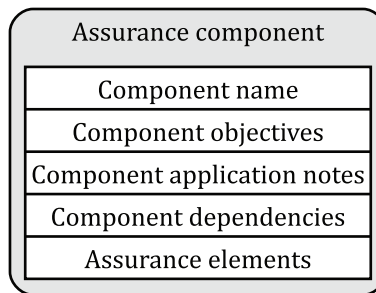


Figure 3 — Assurance component structure

The relationship between components within a family is highlighted using a bolding convention. Those parts of the requirements that are new, enhanced or modified beyond the requirements of the previous component within a hierarchy are bolded.

6.4.2 Component name

The component name provides descriptive information necessary to identify, categorize, register, and reference a component.

Every assurance component is assigned a unique name. The name provides descriptive information about the topics covered by the assurance component. Each assurance component is placed within the assurance family that shares its security objective.

ISO/IEC 15408-3:2026(en)

A unique short form of the assurance component name is also provided. This is the primary means used to reference the assurance component. The convention used is that the short form of the family name is used, followed by a period, and then a numeric character. The numeric characters for the components within each family are assigned sequentially, starting from 1.

6.4.3 Component objectives

The objectives subclause of the assurance component, if present, contains specific objectives for the particular assurance component. For those assurance components that have this subclause, it presents the specific intent of the component and a more detailed explanation of the objectives.

6.4.4 Component application notes

The application notes subclause of an assurance component, if present, contains additional information to facilitate the use of the component.

6.4.5 Component dependencies

Dependencies among assurance components arise when a component is not self-sufficient and relies upon the presence of another component.

Each assurance component provides a complete list of dependencies to other assurance components. In some cases, two or more assurance components can be selected alternatively which is indicated by square brackets. Within the square brackets the selectable assurance components are listed. Furthermore, some components may list "No dependencies", to indicate that no dependencies have been identified. The components depended upon may have dependencies on other components.

The dependency list identifies the minimum set of assurance components which are relied upon. Components which are hierarchical to a component in the dependency list may also be used to satisfy the dependency.

In specific situations, it is possible that the indicated dependencies will not be applicable. The PP, PP-Module, PP-Configuration or ST author, by providing rationale for why a given dependency is not applicable, may elect not to satisfy that dependency.

The dependencies of assurance components are shown in a table at the class introduction clause, where each of the components that is a dependency of some other assurance component is allocated a column. Each assurance component is allocated a row. The value in the table cell indicates whether the column label component is directly required (indicated by a cross "X"), indirectly required (indicated by a dash "-"), or optionally required (indicated by a "O") by the row label component. Sets of optional requirements are indicated by using an entry of type O^x (e.g. O^1) whereby a SAR with such an entry requires at minimum only one of the SARs contained in that O^x -set.

6.4.6 Assurance elements

A set of assurance elements is provided for each assurance component. An assurance element is a security requirement which, if further divided, would not yield a meaningful evaluation result. It is the smallest security requirement recognized in the ISO/IEC 15408 series.

Each assurance element is identified as belonging to one of the three sets of assurance elements:

- developer action elements: the activities that shall be performed by the developer. This set of actions is further qualified by evidential material referenced in the following set of elements. Requirements for developer actions are identified by appending the letter "D" to the element number;
- content and presentation of evidence elements: the evidence required, what the evidence shall demonstrate, and what information the evidence shall convey. Requirements for content and presentation of evidence are identified by appending the letter "C" to the element number;
- evaluator action elements: the activities that shall be performed by the evaluator. This set of actions explicitly includes confirmation that the requirements prescribed in the content and presentation of

ISO/IEC 15408-3:2026(en)

evidence elements have been met. It also includes explicit actions and analysis that shall be performed in addition to that already performed by the developer. Implicit evaluator actions shall also be performed as a result of developer action elements which are not covered by content and presentation of evidence requirements. Requirements for evaluator actions are identified by appending the letter "E" to the element number.

The developer actions and content and presentation of evidence define the assurance requirements that are used to represent a developer's responsibilities in demonstrating assurance in the TOE meeting the SFRs of a PP, PP-Module, PP-Configuration or ST.

The evaluator actions define the evaluator's responsibilities in two aspects of evaluation. The first aspect is validation of the applicable PP, PP-Module, PP-Configuration or ST, according to class ACE (Protection Profile Configuration evaluation) (Clause 8), class APE (Protection Profile (PP) evaluation) (Clause 7) and class ASE (Security Target (ST) evaluation) (Clause 9). The second aspect is verification of the TOE's conformance with its SFRs and SARs. By demonstrating that the PP, PP-Module, PP-Configuration or ST is valid and that the requirements are met by the TOE, the evaluator can provide a basis for confidence that the TOE in its operational environment solves the defined security problem.

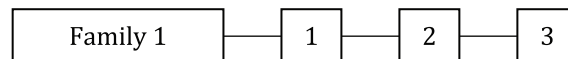
The developer action elements, content and presentation of evidence elements, and explicit evaluator action elements, identify the evaluator effort that shall be expended in verifying the security claims made in the ST of the TOE.

6.5 Assurance elements

Each element represents a requirement to be met. These statements of requirements are intended to be clear, concise and unambiguous. Therefore, there are no compound sentences: each separable requirement is stated as an individual element.

6.6 Component taxonomy

This document contains classes of families and components that are grouped on the basis of related assurance. At the start of each class is a diagram that indicates the families in the class and the components in each family.



Key

1,2,3 components

Figure 4 — Sample class decomposition diagram

In [Figure 4](#), the class as shown contains a single family. The family contains three components that are linearly hierarchical (i.e. component 2 requires more than component 1, in terms of specific actions, specific evidence, or rigour of the actions or evidence). The assurance families in this document are all linearly hierarchical, although linearity is not a required criterion for assurance families that can be added in the future.

7 Class APE Protection Profile (PP) evaluation

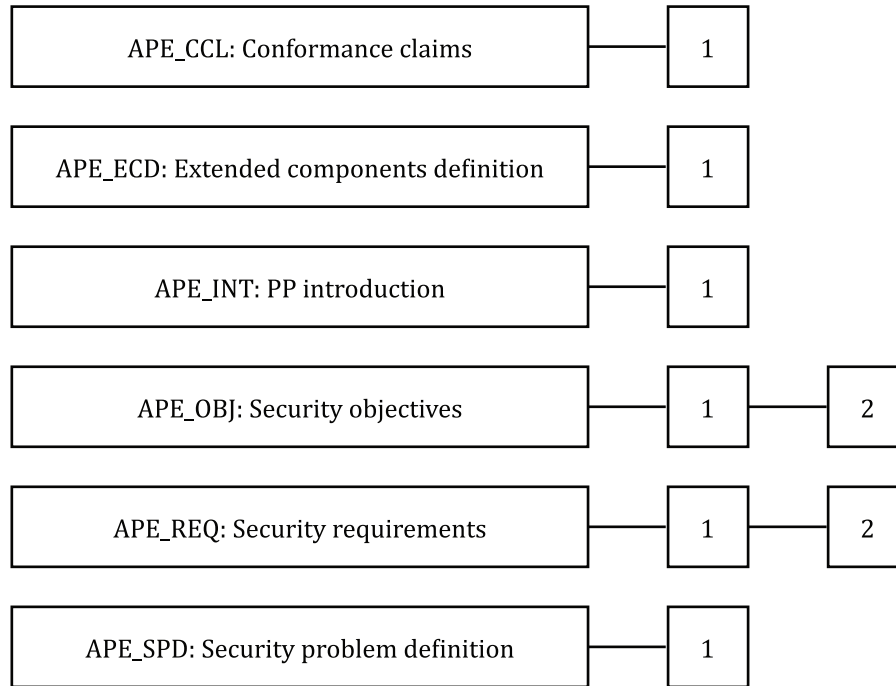
7.1 General

Evaluating a PP is required to demonstrate that the PP is sound and internally consistent, and, if the PP is based on one or more additional PPs or on packages, that the PP is a correct instantiation of these PPs and packages. These properties are necessary for the PP to be suitable for use as the basis for writing an ST or another PP.

This clause should be used in conjunction with ISO/IEC 15408-1:2026, Annexes B and D, as these annexes clarify the concepts discussed and provide many examples.

ISO/IEC 15408-3:2026(en)

Figure 5 describes the family hierarchy of class APE.



Key
 1,2 components

Figure 5 — APE: Protection Profile (PP) evaluation, class decomposition

Direct, indirect and optional dependencies for the components of class APE are shown in Table 1.

Table 1 — Dependency table for class APE: Protection Profile (PP) evaluation

	APE_ECD.1 (7.6.2)	APE_INT.1 (7.2.2)	APE_OBJ.1 (7.5.3)	APE_OBJ.2 (7.5.4)	APE_REQ.1 (7.7.3)	APE_SPD.1 (7.4.2)
APE_CCL.1 (7.3.2)	X	X	-		X	-
APE_ECD.1 (7.6.2)						
APE_INT.1 (7.2.2)						
APE_OBJ.1 (7.5.3)						X
APE_OBJ.2 (7.5.4)						X
APE_REQ.1 (7.7.3)	X		X			X
APE_REQ.2 (7.7.4)	X			X		-
APE_SPD.1 (7.4.2)						

7.2 PP introduction (APE_INT)

7.2.1 Objectives

The objective of this family is to describe the TOE in a narrative way.

Evaluation of the PP introduction is required to demonstrate that the PP is correctly identified, and that the PP reference and TOE overview are consistent with each other.

7.2.2 PP introduction (APE_INT.1)

[Table 2](#) specifies the developer action elements for APE_INT.1, with [Table 3](#) indicating their related content and presentation requirements, and [Table 4](#) the corresponding evaluation actions.

Table 2 — Developer action elements for APE_INT.1

APE_INT.1.1D	The developer shall provide a PP introduction.
--------------	--

Table 3 — Content and presentation elements for APE_INT.1

APE_INT.1.1C	The PP introduction shall contain a PP reference and a TOE overview.
APE_INT.1.2C	The PP reference shall uniquely identify the PP.
APE_INT.1.3C	The TOE overview shall summarize the usage and major security features of the TOE.
APE_INT.1.4C	The TOE overview shall identify the TOE type.
APE_INT.1.5C	The TOE overview shall identify any non-TOE hardware, software, firmware available to the TOE.

Table 4 — Evaluator action elements for APE_INT.1

APE_INT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

7.3 Conformance claims (APE_CCL)

7.3.1 Objectives

The objective of this family is to determine the validity of the conformance claim and conformance statement. These may be based on different editions of the ISO/IEC 15408 series and derived PPs, and packages. In addition, this family specifies how STs and other PPs are to claim conformance with the PP.

NOTE Previous editions can refer to withdrawn and replaced documents.

7.3.2 Conformance claims (APE_CCL.1)

7.3.2.1 Component relationships

Dependencies: APE_INT.1 (PP introduction) ([7.2.2](#))
APE_ECD.1 (Extended components definition) ([7.6.2](#))
APE_REQ.1 (Direct rationale security requirements) ([7.7.3](#))

7.3.2.2 Component elements

[Table 5](#) specifies the developer action elements for APE_CCL.1, with [Table 6](#) indicating their related content and presentation requirements, and [Table 7](#) the corresponding evaluation actions.

Table 5 — Developer action elements for APE_CCL.1

APE_CCL.1.1D	The developer shall provide a conformance claim.
APE_CCL.1.2D	The developer shall provide a conformance claim rationale.
APE_CCL.1.3D	The developer shall provide a conformance statement.

ISO/IEC 15408-3:2026(en)

Table 6 — Content and presentation elements for APE_CCL.1

APE_CCL.1.1C	The conformance claim shall identify the edition of the ISO/IEC 15408 series to which the PP claims conformance.
APE_CCL.1.2C	The conformance claim shall describe the conformance of the PP to ISO/IEC 15408-2 as either "ISO/IEC 15408-2 conformant" or "ISO/IEC 15408-2 extended".
APE_CCL.1.3C	The conformance claim shall describe the conformance of the PP to ISO/IEC 15408-3 as either "ISO/IEC 15408-3 conformant" or "ISO/IEC 15408-3 extended".
APE_CCL.1.4C	The conformance claim shall be consistent with the extended components definition.
APE_CCL.1.5C	The conformance claim shall identify all PPs and packages to which the PP claims conformance.
APE_CCL.1.6C	The conformance claim shall describe any conformance of the PP to a functional package as one of package-conformant, package-augmented, or package-tailored.
APE_CCL.1.7C	The conformance claim shall describe any conformance of the PP to an assurance package as either package-conformant or package-augmented.
APE_CCL.1.8C	The conformance claim shall describe any conformance of the PP to another PP as PP-Conformant.
APE_CCL.1.9C	The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PP(s) to which conformance is being claimed.
APE_CCL.1.10C	The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs and any functional packages for which conformance is being claimed.
APE_CCL.1.11C	The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs and any functional packages for which conformance is being claimed.
APE_CCL.1.12C	The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs and any functional packages for which conformance is being claimed.
APE_CCL.1.13C	The conformance statement shall describe the conformance required of any PPs and STs to the PP as one of exact, strict, or demonstrable conformance.
APE_CCL.1.14C	For an exact conformance PP, the conformance statement shall contain an allowed-with statement that identifies the set of PPs (if any) to which, in combination with the PP under evaluation, exact conformance is allowed to be claimed.
APE_CCL.1.15C	For an exact conformance PP, the conformance statement shall contain an allowed-with statement that identifies the set of PP-Modules (if any) that are allowed to be used with the PP under evaluation in a PP-Configuration.
APE_CCL.1.16C	The conformance statement shall identify the set of derived evaluation methods and evaluation activities (if any) that shall be used with the PP under evaluation. This list shall contain: <ul style="list-style-type: none"> — any evaluation methods and evaluation activities that are specified for the PP under evaluation; — any evaluation methods and evaluation activities specified in conformance statements of PPs to which conformance is being claimed by the PP under evaluation; — any evaluation methods and evaluation activities specified in the Security Requirements sections of packages to which conformance is being claimed by the PP under evaluation.

Table 7 — Evaluator action elements for APE_CCL.1

APE_CCL.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

7.4 Security problem definition (APE_SPD)

7.4.1 Objectives

This part of the PP describes the security problem to be addressed by the TOE and the operational environment of the TOE.

Evaluation of the security problem definition is required to demonstrate that the security problem intended to be addressed by the TOE and its operational environment, is clearly defined.

7.4.2 Security problem definition (APE_SPD.1)

[Table 8](#) specifies the developer action elements for APE_SPD.1, with [Table 9](#) indicating their related content and presentation requirements, and [Table 10](#) the corresponding evaluation actions.

Table 8 — Developer action elements for APE_SPD.1

APE_SPD.1.1D	The developer shall provide a security problem definition.
--------------	---

Table 9 — Content and presentation elements for APE_SPD.1

APE_SPD.1.1C	The security problem definition shall describe the threats.
APE_SPD.1.2C	All threats shall be described in terms of a threat agent, an asset, and an adverse action.
APE_SPD.1.3C	The security problem definition shall describe the organizational security policies (OSPs).
APE_SPD.1.4C	The security problem definition shall describe the assumptions about the operational environment of the TOE.

Table 10 — Evaluator action elements for APE_SPD.1

APE_SPD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

7.5 Security objectives (APE_OBJ)

7.5.1 Objectives

The security objectives are a concise statement of the intended response to the security problem defined through the APE_SPD (Security problem definition) ([7.4](#)) family.

Evaluation of the security objectives is required to demonstrate that the security objectives adequately and completely address the security problem definition and that the division of this problem between the TOE and its operational environment is clearly defined.

7.5.2 Component levelling

The components in this family are levelled on whether they prescribe only security objectives for the operational environment, or also security objectives for the TOE.

7.5.3 Security objectives for the operational environment (APE_OBJ.1)

7.5.3.1 Component relationships

Dependencies: APE_SPD.1 (Security problem definition) ([7.4.2](#))

ISO/IEC 15408-3:2026(en)

7.5.3.2 Component elements

[Table 11](#) specifies the developer action elements for APE_OBJ.1, with [Table 12](#) indicating their related content and presentation requirements, and [Table 13](#) the corresponding evaluation actions.

Table 11 — Developer action elements for APE_OBJ.1

APE_OBJ.1.1D	The developer shall provide a statement of security objectives for the operational environment.
APE_OBJ.1.2D	The developer shall provide a security objectives rationale for the security objectives for the operational environment.

Table 12 — Content and presentation elements for APE_OBJ.1

APE_OBJ.1.1C	The statement of security objectives shall describe the security objectives for the operational environment.
APE_OBJ.1.2C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
APE_OBJ.1.3C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Table 13 — Evaluator action elements for APE_OBJ.1

APE_OBJ.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

7.5.4 Security objectives (APE_OBJ.2)

7.5.4.1 Component relationships

Dependencies: APE_SPD.1 (Security problem definition) ([7.4.2](#))

7.5.4.2 Component elements

[Table 14](#) specifies the developer action elements for APE_OBJ.2, with [Table 15](#) indicating their related content and presentation requirements, and [Table 16](#) the corresponding evaluation actions.

Table 14 — Developer action elements for APE_OBJ.2

APE_OBJ.2.1D	The developer shall provide a statement of security objectives.
APE_OBJ.2.2D	The developer shall provide a security objectives rationale.

ISO/IEC 15408-3:2026(en)

Table 15 — Content and presentation elements for APE_OBJ.2

APE_OBJ.2.1C	The statement of security objectives shall describe the security objectives for the TOE and the security objectives for the operational environment.
APE_OBJ.2.2C	The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.
APE_OBJ.2.3C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
APE_OBJ.2.4C	The security objectives rationale shall demonstrate that the security objectives counter all threats.
APE_OBJ.2.5C	The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.
APE_OBJ.2.6C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Table 16 — Evaluator action elements for APE_OBJ.2

APE_OBJ.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

7.6 Extended components definition (APE_ECD)

7.6.1 Objectives

Extended security requirements are requirements that are not based on components from ISO/IEC 15408-2 or this document, but which are based on extended components, i.e. components defined by the PP author.

Evaluation of the definition of extended components is necessary to determine that they are clear and unambiguous, and that they are necessary, i.e. it is not possible to express them clearly by using existing components from ISO/IEC 15408-2 or this document.

7.6.2 Extended components definition (APE_ECD.1)

[Table 17](#) specifies the developer action elements for APE_ECD.1, with [Table 18](#) indicating their related content and presentation requirements, and [Table 19](#) the corresponding evaluation actions.

Table 17 — Developer action elements for APE_ECD.1

APE_ECD.1.1D	The developer shall provide a statement of security requirements.
APE_ECD.1.2D	The developer shall provide an extended components definition.

Table 18 — Content and presentation elements for APE_ECD.1

APE_ECD.1.1C	The statement of security requirements shall identify all extended security requirements.
APE_ECD.1.2C	The extended components definition shall define an extended component for each extended security requirement.
APE_ECD.1.3C	The extended components definition shall describe how each extended component is related to the existing components, families, and classes of ISO/IEC 15408-2 and ISO/IEC 15408-3.
APE_ECD.1.4C	The extended components definition shall use the existing components, families, classes, and methodology specified in the ISO/IEC 15408 series as a model for presentation.
APE_ECD.1.5C	The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

ISO/IEC 15408-3:2026(en)**Table 19 — Evaluator action elements for APE_ECD.1**

APE_ECD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
APE_ECD.1.2E	The evaluator <i>shall confirm</i> that it is impossible to clearly express any extended component by using existing components.

7.7 Security requirements (APE_REQ)**7.7.1 Objectives**

The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour of the TOE. The SARs form a clear, unambiguous and well-defined description of the expected activities that will be undertaken to gain assurance in the TOE.

Evaluation of the security requirements is required to ensure that they are clear, unambiguous and well-defined.

7.7.2 Component levelling

The components in this family are levelled on whether the SFRs are derived from SPD, or whether the SFRs are derived from security objectives for the TOE.

7.7.3 Direct rationale security requirements (APE_REQ.1)**7.7.3.1 Component relationships**

Dependencies: APE_ECD.1 (Extended components definition) ([7.6.2](#))
 APE_OBJ.1 (Security objectives for the operational environment) ([7.5.3](#))
 APE_SPD.1 (Security problem definition) ([7.4.2](#))

7.7.3.2 Component elements

[Table 20](#) specifies the developer action elements for APE_REQ.1, with [Table 21](#) indicating their related content and presentation requirements, and [Table 22](#) the corresponding evaluation actions.

Table 20 — Developer action elements for APE_REQ.1

APE_REQ.1.1D	The developer shall provide a statement of security requirements.
APE_REQ.1.2D	The developer shall provide a security requirements rationale.

Table 21 — Content and presentation elements for APE_REQ.1

APE_REQ.1.1C	The statement of security requirements shall describe the SFRs and the SARs.
APE_REQ.1.2C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
APE_REQ.1.3C	The statement of security requirements shall identify all operations on the security requirements.
APE_REQ.1.4C	All operations shall be performed correctly.
APE_REQ.1.5C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
APE_REQ.1.6C	The security requirements rationale shall trace each SFR back to the threats countered by that SFR and the OSPs enforced by that SFR.
APE_REQ.1.7C	The security requirements rationale shall demonstrate that the SFRs, in conjunction with the security objectives for the operational environment, counter all threats for the TOE.

ISO/IEC 15408-3:2026(en)

Table 21 (continued)

APE_REQ.1.8C	The security requirements rationale shall demonstrate that the SFRs, in conjunction with the security objectives for the operational environment, enforce all OSPs for the TOE.
APE_REQ.1.9C	The security requirements rationale shall explain why the SARs were chosen.
APE_REQ.1.10C	The statement of security requirements shall be internally consistent.

Table 22 — Evaluator action elements for APE_REQ.1

APE_REQ.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

7.7.4 Derived security requirements (APE_REQ.2)

7.7.4.1 Component relationships

Dependencies: APE_OBJ.2 (Security objectives) (7.5.4)
 APE_ECD.1 (Extended components definition) (7.6.2)

7.7.4.2 Component elements

Table 23 specifies the developer action elements for APE_REQ.2, with Table 24 indicating their related content and presentation requirements, and Table 25 the corresponding evaluation actions.

Table 23 — Developer action elements for APE_REQ.2

APE_REQ.2.1D	The developer shall provide a statement of security requirements.
APE_REQ.2.2D	The developer shall provide a security requirements rationale.

Table 24 — Content and presentation elements for APE_REQ.2

APE_REQ.2.1C	The statement of security requirements shall describe the SFRs and the SARs.
APE_REQ.2.2C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
APE_REQ.2.3C	The statement of security requirements shall identify all operations on the security requirements.
APE_REQ.2.4C	All operations shall be performed correctly.
APE_REQ.2.5C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
APE_REQ.2.6C	The security requirements rationale shall trace each SFR back to the security objectives for the TOE enforced by that SFR.
APE_REQ.2.7C	The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.
APE_REQ.2.8C	The security requirements rationale shall explain why the SARs were chosen.
APE_REQ.2.9C	The statement of security requirements shall be internally consistent.

Table 25 — Evaluator action elements for APE_REQ.2

APE_REQ.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

8 Class ACE Protection Profile Configuration evaluation

8.1 General

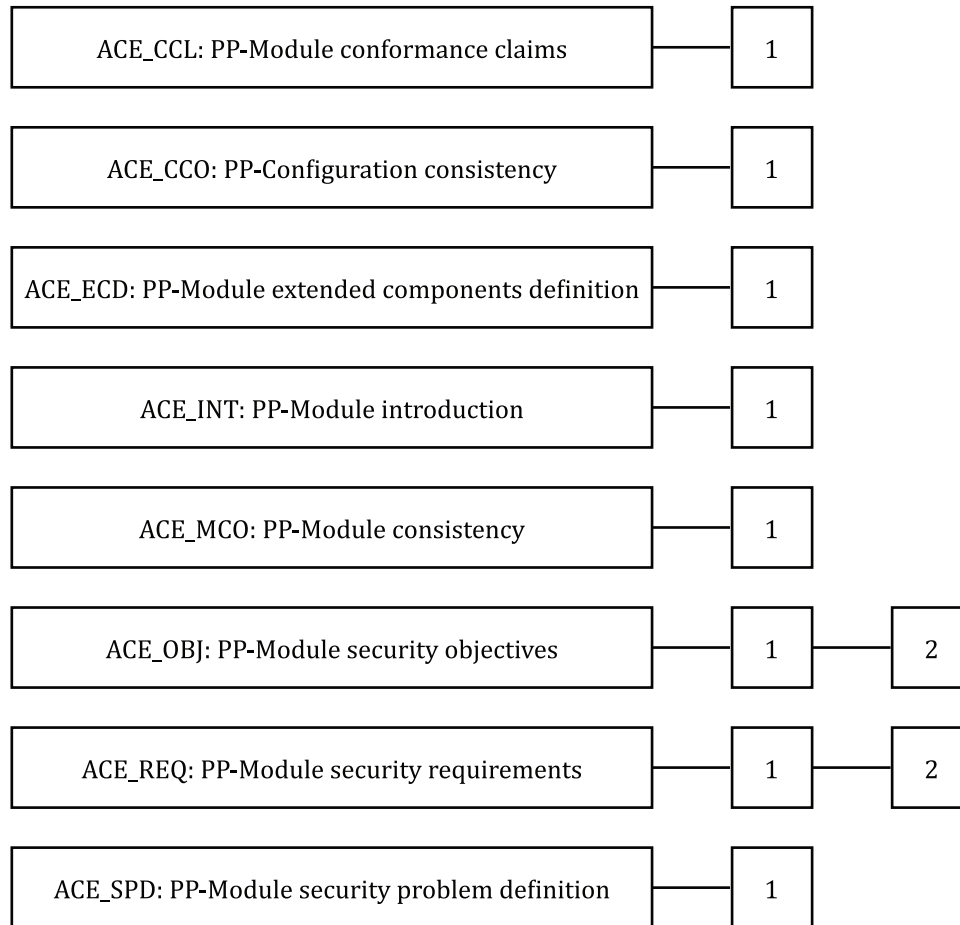
Evaluating a PP-Configuration is required to demonstrate that the PP-Configuration is sound and consistent. These properties are necessary for the PP-Configuration to be suitable for use as the basis for writing an ST.

ACE (Protection Profile Configuration evaluation) is defined for the evaluation of a PP-Configuration composed of at least one PP and one other component (either PPs or PP-Modules or both). The evaluation of PPs is addressed in class APE (Protection Profile (PP) evaluation) ([Clause 7](#)). ACE (Protection Profile Configuration evaluation) specifies the requirements for:

- evaluating the PP-Modules in the framework of their PP-Modules Base(s) components:
 - ACE_INT.1 (PP-Module introduction) ([8.2.2](#)),
 - ACE_CCL.1 (PP-Module conformance claims) ([8.3.2](#)),
 - ACE_SPD.1 (PP-Module security problem definition) ([8.4.2](#)),
 - ACE_OBJ.1 (PP-Module security objectives for the operational environment) ([8.5.3](#)) or ACE_OBJ.2 (PP-Module security objectives) ([8.5.4](#)),
 - ACE_REQ.1 (PP-Module direct rationale security requirements) ([8.7.3](#)) or ACE_REQ.2 (PP-Module derived security requirements) ([8.7.4](#)), and
 - ACE_MCO.1 (PP-Module consistency) ([8.8.2](#)).
- evaluating the consistency of the combination of all the PPs and PP-Modules that belong to the PP-Configuration (see ACE_CCO.1 (PP-Configuration consistency) ([8.9.2](#))).

[Figure 6](#) describes the family hierarchy of class ACE.

ISO/IEC 15408-3:2026(en)



Key
 1,2 components

Figure 6 — ACE: Protection Profile Configuration evaluation, class decomposition

Direct, indirect and optional dependencies for the components of class ACE are shown in [Table 26](#).

Table 26 — Dependency table for class ACE: Protection Profile Configuration evaluation

	ACE_CCL.1 (8.3.2)	ACE_ECD.1 (8.6.2)	ACE_INT.1 (8.2.2)	ACE_MCO.1 (8.8.2)	ACE_OBJ.1 (8.5.3)	ACE_OBJ.2 (8.5.4)	ACE_REQ.1 (8.7.3)	ACE_REQ.2 (8.7.4)	ACE_SPD.1 (8.4.2)
ACE_CCL.1 (8.3.2)		X	X		-	-	O ¹	O ¹	-
ACE_CCO.1 (8.9.2)	X	X	X	X	O ¹	O ¹	O ²	O ²	X
ACE_ECD.1 (8.6.2)									
ACE_INT.1 (8.2.2)									
ACE_MCO.1 (8.8.2)		-	X		O ¹	O ¹	O ²	O ²	X
ACE_OBJ.1 (8.5.3)									X
ACE_OBJ.2 (8.5.4)									X
ACE_REQ.1 (8.7.3)		X			X				X
ACE_REQ.2 (8.7.4)		X				X			-
ACE_SPD.1 (8.4.2)									

ISO/IEC 15408-3:2026(en)

8.2 PP-Module introduction (ACE_INT)

8.2.1 Objectives

The objective of this family is to describe the TOE in a narrative way.

The evaluation of the PP-Module introduction is required to demonstrate that the PP-Module is correctly identified, and that the PP-Module reference and TOE overview are consistent with each other.

8.2.2 PP-Module introduction (ACE_INT.1)

[Table 27](#) specifies the developer action elements for ACE_INT.1, with [Table 28](#) indicating their related content and presentation requirements, and [Table 29](#) the corresponding evaluation actions.

Table 27 — Developer action elements for ACE_INT.1

ACE_INT.1.1D	The developer shall provide a PP-Module introduction.
--------------	---

Table 28 — Content and presentation elements for ACE_INT.1

ACE_INT.1.1C	The PP-Module introduction shall contain a PP-Module reference, the identification of the PP-Module Base(s) and a TOE overview.
ACE_INT.1.2C	The PP-Module reference shall uniquely identify the PP-Module.
ACE_INT.1.3C	The identification of the PP-Module Base shall consist of a list of at least one PP and possibly other PPs and PP-Modules on which the PP-Module depends.
ACE_INT.1.4C	The identification of the PP-Module Base(s) shall describe the dependency structure of the PP-Module Base(s).
ACE_INT.1.5C	The PP-Module introduction shall contain as many TOE overviews as alternative PP-Module Bases.
ACE_INT.1.6C	The TOE overview shall summarize the usage and major security features of the TOE.
ACE_INT.1.7C	The TOE overview shall identify the TOE type.
ACE_INT.1.8C	The TOE overview shall identify any non-TOE hardware, software, firmware available to the TOE.
ACE_INT.1.9C	The TOE overview shall describe the differences of the TOE with regard to the TOEs defined in the PP-Module Base(s).

Table 29 — Evaluator action elements for ACE_INT.1

ACE_INT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

8.3 PP-Module conformance claims (ACE_CCL)

8.3.1 Objectives

The objective of this family is to determine the validity of the conformance claim and conformance statement. These may be based on different editions of the ISO/IEC 15408 series and derived PPs, PP-Configurations or PP-Modules. A PP-Module cannot claim conformance to any PP, PP-Configuration, or another PP-Module.

NOTE Previous editions can refer to withdrawn and replaced documents.

ISO/IEC 15408-3:2026(en)

8.3.2 PP-Module conformance claims (ACE_CCL.1)

8.3.2.1 Component relationships

Dependencies: ACE_INT.1 (PP-Module introduction) (8.2.2)
 ACE_ECD.1 (PP-Module extended components definition) (8.6.2)
 [ACE_REQ.1 (PP-Module direct rationale security requirements) (8.7.3), or
 ACE_REQ.2 (PP-Module derived security requirements) (8.7.4)]

8.3.2.2 Component elements

Table 30 specifies the developer action elements for ACE_CCL.1, with Table 31 indicating their related content and presentation requirements, and Table 32 the corresponding evaluation actions.

Table 30 — Developer action elements for ACE_CCL.1

ACE_CCL.1.1D	The developer shall provide a conformance claim.
ACE_CCL.1.2D	The developer shall provide a conformance statement.

Table 31 — Content and presentation elements for ACE_CCL.1

ACE_CCL.1.1C	The conformance claim shall identify the edition of the ISO/IEC 15408 series to which the PP-Module claims conformance.
ACE_CCL.1.2C	The conformance claim shall describe the conformance of the PP-Module to ISO/IEC 15408-2 as either "ISO/IEC 15408-2 conformant" or "ISO/IEC 15408-2 extended".
ACE_CCL.1.3C	The conformance statement shall describe the conformance type required of any ST to the PP-Module (as part of a PP-Configuration) as one of exact, strict, or demonstrable.
ACE_CCL.1.4C	The conformance claim shall describe the conformance of the PP-Module to ISO/IEC 15408-3 as either "ISO/IEC 15408-3 conformant" or "ISO/IEC 15408-3 extended".
ACE_CCL.1.5C	The conformance claim shall be consistent with the extended components definition.
ACE_CCL.1.6C	The conformance claim shall identify all functional packages to which the PP-Module claims conformance.
ACE_CCL.1.7C	The conformance claim shall describe any conformance of the PP-Module to a functional package as either package-conformant, package-augmented or package-tailored.
ACE_CCL.1.8C	The conformance claim shall identify all assurance packages to which the PP-Module claims conformance.
ACE_CCL.1.9C	The conformance claim shall describe any conformance of the PP-Module to an assurance package as either package-conformant or package-augmented.
ACE_CCL.1.10C	For exact conformance, the PP-Module's conformance statement shall contain an allowed-with statement that identifies the set of PPs and PP-Modules (exclusive of those PPs and PP-Modules that are included in the PP-Module Base) to which, in combination with the PP-Module under evaluation, exact conformance is allowed to be claimed.
ACE_CCL.1.11C	The conformance statement may identify the set of ISO/IEC 18045-derived evaluation methods and evaluation activities that shall be used with the PP-Module under evaluation. This list shall contain any evaluation methods and evaluation activities that are specified in the PP-Module but also any evaluation methods and evaluation activities specified in either the PP-Module Base(s) or in the packages (if any) or both, for which conformance is being claimed by the PP-Module under evaluation.

Table 32 — Evaluator action elements for ACE_CCL.1

ACE_CCL.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	--

8.4 PP-Module security problem definition (ACE_SPD)

8.4.1 Objectives

This part of the PP-Module specifies the security problem to be addressed by the TOE and the operational environment of the TOE.

Evaluation of the security problem definition is required to demonstrate that the security problem intended to be addressed by the TOE and its operational environment, is clearly defined.

8.4.2 PP-Module security problem definition (ACE_SPD.1)

[Table 33](#) specifies the developer action elements for ACE_SPD.1, with [Table 34](#) indicating their related content and presentation requirements, and [Table 35](#) the corresponding evaluation actions.

Table 33 — Developer action elements for ACE_SPD.1

ACE_SPD.1.1D	The developer shall provide a security problem definition.
--------------	--

Table 34 — Content and presentation elements for ACE_SPD.1

ACE_SPD.1.1C	The security problem definition shall describe the threats.
ACE_SPD.1.2C	All threats shall be described in terms of a threat agent, an asset, and an adverse action.
ACE_SPD.1.3C	The security problem definition shall describe the OSPs.
ACE_SPD.1.4C	The security problem definition shall describe the assumptions about the operational environment of the TOE.

Table 35 — Evaluator action elements for ACE_SPD.1

ACE_SPD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

8.5 PP-Module security objectives (ACE_OBJ)

8.5.1 Objectives

The security objectives are a concise statement of the intended response to the security problem defined through the ACE_SPD (PP-Module security problem definition) ([8.4](#)) family.

Evaluation of the security objectives is required to demonstrate that the security objectives adequately and completely address the security problem definition and that the division of this problem between the TOE and its operational environment is clearly defined.

8.5.2 Component levelling

The components in this family are levelled on whether they prescribe only security objectives for the operational environment (see ACE_OBJ.1 (PP-Module security objectives for the operational environment) ([8.5.3](#))), or also security objectives for the TOE (see ACE_OBJ.2 (PP-Module security objectives) ([8.5.4](#))).

8.5.3 PP-Module security objectives for the operational environment (ACE_OBJ.1)

8.5.3.1 Component relationships

Dependencies: ACE_SPD.1 (PP-Module security problem definition) ([8.4.2](#))

ISO/IEC 15408-3:2026(en)

8.5.3.2 Component elements

[Table 36](#) specifies the developer action elements for ACE_OBJ.1, with [Table 37](#) indicating their related content and presentation requirements, and [Table 38](#) the corresponding evaluation actions.

Table 36 — Developer action elements for ACE_OBJ.1

ACE_OBJ.1.1D	The developer shall provide a statement of security objectives for the operational environment of the PP-Module.
ACE_OBJ.1.2D	The developer shall provide a security objectives rationale for the operational environment of the PP-Module.

Table 37 — Content and presentation elements for ACE_OBJ.1

ACE_OBJ.1.1C	The statement of security objectives shall describe the security objectives for the operational environment.
ACE_OBJ.1.2C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
ACE_OBJ.1.3C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Table 38 — Evaluator action elements for ACE_OBJ.1

ACE_OBJ.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

8.5.4 PP-Module security objectives (ACE_OBJ.2)

8.5.4.1 Component relationships

Dependencies: ACE_SPD.1 (PP-Module security problem definition) ([8.4.2](#))

8.5.4.2 Component elements

[Table 39](#) specifies the developer action elements for ACE_OBJ.2, with [Table 40](#) indicating their related content and presentation requirements, and [Table 41](#) the corresponding evaluation actions.

Table 39 — Developer action elements for ACE_OBJ.2

ACE_OBJ.2.1D	The developer shall provide a statement of security objectives for the PP-Module.
ACE_OBJ.2.2D	The developer shall provide a security objectives rationale for the PP-Module.

ISO/IEC 15408-3:2026(en)

Table 40 — Content and presentation elements for ACE_OBJ.2

ACE_OBJ.2.1C	The statement of security objectives shall describe the security objectives for the TOE and the security objectives for the operational environment.
ACE_OBJ.2.2C	The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.
ACE_OBJ.2.3C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
ACE_OBJ.2.4C	The security objectives rationale shall demonstrate that the security objectives counter all threats.
ACE_OBJ.2.5C	The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.
ACE_OBJ.2.6C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Table 41 — Evaluator action elements for ACE_OBJ.2

ACE_OBJ.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	---

8.6 PP-Module extended components definition (ACE_ECD)

8.6.1 Objectives

Extended SFRs are requirements that are not based on components from ISO/IEC 15408-2 or this document, but which are based on extended components, i.e. components defined by the PP-Module author.

Evaluation of the definition of extended functional components is necessary to determine that they are clear and unambiguous, and that they are necessary, i.e. it is not possible to express them clearly by using existing components from ISO/IEC 15408-2 or this document.

8.6.2 PP-Module extended components definition (ACE_ECD.1)

[Table 42](#) specifies the developer action elements for ACE_ECD.1, with [Table 43](#) indicating their related content and presentation requirements, and [Table 44](#) the corresponding evaluation actions.

Table 42 — Developer action elements for ACE_ECD.1

ACE_ECD.1.1D	The developer shall provide a statement of security requirements for the PP-Module.
ACE_ECD.1.2D	The developer shall provide an extended components definition for the PP-Module.

Table 43 — Content and presentation elements for ACE_ECD.1

ACE_ECD.1.1C	The statement of security requirements shall identify all the extended security requirements.
ACE_ECD.1.2C	The extended components definition shall define an extended component for each extended security requirement.
ACE_ECD.1.3C	The extended components definition shall describe how each extended component is related to the existing ISO/IEC 15408 series components, families, and classes.
ACE_ECD.1.4C	The extended components definition shall use the existing ISO/IEC 15408 series components, families, classes, and methodology as a model for presentation.
ACE_ECD.1.5C	The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

ISO/IEC 15408-3:2026(en)

Table 44 — Evaluator action elements for ACE_ECD.1

ACE_ECD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ACE_ECD.1.2E	The evaluator <i>shall confirm</i> that it is impossible to clearly express any extended component by using existing components.

8.7 PP-Module security requirements (ACE_REQ)

8.7.1 Objectives

The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour of the TOE. The SARs form a clear, unambiguous and well-defined description of the expected activities that will be undertaken to gain assurance in the TOE.

Evaluation of the security requirements is required to ensure that they are clear, unambiguous and well-defined.

8.7.2 Component levelling

The components in this family are levelled on whether the SFRs are derived from SPD (see ACE_REQ.1 (PP-Module direct rationale security requirements) (8.7.3)), or whether the SFRs are derived from the security objectives for the TOE (see ACE_REQ.2 (PP-Module derived security requirements) (8.7.4)).

8.7.3 PP-Module direct rationale security requirements (ACE_REQ.1)

8.7.3.1 Component relationships

Dependencies: ACE_ECD.1 (PP-Module extended components definition) (8.6.2)
 ACE_OBJ.1 (PP-Module security objectives for the operational environment) (8.5.3)
 ACE_SPD.1 (PP-Module security problem definition) (8.4.2)

8.7.3.2 Component elements

Table 45 specifies the developer action elements for ACE_REQ.1, with Table 46 indicating their related content and presentation requirements, and Table 47 the corresponding evaluation actions.

Table 45 — Developer action elements for ACE_REQ.1

ACE_REQ.1.1D	The developer shall provide a statement of security requirements for the PP-Module.
ACE_REQ.1.2D	The developer shall provide a security requirements rationale for the PP-Module.

Table 46 — Content and presentation elements for ACE_REQ.1

ACE_REQ.1.1C	The statement of security requirements shall describe the SFRs and SARs.
ACE_REQ.1.2C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
ACE_REQ.1.3C	The statement of security requirements shall identify all operations on the security requirements.
ACE_REQ.1.4C	All operations shall be performed correctly.
ACE_REQ.1.5C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
ACE_REQ.1.6C	The security requirements rationale shall trace each SFR back to the threats countered by that SFR and the OSPs enforced by that SFR.

ISO/IEC 15408-3:2026(en)

Table 46 (continued)

ACE_REQ.1.7C	The security requirements rationale shall demonstrate that the SFRs (in conjunction with the security objectives for the operational environment) counter all the threats for the TOE.
ACE_REQ.1.8C	The security requirements rationale shall demonstrate that the SFRs (in conjunction with the security objectives for the operational environment) enforce all the OSPs for the TOE.
ACE_REQ.1.9C	The security requirements rationale shall explain why the SARs were chosen.
ACE_REQ.1.10C	The statement of security requirements shall be internally consistent.

Table 47 — Evaluator action elements for ACE_REQ.1

ACE_REQ.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

8.7.4 PP-Module derived security requirements (ACE_REQ.2)

8.7.4.1 Component relationships

Dependencies: ACE_ECD.1 (PP-Module extended components definition) (8.6.2)
 ACE_OBJ.2 (PP-Module security objectives) (8.5.4)

8.7.4.2 Component elements

Table 48 specifies the developer action elements for ACE_REQ.2, with Table 49 indicating their related content and presentation requirements, and Table 50 the corresponding evaluation actions.

Table 48 — Developer action elements for ACE_REQ.2

ACE_REQ.2.1D	The developer shall provide a statement of security requirements for the PP-Module.
ACE_REQ.2.2D	The developer shall provide a security requirements rationale for the PP-Module.

Table 49 — Content and presentation elements for ACE_REQ.2

ACE_REQ.2.1C	The statement of security requirements shall describe the SFRs and SARs (the SARs that apply to the PP-Module can be explicitly stated or inherited from the PP-Module Base(s)).
ACE_REQ.2.2C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
ACE_REQ.2.3C	The statement of security requirements shall identify all operations on the security requirements.
ACE_REQ.2.4C	All operations shall be performed correctly.
ACE_REQ.2.5C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
ACE_REQ.2.6C	The security requirements rationale shall trace each SFR back to the security objectives for the TOE enforced by that SFR.
ACE_REQ.2.7C	The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.
ACE_REQ.2.8C	The security requirements rationale shall explain why the SARs were chosen.
ACE_REQ.2.9C	The statement of security requirements shall be internally consistent.

Table 50 — Evaluator action elements for ACE_REQ.2

ACE_REQ.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

8.8 PP-Module consistency (ACE_MCO)

8.8.1 Objectives

The objective of this family is to determine the consistency of the PP-Module and to state the correspondence between the PP-Module and its PP-Module Base(s).

8.8.2 PP-Module consistency (ACE_MCO.1)

8.8.2.1 Component relationships

Dependencies: ACE_INT.1 (PP-Module introduction) (8.2.2)
ACE_SPD.1 (PP-Module security problem definition) (8.4.2)
[ACE_OBJ.1 (PP-Module security objectives for the operational environment) (8.5.3), or
ACE_OBJ.2 (PP-Module security objectives) (8.5.4)]
[ACE_REQ.1 (PP-Module direct rationale security requirements) (8.7.3), or
ACE_REQ.2 (PP-Module derived security requirements) (8.7.4)]

8.8.2.2 Component elements

Table 51 specifies the developer action elements for ACE_MCO.1, with Table 52 indicating their related content and presentation requirements, and Table 53 the corresponding evaluation actions.

Table 51 — Developer action elements for ACE_MCO.1

ACE_MCO.1.1D	The developer shall provide a consistency rationale of the PP-Module for each of the alternative PP-Module Bases identified in the PP-Module introduction.
ACE_MCO.1.2D	The developer shall provide an assurance rationale of the PP-Module for each of the alternative PP-Module Bases identified in the PP-Module introduction.

Table 52 — Content and presentation elements for ACE_MCO.1

ACE_MCO.1.1C	The consistency rationale shall demonstrate that the TOE type of the PP-Module and the TOE types of its PP-Module Base(s) are consistent.
ACE_MCO.1.2C	The consistency rationale shall identify the assets of the PP-Module's SPD that also belong to any of the PP-Module Base(s) and amongst them those for which the PP-Module and the PP-Module Base(s) define different security problems.
ACE_MCO.1.3C	The consistency rationale shall demonstrate that: <ul style="list-style-type: none"> — the statement of the security problem definition is consistent with the statement of the security problem definition of its PP-Module Base(s); — the statement of the security problem definition is consistent with the statement of the security problem definition of any functional package for which conformance is being claimed.
ACE_MCO.1.4C	The consistency rationale shall demonstrate that: <ul style="list-style-type: none"> — the security objectives are consistent with the security objectives of its PP-Module Base(s); — the security objectives are consistent with the security objectives of any functional package for which conformance is being claimed.
ACE_MCO.1.5C	The consistency rationale shall demonstrate that:

ISO/IEC 15408-3:2026(en)

Table 52 (continued)

	<ul style="list-style-type: none"> — the security functional requirements are consistent with the security functional requirements of its PP-Modules Base(s); — the security functional requirements are consistent with the security functional requirements of any functional package for which conformance is being claimed.
ACE_MCO.1.6C	The assurance rationale shall demonstrate the internal consistency of the set of security assurance requirements of the PP-Module with regard to its security problem definition.
ACE_MCO.1.7C	The assurance rationale shall demonstrate the consistency of the set of security assurance requirements of the PP-Module with regard to the security assurance requirements of the PP-Module Base(s).

Table 53 — Evaluator action elements for ACE_MCO.1

ACE_MCO.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence. If the PP-Module specifies alternative PP-Module Bases, the evaluator <i>shall perform</i> this action for each consistency rationale.
--------------	--

8.9 PP-Configuration consistency (ACE_CCO)

8.9.1 Objectives

The objective of this family is to determine the well-formedness and the consistency of the PP-Configuration.

8.9.2 PP-Configuration consistency (ACE_CCO.1)

8.9.2.1 Component relationships

Dependencies: ACE_INT.1 (PP-Module introduction) (8.2.2)
 ACE_CCL.1 (PP-Module conformance claims) (8.3.2)
 ACE_SPD.1 (PP-Module security problem definition) (8.4.2)
 [ACE_OBJ.1 (PP-Module security objectives for the operational environment) (8.5.3), or
 ACE_OBJ.2 (PP-Module security objectives) (8.5.4)]
 ACE_ECD.1 (PP-Module extended components definition) (8.6.2)
 [ACE_REQ.1 (PP-Module direct rationale security requirements) (8.7.3), or
 ACE_REQ.2 (PP-Module derived security requirements) (8.7.4)]
 ACE_MCO.1 (PP-Module consistency) (8.8.2)

8.9.2.2 Component elements

Table 54 specifies the developer action elements for ACE_CCO.1, with Table 55 indicating their related content and presentation requirements, and Table 56 the corresponding evaluation actions.

Table 54 — Developer action elements for ACE_CCO.1

ACE_CCO.1.1D	The developer shall provide the reference of the PP-Configuration.
ACE_CCO.1.2D	The developer shall provide a components statement.
ACE_CCO.1.3D	The developer shall provide a TOE overview.
ACE_CCO.1.4D	The developer shall provide a conformance claim.
ACE_CCO.1.5D	The developer shall provide a conformance statement within the conformance claim.

ISO/IEC 15408-3:2026(en)

Table 54 (continued)

ACE_CCO.1.6D	The developer shall provide a consistency rationale covering all elements of the PP-Configuration.
ACE_CCO.1.7D	The developer shall provide a SAR statement.
ACE_CCO.1.8D	The developer shall provide the set of evaluation methods or activities, or both, that are applicable to the PP-Configuration.

Table 55 — Content and presentation elements for ACE_CCO.1

ACE_CCO.1.1C	The PP-Configuration reference shall uniquely identify the PP-Configuration.
ACE_CCO.1.2C	The PP-Configuration components statement shall uniquely identify the PPs and PP-Modules that compose the PP-Configuration.
ACE_CCO.1.3C	For each PP-Module identified in the PP-Configuration components statement, the components statement shall include the PP-Module Base required by the identified PP-Module. If the PP-Module specifies alternative PP-Module Bases, only one of these PP-Module Bases shall be referred to in the PP-Configuration.
ACE_CCO.1.4C	For a multi-assurance PP-Configuration, the components statement shall describe the organization of the TSF in terms of the sub-TSFs defined in the PPs and PP-Modules defined in the PP-Configuration.
ACE_CCO.1.5C	The TOE overview shall identify the TOE type.
ACE_CCO.1.6C	The TOE overview shall describe the usage and major security features of the TOE.
ACE_CCO.1.7C	The TOE overview shall identify any non-TOE: <ul style="list-style-type: none"> — hardware, — software, — firmware, available to the TOE.
ACE_CCO.1.8C	The conformance claim shall identify the ISO/IEC 15048 edition(s) to which the PP-Configuration components claim conformance.
ACE_CCO.1.9C	The conformance claim shall describe the conformance of the PP-Configuration to ISO/IEC 15408-2 as either "ISO/IEC 15408-2 conformant" or "ISO/IEC 15408-2 extended".
ACE_CCO.1.10C	The conformance claim shall describe the conformance of the PP-Configuration to ISO/IEC 15408-3 as either "ISO/IEC 15408-3 conformant" or "ISO/IEC 15408-3 extended".
ACE_CCO.1.11C	The conformance claim shall be consistent with the conformance claims of the PP-Configuration components.
ACE_CCO.1.12C	The conformance claim of a PP-Configuration shall include an assurance package conformance claim consisting of statements describing any conformance of the PP-Configuration to an assurance package as either package-conformant or package-augmented.
ACE_CCO.1.13C	The conformance statement shall specify the required conformance to the PP-Configuration as one of exact, strict, demonstrable, or it shall provide the list of conformance types that are required by each of the PP-Configuration components.
ACE_CCO.1.14C	For the exact conformance case, the allowed-with statement of the conformance statement of each PP included in the components statement of the PP-Configuration shall identify all the PP-Configuration components as being allowed to be used in combination with the PP in a PP-Configuration.
ACE_CCO.1.15C	For the exact conformance case, the allowed-with statement of the conformance statement of each PP-Module included in the components statement of the PP-Configuration shall identify all the PP-Configuration components that are not in the PP-Module Base(s) for that particular PP-Module as being allowed to be used in combination with the PP-Module in a PP-Configuration.
ACE_CCO.1.16C	For PP-Configurations that are not of exact conformance type (i.e. for PP-Configurations of strict or demonstrable conformance type), the conformance statement of a PP-Configuration may include an evaluation methods and evaluation activities reference statement that identifies the set of ISO/IEC 18045-derived evaluation methods and evaluation activities that are applicable to the PP-Configuration under evaluation.

ISO/IEC 15408-3:2026(en)

Table 55 (continued)

ACE_CCO.1.17C	The consistency rationale shall demonstrate that the TOE type defined in the PP-Configuration is consistent with the TOE types defined in the PPs and PP-Modules that belong to the PP-Configuration components statement.
ACE_CCO.1.18C	The consistency rationale shall demonstrate that the union of all the SPDs, security objectives and security functional requirements defined in the PP-Configuration components is consistent.
ACE_CCO.1.19C	For a single-assurance PP-Configuration, the SAR statement shall define a single set of SARs that applies to the entire TOE. For strict and demonstrable conformance, the set of SARs shall include the SARs identified in each of the PP-Configuration components. For exact conformance, the set of SARs shall be identical to the set of SARs identified in each of the PP-Configuration components.
ACE_CCO.1.20C	For a multi-assurance PP-Configuration, the SAR statement shall define the global set of SARs that applies to the entire TOE and the SARs that apply to each sub-TSF. For strict and demonstrable conformance, the global assurance set of SARs shall include the set of common SARs among the PP-Configuration components, and each set of SARs that apply to a sub-TSF shall include those identified for the PP-Configuration components associated with that sub-TSF. For exact conformance, the global assurance set of SARs shall be the set of common SARs among the PP-Configuration components, and each set of SARs that apply to a sub-TSF shall be identical to those identified for the PP-Configuration components associated with that sub-TSF.
ACE_CCO.1.21C	The SAR statement of a PP-Configuration shall include an assurance rationale that demonstrates the consistency of the applicable set of SARs with those defined in the components of the PP-Configuration under evaluation and their associated evaluation methods and evaluation activities. For a multi-assurance PP-Configuration, the assurance rationale shall demonstrate: <ul style="list-style-type: none"> — that the global set of SARs is consistent with the threats as defined in the SPDs of the PP-Configuration components, and — that the global set of SARs and the sets of SARs for each sub-TSF are consistent with each other.

Table 56 — Evaluator action elements for ACE_CCO.1

ACE_CCO.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ACE_CCO.1.2E	The evaluator <i>shall check</i> that the PP-Configuration consisting of all the PPs and PP-Modules identified in the component statement is consistent.

9 Class ASE Security Target (ST) evaluation

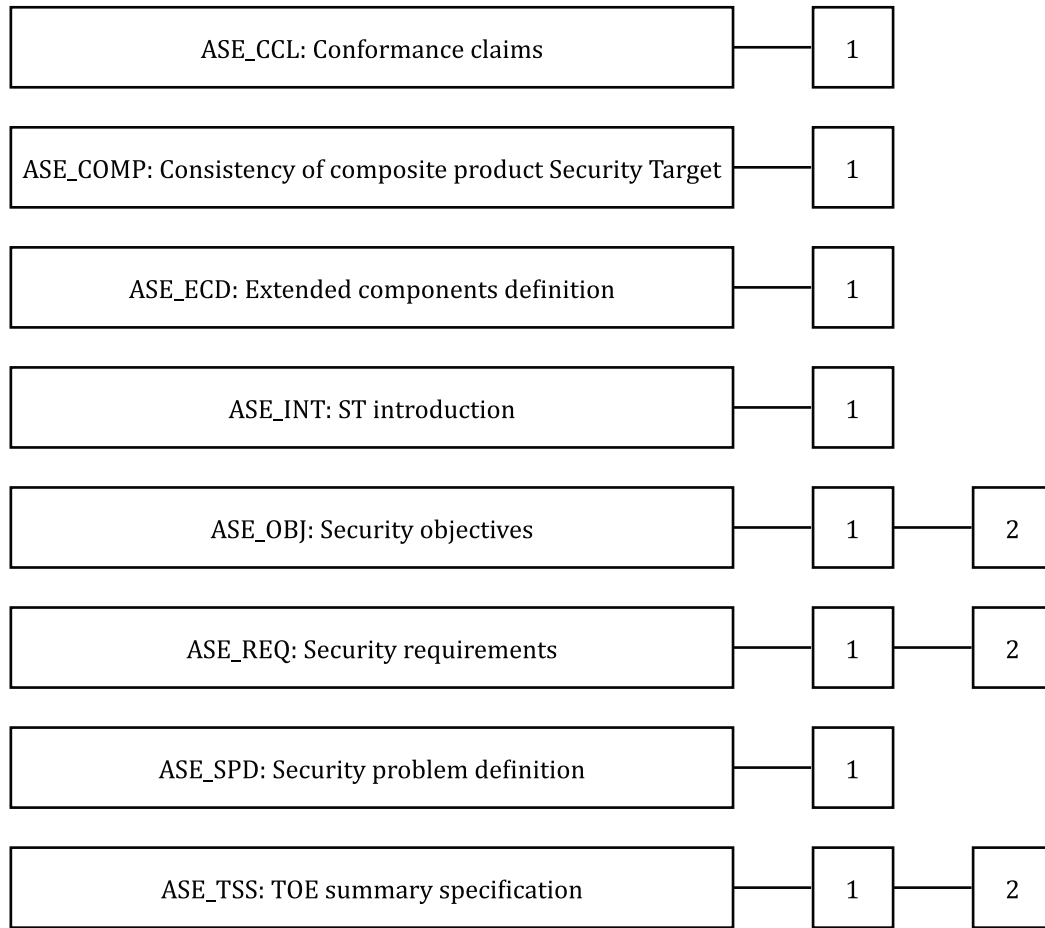
9.1 General

When evaluating an ST, it is required to demonstrate that the ST is sound and internally consistent. If the ST is based on a PP-Configuration, or one or more PPs or packages, it is also required to demonstrate that the ST is a correct instantiation of the PP-Configuration, PPs, and packages. These properties are necessary for the ST to be suitable for use as the basis for a TOE evaluation.

This clause should be used in conjunction with ISO/IEC 15408-1:2026, Annexes B and D, as these annexes clarify the concepts here and provide many examples.

[Figure 7](#) describes the family hierarchy of class ASE.

ISO/IEC 15408-3:2026(en)



Key
 1,2 components

Figure 7 — ASE: Security Target (ST) evaluation, class decomposition

Direct, indirect and optional dependencies for the components of class ASE are shown in [Table 57](#).

Table 57 — Dependency table for class ASE: Security Target (ST) evaluation

	ADV_ARC.1 (10.2.4)	ADV_FSP.1 (10.3.4)	ASE_ECD.1 (9.6.2)	ASE_INT.1 (9.2.2)	ASE_OBJ.1 (9.5.3)	ASE_OBJ.2 (9.5.4)	ASE_REQ.1 (9.7.3)	ASE_SPD.1 (9.4.2)
ASE_CCL.1 (9.3.2)			X	X	-		X	-
ASE_COMP.1 (9.9.4)								
ASE_ECD.1 (9.6.2)								
ASE_INT.1 (9.2.2)								
ASE_OBJ.1 (9.5.3)								X
ASE_OBJ.2 (9.5.4)								X
ASE_REQ.1 (9.7.3)			X		X			X
ASE_REQ.2 (9.7.4)			X			X		-
ASE_SPD.1 (9.4.2)								
ASE_TSS.1 (9.8.3)		X	-	X	-		X	-
ASE_TSS.2 (9.8.4)	X		-	X	-		X	-

9.2 ST introduction (ASE_INT)

9.2.1 Objectives

The objective of this family is to describe the TOE in a narrative way on three levels of abstraction: TOE reference, TOE overview and TOE description.

Evaluation of the ST introduction is required to demonstrate that the ST and the TOE are correctly identified, that the TOE is correctly described at three levels of abstraction and that these three descriptions are consistent with each other.

9.2.2 ST introduction (ASE_INT.1)

[Table 58](#) specifies the developer action elements for ASE_INT.1, with [Table 59](#) indicating their related content and presentation requirements, and [Table 60](#) the corresponding evaluation actions.

Table 58 — Developer action elements for ASE_INT.1

ASE_INT.1.1D	The developer shall provide an ST introduction.
--------------	---

Table 59 — Content and presentation elements for ASE_INT.1

ASE_INT.1.1C	The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.
ASE_INT.1.2C	The ST reference shall uniquely identify the ST.
ASE_INT.1.3C	The TOE reference shall uniquely identify the TOE.
ASE_INT.1.4C	The TOE overview shall summarize the usage and major security features of the TOE.
ASE_INT.1.5C	The TOE overview shall identify the TOE type.
ASE_INT.1.6C	The TOE overview shall identify any non-TOE hardware or software or firmware required by the TOE.
ASE_INT.1.7C	For a multi-assurance ST, the TOE overview shall describe the TSF organization in terms of the sub-TSFs defined in the PP-Configuration the ST claims conformance to.
ASE_INT.1.8C	The TOE description shall describe the physical scope of the TOE.
ASE_INT.1.9C	The TOE description shall describe the logical scope of the TOE.

Table 60 — Evaluator action elements for ASE_INT.1

ASE_INT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ASE_INT.1.2E	The evaluator <i>shall confirm</i> that the TOE reference, the TOE overview, and the TOE description are consistent with each other.

9.3 Conformance claims (ASE_CCL)

9.3.1 Objectives

The objective of this family is to determine the validity of the conformance claim and conformance statement. These may be based on different editions of the ISO/IEC 15408 series and derived PP-Configurations, PPs and packages. In addition, this family specifies how STs are to claim conformance to a PP-Configuration, PPs and packages.

NOTE Previous editions can refer to withdrawn and replaced documents.

ISO/IEC 15408-3:2026(en)

9.3.2 Conformance claims (ASE_CCL.1)

9.3.2.1 Component relationships

Dependencies: ASE_INT.1 (ST introduction) (9.2.2)
 ASE_ECD.1 (Extended components definition) (9.6.2)
 ASE_REQ.1 (Direct rationale security requirements) (9.7.3)

9.3.2.2 Component elements

Table 61 specifies the developer action elements for ASE_CCL.1, with Table 62 indicating their related content and presentation requirements, and Table 63 the corresponding evaluation actions.

Table 61 — Developer action elements for ASE_CCL.1

ASE_CCL.1.1D	The developer shall provide a conformance claim.
ASE_CCL.1.2D	The developer shall provide a conformance claim rationale.

Table 62 — Content and presentation elements for ASE_CCL.1

ASE_CCL.1.1C	The conformance claim shall identify the edition of the ISO/IEC 15408 series to which the ST and the TOE claim conformance.
ASE_CCL.1.2C	The conformance claim shall describe the conformance of the ST to ISO/IEC 15408-2 as either "ISO/IEC 15408-2 conformant" or "ISO/IEC 15408-2 extended".
ASE_CCL.1.3C	The conformance claim shall describe the conformance of the ST to ISO/IEC 15408-3 as either "ISO/IEC 15408-3 conformant" or "ISO/IEC 15408-3 extended".
ASE_CCL.1.4C	The conformance claim shall be consistent with the extended components definition.
ASE_CCL.1.5C	The conformance claim shall identify a PP-Configuration, or all PPs and security requirement packages to which the ST claims conformance.
ASE_CCL.1.6C	The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.
ASE_CCL.1.7C	The conformance claim shall describe any conformance of the ST to a PP as PP-Conformant.
ASE_CCL.1.8C	The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PP-Configuration or PPs for which conformance is being claimed.
ASE_CCL.1.9C	The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PP-Configuration, PPs and any functional packages for which conformance is being claimed.
ASE_CCL.1.10C	The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PP-Configuration, PPs, and any functional package for which conformance is being claimed.
ASE_CCL.1.11C	The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PP-Configuration, PPs, and any functional packages for which conformance is being claimed.
ASE_CCL.1.12C	The conformance claim for PP(s) or a PP-Configuration shall be exact, strict, or demonstrable or a list of conformance types.
ASE_CCL.1.13C	If the conformance claim identifies a set of evaluation methods and evaluation activities derived from ISO/IEC 18045 work units that shall be used to evaluate the TOE then this set shall include all those that are included in any package, PP, or PP-Module in a PP-Configuration to which the ST claims conformance, and no others.

Table 63 — Evaluator action elements for ASE_CCL.1

ASE_CCL.1.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	--

9.4 Security problem definition (ASE_SPD)

9.4.1 Objectives

This part of the ST specifies the security problem to be addressed by the TOE and the operational environment of the TOE.

Evaluation of the security problem definition is required to demonstrate that the security problem intended to be addressed by the TOE and its operational environment is clearly defined.

9.4.2 Security problem definition (ASE_SPD.1)

[Table 64](#) specifies the developer action elements for ASE_SPD.1, with [Table 65](#) indicating their related content and presentation requirements, and [Table 66](#) the corresponding evaluation actions.

Table 64 — Developer action elements for ASE_SPD.1

ASE_SPD.1.1D	The developer shall provide a security problem definition.
--------------	--

Table 65 — Content and presentation elements for ASE_SPD.1

ASE_SPD.1.1C	The security problem definition shall describe the threats.
ASE_SPD.1.2C	All threats shall be described in terms of a threat agent, an asset, and an adverse action.
ASE_SPD.1.3C	The security problem definition shall describe the OSPs.
ASE_SPD.1.4C	The security problem definition shall describe the assumptions about the operational environment of the TOE.

Table 66 — Evaluator action elements for ASE_SPD.1

ASE_SPD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

9.5 Security objectives (ASE_OBJ)

9.5.1 Objectives

The security objectives are a concise statement of the intended response to the security problem defined through the ASE_SPD (Security problem definition) ([9.4](#)) family.

Evaluation of the security objectives is required to demonstrate that the security objectives adequately and completely address the security problem definition, that the division of this problem between the TOE and its operational environment is clearly defined.

9.5.2 Component levelling

The components in this family are levelled on whether they prescribe only security objectives for the operational environment ASE_OBJ.1 (Security objectives for the operational environment) ([9.5.3](#)), or also security objectives for the TOE ASE_OBJ.2 (Security objectives) ([9.5.4](#)).

9.5.3 Security objectives for the operational environment (ASE_OBJ.1)

9.5.3.1 Component relationships

Dependencies: ASE_SPD.1 (Security problem definition) ([9.4.2](#))

9.5.3.2 Component elements

[Table 67](#) specifies the developer action elements for ASE_OBJ.1, with [Table 68](#) indicating their related content and presentation requirements, and [Table 69](#) the corresponding evaluation actions.

Table 67 — Developer action elements for ASE_OBJ.1

ASE_OBJ.1.1D	The developer shall provide a statement of security objectives for the operational environment.
ASE_OBJ.1.2D	The developer shall provide a security objectives rationale for the operational environment.

Table 68 — Content and presentation elements for ASE_OBJ.1

ASE_OBJ.1.1C	The statement of security objectives shall describe the security objectives for the operational environment.
ASE_OBJ.1.2C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
ASE_OBJ.1.3C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Table 69 — Evaluator action elements for ASE_OBJ.1

ASE_OBJ.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

9.5.4 Security objectives (ASE_OBJ.2)

9.5.4.1 Component relationships

Dependencies: ASE_SPD.1 (Security problem definition) ([9.4.2](#))

9.5.4.2 Component elements

[Table 70](#) specifies the developer action elements for ASE_OBJ.2, with [Table 71](#) indicating their related content and presentation requirements, and [Table 72](#) the corresponding evaluation actions.

Table 70 — Developer action elements for ASE_OBJ.2

ASE_OBJ.2.1D	The developer shall provide a statement of security objectives.
ASE_OBJ.2.2D	The developer shall provide a security objectives rationale.

ISO/IEC 15408-3:2026(en)

Table 71 — Content and presentation elements for ASE_OBJ.2

ASE_OBJ.2.1C	The statement of security objectives shall describe the security objectives for the TOE and the security objectives for the operational environment.
ASE_OBJ.2.2C	The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.
ASE_OBJ.2.3C	The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.
ASE_OBJ.2.4C	The security objectives rationale shall demonstrate that the security objectives counter all threats.
ASE_OBJ.2.5C	The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.
ASE_OBJ.2.6C	The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.

Table 72 — Evaluator action elements for ASE_OBJ.2

ASE_OBJ.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	---

9.6 Extended components definition (ASE_ECD)

9.6.1 Objectives

Extended security requirements are requirements that are not based on components from ISO/IEC 15408-2 or this document, but which are based on extended components, i.e. components defined by the ST author.

Evaluation of the definition of extended components is necessary to determine that they are clear and unambiguous, and that they are necessary, i.e. it is not possible to express them clearly by using existing components from ISO/IEC 15408-2 or this document.

9.6.2 Extended components definition (ASE_ECD.1)

[Table 73](#) specifies the developer action elements for ASE_ECD.1, with [Table 74](#) indicating their related content and presentation requirements, and [Table 75](#) the corresponding evaluation actions.

Table 73 — Developer action elements for ASE_ECD.1

ASE_ECD.1.1D	The developer shall provide a statement of security requirements.
ASE_ECD.1.2D	The developer shall provide an extended components definition.

Table 74 — Content and presentation elements for ASE_ECD.1

ASE_ECD.1.1C	The statement of security requirements shall identify all extended security requirements.
ASE_ECD.1.2C	The extended components definition shall define an extended component for each extended security requirement.
ASE_ECD.1.3C	The extended components definition shall describe how each extended component is related to the existing ISO/IEC 15408 series components, families, and classes.
ASE_ECD.1.4C	The extended components definition shall use the existing ISO/IEC 15408 series components, families, classes, and methodology as a model for presentation.
ASE_ECD.1.5C	The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements can be demonstrated.

ISO/IEC 15408-3:2026(en)**Table 75 — Evaluator action elements for ASE_ECD.1**

ASE_ECD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ASE_ECD.1.2E	The evaluator <i>shall confirm</i> that it is impossible to clearly express any extended component by using existing components.

9.7 Security requirements (ASE_REQ)**9.7.1 Objectives**

The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities that will be undertaken to gain assurance in the TOE.

Evaluation of the security requirements is required to ensure that they are clear, unambiguous and well-defined.

9.7.2 Component levelling

The components in this family are levelled on whether they are stated as is (see ASE_REQ.1 (Direct rationale security requirements) (9.7.3)), or whether the SFRs are derived from security objectives for the TOE (see ASE_REQ.2 (Derived security requirements) (9.7.4)).

9.7.3 Direct rationale security requirements (ASE_REQ.1)**9.7.3.1 Component relationships**

Dependencies: ASE_ECD.1 (Extended components definition) (9.6.2)
 ASE_OBJ.1 (Security objectives for the operational environment) (9.5.3)
 ASE_SPD.1 (Security problem definition) (9.4.2)

9.7.3.2 Component elements

Table 76 specifies the developer action elements for ASE_REQ.1, with Table 77 indicating their related content and presentation requirements, and Table 78 the corresponding evaluation actions.

Table 76 — Developer action elements for ASE_REQ.1

ASE_REQ.1.1D	The developer shall provide a statement of security requirements.
ASE_REQ.1.2D	The developer shall provide a security requirements rationale.

Table 77 — Content and presentation elements for ASE_REQ.1

ASE_REQ.1.1C	The statement of security requirements shall describe the SFRs and the SARs.
ASE_REQ.1.2C	For a single-assurance ST, the statement of security requirements shall define the global set of SARs that apply to the entire TOE. The sets of SARs shall be consistent with the PPs or PP-Configuration to which the ST claims conformance.
ASE_REQ.1.3C	For a multi-assurance ST, the statement of security requirements shall define the global set of SARs that apply to the entire TOE and the sets of SARs that apply to each sub-TSF. The sets of SARs shall be consistent with the multi-assurance PP-Configuration to which the ST claims conformance.
ASE_REQ.1.4C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
ASE_REQ.1.5C	The statement of security requirements shall identify all operations on the security requirements.

ISO/IEC 15408-3:2026(en)

Table 77 (continued)

ASE_REQ.1.6C	All operations shall be performed correctly.
ASE_REQ.1.7C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
ASE_REQ.1.8C	The security requirements rationale shall trace each SFR back to the threats countered by that SFR and the OSPs enforced by that SFR. The security requirements rationale shall demonstrate that the SFRs (in conjunction with the security objectives for the operational environment) counter all threats for the TOE.
ASE_REQ.1.9C	The security requirements rationale shall demonstrate that the SFRs (in conjunction with the security objectives for the operational environment) enforce all OSPs.
ASE_REQ.1.10C	The security requirements rationale shall explain why the SARs were chosen.
ASE_REQ.1.11C	The statement of security requirements shall be internally consistent.
ASE_REQ.1.12C	If the ST defines sets of SARs that expand the sets of SARs of the PPs or PP-Configuration to which it claims conformance, the security requirements rationale shall include an assurance rationale that justifies the consistency of the extension and provides a rationale for the disposition of any evaluation methods and evaluation activities identified in the conformance statement that are affected by the extension of the sets of SARs.

Table 78 — Evaluator action elements for ASE_REQ.1

ASE_REQ.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

9.7.4 Derived security requirements (ASE_REQ.2)

9.7.4.1 Component relationships

Dependencies: ASE_OBJ.2 (Security objectives) (9.5.4)
 ASE_ECD.1 (Extended components definition) (9.6.2)

9.7.4.2 Component elements

Table 79 specifies the developer action elements for ASE_REQ.2, with Table 80 indicating their related content and presentation requirements, and Table 81 the corresponding evaluation actions.

Table 79 — Developer action elements for ASE_REQ.2

ASE_REQ.2.1D	The developer shall provide a statement of security requirements.
ASE_REQ.2.2D	The developer shall provide a security requirements rationale.

Table 80 — Content and presentation elements for ASE_REQ.2

ASE_REQ.2.1C	The statement of security requirements shall describe the SFRs and the SARs.
ASE_REQ.2.2C	For a single-assurance ST, the statement of security requirements shall define the global set of SARs that apply to the entire TOE. The sets of SARs shall be consistent with the PPs or PP-Configuration to which the ST claims conformance.
ASE_REQ.2.3C	For a multi-assurance ST, the statement of security requirements shall define the global set of SARs that apply to the entire TOE and the sets of SARs that apply to each sub-TSF. The sets of SARs shall be consistent with the multi-assurance PP-Configuration to which the ST claims conformance.
ASE_REQ.2.4C	All subjects, objects, operations, security attributes, external entities and other terms that are used in the SFRs and the SARs shall be defined.
ASE_REQ.2.5C	The statement of security requirements shall identify all operations on the security requirements.
ASE_REQ.2.6C	All operations shall be performed correctly.

ISO/IEC 15408-3:2026(en)

Table 80 (continued)

ASE_REQ.2.7C	Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.
ASE_REQ.2.8C	The security requirements rationale shall trace each SFR back to the security objectives for the TOE . The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.
ASE_REQ.2.9C	The security requirements rationale shall explain why the SARs were chosen.
ASE_REQ.2.10C	The statement of security requirements shall be internally consistent.
ASE_REQ.2.11C	If the ST specifies sets of SARs that expand the sets of SARs of the PPs or PP-Configuration it claims conformance to , the security requirements rationale shall include an assurance rationale that justifies the consistency of the extension and provides a rationale for the disposition of any evaluation methods and evaluation activities identified in the conformance statement that are affected by the extension of the sets of SARs.

Table 81 — Evaluator action elements for ASE_REQ.2

ASE_REQ.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	---

9.8 TOE summary specification (ASE_TSS)

9.8.1 Objectives

The TOE summary specification enables evaluators and potential consumers to gain a general understanding of how the TOE is implemented.

Evaluation of the TOE summary specification is necessary to determine whether it is adequately described how the TOE:

- meets its SFRs;
- protects itself against interference, logical tampering and bypass;

and whether the TOE summary specification is consistent with other narrative descriptions of the TOE.

9.8.2 Component levelling

The components in this family are levelled on whether the TOE summary specification shall describe how the TOE meets the SFRs, or whether the TOE summary specification also shall describe how the TOE protects itself against logical tampering and bypass. This additional description can be used in special circumstances where there can be a specific concern regarding the TOE security architecture.

9.8.3 TOE summary specification (ASE_TSS.1)

9.8.3.1 Component relationships

Dependencies: ASE_INT.1 (ST introduction) ([9.2.2](#))
 ASE_REQ.1 (Direct rationale security requirements) ([9.7.3](#))
 ADV_FSP.1 (Basic functional specification) ([10.3.4](#))

9.8.3.2 Component elements

[Table 82](#) specifies the developer action elements for ASE_TSS.1, with [Table 83](#) indicating their related content and presentation requirements, and [Table 84](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 82 — Developer action elements for ASE_TSS.1

ASE_TSS.1.1D	The developer shall provide a TOE summary specification.
--------------	--

Table 83 — Content and presentation elements for ASE_TSS.1

ASE_TSS.1.1C	The TOE summary specification shall describe how the TOE meets each SFR.
--------------	--

Table 84 — Evaluator action elements for ASE_TSS.1

ASE_TSS.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ASE_TSS.1.2E	The evaluator <i>shall confirm</i> that the TOE summary specification is consistent with the TOE overview and the TOE description.

9.8.4 TOE summary specification with architectural design summary (ASE_TSS.2)

9.8.4.1 Component relationships

Dependencies: ASE_INT.1 (ST introduction) ([9.2.2](#))
 ASE_REQ.1 (Direct rationale security requirements) ([9.7.3](#))
 ADV_ARC.1 (Security architecture description) ([10.2.4](#))

9.8.4.2 Component elements

[Table 85](#) specifies the developer action elements for ASE_TSS.2, with [Table 86](#) indicating their related content and presentation requirements, and [Table 87](#) the corresponding evaluation actions.

Table 85 — Developer action elements for ASE_TSS.2

ASE_TSS.2.1D	The developer shall provide a TOE summary specification.
--------------	--

Table 86 — Content and presentation elements for ASE_TSS.2

ASE_TSS.2.1C	The TOE summary specification shall describe how the TOE meets each SFR.
ASE_TSS.2.2C	The TOE summary specification shall describe how the TOE protects itself against interference and logical tampering.
ASE_TSS.2.3C	The TOE summary specification shall describe how the TOE protects itself against bypass.

Table 87 — Evaluator action elements for ASE_TSS.2

ASE_TSS.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ASE_TSS.2.2E	The evaluator <i>shall confirm</i> that the TOE summary specification is consistent with the TOE overview and the TOE description.

9.9 Consistency of composite product Security Target (ASE_COMP)

9.9.1 Objectives

The aim of this family is to determine whether the ST of the composite product does not contradict the ST of the related base component.

NOTE 1 The ST of the composite product is denoted by composite product Security Target or composite-ST.

NOTE 2 The ST of the related base component is denoted by base component Security Target or base-ST.

ISO/IEC 15408-3:2026(en)

NOTE 3 Generally, a Security Target expresses a security policy for the TOE defined.

9.9.2 Component levelling

This family contains only one component.

9.9.3 Application notes

An ST for the composite product shall be written and evaluated.

The composite product evaluator shall examine that the ST of the composite product does not contradict the ST of the related base component. In particular, it means that the composite product evaluator shall examine the composite product ST and the base component ST for any conflicting assumptions, compatibility of security objectives, security requirements and security functionality needed by the dependent component.

The composite product evaluation sponsor shall ensure that the ST of the base component is available for the dependent component developer, for the composite product evaluator and for the composite product evaluation authority. It is possible, that the information available in the public version of the base component ST is not sufficient.

These application notes aid the developer to create, as well as the evaluator to analyse, a composite product ST and describe a general methodology for it.

In order to create a composite product ST, the developer should perform the following steps.

Step 1: The developer formulates a preliminary ST for the composite product (the composite-ST) using the standard code of practice. The composite-ST can be formulated independently of the ST of the composite product's related base component (the base-ST), at least as long as there are no formal PP conformance claims.

Step 2: The developer determines the overlap between the base-ST and the composite-ST through analysing and comparing their respective TOE Security Functionality (TSF).

NOTE This is necessary because the TSF enforce the Security Target (together with the organizational measures enforcing the security objectives for the operational environment of the TOE). The comparison is expected to be performed on the abstraction level of SFRs. If the developer defined security functionality groups (TSF-groups) in the TSS part of the Security Target, the evaluator has also to consider them in order to get a better understanding for the context of the security services offered by the TOE.

Step 3: The developer determines under which conditions they can trust in and rely on the base component-TSF being used by the composite-ST without a new examination.

Having undertaken these steps, the developer completes the preliminary ST for the composite product.

It is not required that the composite product and its related base component are being evaluated according to the same edition of ISO/IEC 15408 because the dependent component of the composite product can rely on some security services of the base component, if:

- the assurance level of the base component covers the intended assurance level of the composite product and
- the base component evaluation is valid (i.e. accepted by the base component evaluation authority) and up-to-date.

Equivalence of single-assurance components (and, hence, of assurance levels) belonging to different ISO/IEC 15408 editions shall be established and acknowledged by the composite product evaluation authority.

If conformance to a PP is claimed, e.g. a composite product ST claims conformance to a PP (that possibly claims conformance to a further PP), the consistency check can be reduced to the elements of the ST having not already been covered by these PPs. However, in general the fact of compliance to a PP is not sufficient to avoid inconsistencies. Assume the following situation, where → stands for "complies with":

ISO/IEC 15408-3:2026(en)

composite-ST → PP 1 → PP 2 ← base-ST

PP 1 may require any kind of conformance (e.g. "strict", "exact" or "demonstrable" according to the ISO/IEC 15408-1:2026), but this does not affect the "additional elements" that the base-ST can introduce beyond PP 2. In conclusion, these additions are not necessarily consistent with the composite-ST's additions chosen beyond PP 1. There is no scenario that ensures their consistency "by construction".

Note that objectives for the base component's environment can become objectives for the composite TOE.

9.9.4 Consistency of Security Target (ST) (ASE_COMP.1)

[Table 88](#) specifies the developer action elements for ASE_COMP.1, with [Table 89](#) indicating their related content and presentation requirements, and [Table 90](#) the corresponding evaluation actions.

Table 88 — Developer action elements for ASE_COMP.1

ASE_COMP.1.1D	The developer shall provide a statement of compatibility between the composite product Security Target and the base component Security Target. This statement may be provided within the composite product Security Target.
---------------	--

Table 89 — Content and presentation elements for ASE_COMP.1

ASE_COMP.1.1C	The statement of compatibility shall describe the separation of the base component-TSF into relevant base component-TSF being used by the composite product Security Target and others.
ASE_COMP.1.2C	The statement of compatibility between the composite product Security Target and the base component Security Target shall show (e.g. in form of a mapping) that the Security Targets of the composite product and of the related base component match. That is, this statement shall show that there is no conflict between security environments, security objectives, and security requirements of the composite product Security Target and the base component Security Target. It can be provided by indicating the concerned elements directly in the composite product Security Target followed by explanatory text, if necessary.

Table 90 — Evaluator action elements for ASE_COMP.1

ASE_COMP.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
---------------	--

10 Class ADV Development

10.1 General

The requirements of the development class provide information about the TOE. The knowledge obtained by this information is used as the basis for conducting vulnerability analysis and testing upon the TOE, as described in class AVA (Vulnerability assessment) ([Clause 14](#)) and class ATE (Tests) ([Clause 13](#)).

The Development class encompasses seven families of requirements for structuring and representing the TSF at various levels and varying forms of abstraction. These families include:

- requirements for the description (at the various levels of abstraction) of the design and implementation of the SFRs (ADV_FSP (Functional specification) ([10.3](#)), ADV_TDS (TOE design) ([10.7](#)), ADV_IMP (Implementation representation) ([10.4](#)) and ADV_COMP (Composite design compliance) ([10.8](#)));
- requirements for the description of the architecture-oriented features of domain separation, TSF self-protection and non-bypassability of the security functionality (ADV_ARC (Security architecture) ([10.2](#)));
- requirements for a security policy model and for correspondence mappings between security policy model and the functional specification (ADV_SPM (Formal TSF model) ([10.6](#)));

ISO/IEC 15408-3:2026(en)

- requirements on the internal structure of the TSF, which covers aspects such as modularity, layering, and minimization of complexity (ADV_INT (TSF internals) (10.5)).

When documenting the security functionality of a TOE, there are two properties that shall be demonstrated. The first property is that the security functionality works correctly, i.e. it performs as specified. The second property, and one that is arguably harder to demonstrate, is that the TOE cannot be used in a way such that the security functionality can be corrupted or bypassed. These two properties require somewhat different approaches in analysis, and so the families in ADV (Development) are structured to support these different approaches.

The families ADV_FSP (Functional specification) (10.3), ADV_TDS (TOE design) (10.7), ADV_IMP (Implementation representation) (10.4), and ADV_SPM (Formal TSF model) (10.6) deal with the first property: the specification of the security functionality. The families ADV_ARC (Security architecture) (10.2) and ADV_INT (TSF internals) (10.5) deal with the second property: the specification of the design of the TOE demonstrating the security functionality cannot be corrupted or bypassed. It should be noted that both properties are required to be realized: the more confidence one has that the properties are satisfied, the more trustworthy the TOE is.

The TSF of a composite product are represented at various levels of abstraction in the families of ADV (Development). The family ADV_COMP (Composite design compliance) (10.8) determines whether the requirements on the dependent component, imposed by the related base component, are fulfilled in a composite product.

Due to the distribution of the TSF of a composite product to various levels in the families of ADV (Development), this family is not represented in Figure 8. The components in the families are designed so that more assurance can be gained as the components hierarchically increase.

The paradigm for the families targeted at the first property is one of design decomposition. At the highest level, there is a functional specification of the TSF in terms of:

- its interfaces (describing what the TSF does in terms of requests to the TSF for services and resulting responses);
- decomposing the TSF into smaller units (dependent on the assurance desired and the complexity of the TOE) and describing how the TSF accomplishes its functions (to a level of detail commensurate with the assurance level);
- showing the implementation of the TSF.

A formal model of the security behaviour can also be given. All levels of decomposition are used in determining the completeness and accuracy of all other levels, ensuring that the levels are mutually supportive. The requirements for the various TSF representations are separated into different families, to allow the PP or ST author to specify which TSF representations are required. The level chosen will dictate the assurance desired/gained.

Figure 8 indicates the relationships among the various TSF representations of ADV (Development), as well as their relationships with other classes. As the figure indicates, class APE (Protection Profile (PP) evaluation) (Clause 7) and class ASE (Security Target (ST) evaluation) (Clause 9) define the requirements for the correspondence between the SFRs and the security objectives for the TOE. class ASE (Security Target (ST) evaluation) (Clause 9) also specifies requirements for the correspondence between both the security objectives and SFRs, and for the TOE summary specification which explains how the TOE meets its SFRs. The activities of ALC_CMC.5.2E (12.2.8) include the verification that the TSF that is tested under class ATE (Tests) (Clause 13) and class AVA (Vulnerability assessment) (Clause 14) is in fact the one described by all ADV (Development) decomposition levels.

ISO/IEC 15408-3:2026(en)

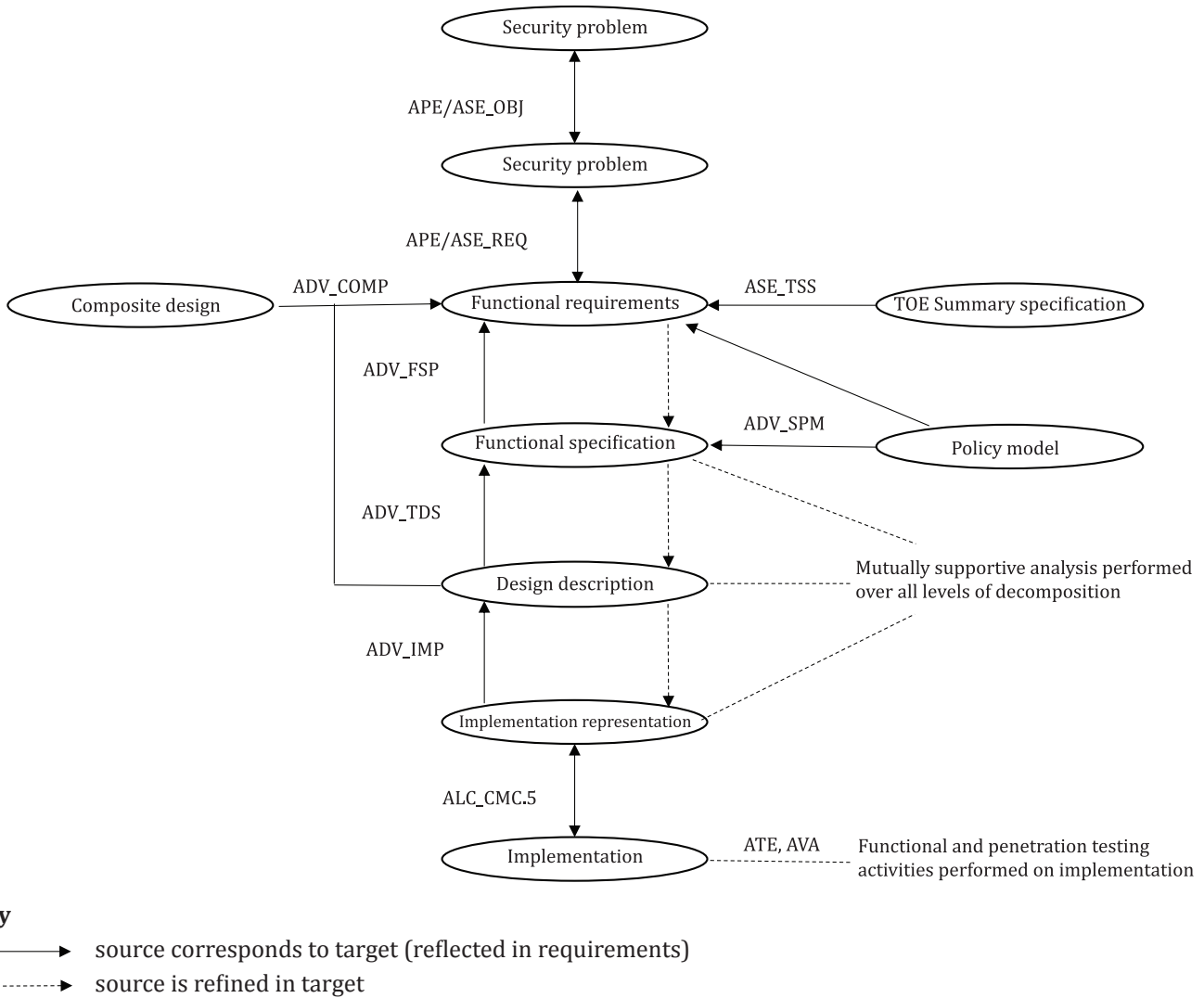


Figure 8 — Relationships of ADV constructs to one another and to other families

The requirements for all other correspondence shown in [Figure 8](#) are defined in ADV (Development) for the TOE. The ADV_SPM (Formal TSF model) (10.6) family specifies the requirements for formally modelling selected SFRs and providing correspondence between the functional specification and the formal model. Each assurance family specific to a TSF representation (i.e. ADV_FSP (Functional specification) (10.3), ADV_TDS (TOE design) (10.7) and ADV_IMP (Implementation representation) (10.4)) specifies requirements relating that TSF representation to the SFRs. All decompositions shall accurately reflect all other decompositions (i.e. be mutually supportive); the developer supplies the tracings in the last .C elements of the components. Assurance relating to this factor is obtained during the analysis for each of the levels of decomposition by referring to other levels of decomposition (in a recursive fashion) while the analysis of a particular level of decomposition is being performed; the evaluator verifies the correspondence as part of the second E element. The understanding gained from these levels of decomposition form the basis of the functional and penetration testing efforts.

The ADV_INT (TSF internals) (10.5) family is not represented in this [Figure 8](#), as it is related to the internal structure of the TSF, and is only indirectly related to the process of refinement of the TSF representations. Similarly, the ADV_ARC (Security architecture) (10.2) family is not represented in the [Figure 8](#) because it relates to the architectural soundness, rather than representation, of the TSF. Both ADV_INT (TSF internals) (10.5) and ADV_ARC (Security architecture) (10.2) relate to the analysis of the property that the TOE cannot be made to circumvent or corrupt its security functionality.

The TOE security functionality (TSF) consists of all parts of the TOE that shall be relied upon for enforcement of the SFRs. The TSF includes both functionality that directly enforces the SFRs, as well as functionality that,

ISO/IEC 15408-3:2026(en)

while not directly enforcing the SFRs, contributes to their enforcement in a more indirect manner, including functionality with the capability to cause the SFRs to be violated. This includes portions of the TOE that are invoked on start-up that are responsible for putting the TSF into its initial secure state.

Several important concepts were used in the development of the components of the ADV (Development) families. These concepts, while introduced briefly here, are explained more fully in the application notes for the families.

One over-riding notion is that, as more information becomes available, greater assurance can be obtained that the security functionality:

- is correctly implemented;
- cannot be corrupted; and
- cannot be bypassed.

This is done through the verification that the documentation is correct and consistent with other documentation, and by providing information that can be used to ensure that the testing activities (both functional and penetration testing) are comprehensive. This is reflected in the levelling of the components of the families. In general, components are levelled based on the amount of information that is required to be provided (and subsequently analysed).

While not true for all TOEs, it is generally the case that the TSF is sufficiently complex that there are portions of the TSF that deserve more intense examination than other portions of the TSF. Determining those portions is unfortunately somewhat subjective, thus terminology and components have been defined such that as the level of assurance increases, the responsibility for determining what portions of the TSF require detailed examination shifts from the developer to the evaluator. To aid in expressing this concept, the following terminology is introduced. It should be noted that in the families of the class, this terminology is used when expressing SFR-related portions of the TOE (i.e. elements and work units embodied in the ADV_FSP (Functional specification) (10.3), ADV_TDS (TOE design) (10.7), and ADV_IMP (Implementation representation) (10.4) families). While the general concept (that some portions of the TOE are more interesting than others) applies to other families, the criteria are expressed differently in order to obtain the assurance required.

All portions of the TSF are security relevant, meaning that they shall preserve the security of the TOE as expressed by the SFRs and requirements for domain separation and non-bypassability. One aspect of security relevance is the degree to which a portion of the TSF enforces a security requirement. Since different portions of the TOE play different roles (or no apparent role at all) in enforcing security requirements, this creates a continuum of SFR relevance: at one end of this continuum are portions of the TOE that are termed SFR-enforcing.

Such portions play a direct role in implementing any SFR on the TOE. Such SFRs refer to any functionality provided by one of the SFRs contained in the ST. It should be noted that the definition of "plays a role in" for SFR-enforcing functionality is impossible to express quantitatively.

For example, in the implementation of a Discretionary Access Control (DAC) mechanism, a very narrow view of SFR-enforcing can be the several lines of code that actually perform the check of a subject's attributes against the object's attributes.

A broader view would include the software entity (e.g. C function) that contained the several lines of code. A broader view still would include callers of the C function, since they would be responsible for enforcing the decision returned by the attribute check.

A still broader view would include any code in the call tree (or programming equivalent for the implementation language used) for that C function (e.g. a function that sorted access control list entries in a first-match algorithm implementation). At some point, the component is not so much enforcing the security policy but rather plays a supporting role; such components are termed SFR supporting. One of the characteristics of SFR-supporting functionality is that it is trusted to preserve the correctness of the SFR implementation by operating without error.

ISO/IEC 15408-3:2026(en)

Such functionality can be depended on by SFR-enforcing functionality, but the dependence is generally at a functional level, e.g. memory management or buffer management.

Further down on the security relevance continuum is functionality termed SFR non-interfering. Such functionality has no role in implementing the SFRs and is likely part of the TSF because of its environment, e.g. any code running in a privileged hardware mode on an operating system. It shall be considered part of the TSF because, if compromised (or replaced by malicious code), it can compromise the correct operation of an SFR by virtue of its operating in the privileged hardware mode.

An example of SFR non-interfering functionality can be a set of mathematical floating-point operations implemented in kernel mode for speed considerations.

The family ADV_ARC (Security architecture) (10.2) provides for requirements and analysis of the TOE based on properties of domain separation, self-protection, and non-bypassability. These properties relate to the SFRs in that, if these properties are not present, it will likely lead to the failure of mechanisms implementing SFRs. Functionality and design relating to these properties is not considered a part of the continuum described above, but instead is treated separately due to its fundamentally different nature and analysis requirements.

The difference in analysis of the implementation of SFRs (SFR-enforcing and SFR-supporting functionality) and the implementation of somewhat fundamental security properties of the TOE, which include the initialisation, self-protection, and non-bypassability concerns, is that the SFR-related functionality is more or less directly visible and relatively easy to test, while the above-mentioned properties require varying degrees of analysis on a much broader set of functionalities. Further, the depth of analysis for such properties will vary depending on the design of the TOE. The ADV (Development) families are constructed to address this by a separate family (ADV_ARC (Security architecture) (10.2)) devoted to analysis of the initialisation, self-protection, and non-bypassability requirements, while the other families are concerned with analysis of the functionality supporting SFRs.

Even in cases where different descriptions are necessary for the multiple levels of abstraction, it is not absolutely necessary for each and every TSF representation to be in a separate document. Indeed, it is possible that a single document meets the documentation requirements for more than one TSF representation, since it is the information about each of these TSF representations that is required, rather than the resulting document structure. In cases where multiple TSF representations are combined within a single document, the developer should indicate which portions of the documents meet which requirements.

Three types of specification style are mandated by this class: informal, semi-formal and formal. The functional specification and TOE design documentation are always written in either informal or semi-formal style. A semi-formal style reduces the ambiguity in these documents over an informal presentation. A formal specification can also be required in addition to the semi-formal presentation; the value is that a description of the TSF in more than one way will add increased assurance that the TSF has been completely and accurately specified.

An informal specification is written as prose in natural language. Natural language is used here as meaning communication in any commonly spoken tongue (e.g. Spanish, German, French, English, Dutch). An informal specification is not subject to any notational or special restrictions other than those required as ordinary conventions for that language (e.g. grammar and syntax). While no notational restrictions apply, the informal specification is also required to provide defined meanings for terms that are used in a context other than that accepted by normal usage.

The difference between semi-formal and informal documents is only a matter of formatting or presentation: a semi-formal notation includes, e.g. an explicit glossary of terms, a standardized presentation format. A semi-formal specification is written to a standard presentation template. The presentation should use terms consistently if written in a natural language. The presentation can also use more structured languages/diagrams (e.g. data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). Whether based on diagrams or natural language, a set of conventions shall be used in the presentation. The glossary explicitly identifies the words that are being used in a precise and constant manner; similarly, the standardized format implies that extreme care has been taken in methodically preparing the document in a manner that maximizes clarity. It should be noted that fundamentally different portions of the TSF can have different semi-formal notation conventions

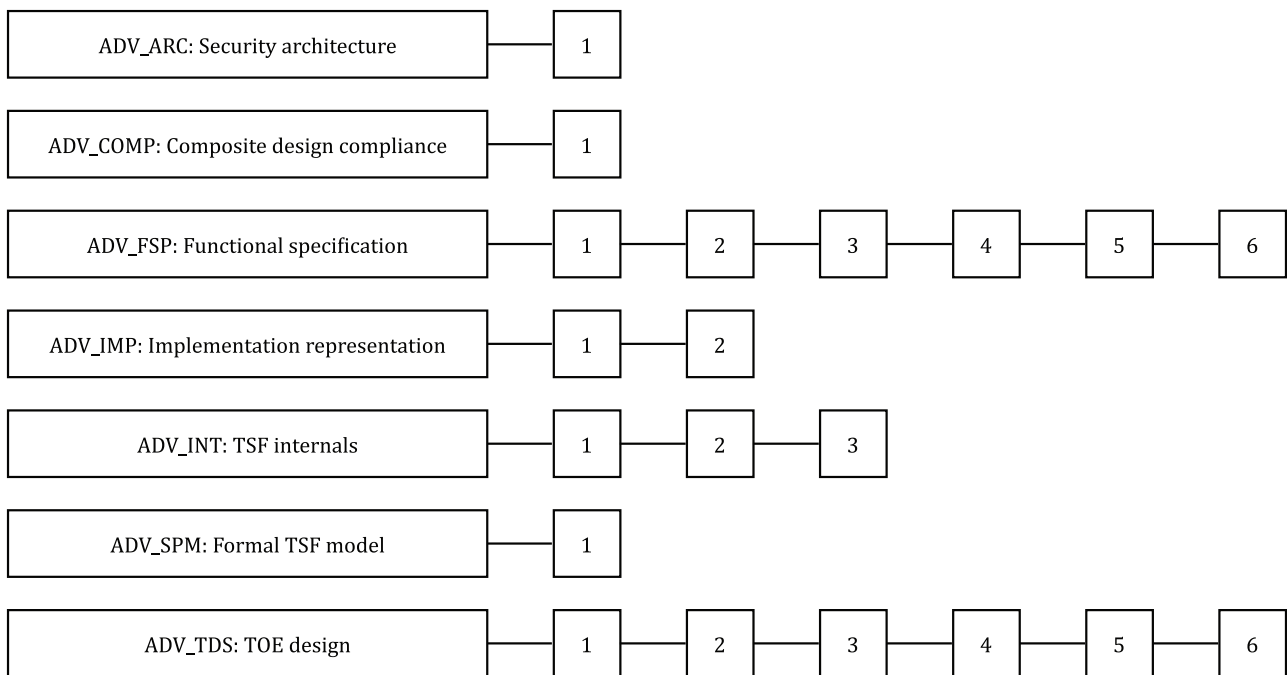
ISO/IEC 15408-3:2026(en)

and presentation styles (as long as the number of different "semi-formal notations" is small); this still conforms to the concept of a semi-formal presentation.

A formal specification is written in a notation based upon well-established mathematical concepts and is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts are used to define the syntax and semantics of the notation and the proof rules that support logical reasoning. The syntactic and semantic rules supporting a formal notation should define how to recognize constructs unambiguously and determine their meaning. There shall be evidence that it is impossible to derive contradictions, and all rules supporting the notation shall be defined or referenced.

In case of a multi-assurance evaluation, the requirements for the description (at the various levels of abstraction) of the design and implementation of the SFRs (ADV_FSP (Functional specification) (10.3), ADV_TDS (TOE design) (10.7), ADV_IMP (Implementation representation) (10.4) and ADV_COMP (Composite design compliance) (10.8)) will be presented for the sub-TSF of the TOE. ADV_ARC (Security architecture) (10.2) provides for requirements and analysis of the TOE based on properties of domain separation, self-protection, and non-bypassability which can also hold for boundaries between the sub-TSF.

Figure 9 describes the family hierarchy of class ADV.



Key

1,2,3,4,5,6 components

Figure 9 — ADV: Development, class decomposition

Direct, indirect and optional dependencies for the components of class ADV are shown in Table 91.

ISO/IEC 15408-3:2026(en)

Table 91 — Dependency table for class ADV: Development

	ADV_FSP.1 (10.3.4)	ADV_FSP.2 (10.3.5)	ADV_FSP.3 (10.3.6)	ADV_FSP.4 (10.3.7)	ADV_FSP.5 (10.3.8)	ADV_FSP.6 (10.3.9)	ADV_IMP.1 (10.4.4)	ADV_TDS.1 (10.7.4)	ADV_TDS.3 (10.7.6)	ALC_CMC.5 (12.2.8)	ALC_TAT.1 (12.9.4)	ASE_OBJ.2 (9.5.4)	ASE_REQ.2 (9.7.4)
ADV_ARC.1 (10.2.4)	X	-						X					
ADV_COMP.1 (10.8.4)													
ADV_FSP.1 (10.3.4)													
ADV_FSP.2 (10.3.5)		-						X					
ADV_FSP.3 (10.3.6)		-						X					
ADV_FSP.4 (10.3.7)		-						X					
ADV_FSP.5 (10.3.8)		-		-			X	X	-		-		
ADV_FSP.6 (10.3.9)		-		-			X	X	-		-		
ADV_IMP.1 (10.4.4)		-		-				-	X		X		
ADV_IMP.2 (10.4.5)		-		-				-	X	X	X		
ADV_INT.1 (10.5.4)		-		-			X	-	X		X		
ADV_INT.2 (10.5.5)		-		-			X	-	X		X		
ADV_INT.3 (10.5.6)		-		-			X	-	X		X		
ADV_SPM.1 (10.6.4)		-		X				-				X	X
ADV_TDS.1 (10.7.4)		X						-					
ADV_TDS.2 (10.7.5)		-	X					-					
ADV_TDS.3 (10.7.6)		-		X				-					
ADV_TDS.4 (10.7.7)		-		-	X		-	-	-		-		
ADV_TDS.5 (10.7.8)		-		-	X		-	-	-		-		
ADV_TDS.6 (10.7.9)		-		-		X	-	-	-		-		

10.2 Security architecture (ADV_ARC)

10.2.1 Objectives

The objective of this family is for the developer to provide a description of the security architecture of the TSF. This allows analysis of the information that, when coupled with the other evidence presented for the TSF, shall confirm that the TSF achieves the desired properties. The security architecture descriptions support the implicit claim that security analysis of the TOE can be achieved by examining the TSF; without a sound architecture, the entire TOE functionality shall be examined.

10.2.2 Component levelling

This family contains only one component.

10.2.3 Application notes

The properties of self-protection, domain separation, and non-bypassability are distinct from security functionality expressed by ISO/IEC 15408-2:2026 SFRs because self-protection and non-bypassability largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are achieved through the design of the TOE and TSF and enforced by the correct implementation of that design.

The approach used in this family is for the developer to design and provide a TSF that exhibits the properties of self-protection, domain separation, and non-bypassability, and to provide evidence (in the form of documentation) that explains these properties of the TSF. This explanation is provided at the same level of detail as the description of the SFR-enforcing elements of the TOE in the TOE design document. The evaluator

ISO/IEC 15408-3:2026(en)

has the responsibility for looking at the evidence and, coupled with other evidence delivered for the TOE and TSF, determining that the properties are achieved.

Specification of security functionality implementing the SFRs in ADV_FSP (Functional specification) (10.3) and ADV_TDS (TOE design) (10.7) will not necessarily describe mechanisms employed in implementing self-protection and non-bypassability (e.g. memory management mechanisms). Therefore, the material needed to provide the assurance that these requirements are being achieved is better suited to a presentation separate from the design decomposition of the TSF as embodied in ADV_FSP (Functional specification) (10.3) and ADV_TDS (TOE design) (10.7). This is not to imply that the security architecture description called for by this component cannot reference or make use of the design decomposition material; but it is likely that much of the detail present in the decomposition documentation will not be relevant to the argument being provided for the security architecture description document.

The description of architectural soundness can be thought of as a developer's vulnerability analysis, in that it provides the justification for why the TSF is sound and enforces all of its SFRs. Where the soundness is achieved through specific security mechanisms, these will be tested as part of the ATE_DPT (Depth) (13.3) requirements; where the soundness is achieved solely through the architecture, the behaviour will be tested as part of the class AVA (Vulnerability assessment) (Clause 14) requirements.

This family consists of requirements for a security architecture description that describes the self-protection, domain separation, non-bypassability principles, including a description of how these principles are supported by the parts of the TOE that are used for TSF initialisation.

In case of a multi-assurance evaluation the properties of self-protection, domain separation, and non-bypassability can also be described for boundaries between the sub-TSF.

Additional information on the security architecture properties of self-protection, domain separation, and non-bypassability can be found in A.1.

10.2.4 Security architecture description (ADV_ARC.1)

10.2.4.1 Component relationships

Dependencies: ADV_FSP.1 (Basic functional specification) (10.3.4)
 ADV_TDS.1 (Basic design) (10.7.4)

10.2.4.2 Component elements

Table 92 specifies the developer action elements for ADV_ARC.1, with Table 93 indicating their related content and presentation requirements, and Table 94 the corresponding evaluation actions.

Table 92 — Developer action elements for ADV_ARC.1

ADV_ARC.1.1D	The developer shall design and implement the TOE so that the security features of the TSF cannot be bypassed.
ADV_ARC.1.2D	The developer shall design and implement the TSF so that it is able to protect itself from tampering by untrusted active entities.
ADV_ARC.1.3D	The developer shall provide a security architecture description of the TSF.

Table 93 — Content and presentation elements for ADV_ARC.1

ADV_ARC.1.1C	The security architecture description shall be at a level of detail commensurate with the description of the SFR-enforcing abstractions described in the TOE design document.
ADV_ARC.1.2C	The security architecture description shall describe the security domains maintained by the TSF consistently with the SFRs.
ADV_ARC.1.3C	The security architecture description shall describe how the TSF initialisation process is secure.
ADV_ARC.1.4C	The security architecture description shall demonstrate that the TSF protects itself from tampering.
ADV_ARC.1.5C	The security architecture description shall demonstrate that the TSF prevents bypass of the SFR-enforcing functionality.

Table 94 — Evaluator action elements for ADV_ARC.1

ADV_ARC.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

10.3 Functional specification (ADV_FSP)

10.3.1 Objectives

This family levies requirements upon the functional specification, which describes the TSF interfaces (TSFIs). The TSFI consists of all means by which external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data from the TSF or invoke services from the TSF. It does not describe how the TSF processes those service requests, nor does it describe the communication when the TSF invokes services from its operational environment; this information is addressed by the ADV_TDS (TOE design) (10.7) and ACO_REL (Reliance of dependent component) (15.4) families, respectively.

This family provides assurance directly by allowing the evaluator to understand how the TSF meets the claimed SFRs. It also provides assurance indirectly, as input to other assurance families and classes:

- ADV_ARC (Security architecture) (10.2), where the description of the TSFIs can be used to gain better understanding of how the TSF is protected against either corruption (i.e. subversion of self-protection or domain separation), or bypass, or both;
- Class ATE (Tests) (Clause 13), where the description of the TSFIs is an important input for both developer and evaluator testing;
- Class AVA (Vulnerability assessment) (Clause 14), where the description of the TSFIs is used to search for vulnerabilities.

10.3.2 Component levelling

The components in this family are levelled on the degree of detail required of the description of the TSFIs, and the degree of formalism required of the description of the TSFIs.

10.3.3 Application notes

10.3.3.1 General

Once the TSFIs are determined (see A.2.3 for guidance, requirements and examples of determining TSFI), they are described. At lower-level components, developers focus their documentation (and evaluators focus

ISO/IEC 15408-3:2026(en)

their analysis) on the more security-relevant aspects of the TOE. Three categories of TSFIs are defined based upon the relevance that the services available through them have for the SFRs being claimed:

- if a service available through an interface can be traced to one of the SFRs levied on the TSF, then that interface is termed SFR-enforcing. It is possible that an interface can have various services and results, some of which can be SFR-enforcing and some of which not;
- interfaces to (or services available through an interface relating to) services that SFR-enforcing functionality depend upon, but need only to function correctly in order for the security policies of the TOE to be preserved, are termed SFR-supporting;
- interfaces to services on which SFR-enforcing functionality has no dependence are termed SFR non-interfering.

It should be noted that in order for an interface to be SFR-supporting or SFR non-interfering, it shall have no SFR-enforcing services or results. In contrast, an SFR-enforcing interface can have SFR-supporting services. For example, the ability to set the system clock can be an SFR-enforcing service of an interface, but if that same interface is used to display the system date that service can be only SFR-supporting. An example of a purely SFR-supporting interface is a system call interface that is used both by users and by a portion of the TSF that is running on behalf of users.

As more information about the TSFIs becomes available, the greater the assurance that can be gained that the interfaces are correctly categorized/analysed. The requirements are structured such that, at the lowest level, the information required for SFR non-interfering interfaces is the minimum necessary in order for the evaluator to make this determination in an effective manner. At higher levels, more information becomes available so that the evaluator has greater confidence in the designation.

The purpose in defining these labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) and for levying different requirements upon each (at the lower assurance components) is to provide a first approximation of where to focus the analysis and the evidence upon which that analysis is performed. If the developer's documentation of the TSF interfaces describes all the interfaces to the degree specified in the requirements for the SFR-enforcing interfaces (i.e. if the documentation exceeds the requirements), there is no need for the developer to create new evidence to match the requirements. Similarly, because the labels are merely a means of differentiating the interface types within the requirements, there is no need for the developer to update the evidence solely to label the interfaces as SFR-enforcing, SFR-supporting, and SFR-non-interfering. The primary purpose of this labelling is to allow developers with less mature development methodologies (and associated artefacts, such as detailed interface and design documentation) to provide only the necessary evidence without undue cost.

The last C element of each component within this family provides a direct correspondence between the SFRs and the functional specification, i.e. an indication of which interfaces are used to invoke each of the claimed SFRs. In the cases where the ST contains such functional requirements as ISO/IEC 15408-2, whose functionality does not manifest itself at the TSFIs, either the functional specification, or the tracing, or both, is expected to identify these SFRs; including them in the functional specification helps to ensure that they are not lost at lower levels of decomposition, where they will be relevant.

10.3.3.2 Detail about the interfaces

The requirements define collections of details about TSFIs to be provided. For the purposes of the requirements, interfaces are specified (in varying degrees of detail) in terms of their purpose, method of use, parameters, parameter descriptions, and error messages.

The purpose of an interface is a high-level description of the general goal of the interface (e.g. process GUI commands, receive network packets, provide printer output, etc.).

The interface's method of use describes how the interface is supposed to be used. This description should be built around the various interactions available at that interface. For instance, if the interface were a Unix-like command shell, the Unix-like commands `ls`, `mv` and `cp` would be interactions for that interface. For each interaction the method of use describes what the interaction does, both for behaviour seen at the interface (e.g. the programmer calling the API) as well as behaviour at other interfaces (e.g. generating an audit record).

ISO/IEC 15408-3:2026(en)

Parameters are explicit inputs to and outputs from an interface that control the behaviour of that interface. For example, parameters are the arguments supplied to an API; the various fields in a packet for a given network protocol; the individual key values in an operating system registry; the signals across a set of pins on a chip; the flags that can be set for the I/O, etc. The parameters are "identified" with a simple list of what they are.

A parameter description tells what the parameter is in some meaningful way. For instance, an acceptable parameter description for interface *foo(i)* would be "parameter *i* is an integer that indicates the number of users currently logged in to the system". A description such as "parameter *i* is an integer" is not acceptable.

The description of an interface's actions describes what the interface does. This is more detailed than the purpose in that, while the "purpose" reveals why it is used, the "actions" reveals everything that it does. These actions can be related to the SFRs or not. In cases where the interface's action is not related to SFRs, its description is said to be summarized, meaning that the description merely makes clear that it is indeed not SFR-related.

The error message description identifies the condition that generated it, what the message is, and the meaning of any error codes. An error message is generated by the TSF to signify that a problem or irregularity of some degree has been encountered. The requirements in this family refer to different kinds of error messages:

- a direct error message is a security-relevant response through a specific TSFI invocation;
- an indirect error cannot be tied to a specific TSFI invocation because it results from system-wide conditions (e.g. resource exhaustion, connectivity interruptions, etc.). Error messages that are not security-relevant are also considered "indirect";
- "remaining" errors are any other errors, such as those that can be referenced within the code. For example, the use of condition-checking code that checks for conditions that would not logically occur (e.g. a final "else" after a list of "case" statements), would provide for generating a catch-all error message; in an operational TOE, these error messages are never expected to be seen.

An example functional specification is provided in [A.2.4](#).

10.3.3.3 Components of this family

Increasing assurance through increased completeness and accuracy in the interface specification is reflected in the documentation required from the developer as detailed in the various hierarchical components of this family.

At ADV_FSP.1 (Basic functional specification) ([10.3.4](#)), the only documentation required is a characterization of all TSFIs and a high-level description of SFR-enforcing and SFR-supporting TSFIs. To provide some assurance that the "important" aspects of the TSF have been correctly characterized at the TSFIs, the developer is required to provide the purpose and method of use, parameters for the SFR-enforcing and SFR-supporting TSFIs.

At ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#)), the developer is required to provide the purpose, method of use, parameters, and parameter descriptions for all TSFIs. Additionally, for the SFR-enforcing TSFIs the developer shall describe the SFR-enforcing actions and direct error messages.

At ADV_FSP.3 (Functional specification with complete summary) ([10.3.6](#)), the developer shall, in addition to the information required at ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#)), provide enough information about the SFR-supporting and SFR-non-interfering actions to show that they are not SFR-enforcing. Further, the developer shall now document all of the direct error messages resulting from the invocation of SFR-enforcing TSFIs.

At ADV_FSP.4 (Complete functional specification) ([10.3.7](#)), all TSFIs, whether SFR-enforcing, SFR-supporting or SFR-non-interfering, shall be described to the same degree, including all of the direct error messages.

ISO/IEC 15408-3:2026(en)

At ADV_FSP.5 (Complete semi-formal functional specification with additional error information) (10.3.8) Complete semi-formal functional specification with additional error information, the TSFIs descriptions also include error messages that do not result from an invocation of a TSFI.

At ADV_FSP.6 (Complete semi-formal functional specification with additional formal specification) (10.3.9) Complete semi-formal functional specification with additional formal specification, in addition to the information required by ADV_FSP.5 (Complete semi-formal functional specification with additional error information) (10.3.8), all remaining error messages are included. The developer shall also provide a formal description of the TSFI. This provides an alternative view of the TSFI that can expose inconsistencies or incomplete specification.

10.3.4 Basic functional specification (ADV_FSP.1)

Table 95 specifies the developer action elements for ADV_FSP.1, with Table 96 indicating their related content and presentation requirements, and Table 97 the corresponding evaluation actions.

Table 95 — Developer action elements for ADV_FSP.1

ADV_FSP.1.1D	The developer shall provide a functional specification.
ADV_FSP.1.2D	The developer shall provide a tracing from the functional specification to the SFRs.

Table 96 — Content and presentation elements for ADV_FSP.1

ADV_FSP.1.1C	The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.2C	The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.3C	The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.
ADV_FSP.1.4C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Table 97 — Evaluator action elements for ADV_FSP.1

ADV_FSP.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.1.2E	The evaluator <i>shall determine</i> that the functional specification is an accurate and complete instantiation of the SFRs.

10.3.5 Security-enforcing functional specification (ADV_FSP.2)

10.3.5.1 Component relationships

Dependencies: ADV_TDS.1 (Basic design) (10.7.4)

10.3.5.2 Component elements

Table 98 specifies the developer action elements for ADV_FSP.2, with Table 99 indicating their related content and presentation requirements, and Table 100 the corresponding evaluation actions.

Table 98 — Developer action elements for ADV_FSP.2

ADV_FSP.2.1D	The developer shall provide a functional specification.
ADV_FSP.2.2D	The developer shall provide a tracing from the functional specification to the SFRs.

ISO/IEC 15408-3:2026(en)

Table 99 — Content and presentation elements for ADV_FSP.2

ADV_FSP.2.1C	The functional specification shall completely represent the TSF.
ADV_FSP.2.2C	The functional specification shall describe the purpose and method of use for all TSFIs .
ADV_FSP.2.3C	The functional specification shall identify and describe all parameters associated with each TSFI.
ADV_FSP.2.4C	For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions associated with the TSFI.
ADV_FSP.2.5C	For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting from processing associated with the SFR-enforcing actions.
ADV_FSP.2.6C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Table 100 — Evaluator action elements for ADV_FSP.2

ADV_FSP.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.2.2E	The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

10.3.6 Functional specification with complete summary (ADV_FSP.3)

10.3.6.1 Component relationships

Dependencies: ADV_TDS.1 (Basic design) ([10.7.4](#))

10.3.6.2 Component elements

[Table 101](#) specifies the developer action elements for ADV_FSP.3, with [Table 102](#) indicating their related content and presentation requirements, and [Table 103](#) the corresponding evaluation actions.

Table 101 — Developer action elements for ADV_FSP.3

ADV_FSP.3.1D	The developer shall provide a functional specification.
ADV_FSP.3.2D	The developer shall provide a tracing from the functional specification to the SFRs.

Table 102 — Content and presentation elements for ADV_FSP.3

ADV_FSP.3.1C	The functional specification shall completely represent the TSF.
ADV_FSP.3.2C	The functional specification shall describe the purpose and method of use for all TSFIs.
ADV_FSP.3.3C	The functional specification shall identify and describe all parameters associated with each TSFI.
ADV_FSP.3.4C	For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions associated with the TSFI.
ADV_FSP.3.5C	For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting from SFR-enforcing actions and exceptions associated with invocation of the TSFI .
ADV_FSP.3.6C	The functional specification shall summarize the SFR-supporting and SFR-non-interfering actions associated with each TSFI.
ADV_FSP.3.7C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

ISO/IEC 15408-3:2026(en)

Table 103 — Evaluator action elements for ADV_FSP.3

ADV_FSP.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.3.2E	The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

10.3.7 Complete functional specification (ADV_FSP.4)

10.3.7.1 Component relationships

Dependencies: ADV_TDS.1 (Basic design) ([10.7.4](#))

10.3.7.2 Component elements

[Table 104](#) specifies the developer action elements for ADV_FSP.4, with [Table 105](#) indicating their related content and presentation requirements, and [Table 106](#) the corresponding evaluation actions.

Table 104 — Developer action elements for ADV_FSP.4

ADV_FSP.4.1D	The developer shall provide a functional specification.
ADV_FSP.4.2D	The developer shall provide a tracing from the functional specification to the SFRs.

Table 105 — Content and presentation elements for ADV_FSP.4

ADV_FSP.4.1C	The functional specification shall completely represent the TSF.
ADV_FSP.4.2C	The functional specification shall describe the purpose and method of use for all TSFIs.
ADV_FSP.4.3C	The functional specification shall identify and describe all parameters associated with each TSFI.
ADV_FSP.4.4C	The functional specification shall describe all actions associated with each TSFI.
ADV_FSP.4.5C	The functional specification shall describe all direct error messages that can result from an invocation of each TSFI.
ADV_FSP.4.6C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Table 106 — Evaluator action elements for ADV_FSP.4

ADV_FSP.4.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.4.2E	The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

10.3.8 Complete semi-formal functional specification with additional error information (ADV_FSP.5)

10.3.8.1 Component relationships

Dependencies: ADV_TDS.1 (Basic design) ([10.7.4](#))
 ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))

10.3.8.2 Component elements

[Table 107](#) specifies the developer action elements for ADV_FSP.5, with [Table 108](#) indicating their related content and presentation requirements, and [Table 109](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 107 — Developer action elements for ADV_FSP.5

ADV_FSP.5.1D	The developer shall provide a functional specification.
ADV_FSP.5.2D	The developer shall provide a tracing from the functional specification to the SFRs.

Table 108 — Content and presentation elements for ADV_FSP.5

ADV_FSP.5.1C	The functional specification shall completely represent the TSF.
ADV_FSP.5.2C	The functional specification shall describe the TSFIs using a semi-formal style.
ADV_FSP.5.3C	The functional specification shall describe the purpose and method of use for all TSFIs.
ADV_FSP.5.4C	The functional specification shall identify and describe all parameters associated with each TSFI.
ADV_FSP.5.5C	The functional specification shall describe all actions associated with each TSFI.
ADV_FSP.5.6C	The functional specification shall describe all direct error messages that can result from an invocation of each TSFI.
ADV_FSP.5.7C	The functional specification shall describe all error messages that do not result from an invocation of a TSFI.
ADV_FSP.5.8C	The functional specification shall provide a rationale for each error message contained in the TSF implementation yet does not result from an invocation of a TSFI.
ADV_FSP.5.9C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Table 109 — Evaluator action elements for ADV_FSP.5

ADV_FSP.5.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.5.2E	The evaluator <i>shall determine</i> that the functional specification is an accurate and complete instantiation of the SFRs.

10.3.9 Complete semi-formal functional specification with additional formal specification (ADV_FSP.6)

10.3.9.1 Component relationships

Dependencies: ADV_TDS.1 (Basic design) ([10.7.4](#))
 ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))

10.3.9.2 Component elements

[Table 110](#) specifies the developer action elements for ADV_FSP.6, with [Table 111](#) indicating their related content and presentation requirements, and [Table 112](#) the corresponding evaluation actions.

Table 110 — Developer action elements for ADV_FSP.6

ADV_FSP.6.1D	The developer shall provide a functional specification.
ADV_FSP.6.2D	The developer shall provide a formal presentation of the functional specification of the TSF.
ADV_FSP.6.3D	The developer shall provide a tracing from the functional specification to the SFRs.

ISO/IEC 15408-3:2026(en)

Table 111 — Content and presentation elements for ADV_FSP.6

ADV_FSP.6.1C	The functional specification shall completely represent the TSF.
ADV_FSP.6.2C	The functional specification shall describe the TSFIs using a semi-formal style.
ADV_FSP.6.3C	The functional specification shall describe the purpose and method of use for all TSFIs.
ADV_FSP.6.4C	The functional specification shall identify and describe all parameters associated with each TSFI.
ADV_FSP.6.5C	The functional specification shall describe all actions associated with each TSFI.
ADV_FSP.6.6C	The functional specification shall describe all direct error messages that can result from an invocation of each TSFI.
ADV_FSP.6.7C	The functional specification shall describe all error messages contained in the TSF implementation representation.
ADV_FSP.6.8C	The functional specification shall provide a rationale for each error message contained in the TSF implementation that is not otherwise described in the functional specification justifying why it is not associated with a TSFI.
ADV_FSP.6.9C	The formal presentation of the functional specification of the TSF shall describe the TSFIs using a formal style, supported by informal, explanatory text where appropriate.
ADV_FSP.6.10C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Table 112 — Evaluator action elements for ADV_FSP.6

ADV_FSP.6.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_FSP.6.2E	The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

10.4 Implementation representation (ADV_IMP)

10.4.1 Objectives

The function of the ADV_IMP (Implementation representation) family is for the developer to make available the implementation representation (and, at higher levels, the implementation itself) of the TOE in a form that can be analysed by the evaluator. The implementation representation is used in analysis activities for other families (analysing the TOE design, for instance) to demonstrate that the TOE conforms its design and to provide a basis for analysis in other areas of the evaluation (e.g. the search for vulnerabilities). The implementation representation is expected to be in a form that captures the detailed internal workings of the TSF. This can be software source code, firmware source code, hardware diagrams, IC hardware design language code or layout data, or a combination.

10.4.2 Component levelling

The components in this family are levelled on the amount of implementation that is mapped to the TOE design description.

10.4.3 Application notes

Source code, hardware diagrams, IC hardware design language code and layout data that are used to build the actual hardware are examples of parts of an implementation representation. It is important to note that while the implementation representation shall be made available to the evaluator, this does not imply that the evaluator is required to possess that representation. For instance, the developer can require that the evaluator review the implementation representation at a site of the developer's choosing.

The entire implementation representation is made available to ensure that analysis activities are not curtailed due to lack of information. This does not, however, imply that all the representation is examined when the analysis activities are being performed. This is likely impractical in almost all cases, in addition to the fact that it most likely will not result in a higher-assurance TOE vs. targeted sampling of the implementation representation. The implementation representation is made available to allow analysis of

ISO/IEC 15408-3:2026(en)

other TOE design decompositions (e.g. functional specification, TOE design), and to gain confidence that the security functionality described at a higher level in the design actually appear to be implemented in the TOE. Conventions in some forms of the implementation representation can make it difficult or impossible to determine from just the implementation representation itself what the actual result of the compilation or run-time interpretation will be. For example, compiler directives for C language compilers will cause the compiler to exclude or include entire portions of the code. For this reason, it is important that such "extra" information or related tools (e.g. scripts, compilers) be provided so that the implementation representation can be accurately determined.

The purpose of the mapping between the implementation representation and the TOE design description is to aid the evaluator's analysis. The internal workings of the TOE can be better understood when the TOE design is analysed with corresponding portions of the implementation representation. The mapping serves as an index into the implementation representation. At the lower component, only a subset of the implementation representation is mapped to the TOE design description. As it is uncertain as to which portions of the implementation representation will require such a mapping, the developer may choose either to map the entire implementation representation beforehand, or to wait to see which portions of the implementation representation the evaluator requires to be mapped.

The implementation representation is manipulated by the developer in a form that is suitable for transformation to the actual implementation. For instance, the developer can work with files containing source code, which is eventually compiled to become part of the TSF. The developer makes available the implementation representation in the form used by the developer, so that the evaluator can use automated techniques in the analysis. This also increases the confidence that the implementation representation examined is actually the one used in the production of the TSF (as opposed to the case where it is supplied in an alternate presentation format, such as a word processor document). It should be noted that other forms of the implementation representation can also be used by the developer; these forms are supplied as well. The overall goal is to supply the evaluator with the information that will maximize the effectiveness of the evaluator's analysis efforts.

Some forms of the implementation representation can require additional information because they introduce significant barriers to understanding and analysis. Examples include "shrouded" source code or source code that has been obfuscated in other ways such that it prevents either understanding or analysis, or both. These forms of implementation representation typically result from the TOE developer taking a version of the implementation representation and running a shrouding or obfuscation program on it. While the shrouded representation is what is compiled and can be closer to the implementation (in terms of structure) than the original, un-shrouded representation, supplying such obfuscated code can cause significantly more time to be spent in analysis tasks involving the representation. When such forms of representation are created, the components require details on the shrouding tools and algorithms used so that the un-shrouded representation can be supplied, and the additional information can be used to gain confidence that the shrouding process does not compromise any security functionality.

10.4.4 Implementation representation of the TSF (ADV_IMP.1)

10.4.4.1 Component relationships

Dependencies: ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ALC_TAT.1 (Well-defined development tools) ([12.9.4](#))

10.4.4.2 Component elements

[Table 113](#) specifies the developer action elements for ADV_IMP.1, with [Table 114](#) indicating their related content and presentation requirements, and [Table 115](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 113 — Developer action elements for ADV_IMP.1

ADV_IMP.1.1D	The developer shall make available the implementation representation for the entire TSF.
ADV_IMP.1.2D	The developer shall provide a mapping between the TOE design description and the sample of the implementation representation.

Table 114 — Content and presentation elements for ADV_IMP.1

ADV_IMP.1.1C	The implementation representation shall define the TSF to a level of detail such that the TSF can be generated without further design decisions.
ADV_IMP.1.2C	The implementation representation shall be in the form used by the development personnel.
ADV_IMP.1.3C	The mapping between the TOE design description and the sample of the implementation representation shall demonstrate their correspondence.

Table 115 — Evaluator action elements for ADV_IMP.1

ADV_IMP.1.1E	The evaluator <i>shall confirm</i> that, for the selected sample of the implementation representation, the information provided meets all requirements for content and presentation of evidence.
--------------	--

10.4.5 Complete mapping of the implementation representation of the TSF (ADV_IMP.2)

10.4.5.1 Component relationships

Dependencies: ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ALC_TAT.1 (Well-defined development tools) ([12.9.4](#))
 ALC_CMC.5 (Advanced support) ([12.2.8](#))

10.4.5.2 Component elements

[Table 116](#) specifies the developer action elements for ADV_IMP.2, with [Table 117](#) indicating their related content and presentation requirements, and [Table 118](#) the corresponding evaluation actions.

Table 116 — Developer action elements for ADV_IMP.2

ADV_IMP.2.1D	The developer shall make available the implementation representation for the entire TSF.
ADV_IMP.2.2D	The developer shall provide a mapping between the TOE design description and the entire implementation representation.

Table 117 — Content and presentation elements for ADV_IMP.2

ADV_IMP.2.1C	The implementation representation shall define the TSF to a level of detail such that the TSF can be generated without further design decisions.
ADV_IMP.2.2C	The implementation representation shall be in the form used by the development personnel.
ADV_IMP.2.3C	The mapping between the TOE design description and the entire implementation representation shall demonstrate their correspondence.

Table 118 — Evaluator action elements for ADV_IMP.2

ADV_IMP.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

10.5 TSF internals (ADV_INT)

10.5.1 Objectives

This family addresses the assessment of the internal structure of the TSF. A TSF whose internals are well-structured is easier to implement and less likely to contain flaws that can lead to vulnerabilities; it is also easier to maintain without the introduction of flaws.

10.5.2 Component levelling

The components in this family are levelled on the basis of the amount of structure and minimization of complexity required. ADV_INT.1 (Well-structured subset of TSF internals) ([10.5.4](#)) places requirements for well-structured internals on only selected parts of the TSF. This component is not included in an EAL because this component is viewed for use in special circumstances (e.g. the sponsor has a specific concern regarding a cryptographic module, which is isolated from the rest of the TSF) and would not be widely applicable.

At the next level, the requirements for well-structured internals are placed on the entire TSF. Finally, minimization of complexity is introduced in the highest component.

10.5.3 Application notes

These requirements, when applied to the internal structure of the TSF, typically result in improvements that aid both the developer and the evaluator in understanding the TSF, and also provide the basis for designing and evaluating test suites. Further, improving understandability of the TSF is expected to assist the developer in simplifying its maintainability.

The requirements in this family are presented at an abstract level. The wide variety of TOEs makes it impossible to codify anything more specific than "well-structured" or "minimum complexity". Judgements on structure and complexity are expected to be derived from the specific technologies used in the TOE. For example, software is likely to be considered well-structured if it exhibits the characteristics cited in the software engineering disciplines. The components within this family call for identifying the standards for measuring the characteristic of being well-structured and not overly complex.

10.5.4 Well-structured subset of TSF internals (ADV_INT.1)

10.5.4.1 Component relationships

Dependencies: ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
 ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ALC_TAT.1 (Well-defined development tools) ([12.9.4](#))

10.5.4.2 Objectives

The objective of this component is to provide a means for requiring specific portions of the TSF to be well-structured. The intent is that the entire TSF has been designed and implemented using sound engineering principles, but the analysis is performed upon only a specific subset.

10.5.4.3 Application notes

This component requires the PP or ST author to fill in an assignment with the subset of the TSF. This subset can be identified in terms of the internals of the TSF at any layer of abstraction. For example:

- the structural elements of the TSF as identified in the TOE design (e.g. "The developer shall design and implement the audit subsystem such that it has well-structured internals.");
- the implementation (e.g. "The developer shall design and implement the encrypt.c and decrypt.c files such that it has well-structured internals." or "The developer shall design and implement the 6227 IC chip such that it has well-structured internals.").

ISO/IEC 15408-3:2026(en)

It is likely this would not be readily accomplished by referencing the claimed SFRs (e.g. “The developer shall design and implement the portion of the TSF that provide anonymity as defined in FPR_ANO.2 (Anonymity without soliciting information) (see ISO/IEC 15408-2:2026, 14.3.7) such that it has well-structured internals.”) because this does not indicate where to focus the analysis.

This component has limited value and would be suitable in cases where potentially malicious users/subjects have limited or strictly controlled access to the TSFIs or where there is another means of protection (e.g. domain separation) that ensures the chosen subset of the TSF cannot be adversely affected by the rest of the TSF (e.g. the cryptographic functionality, which is isolated from the rest of the TSF, is well-structured).

10.5.4.4 Component elements

[Table 119](#) specifies the developer action elements for ADV_INT.1, with [Table 120](#) indicating their related content and presentation requirements, and [Table 121](#) the corresponding evaluation actions.

Table 119 — Developer action elements for ADV_INT.1

ADV_INT.1.1D	The developer shall design and implement [assignment ^(a1) : <i>subset of the TSF</i>] such that it has well-structured internals.
ADV_INT.1.2D	The developer shall provide an internals description and justification.

Table 120 — Content and presentation elements for ADV_INT.1

ADV_INT.1.1C	The justification shall explain the characteristics used to judge the meaning of "well-structured".
ADV_INT.1.2C	The TSF internals description shall demonstrate that the assigned subset of the TSF is well-structured.

Table 121 — Evaluator action elements for ADV_INT.1

ADV_INT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ADV_INT.1.2E	The evaluator <i>shall perform</i> an internals analysis on the assigned subset of the TSF.

10.5.5 Well-structured internals (ADV_INT.2)

10.5.5.1 Component relationships

Dependencies: ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
 ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ALC_TAT.1 (Well-defined development tools) ([12.9.4](#))

10.5.5.2 Objectives

The objective of this component is to provide a means for requiring the TSF to be well-structured. The intent is that the entire TSF has been designed and implemented using sound engineering principles.

10.5.5.3 Application notes

Judgements on the adequacy of the structure are expected to be derived from the specific technologies used in the TOE. This component calls for identifying the standards for measuring the characteristic of being well-structured.

10.5.5.4 Component elements

[Table 122](#) specifies the developer action elements for ADV_INT.2, with [Table 123](#) indicating their related content and presentation requirements, and [Table 124](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 122 — Developer action elements for ADV_INT.2

ADV_INT.2.1D	The developer shall design and implement the entire TSF such that it has well-structured internals.
ADV_INT.2.2D	The developer shall provide an internals description and justification.

Table 123 — Content and presentation elements for ADV_INT.2

ADV_INT.2.1C	The justification shall describe the characteristics used to judge the meaning of "well-structured".
ADV_INT.2.2C	The TSF internals description shall demonstrate that the entire TSF is well-structured.

Table 124 — Evaluator action elements for ADV_INT.2

ADV_INT.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_INT.2.2E	The evaluator shall perform an internals analysis on the TSF.

10.5.6 Minimally complex internals (ADV_INT.3)

10.5.6.1 Component relationships

Dependencies: ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
 ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ALC_TAT.1 (Well-defined development tools) ([12.9.4](#))

10.5.6.2 Objectives

The objective of this component is to provide a means for requiring the TSF to be well-structured and of minimal complexity. The intent is that the entire TSF has been designed and implemented using sound engineering principles.

10.5.6.3 Application notes

Judgements on the adequacy of the structure and complexity are expected to be derived from the specific technologies used in the TOE. This component calls for identifying the standards for measuring the structure and complexity.

10.5.6.4 Component elements

[Table 125](#) specifies the developer action elements for ADV_INT.3, with [Table 126](#) indicating their related content and presentation requirements, and [Table 127](#) the corresponding evaluation actions.

Table 125 — Developer action elements for ADV_INT.3

ADV_INT.3.1D	The developer shall design and implement the entire TSF such that it has well-structured internals.
ADV_INT.3.2D	The developer shall provide an internals description and justification.

Table 126 — Content and presentation elements for ADV_INT.3

ADV_INT.3.1C	The justification shall describe the characteristics used to judge the meaning of "well-structured" and "complex" .
ADV_INT.3.2C	The TSF internals description shall demonstrate that the entire TSF is well-structured and is not overly complex .

Table 127 — Evaluator action elements for ADV_INT.3

ADV_INT.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_INT.3.2E	The evaluator shall perform an internals analysis on the entire TSF.

10.6 Formal TSF model (ADV_SPM)

10.6.1 Objectives

It is the objective of this family to provide additional assurance through the development of a formal representation of the TSF (the formal model) and its properties (the formal properties), as defined by the SFRs and the security objectives of the ST. The following are expected to be established:

- proof that these formal properties hold in the formal model;
- a correspondence rationale that the TOE functional specification preserves the formal properties proven for the formal model.

A formal proof or semi-formal demonstration of preservation of the formal properties in the formal or semi-formal specification is expected if the latter exists (ADV_FSP.6 (Complete semi-formal functional specification with additional formal specification) (10.3.9) or ADV_FSP.5 (Complete semi-formal functional specification with additional error information) (10.3.8), respectively).

10.6.2 Component levelling

This family contains only one component.

10.6.3 Application notes

Inadequacies in a TOE can result either from a failure in understanding the security requirements or from a flawed implementation of those security requirements. Defining the security requirements adequately to ensure their understanding can be problematic because the definition is required to be sufficiently precise to prevent undesired results or subtle flaws during the implementation of the TOE. Throughout the design, implementation, and review processes, a formal representation of the TSF and its properties can be used as precise design and implementation guidance, thereby providing increased assurance that the SFRs and the security objectives of the ST are satisfied by the TOE. The resulting guidance and the precision of the TSF representation and its properties, as defined by the SFRs and the security objectives of the ST, are significantly improved by defining the formal model and specifying the formal properties using a formal language and providing a formal proof that these formal properties hold in the formal model.

The creation of a formal Security Policy Model (SPM) of the TSF shall be complete with respect to the ST; such a model helps to identify and eliminate ambiguous, inconsistent, contradictory, or unenforceable elements and to avoid any misunderstanding on the scope. To this end, the evaluation shall determine whether the formal model and the formal properties completely cover the ST and accept only STs and SPMs that match in scope. Once the TOE has been built, the formal model serves the evaluation effort by contributing to the evaluator's judgement of how well the developer has understood the TSF being implemented and whether there are inconsistencies between the formal properties as defined by the security objectives of the ST and the TOE design. The confidence gained by formally proving properties of the model is accompanied by confidence gained by defining a correspondence rationale between the formal model and the TOE functional specification (as defined for ADV_FSP (Functional specification) (10.3)). The correspondence rationale consists of a formal proof when mapping to formal aspects of the TOE functional specification and semi-formal demonstration otherwise. A combination of different formal systems (modelling languages, tools, proof systems) can be used for different parts of the ST (SFRs and Security Objectives) and correspondence rationales.

ISO/IEC 15408-3:2026(en)

10.6.4 Formal TSF model (ADV_SPM.1)

10.6.4.1 Component relationships

Dependencies: ASE_OBJ.2 (Security objectives) (9.5.4)
 ASE_REQ.2 (Derived security requirements) (9.7.4)
 ADV_FSP.4 (Complete functional specification) (10.3.7)

10.6.4.2 Component elements

Table 128 specifies the developer action elements for ADV_SPM.1, with Table 129 indicating their related content and presentation requirements, and Table 130 the corresponding evaluation actions.

Table 128 — Developer action elements for ADV_SPM.1

ADV_SPM.1.1D	The developer shall provide a formal model for the TSF supported by explanatory text.
ADV_SPM.1.2D	The developer shall provide the set of formal properties for the TOE supported by explanatory text.
ADV_SPM.1.3D	The developer shall provide a formal proof that the model satisfies the formal properties supported by explanatory text.
ADV_SPM.1.4D	The developer shall provide a correspondence rationale between the formal model and the functional specification.
ADV_SPM.1.5D	The developer shall provide a semi-formal demonstration of correspondence between the formal model and any semi-formal functional specification.
ADV_SPM.1.6D	The developer shall provide a formal proof of correspondence between the formal model and any formal functional specification.
ADV_SPM.1.7D	The developer shall provide all the tools used for the formal model, the formal properties, proofs and demonstrations.

Table 129 — Content and presentation elements for ADV_SPM.1

ADV_SPM.1.1C	The formal model, properties and proofs shall be defined using a well-founded mathematical theory.
ADV_SPM.1.2C	The explanatory text shall cover the entire formal model, formal properties and proofs, including instructions for reproducing the proofs and the correspondence rationale, and it shall provide a rationale for the modelling and verification choices.
ADV_SPM.1.3C	The formal model shall cover the complete set of SFRs that define the TSF.
ADV_SPM.1.4C	The formal properties shall cover the complete set of security objectives for the TOE.
ADV_SPM.1.5C	The formal proof shall show that the formal model satisfies all the formal properties and that the consistency of the underlying mathematical theory is preserved.
ADV_SPM.1.6C	The correspondence rationale shall show that the formal properties proven for the formal model hold for the functional specification.
ADV_SPM.1.7C	The semi-formal demonstration of correspondence shall show that the formal properties proven for the formal model hold for any semi-formal functional specification.
ADV_SPM.1.8C	The formal proof of correspondence shall show that the properties proven for the formal model hold for any formal functional specification.
ADV_SPM.1.9C	Any tool used to model or to prove the formal properties or the relationship between the formal model and the functional specification shall be well-defined and unambiguously identified and it shall be accompanied by documentation and a rationale of the tool's suitability and trustworthiness.

Table 130 — Evaluator action elements for ADV_SPM.1

ADV_SPM.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

10.7 TOE design (ADV_TDS)

10.7.1 Objectives

The design description of a TOE provides both context for a description of the TSF, and a thorough description of the TSF. As assurance needs increase, the level of detail provided in the description also increases. As the size and complexity of the TSF increase, multiple levels of decomposition are appropriate. The design requirements are intended to provide information (commensurate with the given assurance level) so that a determination can be made that the SFRs are realized.

10.7.2 Component levelling

The components in this family are levelled on the basis of the amount of information that is required to be presented with respect to the TSF, and on the degree of formalism required of the design description.

10.7.3 Application notes

10.7.3.1 General

The goal of design documentation is to provide sufficient information to determine the TSF boundary, and to describe how the TSF implements the SFRs. The amount and structure of the design documentation will depend on the complexity of the TOE and the number of SFRs; in general, a very complex TOE with a large number of SFRs will require more design documentation than a very simple TOE implementing only a few SFRs. Very complex TOEs will benefit (in terms of the assurance provided) from the production of differing levels of decomposition in describing the design, while very simple TOEs do not require both high-level and low-level descriptions of its implementation.

This family uses two levels of decomposition: the subsystem and the module. A module is the most specific description of functionality: it is a description of the implementation. A developer can implement the part of the TOE described by the module with no further design decisions. A subsystem is a description of the design of the TOE; it helps to provide a high-level description of what a portion of the TOE is doing and how. As such, a subsystem can be further divided into lower-level subsystems, or into modules. Very complex TOEs can require several levels of subsystems in order to adequately convey a useful description of how the TOE works. In contrast, it is possible that very simple TOEs, do not require a subsystem level of description; the module can clearly describe how the TOE works.

The general approach adopted for design documentation is that, as the level of assurance increases, the emphasis of description shifts from the general (subsystem level) to more (module level) detail. In cases where a module-level of abstraction is appropriate because the TOE is simple enough to be described at the module level, yet the level of assurance calls for a subsystem level of description, the module-level description alone will suffice. For complex TOEs, however, this is not the case: an enormous amount of (module-level) detail would be incomprehensible without an accompanying subsystem level of description.

This approach follows the general paradigm that providing additional detail about the implementation of the TSF will result in greater assurance that the SFRs are implemented correctly and provide information that can be used to demonstrate this in testing class ATE (Tests) ([Clause 13](#)).

In the requirements for this family, the term interface is used as the means of communication (between two modules). It describes how the communication is invoked; this is similar to the details of TSFIs (see ADV_FSP (Functional specification) ([10.3](#))). The term interaction is used to identify the purpose for communication; it identifies why two subsystems or modules are communicating.

10.7.3.2 Detail about the subsystems and modules

The requirements identify details about the subsystems and modules that shall be provided, such as the following:

- the subsystems and modules are identified with a simple list of what they are;

ISO/IEC 15408-3:2026(en)

- subsystems and modules can be categorized (either implicitly or explicitly) as "SFR-enforcing", "SFR-supporting", or "SFR-non-interfering"; these terms are used the same as they are used in ADV_FSP (Functional specification) ([10.3](#));
- a subsystem's behaviour is what it does. The behaviour can also be categorized as SFR-enforcing, SFR-supporting, or SFR-non-interfering. The behaviour of the subsystem is never categorized as more SFR-relevant than the category of the subsystem itself. For example, an SFR-enforcing subsystem can have SFR-enforcing behaviour as well as SFR-supporting or SFR-non-interfering behaviour;
- a behaviour summary of a subsystem is an overview of the actions it performs (e.g. "The TCP subsystem assembles IP datagrams into reliable byte streams");
- a behaviour description of a subsystem is an explanation of everything it does. This description should be at a level of detail allowing ready determination of whether the behaviour has any relevance to the enforcement of the SFRs;
- a description of interactions among or between subsystems or modules identifies the reason that subsystems or modules communicate and characterizes the information that is passed. It is not necessary to define the information to the same level of detail as an interface specification. For example, it would be sufficient to say "subsystem X requests a block of memory from the memory manager, which responds with the location of the allocated memory";
- a description of interfaces provides the details of how the interactions among modules are achieved. Rather than describing the reason the modules are communicating or the purpose of their communication (i.e. the description of interactions), the description of interfaces describes the details of how that communication is accomplished, in terms of the structure and contents of the messages, semaphores, internal process communications;
- the purpose describes how a module provides their functionality. It provides sufficient detail that no further design decisions are needed. The correspondence between the implementation representation that implements the module, and the purpose of the module should be readily apparent;
- a module is otherwise described in terms of whatever is identified in the element.

Subsystems and modules, and "SFR-enforcing" are all further explained in greater detail in [A.4](#).

10.7.4 Basic design (ADV_TDS.1)

10.7.4.1 Component relationships

Dependencies: ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#))

10.7.4.2 Component elements

[Table 131](#) specifies the developer action elements for ADV_TDS.1, with [Table 132](#) indicating their related content and presentation requirements, and [Table 133](#) the corresponding evaluation actions.

Table 131 — Developer action elements for ADV_TDS.1

ADV_TDS.1.1D	The developer shall provide the design of the TOE.
ADV_TDS.1.2D	The developer shall provide a mapping from the TSFIs of the functional specification to the lowest level of decomposition available in the TOE design.

ISO/IEC 15408-3:2026(en)

Table 132 — Content and presentation elements for ADV_TDS.1

ADV_TDS.1.1C	The design shall describe the structure of the TOE in terms of subsystems.
ADV_TDS.1.2C	The design shall identify all subsystems of the TSF.
ADV_TDS.1.3C	The design shall provide the behaviour summary of each SFR-supporting or SFR-non-interfering TSF subsystem.
ADV_TDS.1.4C	The design shall summarize the SFR-enforcing behaviour of the SFR-enforcing subsystems.
ADV_TDS.1.5C	The design shall provide a description of the interactions among SFR-enforcing subsystems of the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.
ADV_TDS.1.6C	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.

Table 133 — Evaluator action elements for ADV_TDS.1

ADV_TDS.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ADV_TDS.1.2E	The evaluator <i>shall determine</i> that the design is an accurate and complete instantiation of all security functional requirements.

10.7.5 Architectural design (ADV_TDS.2)

10.7.5.1 Component relationships

Dependencies: ADV_FSP.3 (Functional specification with complete summary) ([10.3.6](#))

10.7.5.2 Component elements

[Table 134](#) specifies the developer action elements for ADV_TDS.2, with [Table 135](#) indicating their related content and presentation requirements, and [Table 136](#) the corresponding evaluation actions.

Table 134 — Developer action elements for ADV_TDS.2

ADV_TDS.2.1D	The developer shall provide the design of the TOE.
ADV_TDS.2.2D	The developer shall provide a mapping from the TSFIs of the functional specification to the lowest level of decomposition available in the TOE design.

Table 135 — Content and presentation elements for ADV_TDS.2

ADV_TDS.2.1C	The design shall describe the structure of the TOE in terms of subsystems.
ADV_TDS.2.2C	The design shall identify all subsystems of the TSF.
ADV_TDS.2.3C	The design shall provide the behaviour summary of each SFR non-interfering subsystem of the TSF.
ADV_TDS.2.4C	The design shall describe the SFR-enforcing behaviour of the SFR-enforcing subsystems.
ADV_TDS.2.5C	The design shall summarize the SFR-supporting and SFR-non-interfering behaviour of the SFR-enforcing subsystems.
ADV_TDS.2.6C	The design shall summarize the behaviour of the SFR-supporting subsystems.
ADV_TDS.2.7C	The design shall provide a description of the interactions among all subsystems of the TSF.
ADV_TDS.2.8C	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.

ISO/IEC 15408-3:2026(en)

Table 136 — Evaluator action elements for ADV_TDS.2

ADV_TDS.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_TDS.2.2E	The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

10.7.6 Basic modular design (ADV_TDS.3)

10.7.6.1 Component relationships

Dependencies: ADV_FSP.4 (Complete functional specification) ([10.3.7](#))

10.7.6.2 Component elements

[Table 137](#) specifies the developer action elements for ADV_TDS.3, with [Table 138](#) indicating their related content and presentation requirements, and [Table 139](#) the corresponding evaluation actions.

Table 137 — Developer action elements for ADV_TDS.3

ADV_TDS.3.1D	The developer shall provide the design of the TOE.
ADV_TDS.3.2D	The developer shall provide a mapping from the TSFIs of the functional specification to the lowest level of decomposition available in the TOE design.

Table 138 — Content and presentation elements for ADV_TDS.3

ADV_TDS.3.1C	The design shall describe the structure of the TOE in terms of subsystems.
ADV_TDS.3.2C	The design shall describe the TSF in terms of modules.
ADV_TDS.3.3C	The design shall identify all subsystems of the TSF.
ADV_TDS.3.4C	The design shall provide a description of each subsystem of the TSF.
ADV_TDS.3.5C	The design shall provide a description of the interactions among all subsystems of the TSF.
ADV_TDS.3.6C	The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
ADV_TDS.3.7C	The design shall describe each SFR-enforcing module in terms of its purpose and relationship with other modules.
ADV_TDS.3.8C	The design shall describe each SFR-enforcing module in terms of its SFR-related interfaces, return values from those interfaces, interaction with other modules and called SFR-related interfaces to other SFR-enforcing modules.
ADV_TDS.3.9C	The design shall describe each SFR-supporting and SFR-non-interfering module in terms of its purpose and interaction with other modules.
ADV_TDS.3.10C	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.

Table 139 — Evaluator action elements for ADV_TDS.3

ADV_TDS.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_TDS.3.2E	The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

ISO/IEC 15408-3:2026(en)

10.7.7 Semi-Formal modular design (ADV_TDS.4)

10.7.7.1 Component relationships

Dependencies: ADV_FSP.5 (Complete semi-formal functional specification with additional error information) ([10.3.8](#))

10.7.7.2 Component elements

[Table 140](#) specifies the developer action elements for ADV_TDS.4, with [Table 141](#) indicating their related content and presentation requirements, and [Table 142](#) the corresponding evaluation actions.

Table 140 — Developer action elements for ADV_TDS.4

ADV_TDS.4.1D	The developer shall provide the design of the TOE.
ADV_TDS.4.2D	The developer shall provide a mapping from the TSFIs of the functional specification to the lowest level of decomposition available in the TOE design.

Table 141 — Content and presentation elements for ADV_TDS.4

ADV_TDS.4.1C	The design shall describe the structure of the TOE in terms of subsystems.
ADV_TDS.4.2C	The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-supporting, or SFR-non-interfering.
ADV_TDS.4.3C	The design shall identify all subsystems of the TSF.
ADV_TDS.4.4C	The design shall provide a semi-formal description of each subsystem of the TSF, supported by informal, explanatory text where appropriate.
ADV_TDS.4.5C	The design shall provide a description of the interactions among all subsystems of the TSF.
ADV_TDS.4.6C	The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
ADV_TDS.4.7C	The design shall describe each SFR-enforcing and SFR-supporting module in terms of its purpose and relationship with other modules.
ADV_TDS.4.8C	The design shall describe each SFR-enforcing and SFR-supporting module in terms of its SFR-related interfaces, return values from those interfaces, interaction with other modules and called SFR-related interfaces to other SFR-enforcing or SFR-supporting modules.
ADV_TDS.4.9C	The design shall describe each SFR-non-interfering module in terms of its purpose and interaction with other modules.
ADV_TDS.4.10C	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.

Table 142 — Evaluator action elements for ADV_TDS.4

ADV_TDS.4.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_TDS.4.2E	The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

10.7.8 Complete semi-formal modular design (ADV_TDS.5)

10.7.8.1 Component relationships

Dependencies: ADV_FSP.5 (Complete semi-formal functional specification with additional error information) ([10.3.8](#))

10.7.8.2 Component elements

[Table 143](#) specifies the developer action elements for ADV_TDS.5, with [Table 144](#) indicating their related content and presentation requirements, and [Table 145](#) the corresponding evaluation actions.

Table 143 — Developer action elements for ADV_TDS.5

ADV_TDS.5.1D	The developer shall provide the design of the TOE.
ADV_TDS.5.2D	The developer shall provide a mapping from the TSFIs of the functional specification to the lowest level of decomposition available in the TOE design.

Table 144 — Content and presentation elements for ADV_TDS.5

ADV_TDS.5.1C	The design shall describe the structure of the TOE in terms of subsystems.
ADV_TDS.5.2C	The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-supporting, or SFR-non-interfering.
ADV_TDS.5.3C	The design shall identify all subsystems of the TSF.
ADV_TDS.5.4C	The design shall provide a semi-formal description of each subsystem of the TSF, supported by informal, explanatory text where appropriate.
ADV_TDS.5.5C	The design shall provide a description of the interactions among all subsystems of the TSF.
ADV_TDS.5.6C	The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
ADV_TDS.5.7C	The design shall provide a semi-formal description of each module in terms of its purpose, interaction , interfaces, return values from those interfaces, and called interfaces to other modules, supported by informal, explanatory text where appropriate .
ADV_TDS.5.8C	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.

Table 145 — Evaluator action elements for ADV_TDS.5

ADV_TDS.5.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_TDS.5.2E	The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

10.7.9 Complete semi-formal modular design with formal high-level design presentation (ADV_TDS.6)

10.7.9.1 Component relationships

Dependencies: ADV_FSP.6 (Complete semi-formal functional specification with additional formal specification) ([10.3.9](#))

10.7.9.2 Component elements

[Table 146](#) specifies the developer action elements for ADV_TDS.6, with [Table 147](#) indicating their related content and presentation requirements, and [Table 148](#) the corresponding evaluation actions.

Table 146 — Developer action elements for ADV_TDS.6

ADV_TDS.6.1D	The developer shall provide the design of the TOE.
ADV_TDS.6.2D	The developer shall provide a mapping from the TSFIs of the functional specification to the lowest level of decomposition available in the TOE design.
ADV_TDS.6.3D	The developer shall provide a formal specification of the TSF subsystems.
ADV_TDS.6.4D	The developer shall provide a proof of correspondence between the formal specifications of the TSF subsystems and of the functional specification.

ISO/IEC 15408-3:2026(en)

Table 147 — Content and presentation elements for ADV_TDS.6

ADV_TDS.6.1C	The design shall describe the structure of the TOE in terms of subsystems.
ADV_TDS.6.2C	The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-supporting, or SFR-non-interfering.
ADV_TDS.6.3C	The design shall identify all subsystems of the TSF.
ADV_TDS.6.4C	The design shall provide a semi-formal description of each subsystem of the TSF, supported by informal, explanatory text where appropriate.
ADV_TDS.6.5C	The design shall provide a description of the interactions among all subsystems of the TSF.
ADV_TDS.6.6C	The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
ADV_TDS.6.7C	The design shall describe each module in semi-formal style in terms of its purpose, interaction, interfaces, return values from those interfaces, and called interfaces to other modules, supported by informal, explanatory text where appropriate.
ADV_TDS.6.8C	The formal specification of the TSF subsystems shall describe the TSF using a formal style, supported by informal, explanatory text where appropriate.
ADV_TDS.6.9C	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that they invoke.
ADV_TDS.6.10C	The proof of correspondence between the formal specifications of the TSF subsystems and of the functional specification shall demonstrate that all behaviour described in the TOE design is a correct and complete refinement of the TSFI that invoked it.

Table 148 — Evaluator action elements for ADV_TDS.6

ADV_TDS.6.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ADV_TDS.6.2E	The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.

10.8 Composite design compliance (ADV_COMP)

10.8.1 Objectives

The aim of this family is to determine whether the requirements on the dependent component, imposed by the related base component, are fulfilled in the composite product.

10.8.2 Component levelling

This family contains only one component.

10.8.3 Application notes

The requirements on the dependent component, imposed by the related base component, can be formulated in the relevant base component-related user guidance, ETR for composite evaluation (in form of observations and recommendations) and report of the base component evaluation authority (e.g. in form of constraints and recommendations). The developer of the dependent component shall regard each of these sources, if available, and implement the dependent component in such a way that the applicable requirements are fulfilled. The composite product evaluator shall verify that all stipulations for the dependent component that are imposed by the base component and provided in its evaluation related documentation are fulfilled by the composite product, i.e. have been taken into account by the dependent component developer.

The composite product evaluation sponsor shall ensure that the following is made available for the composite product evaluator:

- the base component-related user guidance,
- the base component-related ETR for composite evaluation prepared by the base component evaluator,

ISO/IEC 15408-3:2026(en)

- the report of the base component evaluation authority,
- a rationale for secure composite product implementation including evidence prepared by the dependent component developer.

The TSF of the composite product are represented at various levels of abstraction in the families of ADV (Development). From experience, the appropriate levels of design representation for examining, whether the requirements of the base component are fulfilled by the composite product, are the TOE design ADV_TDS (TOE design) (10.7), security architecture ADV_ARC (Security architecture) (10.2) and ADV_IMP (Implementation representation) (10.4). If the design representation is not available (e.g. because the assurance package chosen is EAL1 (see ISO/IEC 15408-5:2026, 4.4), the current family is not applicable (see the next paragraph for the reason).

Due to the definition of the composite product the interface between its base component and dependent component is the internal one, hence, a ADV_FSP (Functional specification) (10.3) as representation level is not appropriate for analysing the design compliance.

ADV_ARC (Security architecture) (10.2) as assurance family is dedicated to ensuring that integrative security services like domain separation, self-protection and non-bypassability properly work. It is impossible and not the sense of the composite evaluation to have an insight into the architectural internals of the related base component (it is a matter of the base component evaluation). In the context of the ADV_ARC (Security architecture) (10.2), the composite product evaluator shall:

- determine whether the dependent component uses services of the related base component within its own composite product ST to provide domain separation, self-protection, non-bypassability and protected start-up; if no, there are no further composite activities for ADV_ARC (Security architecture) (10.2); if yes, then
- the evaluator shall determine whether the dependent component uses these services of the base component in an appropriate/secure way (please refer to any base component user/integrator guidance and further relevant sources for requirements).

As consistency of the composite product security policy has already been considered in the context of the ST in the assurance family ASE_COMP (Consistency of composite product Security Target) (9.9), it is not necessary to consider non-contradictoriness of the security policy model (ADV_SPM (Formal TSF model) (10.6)) of the composite product and the security policy model of its related base component.

10.8.4 Design compliance with the base component-related user guidance, ETR for composite evaluation and report of the base component evaluation authority (ADV_COMP.1)

Table 149 specifies the developer action elements for ADV_COMP.1, with Table 150 indicating their related content and presentation requirements, and Table 151 the corresponding evaluation actions.

Table 149 — Developer action elements for ADV_COMP.1

ADV_COMP.1.1D	The developer shall provide a design compliance justification.
---------------	---

Table 150 — Content and presentation elements for ADV_COMP.1

ADV_COMP.1.1C	The design compliance justification shall provide a rationale for design compliance - on an appropriate representation level - of how the requirements on the dependent component that are imposed by the related base component are fulfilled in the composite product.
---------------	---

Table 151 — Evaluator action elements for ADV_COMP.1

ADV_COMP.1.1E	The evaluator shall confirm that the rationale for design compliance is complete, coherent, and internally consistent.
---------------	---

11 Class AGD guidance documents

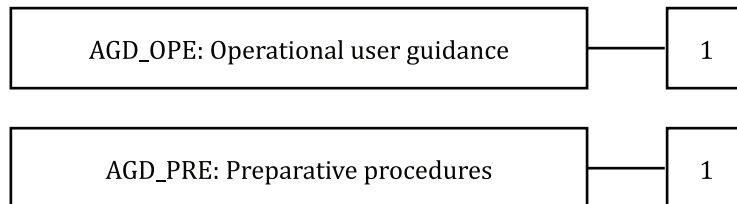
11.1 General

The guidance documents class provides the requirements for guidance documentation for all user roles. For the secure preparation and operation of the TOE, it is necessary to describe all relevant aspects for the secure handling of the TOE. The class also addresses the possibility of unintended incorrect configuration or handling of the TOE.

In many cases it can be appropriate to provide guidance in separate documents for preparation and operation of the TOE, or even separate for different user roles as, e.g. end-users, administrators, application programmers using software or hardware interfaces.

The guidance documents class is subdivided into two families which are concerned with the preparative user guidance (what must be done to transform the delivered TOE into its evaluated configuration in the operational environment as described in the ST) and with the operational user guidance (what must be done during the operation of the TOE in its evaluated configuration).

[Figure 10](#) describes the family hierarchy of class AGD.



Key

1 component

Figure 10 — AGD: Guidance documents, class decomposition

Direct, indirect and optional dependencies for the components of class AGD are shown in [Table 152](#).

Table 152 — Dependency table for class AGD: Guidance documents

	ADV_FSP.1 (10.3.4)
AGD_OPE.1 (11.2.4)	X
AGD_PRE.1 (11.3.4)	

11.2 Operational user guidance (AGD_OPE)

11.2.1 Objectives

Operational user guidance refers to written material that is intended to be used by all types of users of the TOE in its evaluated configuration, including:

- end-users,
- persons responsible for maintaining and administering the TOE in a correct manner for maximum security,
- others (e.g. programmers) using the TOE's external interfaces.

ISO/IEC 15408-3:2026(en)

Operational user guidance:

- describes the security functionality provided by the TSF,
- provides instructions and guidelines (including warnings),
- facilitates understanding of the TSF
- includes the security-critical information, and the security-critical actions required, for its secure use.

Misleading and unreasonable guidance should be absent from the guidance documentation, and secure procedures for all modes of operation shall be addressed. Insecure states should be easy to detect.

The operational user guidance provides a measure of confidence that non-malicious users, administrators, application providers and others exercising the external interfaces of the TOE will understand the secure operation of the TOE and will use it as intended. The evaluation of the user guidance includes investigating whether the TOE can be used in a manner that is insecure but that the user of the TOE would reasonably believe to be secure. The objective is to minimize the risk of human or other errors in operation that can deactivate, disable, or fail to activate security functionality, resulting in an undetected insecure state.

11.2.2 Component levelling

This family contains only one component.

11.2.3 Application notes

There may be different user roles or groups that are recognized by the TOE and that can interact with the TSF. These user roles and groups shall be taken into consideration by the operational user guidance. They may be roughly grouped into administrators and non-administrative users, or more specifically grouped into persons responsible for receiving, accepting, installing and maintaining the TOE, application programmers, revisors, auditors, daily-management, end-users. Each role can encompass an extensive set of capabilities or only one.

The requirement AGD_OPE.1.1C ([11.2.4](#)) encompasses the aspect that any warnings to the users during operation of a TOE with regard to the security problem definition and the security objectives for the operational environment described in the PP or ST are appropriately covered in the user guidance.

The concept of secure values, as employed in AGD_OPE.1.3C ([11.2.4](#)), has relevance where a user has control over security parameters. Guidance shall be provided on secure and insecure settings for such parameters.

AGD_OPE.1.4C ([11.2.4](#)) requires that the user guidance describes the appropriate reactions to all security-relevant events. Although many security-relevant events are the result of performing functions, this is not always the case (e.g. the audit log fills up, an intrusion is detected). Furthermore, a security-relevant event can happen as a result of a specific chain of functions or, conversely, several security-relevant events can be triggered by one function.

AGD_OPE.1.7C ([11.2.4](#)) requires that the user guidance is clear and reasonable. Misleading or unreasonable guidance can result in a user of the TOE believing that the TOE is secure when it is not.

An example of misleading guidance would be the description of a single guidance instruction that can be parsed in more than one way, one of which can result in an insecure state.

An example of unreasonable guidance would be a recommendation to follow a procedure that is so complicated that it cannot reasonably be expected that users will follow this guidance.

11.2.4 Operational user guidance (AGD_OPE.1)

11.2.4.1 Component relationships

Dependencies: ADV_FSP.1 (Basic functional specification) ([10.3.4](#))

11.2.4.2 Component elements

[Table 153](#) specifies the developer action elements for AGD_OPE.1, with [Table 154](#) indicating their related content and presentation requirements, and [Table 155](#) the corresponding evaluation actions.

Table 153 — Developer action elements for AGD_OPE.1

AGD_OPE.1.1D	The developer shall provide operational user guidance.
--------------	--

Table 154 — Content and presentation elements for AGD_OPE.1

AGD_OPE.1.1C	The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.
AGD_OPE.1.2C	The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.
AGD_OPE.1.3C	The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.
AGD_OPE.1.4C	The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that must be performed, including changing the security characteristics of entities under the control of the TSF.
AGD_OPE.1.5C	The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.
AGD_OPE.1.6C	The operational user guidance shall, for each user role, describe the security controls to be followed in order to fulfil the security objectives for the operational environment as described in the ST.
AGD_OPE.1.7C	The operational user guidance shall be clear and reasonable.

Table 155 — Evaluator action elements for AGD_OPE.1

AGD_OPE.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

11.3 Preparative procedures (AGD_PRE)

11.3.1 Objectives

Preparative procedures are useful for ensuring that the TOE has been received and installed in a secure manner as intended by the developer. The requirements for preparation call for a secure transition from the delivered TOE to its initial operational environment. This includes investigating whether the TOE can be configured or installed in a manner that is insecure but that the user of the TOE would reasonably believe to be secure.

11.3.2 Component levelling

This family contains only one component.

11.3.3 Application notes

It is recognized that the application of these requirements will vary depending on aspects, e.g. whether the TOE is delivered in an operational state, or whether it must be installed at the TOE owner's site.

The first process covered by the preparative procedures is the consumer's secure acceptance of the received TOE in accordance with the developer's delivery procedures. If the developer has not defined delivery procedures, security of the acceptance shall be ensured otherwise.

ISO/IEC 15408-3:2026(en)

Installation of the TOE includes transforming its operational environment into a state that conforms to the security objectives for the operational environment provided in the ST.

It can also be the case that no installation is necessary, for example a smart card. In this case it can be inappropriate to require and analyse installation procedures.

The requirements in this assurance family are presented separately from those in the AGD_OPE (Operational user guidance) (11.2) family, due to the infrequent, possibly one-time use of the preparative procedures.

11.3.4 Preparative procedures (AGD_PRE.1)

Table 156 specifies the developer action elements for AGD_PRE.1, with Table 157 indicating their related content and presentation requirements, and Table 158 the corresponding evaluation actions.

Table 156 — Developer action elements for AGD_PRE.1

AGD_PRE.1.1D	The developer shall provide the TOE including its preparative procedures.
--------------	--

Table 157 — Content and presentation elements for AGD_PRE.1

AGD_PRE.1.1C	The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.
AGD_PRE.1.2C	The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Table 158 — Evaluator action elements for AGD_PRE.1

AGD_PRE.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
AGD_PRE.1.2E	The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

12 Class ALC life cycle support

12.1 General

life cycle support is an aspect of establishing appropriate security controls in the development, production, delivery and maintenance of the TOE. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development, production, delivery and maintenance activities.

During the life cycle of the TOE it is distinguished whether the TOE is under the responsibility of the TOE developer or the user rather than whether it is located in the development or the user environment. The point of transition is when the TOE is accepted by the user. User in this context relates to the end-user as well as product and system integrators.

ALC (life cycle support) consists of nine families:

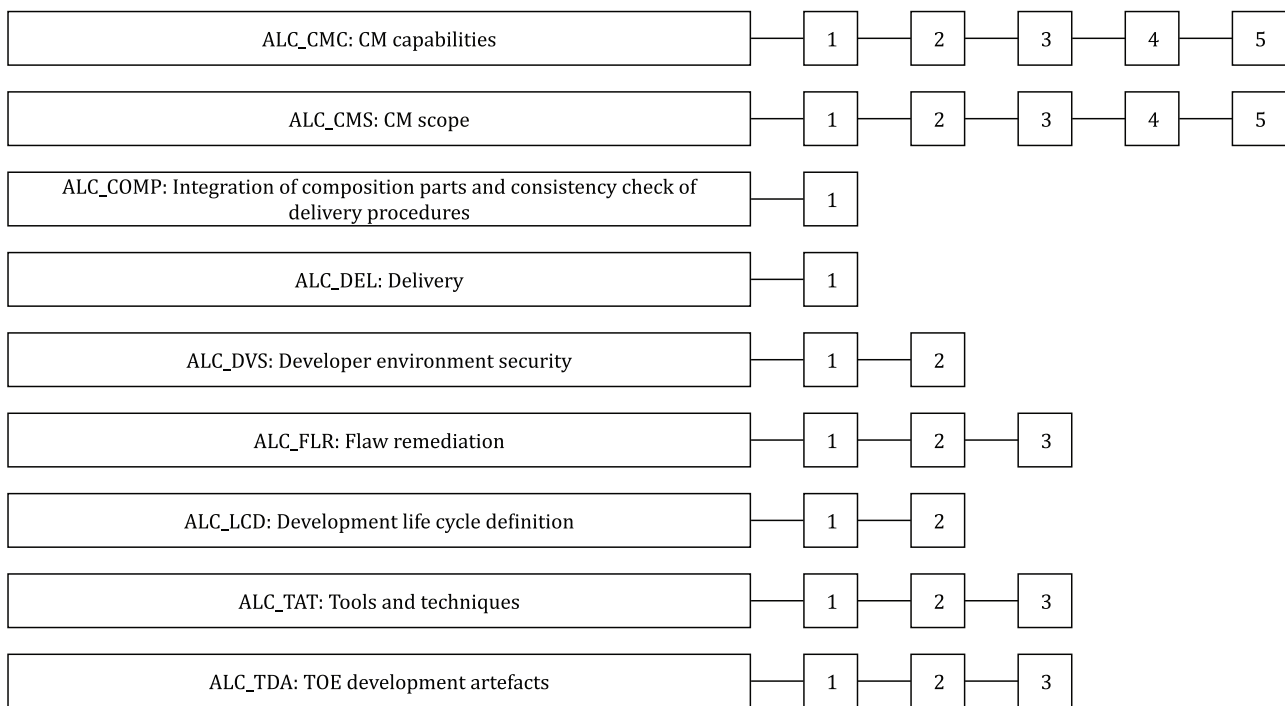
- ALC_LCD (Development life cycle definition) (12.7) provides requirements for the developer's description of the life cycle model used in the development, production, delivery and maintenance life cycle of the TOE;
- ALC_CMC (CM capabilities) (12.2) provides requirements for the management of the configuration items;
- ALC_CMS (CM scope) (12.3) requires a minimum set of configuration items to be managed in the defined way;

ISO/IEC 15408-3:2026(en)

- ALC_DVS (Developer environment security) (12.5) is concerned with the developer's physical, logical, procedural, personnel, and other security controls;
- ALC_TAT (Tools and techniques) (12.9) provides requirements for the development tools and implementation standards used by the developer;
- ALC_FLR (Flaw remediation) (12.6) provides requirements for the handling of security flaws;
- ALC_DEL (Delivery) (12.4) provides requirements for the procedures used for the delivery of the TOE to the recipient (A user of the TOE and related information, whether an end user or system integrator). Processes occurring during the development of the TOE are denoted as transfers, and are handled in the context of integration and acceptance procedures in other families of this class;
- ALC_TDA (TOE development artefacts) (12.8) is concerned with the generation of certain artefacts during the development process;
- ALC_COMP (Integration of composition parts and consistency check of delivery procedures) (12.10) is concerned with the integration of composition parts and a consistency check of delivery procedures.

Throughout this class, development and related terms (developer, develop) are meant in the more general sense to comprise development and production, whereas production specifically means the process of transforming the implementation representation into the final TOE.

Figure 11 describes the family hierarchy of class ALC.



Key

1,2,3,4,5 components

Figure 11 — ALC: life cycle support, class decomposition

Direct, indirect and optional dependencies for the components of class ALC are shown in [Table 159](#).

ISO/IEC 15408-3:2026(en)

Table 159 — Dependency table for class ALC: life cycle support

	ADV_IMP1 (10.4.4)	ALC_CMS.1 (12.3.4)	ALC_CMS.3 (12.3.6)	ALC_DVS.1 (12.5.4)	ALC_DVS.2 (12.5.5)	ALC_LCD.1 (12.7.4)	ALC_TAT.1 (12.9.4)
ALC_CMC.1 (12.2.4)		X					
ALC_CMC.2 (12.2.5)		X					
ALC_CMC.3 (12.2.6)		X		X		X	
ALC_CMC.4 (12.2.7)		X		X		X	
ALC_CMC.5 (12.2.8)		X			X	X	
ALC_CMS.1 (12.3.4)							
ALC_CMS.2 (12.3.5)							
ALC_CMS.3 (12.3.6)							
ALC_CMS.4 (12.3.7)							
ALC_CMS.5 (12.3.8)							
ALC_COMP.1 (12.10.4)							
ALC_DEL.1 (12.4.4)							
ALC_DVS.1 (12.5.4)							
ALC_DVS.2 (12.5.5)							
ALC_FLR.1 (12.6.4)							
ALC_FLR.2 (12.6.5)							
ALC_FLR.3 (12.6.6)							
ALC_LCD.1 (12.7.4)							
ALC_LCD.2 (12.7.5)							
ALC_TAT.1 (12.9.4)	X						
ALC_TAT.2 (12.9.5)	X						
ALC_TAT.3 (12.9.6)	X						
ALC_TDA.1 (12.8.4)							
ALC_TDA.2 (12.8.5)			X				
ALC_TDA.3 (12.8.6)	X		X				X

12.2 CM capabilities (ALC_CMC)

12.2.1 Objectives

Configuration management (CM) techniques, properly defined as part of the development life cycle model, contribute to the assurance argument that the TOE meets the SFRs. A CM system that is managed and operated correctly will help ensure the integrity of the portions of the TOE that are controlled, by providing a method of tracking any changes to the TOE, and to help ensure that all changes to the TOE are authorized.

The objective of this family is to require the TOE developer's CM system to have certain capabilities. These capabilities are intended to reduce the likelihood that accidental or unauthorized modifications of the configuration items will occur. The CM system shall support maintaining the integrity of the TOE throughout the part of the TOE's life cycle that is under the control of the developer.

The objective of introducing automated CM tools is to increase the effectiveness of the CM system. While both automated and manual CM systems can be bypassed, ignored, or proven insufficient to prevent unauthorized modification, automated systems are less susceptible to human error or negligence.

ISO/IEC 15408-3:2026(en)

The objectives of this family include the following:

- ensuring that the TOE is identifiable and complete before it is sent to the recipient;
- ensuring that no configuration items are missed during evaluation;
- preventing unauthorized modification, addition, or deletion of TOE configuration items.

12.2.2 Component levelling

The components in this family are levelled on the basis of the CM system capabilities, the scope of the CM documentation and the evidence provided by the developer.

12.2.3 Application notes

In the case where the TOE is a subset of a product, the requirements of this family apply only to the TOE configuration items, not to the entire product.

For developer organizations that specify more than one CM application or include different instances of a CM application within the scope of the TOEs design, development, production and maintenance, it is required to document all of them. For evaluation purposes, the set of CM applications should be regarded as parts of an overall CM system, applicable to the TOE, which is addressed in the criteria.

The overall CM system should address any aspects of integration between component CM applications.

Several elements of this family refer to configuration items. These elements identify CM requirements to be imposed on all items identified in the configuration list, but leave the contents of the list to the discretion of the developer. ALC_CMS (CM scope) (12.3) can be used to narrow this discretion by identifying specific items that shall be included in the configuration list, and hence within the scope of the overall CM system.

ALC_CMC.2.3C (12.2.5) introduces a requirement that the CM system uniquely identify all configuration items. This also requires that modifications to configuration items result in a new, unique identifier being assigned to the configuration item.

ALC_CMC.3.8C (12.2.6) introduces the requirement that the evidence shall demonstrate that the CM system operates in accordance with the CM plan. Examples of such evidence can be documentation such as screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM system by the developer. The evaluator is responsible for determining that this evidence is sufficient to show that the CM system operates in accordance with the CM plan.

ALC_CMC.4.5C (12.2.7) introduces a requirement that the CM system provide an automated means to support the production of the TOE. This requires that the CM system provide an automated means to assist in determining that the correct configuration items are used in generating the TOE.

ALC_CMC.5.10C (12.2.8) introduces a requirement that the CM system provide an automated means to ascertain the changes between the TOE and its preceding version. If no previous version of the TOE exists, the developer shall provide an automated means to ascertain the changes between the TOE and a future version of the TOE.

12.2.4 Labelling of the TOE (ALC_CMC.1)

12.2.4.1 Component relationships

Dependencies: ALC_CMS.1 (TOE CM coverage) (12.3.4)

12.2.4.2 Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

12.2.4.3 Component elements

[Table 160](#) specifies the developer action elements for ALC_CMC.1, with [Table 161](#) indicating their related content and presentation requirements, and [Table 162](#) the corresponding evaluation actions.

Table 160 — Developer action elements for ALC_CMC.1

ALC_CMC.1.1D	The developer shall provide the TOE and a unique reference for the TOE.
--------------	--

Table 161 — Content and presentation elements for ALC_CMC.1

ALC_CMC.1.1C	The TOE shall be labelled with its unique reference.
--------------	---

Table 162 — Evaluator action elements for ALC_CMC.1

ALC_CMC.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

12.2.5 Use of the CM system (ALC_CMC.2)

12.2.5.1 Component relationships

Dependencies: ALC_CMS.1 (TOE CM coverage) ([12.3.4](#))

12.2.5.2 Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

The use of a CM system increases assurance that the configuration items are maintained in a controlled manner.

12.2.5.3 Component elements

[Table 163](#) specifies the developer action elements for ALC_CMC.2, with [Table 164](#) indicating their related content and presentation requirements, and [Table 165](#) the corresponding evaluation actions.

Table 163 — Developer action elements for ALC_CMC.2

ALC_CMC.2.1D	The developer shall provide the TOE and a unique reference for the TOE.
ALC_CMC.2.2D	The developer shall provide the CM documentation.
ALC_CMC.2.3D	The developer shall use a CM system.

Table 164 — Content and presentation elements for ALC_CMC.2

ALC_CMC.2.1C	The TOE shall be labelled with its unique reference.
ALC_CMC.2.2C	The CM documentation shall describe the method used to uniquely identify the configuration items.
ALC_CMC.2.3C	The CM system shall uniquely identify all configuration items.

ISO/IEC 15408-3:2026(en)

Table 165 — Evaluator action elements for ALC_CMC.2

ALC_CMC.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.2.6 Authorization controls (ALC_CMC.3)

12.2.6.1 Component relationships

Dependencies: ALC_CMS.1 (TOE CM coverage) ([12.3.4](#))
 ALC_DVS.1 (Identification of security controls) ([12.5.4](#))
 ALC_LCD.1 (Developer defined life cycle processes) ([12.7.4](#))

12.2.6.2 Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

The use of a CM system increases assurance that the configuration items are maintained in a controlled manner.

Providing controls to ensure that unauthorized modifications are not made to the TOE ("CM access control") and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

12.2.6.3 Component elements

[Table 166](#) specifies the developer action elements for ALC_CMC.3, with [Table 167](#) indicating their related content and presentation requirements, and [Table 168](#) the corresponding evaluation actions.

Table 166 — Developer action elements for ALC_CMC.3

ALC_CMC.3.1D	The developer shall provide the TOE and a unique reference for the TOE.
ALC_CMC.3.2D	The developer shall provide the CM documentation.
ALC_CMC.3.3D	The developer shall use a CM system.

Table 167 — Content and presentation elements for ALC_CMC.3

ALC_CMC.3.1C	The TOE shall be labelled with its unique reference.
ALC_CMC.3.2C	The CM documentation shall describe the method used to uniquely identify the configuration items.
ALC_CMC.3.3C	The CM system shall uniquely identify all configuration items.
ALC_CMC.3.4C	The CM system shall provide controls such that only authorized changes are made to the configuration items.
ALC_CMC.3.5C	The CM documentation shall include a CM plan.
ALC_CMC.3.6C	The CM plan shall describe how the CM system is used for the development of the TOE.
ALC_CMC.3.7C	The evidence shall demonstrate that all configuration items are being maintained under the CM system.
ALC_CMC.3.8C	The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

ISO/IEC 15408-3:2026(en)

Table 168 — Evaluator action elements for ALC_CMC.3

ALC_CMC.3.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.2.7 Production support, acceptance procedures and automation (ALC_CMC.4)

12.2.7.1 Component relationships

Dependencies: ALC_CMS.1 (TOE CM coverage) ([12.3.4](#))
 ALC_DVS.1 (Identification of security controls) ([12.5.4](#))
 ALC_LCD.1 (Developer defined life cycle processes) ([12.7.4](#))

12.2.7.2 Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

The use of a CM system increases assurance that the configuration items are maintained in a controlled manner.

Providing controls to ensure that unauthorized modifications are not made to the TOE ("CM access control") and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

The purpose of the acceptance procedures is to ensure that the parts of the TOE are of adequate quality and to confirm that any creation or modification of configuration items is authorized. Acceptance procedures are an essential element in integration processes and in the life cycle management of the TOE.

In development environments where the configuration items or their quantity or organization are complex, it is difficult to control changes without the support of automated tools. In particular, these automated tools shall be able to support the numerous changes that occur during development and ensure that those changes are authorized. It is an objective of this component to ensure that the configuration items are controlled through automated means. If the TOE is developed by multiple developers, i.e. integration is required, the use of automatic tools is adequate. Furthermore, in the case where the overall CM system includes more than one CM application then automated tools can also support integration between the CM applications and of the TOE.

Production support procedures help to ensure that the generation of the TOE from a managed set of configuration items is correctly performed in an authorized manner, particularly in the case when different developers are involved and integration processes are carried out.

12.2.7.3 Component elements

[Table 169](#) specifies the developer action elements for ALC_CMC.4, with [Table 170](#) indicating their related content and presentation requirements, and [Table 171](#) the corresponding evaluation actions.

Table 169 — Developer action elements for ALC_CMC.4

ALC_CMC.4.1D	The developer shall provide the TOE and a unique reference for the TOE.
ALC_CMC.4.2D	The developer shall provide the CM documentation.
ALC_CMC.4.3D	The developer shall use a CM system.

ISO/IEC 15408-3:2026(en)

Table 170 — Content and presentation elements for ALC_CMC.4

ALC_CMC.4.1C	The TOE shall be labelled with its unique reference.
ALC_CMC.4.2C	The CM documentation shall describe the method or methods used to uniquely identify the configuration items.
ALC_CMC.4.3C	The CM system shall uniquely identify all configuration items.
ALC_CMC.4.4C	The CM system shall provide automated controls such that only authorized changes are made to the configuration items.
ALC_CMC.4.5C	The CM system shall support the production of the TOE by automated means.
ALC_CMC.4.6C	The CM documentation shall include a CM plan.
ALC_CMC.4.7C	The CM plan shall describe how the CM system is used for the development of the TOE.
ALC_CMC.4.8C	The CM plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.
ALC_CMC.4.9C	The evidence shall demonstrate that all configuration items are being maintained under the CM system.
ALC_CMC.4.10C	The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

Table 171 — Evaluator action elements for ALC_CMC.4

ALC_CMC.4.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.2.8 Advanced support (ALC_CMC.5)

12.2.8.1 Component relationships

Dependencies: ALC_CMS.1 (TOE CM coverage) ([12.3.4](#))
 ALC_DVS.2 (Sufficiency of security controls) ([12.5.5](#))
 ALC_LCD.1 (Developer defined life cycle processes) ([12.7.4](#))

12.2.8.2 Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

The use of a CM system increases assurance that the configuration items are maintained in a controlled manner.

Providing controls to ensure that unauthorized modifications are not made to the TOE ("CM access control") and ensuring proper functionality and use of the CM system, helps to maintain the integrity of the TOE.

The purpose of the acceptance procedures is to ensure that the parts of the TOE are of adequate quality and to confirm that any creation or modification of configuration items is authorized. Acceptance procedures are an essential element in integration processes and in the life cycle management of the TOE.

In development environments where the configuration items are complex, it is difficult to control changes without the support of automated tools. In particular, these automated tools shall be able to support the numerous changes that occur during development and ensure that those changes are authorized. It is an objective of this component to ensure that the configuration items are controlled through automated means. If the TOE is developed by multiple developers, i.e. integration is required, the use of automatic tools is adequate.

ISO/IEC 15408-3:2026(en)

Production support procedures help to ensure that the generation of the TOE from a managed set of configuration items is correctly performed in an authorized manner, particularly in the case when different developers are involved and integration processes are carried out.

The CM system shall be able to identify the version of the implementation representation from which the TOE is generated. This helps to ensure that the integrity of this material is preserved by the appropriate technical, physical and procedural safeguards.

Automated means of ascertaining changes between versions of the TOE and identifying which configuration items are affected by modifications to other configuration items assist in determining the impact of the changes between successive versions of the TOE. This in turn can provide valuable information in determining whether changes to the TOE result in all configuration items being consistent with one another.

12.2.8.3 Component elements

[Table 172](#) specifies the developer action elements for ALC_CMC.5, with [Table 173](#) indicating their related content and presentation requirements, and [Table 174](#) the corresponding evaluation actions.

Table 172 — Developer action elements for ALC_CMC.5

ALC_CMC.5.1D	The developer shall provide the TOE and a unique reference for the TOE.
ALC_CMC.5.2D	The developer shall provide the CM documentation.
ALC_CMC.5.3D	The developer shall use a CM system.

Table 173 — Content and presentation elements for ALC_CMC.5

ALC_CMC.5.1C	The TOE shall be labelled with its unique reference.
ALC_CMC.5.2C	The CM documentation shall describe the method used to uniquely identify the configuration items.
ALC_CMC.5.3C	The CM documentation shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to all configuration items.
ALC_CMC.5.4C	The CM system shall uniquely identify all configuration items.
ALC_CMC.5.5C	The CM system shall provide automated controls such that only authorized changes are made to the configuration items.
ALC_CMC.5.6C	The CM system shall support the production of the TOE by automated means.
ALC_CMC.5.7C	The CM system shall ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.
ALC_CMC.5.8C	The CM system shall identify the configuration items that comprise the TSF.
ALC_CMC.5.9C	The CM system shall support the audit of all changes to the TOE by automated means, including the originator, date, and time in the audit trail.
ALC_CMC.5.10C	The CM system shall provide an automated means to identify all other configuration items that are affected by the change of a given configuration item.
ALC_CMC.5.11C	The CM system shall be able to identify the version of the implementation representation from which the TOE is generated.
ALC_CMC.5.12C	The CM documentation shall include a CM plan.
ALC_CMC.5.13C	The CM plan shall describe how the CM system is used for the development of the TOE.
ALC_CMC.5.14C	The CM plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.
ALC_CMC.5.15C	The evidence shall demonstrate that all configuration items are being maintained under the CM system.
ALC_CMC.5.16C	The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

ISO/IEC 15408-3:2026(en)

Table 174 — Evaluator action elements for ALC_CMC.5

ALC_CMC.5.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_CMC.5.2E	The evaluator <i>shall determine</i> that the application of the production support procedures results in a TOE as provided by the developer for testing activities.

12.3 CM scope (ALC_CMS)

12.3.1 Objectives

The objective of this family is to identify items to be included as configuration items and hence placed under the CM requirements of ALC_CMC (CM capabilities) (12.2). Applying configuration management to these additional items provides additional assurance that the integrity of TOE is maintained.

12.3.2 Component levelling

The components in this family are levelled on the basis of which of the following are required to be included as configuration items: the TOE and the evaluation evidence required by the SARs; the parts of the TOE; the implementation representation; security flaws; and development tools and related information.

12.3.3 Application notes

While ALC_CMS (CM scope) mandates a list of configuration items and that each item on this list be under CM, ALC_CMC (CM capabilities) (12.2) leaves the contents of the configuration list to the discretion of the developer. ALC_CMS (CM scope) narrows this discretion by identifying items that shall be included in the configuration list, and hence come under the CM requirements of ALC_CMC (CM capabilities) (12.2).

12.3.4 TOE CM coverage (ALC_CMS.1)

12.3.4.1 Objectives

A CM system can control changes only to those items that have been placed under CM (i.e. the configuration items identified in the configuration list). By placing the TOE itself and the evaluation evidence required by the other SARs in the ST under CM ensures that they have been modified in a controlled manner with proper authorizations.

12.3.4.2 Application notes

ALC_CMS.1.1C introduces the requirement that the TOE itself and the evaluation evidence required by the other SARs in the ST be included in the configuration list and hence be subject to the CM requirements of ALC_CMC (CM capabilities) (12.2).

12.3.4.3 Component elements

[Table 175](#) specifies the developer action elements for ALC_CMS.1, with [Table 176](#) indicating their related content and presentation requirements, and [Table 177](#) the corresponding evaluation actions.

Table 175 — Developer action elements for ALC_CMS.1

ALC_CMS.1.1D	The developer shall provide a configuration list for the TOE.
--------------	--

ISO/IEC 15408-3:2026(en)

Table 176 — Content and presentation elements for ALC_CMS.1

ALC_CMS.1.1C	The configuration list shall include: — the TOE itself; — the evaluation evidence required by the SARs.
ALC_CMS.1.2C	The configuration list shall uniquely identify the configuration items.

Table 177 — Evaluator action elements for ALC_CMS.1

ALC_CMS.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

12.3.5 Parts of the TOE CM coverage (ALC_CMS.2)

12.3.5.1 Objectives

A CM system can control changes only to those items that have been placed under CM (i.e. the configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise the TOE, and the evaluation evidence required by the other SARs under CM provides assurance that they have been modified in a controlled manner with proper authorizations.

12.3.5.2 Application notes

ALC_CMS.2.1C introduces the requirement that the parts that comprise the TOE (all parts that are delivered to the consumer, for example hardware parts or executable files) be included in the configuration list and hence be subject to the CM requirements of ALC_CMC (CM capabilities) ([12.2](#)).

ALC_CMS.2.3C introduces the requirement that the configuration list indicate the developer of each TSF relevant configuration item.

12.3.5.3 Component elements

[Table 178](#) specifies the developer action elements for ALC_CMS.2, with [Table 179](#) indicating their related content and presentation requirements, and [Table 180](#) the corresponding evaluation actions.

Table 178 — Developer action elements for ALC_CMS.2

ALC_CMS.2.1D	The developer shall provide a configuration list for the TOE.
--------------	---

Table 179 — Content and presentation elements for ALC_CMS.2

ALC_CMS.2.1C	The configuration list shall include the following : — the TOE itself; — the evaluation evidence required by the SARs; — the parts that comprise the TOE.
ALC_CMS.2.2C	The configuration list shall uniquely identify the configuration items.
ALC_CMS.2.3C	For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

Table 180 — Evaluator action elements for ALC_CMS.2

ALC_CMS.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

ISO/IEC 15408-3:2026(en)

12.3.6 Implementation representation CM coverage (ALC_CMS.3)

12.3.6.1 Objectives

A CM system can control changes only to those items that have been placed under CM (i.e. the configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise the TOE, the TOE implementation representation and the evaluation evidence required by the other SARs under CM provides assurance that they have been modified in a controlled manner with proper authorizations.

12.3.6.2 Application notes

ALC_CMS.3.1C introduces the requirement that the TOE implementation representation be included in the list of configuration items and hence be subject to the CM requirements of ALC_CMC (CM capabilities) (12.2).

12.3.6.3 Component elements

Table 181 specifies the developer action elements for ALC_CMS.3, with Table 182 indicating their related content and presentation requirements, and Table 183 the corresponding evaluation actions.

Table 181 — Developer action elements for ALC_CMS.3

ALC_CMS.3.1D	The developer shall provide a configuration list for the TOE.
--------------	---

Table 182 — Content and presentation elements for ALC_CMS.3

ALC_CMS.3.1C	The configuration list shall include the following: <ul style="list-style-type: none"> — the TOE itself; — the evaluation evidence required by the SARs; — the parts that comprise the TOE; — the implementation representation.
ALC_CMS.3.2C	The configuration list shall uniquely identify the configuration items.
ALC_CMS.3.3C	For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

Table 183 — Evaluator action elements for ALC_CMS.3

ALC_CMS.3.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.3.7 Problem tracking CM coverage (ALC_CMS.4)

12.3.7.1 Objectives

A CM system can control changes only to those items that have been placed under CM (i.e. the configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise the TOE, the TOE implementation representation and the evaluation evidence required by the other SARs under CM provides assurance that they have been modified in a controlled manner with proper authorizations.

Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that access to them is managed, further, it can support developers in tracking security flaws to their resolution.

12.3.7.2 Application notes

ALC_CMS.4.1C introduces the requirement that reports of identified security flaws be included in the configuration list and hence be subject to the CM requirements of ALC_CMC (CM capabilities) (12.2). This

ISO/IEC 15408-3:2026(en)

requires that information regarding previously identified security flaw reports and their resolution be maintained.

12.3.7.3 Component elements

[Table 184](#) specifies the developer action elements for ALC_CMS.4, with [Table 185](#) indicating their related content and presentation requirements, and [Table 186](#) the corresponding evaluation actions.

Table 184 — Developer action elements for ALC_CMS.4

ALC_CMS.4.1D	The developer shall provide a configuration list for the TOE.
--------------	---

Table 185 — Content and presentation elements for ALC_CMS.4

ALC_CMS.4.1C	The configuration list shall include the following: <ul style="list-style-type: none"> — the TOE itself; — the evaluation evidence required by the SARs; — the parts that comprise the TOE; — the implementation representation; — security flaw reports and resolution status.
ALC_CMS.4.2C	The configuration list shall uniquely identify the configuration items.
ALC_CMS.4.3C	For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

Table 186 — Evaluator action elements for ALC_CMS.4

ALC_CMS.4.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.3.8 Development tools CM coverage (ALC_CMS.5)

12.3.8.1 Objectives

A CM system can control changes only to those items that have been placed under CM (i.e. the configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise the TOE, the TOE implementation representation and the evaluation evidence required by the other SARs under CM provides assurance that they have been modified in a controlled manner with proper authorizations.

Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that access to them is managed, further, it can support developers in tracking security flaws to their resolution.

Development tools play an important role in ensuring the production of a quality version of the TOE. Therefore, it is important to control modifications to these tools.

12.3.8.2 Application notes

ALC_CMS.5.1C introduces the requirement that development tools and other related information be included in the list of configuration items and hence be subject to the CM requirements of ALC_CMC (CM capabilities) ([12.2](#)). Examples of development tools are programming languages and compilers. Information pertaining to TOE generation items (such as compiler options, generation options, and build options) is an example of information relating to development tools.

12.3.8.3 Component elements

[Table 187](#) specifies the developer action elements for ALC_CMS.5, with [Table 188](#) indicating their related content and presentation requirements, and [Table 189](#) the corresponding evaluation actions.

Table 187 — Developer action elements for ALC_CMS.5

ALC_CMS.5.1D	The developer shall provide a configuration list for the TOE.
--------------	---

Table 188 — Content and presentation elements for ALC_CMS.5

ALC_CMS.5.1C	The configuration list shall include the following: <ul style="list-style-type: none"> — the TOE itself; — the evaluation evidence required by the SARs; — the parts that comprise the TOE; — the implementation representation; — security flaw reports and resolution status; — development tools and related information.
ALC_CMS.5.2C	The configuration list shall uniquely identify the configuration items.
ALC_CMS.5.3C	For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

Table 189 — Evaluator action elements for ALC_CMS.5

ALC_CMS.5.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.4 Delivery (ALC_DEL)

12.4.1 Objectives

The concern of this family is the secure transfer of the finished TOE from the development environment into the responsibility of the user.

The requirements for delivery call for system control and distribution facilities and procedures that detail the controls necessary to provide assurance that the security of the TOE is maintained during distribution of the TOE to the user. For a valid distribution of the TOE, the procedures used for the distribution of the TOE address the implied or identified objectives identified in the PP or ST relating to the security of the TOE during delivery.

12.4.2 Component levelling

This family contains only one component. An increasing level of protection for the TOE is established by requiring that the delivery procedures are commensurate with the assumed attack potential in the family AVA_VAN (Vulnerability analysis) ([14.3](#)) specified in the ST.

12.4.3 Application notes

Transfers from subcontractors to the developer or between different development sites are not considered here, but in the family ALC_DVS (Developer environment security) ([12.5](#)).

The end of the delivery phase is marked by the acceptance of the transfer of the TOE into the responsibility of the recipient.

NOTE This does not necessarily coincide with the arrival of the TOE at the recipient's location.

ISO/IEC 15408-3:2026(en)

The delivery procedures should consider, if applicable, issues such as:

- ensuring that the TOE received by the recipient corresponds precisely to the evaluated version of the TOE;
- avoiding or detecting any tampering with the actual version of the TOE;
- preventing submission of a counterfeit version of the TOE;
- avoiding unwanted knowledge of distribution of the TOE to the recipient: there can be cases where knowledge of when and how it is delivered must be kept from any potential attackers;
- avoiding or detecting the TOE being intercepted during delivery; and
- avoiding the TOE being delayed or stopped during distribution.

The delivery procedures should include the recipient's actions implied by these issues. The consistent description of these implied actions is examined in the AGD_PRE (Preparative procedures) (11.3) family, if present.

12.4.4 Delivery procedures (ALC_DEL.1)

Table 190 specifies the developer action elements for ALC_DEL.1, with Table 191 indicating their related content and presentation requirements, and Table 192 the corresponding evaluation actions.

Table 190 — Developer action elements for ALC_DEL.1

ALC_DEL.1.1D	The developer shall document and provide procedures for delivery of the TOE or parts of it to the recipient.
ALC_DEL.1.2D	The developer shall use the delivery procedures.

Table 191 — Content and presentation elements for ALC_DEL.1

ALC_DEL.1.1C	The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to the recipient.
--------------	---

Table 192 — Evaluator action elements for ALC_DEL.1

ALC_DEL.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

12.5 Developer environment security (ALC_DVS)

12.5.1 Objectives

Developer environment security is concerned with the determination and specification of security controls relating to the developer provided environment.

NOTE Such controls include coverage of security relevant aspects of asset management, human resources security, physical and environmental security, communications and operations management, access control, information systems acquisition, development and maintenance, information security incident management, and business continuity management.

12.5.2 Component levelling

The components in this family are levelled on the basis of whether justification of the sufficiency of the security controls is required.

ISO/IEC 15408-3:2026(en)

12.5.3 Application notes

This family deals with controls to remove or reduce threads and security risks existing at the developer's site.

The evaluator should visit the site(s) in order to assess evidence for developer environment security. This can include sites of subcontractors involved in the TOE development and production. Any decision not to visit shall be agreed with the evaluation authority.

Although developer environment security deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, the ALC_DVS (Developer environment security) requirements specify only that the developer environment security controls be in place at the time of evaluation. Furthermore, ALC_DVS (Developer environment security) does not contain any requirements related to the sponsor's intention to apply the developer environment security controls in the future, after completion of the evaluation.

It is recognized that confidentiality is not always an issue for the protection of the TOE in its development environment. The use of the word "necessary" allows for the selection of appropriate safeguards.

12.5.4 Identification of security controls (ALC_DVS.1)

[Table 193](#) specifies the developer action elements for ALC_DVS.1, with [Table 194](#) indicating their related content and presentation requirements, and [Table 195](#) the corresponding evaluation actions.

Table 193 — Developer action elements for ALC_DVS.1

ALC_DVS.1.1D	The developer shall produce and provide developer environment security documentation.
--------------	--

Table 194 — Content and presentation elements for ALC_DVS.1

ALC_DVS.1.1C	The developer environment security documentation shall describe all the physical, logical, procedural, personnel, and other security controls that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.
--------------	---

Table 195 — Evaluator action elements for ALC_DVS.1

ALC_DVS.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_DVS.1.2E	The evaluator <i>shall confirm</i> that the security controls are being applied.

12.5.5 Sufficiency of security controls (ALC_DVS.2)

[Table 196](#) specifies the developer action elements for ALC_DVS.2, with [Table 197](#) indicating their related content and presentation requirements, and [Table 198](#) the corresponding evaluation actions.

Table 196 — Developer action elements for ALC_DVS.2

ALC_DVS.2.1D	The developer shall produce and provide developer environment security documentation.
--------------	--

Table 197 — Content and presentation elements for ALC_DVS.2

ALC_DVS.2.1C	The developer environment security documentation shall describe all the physical, logical, procedural, personnel, and other security controls that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.
ALC_DVS.2.2C	The developer environment security documentation shall justify that the security controls provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

ISO/IEC 15408-3:2026(en)

Table 198 — Evaluator action elements for ALC_DVS.2

ALC_DVS.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ALC_DVS.2.2E	The evaluator shall confirm that the security controls are being applied.

12.6 Flaw remediation (ALC_FLR)

12.6.1 Objectives

Flaw remediation requires that discovered security flaws be tracked and corrected by the developer. Although future compliance with flaw remediation procedures cannot be determined at the time of the TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to track and correct flaws, and to distribute the flaw information and corrections.

12.6.2 Component levelling

The components in this family are levelled on the basis of the increasing extent in scope of the flaw remediation procedures and the rigour of the flaw remediation policies.

12.6.3 Application notes

This family provides assurance that the TOE will be maintained and supported in the future, requiring the TOE developer to track and correct flaws in the TOE. Additionally, requirements are included for the distribution of flaw corrections. However, this family does not impose evaluation requirements beyond the current evaluation.

The TOE user is considered to be the focal point in the user organization that is responsible for receiving and implementing fixes to security flaws. This is not necessarily an individual user, but can be an organizational representative who is responsible for the handling of security flaws. The use of the term TOE user recognizes that different organisations have different procedures for handling flaw reporting, which can be done either by an individual user, or by a central administrative body.

The flaw remediation procedures should describe the methods for dealing with all types of flaws encountered. These flaws can be reported by the developer, by users of the TOE, or by other parties with familiarity with the TOE. It is possible that some flaws are not reparable immediately. There can be occasions where a flaw cannot be fixed and other (e.g. procedural) controls shall be taken. The documentation provided should cover the procedures for providing the operational sites with fixes and providing information on flaws where fixes are delayed (and what to do in the interim) or when fixes are not possible.

Changes applied to a TOE after its release render it unevaluated; although some information from the original evaluation can still apply. The phrase "release of the TOE" used in this family therefore refers to a version of a product that is a release of a certified TOE, to which changes have been applied.

12.6.4 Basic flaw remediation (ALC_FLR.1)

[Table 199](#) specifies the developer action elements for ALC_FLR.1, with [Table 200](#) indicating their related content and presentation requirements, and [Table 201](#) the corresponding evaluation actions.

Table 199 — Developer action elements for ALC_FLR.1

ALC_FLR.1.1D	The developer shall document and provide flaw remediation procedures addressed to TOE developers.
--------------	--

ISO/IEC 15408-3:2026(en)

Table 200 — Content and presentation elements for ALC_FLR.1

ALC_FLR.1.1C	The documentation of the flaw remediation procedures shall describe the procedures used to track all reported security flaws in each release of the TOE.
ALC_FLR.1.2C	The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.
ALC_FLR.1.3C	The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.
ALC_FLR.1.4C	The documentation of the flaw remediation procedures shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

Table 201 — Evaluator action elements for ALC_FLR.1

ALC_FLR.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

12.6.5 Flaw reporting procedures (ALC_FLR.2)

12.6.5.1 Objectives

In order for the developer to be able to act appropriately upon security flaw reports from TOE users, and to know to whom to send corrective fixes, it is necessary for TOE users to understand how to submit security flaw reports to the developer. Flaw remediation guidance from the developer to the TOE user ensures that TOE users are aware of this important information.

12.6.5.2 Component elements

[Table 202](#) specifies the developer action elements for ALC_FLR.2, with [Table 203](#) indicating their related content and presentation requirements, and [Table 204](#) the corresponding evaluation actions.

Table 202 — Developer action elements for ALC_FLR.2

ALC_FLR.2.1D	The developer shall document and provide flaw remediation procedures addressed to TOE developers.
ALC_FLR.2.2D	The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.
ALC_FLR.2.3D	The developer shall provide flaw remediation guidance addressed to TOE users.

Table 203 — Content and presentation elements for ALC_FLR.2

ALC_FLR.2.1C	The documentation of the flaw remediation procedures shall describe the procedures used to track all reported security flaws in each release of the TOE.
ALC_FLR.2.2C	The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.
ALC_FLR.2.3C	The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.
ALC_FLR.2.4C	The documentation of the flaw remediation procedures shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.
ALC_FLR.2.5C	The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ISO/IEC 15408-3:2026(en)

Table 203 (continued)

ALC_FLR.2.6C	The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.
ALC_FLR.2.7C	The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.
ALC_FLR.2.8C	The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

Table 204 — Evaluator action elements for ALC_FLR.2

ALC_FLR.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.6.6 Systematic flaw remediation (ALC_FLR.3)

12.6.6.1 Objectives

In order for the developer to be able to act appropriately upon security flaw reports from TOE users, and to know to whom to send corrective fixes, it is necessary for TOE users to understand how to submit security flaw reports to the developer, and how to register themselves with the developer so that they can receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures that TOE users are aware of this important information.

12.6.6.2 Component elements

[Table 205](#) specifies the developer action elements for ALC_FLR.3, with [Table 206](#) indicating their related content and presentation requirements, and [Table 207](#) the corresponding evaluation actions.

Table 205 — Developer action elements for ALC_FLR.3

ALC_FLR.3.1D	The developer shall document and provide flaw remediation procedures addressed to TOE developers.
ALC_FLR.3.2D	The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.
ALC_FLR.3.3D	The developer shall provide flaw remediation guidance addressed to TOE users.

Table 206 — Content and presentation elements for ALC_FLR.3

ALC_FLR.3.1C	The documentation of the flaw remediation procedures shall describe the procedures used to track all reported security flaws in each release of the TOE.
ALC_FLR.3.2C	The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.
ALC_FLR.3.3C	The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.
ALC_FLR.3.4C	The documentation of the flaw remediation procedures shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.
ALC_FLR.3.5C	The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.
ALC_FLR.3.6C	The flaw remediation procedures shall include a procedure requiring timely response and the automatic distribution of security flaw reports and the associated corrections to registered users who can be affected by the security flaw.
ALC_FLR.3.7C	The procedures for processing reported security flaws shall ensure that any reported flaws are remediated and the remediation procedures issued to TOE users.
ALC_FLR.3.8C	The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ISO/IEC 15408-3:2026(en)

Table 206 (continued)

ALC_FLR.3.9C	The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.
ALC_FLR.3.10C	The flaw remediation guidance shall describe a means by which TOE users can register with the developer, to be eligible to receive security flaw reports and corrections.
ALC_FLR.3.11C	The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.

Table 207 — Evaluator action elements for ALC_FLR.3

ALC_FLR.3.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

12.7 Development life cycle definition (ALC_LCD)

12.7.1 Objectives

Poorly defined or uncontrolled processes applied during the development, production and maintenance of the TOE can result in a TOE that does not meet all of its security objectives. Therefore, it is important that well defined and controlled processes be established as early as possible in the TOE's life cycle.

Defining and implementing such processes does not guarantee that the TOE meets all of its SFRs. It is possible that the processes will be insufficient or inadequate.

By adopting a life cycle model, or models that meet the needs of the developer's organization, there is an improved likelihood that the development, production and maintenance processes applied to TOE support the correct design and implementation of a TOE that meets the specified SFRs.

The determination of appropriate process controls in order to support process improvement is a long-established best practice.

12.7.2 Component levelling

The components in this family are levelled on the basis of increasing requirements for measurability of the life cycle model, and for compliance with that model.

12.7.3 Application notes

A life cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE. Aspects of the process that can be covered by such a model include design methods, review procedures, project management controls, change control procedures, test methods and acceptance procedures. An effective life cycle model will address these aspects of the development and maintenance process within an overall management structure that assigns responsibilities and monitors progress.

There are different types of acceptance situations that are dealt with at different locations in the criteria:

- acceptance of parts delivered by subcontractors ("integration") should be treated in this family, ALC_LCD (Development life cycle definition),
- acceptance subsequent to internal transportations in ALC_DVS (Developer environment security) ([12.5](#)),
- acceptance of parts into the CM system in ALC_CMC (CM capabilities) ([12.2](#)), and
- acceptance of the delivered TOE by the recipient in ALC_DEL (Delivery) ([12.4](#)).

The first three types can overlap.

Although life cycle definition deals with the maintenance of the TOE and hence with aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of the life cycle information for the TOE provided at the time of the evaluation.

ISO/IEC 15408-3:2026(en)

A life cycle model provides for the necessary control over the development and maintenance of the TOE, if the model enables sufficient minimization of the danger that the TOE will not meet its security requirement.

A measurable life cycle model is a model using some quantitative valuation (arithmetic parameters, metrics, etc.) of the managed product in order to measure development properties of the product.

Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean time to failure. For the security evaluation all those metrics are of relevance. They are used to increase quality by decreasing the probability of faults, thereby increasing assurance in the security of the TOE.

The evaluator should take into account that there exist standardized life cycle models on the one hand (like the waterfall model) and standardized metrics on the other hand (like error density), which can be combined. The ISO/IEC 15408 series does not require the life cycle to follow exactly one standard defining both aspects.

12.7.4 Developer defined life cycle processes (ALC_LCD.1)

[Table 208](#) specifies the developer action elements for ALC_LCD.1, with [Table 209](#) indicating their related content and presentation requirements, and [Table 210](#) the corresponding evaluation actions.

Table 208 — Developer action elements for ALC_LCD.1

ALC_LCD.1.1D	The developer shall establish a life cycle model to be used in the development and maintenance of the TOE.
ALC_LCD.1.2D	The developer shall provide life cycle definition documentation.

Table 209 — Content and presentation elements for ALC_LCD.1

ALC_LCD.1.1C	The life cycle definition documentation shall describe the processes used to develop and maintain the TOE.
ALC_LCD.1.2C	The life cycle model shall provide for the necessary control over the development and maintenance of the TOE.

Table 210 — Evaluator action elements for ALC_LCD.1

ALC_LCD.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

12.7.5 Measurable life cycle model (ALC_LCD.2)

[Table 211](#) specifies the developer action elements for ALC_LCD.2, with [Table 212](#) indicating their related content and presentation requirements, and [Table 213](#) the corresponding evaluation actions.

Table 211 — Developer action elements for ALC_LCD.2

ALC_LCD.2.1D	The developer shall establish a life cycle model to be used in the development and maintenance of the TOE that is based on a measurable life cycle model.
ALC_LCD.2.2D	The developer shall provide life cycle definition documentation.
ALC_LCD.2.3D	The developer shall measure the TOE development using the measurable life cycle model.
ALC_LCD.2.4D	The developer shall provide life cycle output documentation.

Table 212 — Content and presentation elements for ALC_LCD.2

ALC_LCD.2.1C	The life cycle definition documentation shall describe the processes used to develop and maintain the TOE including the details of its arithmetic parameters, metrics, or a combination, used to measure the quality of either the TOE or its development, or both.
ALC_LCD.2.2C	The life cycle model shall provide for the necessary control over the development and maintenance of the TOE.
ALC_LCD.2.3C	The life cycle output documentation shall provide the results of the measurements of the TOE development using the measurable life cycle model.

Table 213 — Evaluator action elements for ALC_LCD.2

ALC_LCD.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_LCD.2.2E	The evaluator shall confirm that the measurements of the TOE development processes and security relevant properties of the TOE support improvements either in the development processes or the TOE itself, or both.

12.8 TOE development artefacts (ALC_TDA)

12.8.1 Objectives

This family aims to add trust to the development process or a development. It focuses on the generation of certain artefacts in the development process. These artefacts are used at a later point in time to assess the degree to which the development process is trustable. This trust is realized through the validation of the generated artefacts for confirming them as sufficient evidence for trustable development.

This family introduces developer practices within the development process to generate the required artefacts for realizing trustable development. If a requirement in this family does not explicitly specify the use of automation to generate the required artefacts, the developer is free to undertake the corresponding practice manually, or to use some integrated automation in the development process, or to use a hybrid method of both. It is expected that the degree of trust in the development process is proportional to the degree of automation adoption to implement the corresponding practice in the development process.

This family also has a relationship with the ALC_TAT (Tools and techniques) (12.9) family. As ALC_TAT (Tools and techniques) (12.9) focuses on the tools and techniques aspect for developing, analysing, and implementing the TOE, it provides the necessary context when describing the practices of this family being introduced into the development process.

12.8.2 Component levelling

The components in this family are levelled on the basis of increasing cross-checking for consistency with relevant evidence from components of other families of other security assurance classes.

12.8.3 Application notes

The requirements in ALC_TDA.1 (Uniquely identifying implementation representation) (12.8.4) provide a degree of trust in the developer's ability to identify the set of implementation representation which actually has been used during the TOE generation time. This degree of trust helps to positively answer the questions which are potentially relevant in an evaluation:

- is that really the source code for this software?
- is that really the register-transfer level (RTL) design or description for this integrated circuit hardware?
- is that really the set of implementation representation for this TOE?

Such degree of trust is built on:

- the timing of when the set of implementation representation identifiers is recorded or logged,

ISO/IEC 15408-3:2026(en)

- the integrity and authenticity of the record of implementation representation identifiers, and
- the traceability of implementation representation identifiers from the TOE.

In the case where some implementation representation elements are also covered in the configuration list due to ALC_CMS.3 (Implementation representation CM coverage) (12.3.6), the requirements in ALC_TDA.2 (Matching CMS scope of implementation representation) (12.8.5) ensure that these implementation representation elements actually are identifiable through the use of the implementation representation identifiers of ALC_TDA.1 (Uniquely identifying implementation representation) (12.8.4).

With the accurate recording or logging of the actual implementation representation being used by the development tools under the scope of ALC_TAT (Tools and techniques) (12.9), it provides an additional evidence to convince a third party that a regeneration of the TOE is functionally equivalent to the original TOE.

The requirements in ALC_TDA.3 (Regenerate TOE with well-defined development tools) (12.8.6), provide the developer an opportunity to testify the absence of functional differences between the two possibly visibly different TOEs which have been independently generated from the identical set of implementation representation.

12.8.4 Uniquely identifying implementation representation (ALC_TDA.1)

Table 214 specifies the developer action elements for ALC_TDA.1, with Table 215 indicating their related content and presentation requirements, and Table 216 the corresponding evaluation actions.

Table 214 — Developer action elements for ALC_TDA.1

ALC_TDA.1.1D	The developer shall identify individual elements of the TOE implementation representation to record the list of unique TOE implementation representation identifiers, as the development tool generates the TOE.
ALC_TDA.1.2D	The developer shall use the current date and time to timestamp the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.1.3D	The developer shall maintain the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.1.4D	The developer shall ensure the authenticity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time, with the maintenance of the (author) origination information.
ALC_TDA.1.5D	The developer shall be able to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.1.6D	The developer shall produce and provide documentation describing: <ul style="list-style-type: none"> — the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the maintenance of the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information; — the developer's mechanism to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

ISO/IEC 15408-3:2026(en)

Table 215 — Content and presentation elements for ALC_TDA.1

ALC_TDA.1.1C	The list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall demonstrate the correspondence between the TOE implementation representation element identifiers and the TOE implementation representation element names.
ALC_TDA.1.2C	The TOE implementation representation element names shall be in the same form as used or referenced by the development tool to generate the TOE.
ALC_TDA.1.3C	The timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the creation time of the TOE.
ALC_TDA.1.4C	The (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the (author) origination information of the TOE. The author origination information can be the name of an affiliate of an organization.

Table 216 — Evaluator action elements for ALC_TDA.1

ALC_TDA.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_TDA.1.2E	The evaluator <i>shall confirm</i> that the development tool for generating the TOE is capable to use or reference the implementation representation element names.
ALC_TDA.1.3E	The evaluator <i>shall confirm</i> that the timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time is consistent with the creation time of the TOE.
ALC_TDA.1.4E	The evaluator <i>shall confirm</i> that the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time is consistent with the (author) origination information of the TOE.
ALC_TDA.1.5E	The evaluator <i>shall check</i> the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information.
ALC_TDA.1.6E	The evaluator <i>shall confirm</i> the developer's ability to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

12.8.5 Matching CMS scope of implementation representation (ALC_TDA.2)

12.8.5.1 Component relationships

Dependencies: ALC_CMS.3 (Implementation representation CM coverage) ([12.3.6](#))

12.8.5.2 Component elements

[Table 217](#) specifies the developer action elements for ALC_TDA.2, with [Table 218](#) indicating their related content and presentation requirements, and [Table 219](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 217 — Developer action elements for ALC_TDA.2

ALC_TDA.2.1D	The developer shall identify individual elements of the TOE implementation representation to record the list of unique TOE implementation representation identifiers, as the development tool generates the TOE.
ALC_TDA.2.2D	The developer shall use the current date and time to timestamp the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.2.3D	The developer shall maintain the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.2.4D	The developer shall ensure the authenticity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time, with the maintenance of the (author) origination information.
ALC_TDA.2.5D	The developer shall be able to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.2.6D	The developer shall produce and provide documentation describing: <ul style="list-style-type: none"> — the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the maintenance of the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information; — the developer's mechanism to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.2.7D	The developer shall provide evidence that the elements of implementation representation under the configuration scope of ALC_CMS.3 (Implementation representation CM coverage) (12.3.6) are identified by the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

Table 218 — Content and presentation elements for ALC_TDA.2

ALC_TDA.2.1C	The list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall demonstrate the correspondence between the TOE implementation representation element identifiers and the TOE implementation representation element names.
ALC_TDA.2.2C	The TOE implementation representation element names shall be in the same form as used or referenced by the development tool to generate the TOE.
ALC_TDA.2.3C	The timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the creation time of the TOE.
ALC_TDA.2.4C	The (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the (author) origination information of the TOE. The author origination information can be the name of an affiliate of an organization.
ALC_TDA.2.5C	The list of identifiers of the elements of implementation representation under the configuration scope of ALC_CMS.3 (Implementation representation CM coverage) (12.3.6) shall match with the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

ISO/IEC 15408-3:2026(en)

Table 219 — Evaluator action elements for ALC_TDA.2

ALC_TDA.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ALC_TDA.2.2E	The evaluator shall confirm that the development tool for generating the TOE is capable to use or reference the implementation representation element names.
ALC_TDA.2.3E	The evaluator shall confirm that the timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time is consistent with the creation time of the TOE.
ALC_TDA.2.4E	The evaluator shall confirm that the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time is consistent with the (author) origination information of the TOE.
ALC_TDA.2.5E	The evaluator shall check the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information.
ALC_TDA.2.6E	The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.2.7E	The evaluator shall confirm that the list of identifiers of the elements of implementation representation under the configuration scope of ALC_CMS.3 (Implementation representation CM coverage) (12.3.6) matches with the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

12.8.6 Regenerate TOE with well-defined development tools (ALC_TDA.3)

12.8.6.1 Component relationships

Dependencies: ALC_CMS.3 (Implementation representation CM coverage) (12.3.6)
 ALC_TAT.1 (Well-defined development tools) (12.9.4)
 ADV_IMP.1 (Implementation representation of the TSF) (10.4.4)

12.8.6.2 Component elements

Table 220 specifies the developer action elements for ALC_TDA.3, with Table 221 indicating their related content and presentation requirements, and Table 222 the corresponding evaluation actions.

Table 220 — Developer action elements for ALC_TDA.3

ALC_TDA.3.1D	The developer shall identify individual elements of the TOE implementation representation to record the list of unique TOE implementation representation identifiers, as the development tool generates the TOE.
ALC_TDA.3.2D	The developer shall use the current date and time to timestamp the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.3D	The developer shall maintain the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.4D	The developer shall ensure the authenticity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time, with the maintenance of the (author) origination information.
ALC_TDA.3.5D	The developer shall be able to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.6D	The developer shall produce and provide documentation describing:

ISO/IEC 15408-3:2026(en)

Table 220 (continued)

	<ul style="list-style-type: none"> — the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time; — the maintenance of the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information; — the developer's mechanism to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.7D	The developer shall provide evidence that the elements of implementation representation under the configuration scope of ALC_CMS.3 (Implementation representation CM coverage) (12.3.6) are identified by the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.8D	After applying the development tools to another copy of the TOE implementation representation according to the list of unique TOE implementation representation identifiers to regenerate a TOE copy, the developer shall explain any functional differences between the TOE copy and the original TOE.
ALC_TDA.3.9D	The developer shall produce and provide documentation explaining any functional differences between the regenerated TOE copy and the original TOE.

Table 221 — Content and presentation elements for ALC_TDA.3

ALC_TDA.3.1C	The list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall demonstrate the correspondence between the TOE implementation representation element identifiers and the TOE implementation representation element names.
ALC_TDA.3.2C	The TOE implementation representation element names shall be in the same form as used or referenced by the development tool to generate the TOE.
ALC_TDA.3.3C	The timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the creation time of the TOE.
ALC_TDA.3.4C	The (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the (author) origination information of the TOE. The author origination information can be the name of an affiliate of an organization.
ALC_TDA.3.5C	The list of identifiers of the elements of implementation representation under the configuration scope of ALC_CMS.3 (Implementation representation CM coverage) (12.3.6) shall match with the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.6C	The developer's explanation of any functional differences between the regenerated TOE copy and the original TOE shall take into account any visible differences between the regenerated TOE copy and the original TOE

Table 222 — Evaluator action elements for ALC_TDA.3

ALC_TDA.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ALC_TDA.3.2E	The evaluator shall confirm that the development tool for generating the TOE is capable to use or reference the implementation representation element names.
ALC_TDA.3.3E	The evaluator shall confirm that the timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time is consistent with the creation time of the TOE.
ALC_TDA.3.4E	The evaluator shall confirm that the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time is consistent with the (author) origination information of the TOE.
ALC_TDA.3.5E	The evaluator shall check the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information.
ALC_TDA.3.6E	The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.7E	The evaluator shall confirm that the list of identifiers of the elements of implementation representation under the configuration scope of ALC_CMS.3 (Implementation representation CM coverage) (12.3.6) matches with the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.
ALC_TDA.3.8E	The evaluator shall check that the developer's explanation of any functional differences between the regenerated TOE copy and the original TOE takes into account any visible differences between the regenerated TOE copy and the original TOE.

12.9 Tools and techniques (ALC_TAT)

12.9.1 Objectives

Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools from being used to develop the TOE. This includes, but is not limited to, programming languages, documentation, implementation standards, and other parts of the TOE such as supporting runtime libraries.

12.9.2 Component levelling

The components in this family are levelled on the basis of increasing requirements on the description and scope of the implementation standards and the documentation of implementation-dependent options.

12.9.3 Application notes

There is a requirement for well-defined development tools. These are tools that are clearly and completely described. For example, programming languages and computer aided design (CAD) systems that are based on a standard published by a standardization body are assumed to be well-defined. Self-made tools would need further investigation to clarify whether they are well-defined.

The requirement in ALC_TAT.1.2C ([12.9.4](#)) is especially applicable to programming languages so as to ensure that all statements in the source code have an unambiguous meaning.

In ALC_TAT.2 (Compliance with implementation standards) ([12.9.5](#)) and ALC_TAT.3 (Compliance with implementation standards - all parts) ([12.9.6](#)), implementation guidelines can be accepted as an implementation standard if they have been approved by some group of experts (e.g. academic experts, standards bodies). Implementation standards are normally public, well accepted and common practice in a specific industry, but developer-specific implementation guidelines can also be accepted as a standard; the emphasis is on the expertise.

Tools and techniques distinguishes between the implementation standards applied by the developer (ALC_TAT.2.3D ([12.9.5](#))) and the implementation standards for "all parts of the TOE" (ALC_TAT.3.3D ([12.9.6](#)))

ISO/IEC 15408-3:2026(en)

which include third-party software, hardware, or firmware. The configuration list introduced in ALC_CMS (CM scope) (12.3) requires that for each TSF relevant configuration item to indicate if it has been generated by the TOE developer or by third-party developers.

12.9.4 Well-defined development tools (ALC_TAT.1)**12.9.4.1 Component relationships**

Dependencies: ADV_IMP.1 (Implementation representation of the TSF) (10.4.4)

12.9.4.2 Component elements

Table 223 specifies the developer action elements for ALC_TAT.1, with Table 224 indicating their related content and presentation requirements, and Table 225 the corresponding evaluation actions.

Table 223 — Developer action elements for ALC_TAT.1

ALC_TAT.1.1D	The developer shall provide the documentation identifying each development tool being used for the TOE.
ALC_TAT.1.2D	The developer shall document and provide the selected implementation-dependent options of each development tool.

Table 224 — Content and presentation elements for ALC_TAT.1

ALC_TAT.1.1C	Each development tool used for implementation shall be well-defined.
ALC_TAT.1.2C	The documentation of each development tool shall unambiguously define the meaning of all statements as well as all conventions and directives used in the implementation.
ALC_TAT.1.3C	The documentation of each development tool shall unambiguously define the meaning of all implementation-dependent options.

Table 225 — Evaluator action elements for ALC_TAT.1

ALC_TAT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

12.9.5 Compliance with implementation standards (ALC_TAT.2)**12.9.5.1 Component relationships**

Dependencies: ADV_IMP.1 (Implementation representation of the TSF) (10.4.4)

12.9.5.2 Component elements

Table 226 specifies the developer action elements for ALC_TAT.2, with Table 227 indicating their related content and presentation requirements, and Table 228 the corresponding evaluation actions.

Table 226 — Developer action elements for ALC_TAT.2

ALC_TAT.2.1D	The developer shall provide the documentation identifying each development tool being used for the TOE.
ALC_TAT.2.2D	The developer shall document and provide the selected implementation-dependent options of each development tool.
ALC_TAT.2.3D	The developer shall describe and provide the implementation standards that are being applied by the developer.

ISO/IEC 15408-3:2026(en)

Table 227 — Content and presentation elements for ALC_TAT.2

ALC_TAT.2.1C	Each development tool used for implementation shall be well-defined.
ALC_TAT.2.2C	The documentation of each development tool shall unambiguously define the meaning of all statements as well as all conventions and directives used in the implementation.
ALC_TAT.2.3C	The documentation of each development tool shall unambiguously define the meaning of all implementation-dependent options.

Table 228 — Evaluator action elements for ALC_TAT.2

ALC_TAT.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_TAT.2.2E	The evaluator shall confirm that the implementation standards have been applied.

12.9.6 Compliance with implementation standards - all parts (ALC_TAT.3)

12.9.6.1 Component relationships

Dependencies: ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))

12.9.6.2 Component elements

[Table 229](#) specifies the developer action elements for ALC_TAT.3, with [Table 230](#) indicating their related content and presentation requirements, and [Table 231](#) the corresponding evaluation actions.

Table 229 — Developer action elements for ALC_TAT.3

ALC_TAT.3.1D	The developer shall provide the documentation identifying each development tool being used for the TOE.
ALC_TAT.3.2D	The developer shall document and provide the selected implementation-dependent options of each development tool.
ALC_TAT.3.3D	The developer shall describe and provide the implementation standards that are being applied by the developer and by any third-party providers for all parts of the TOE.

Table 230 — Content and presentation elements for ALC_TAT.3

ALC_TAT.3.1C	Each development tool used for implementation shall be well-defined.
ALC_TAT.3.2C	The documentation of each development tool shall unambiguously define the meaning of all statements as well as all conventions and directives used in the implementation.
ALC_TAT.3.3C	The documentation of each development tool shall unambiguously define the meaning of all implementation-dependent options.

Table 231 — Evaluator action elements for ALC_TAT.3

ALC_TAT.3.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_TAT.3.2E	The evaluator <i>shall confirm</i> that the implementation standards have been applied.

ISO/IEC 15408-3:2026(en)

12.10 Integration of composition parts and consistency check of delivery procedures (ALC_COMP)

12.10.1 Objectives

The aim of this family is to determine whether:

- the correct version of the dependent component is installed onto or embedded into the correct version of the related base component, and
- the preparative guidance procedures of the base component developer and the dependent component developer are compatible with the acceptance procedures of the composite product integrator.

12.10.2 Component levelling

This family contains only one component.

12.10.3 Application notes

The composite product evaluator shall verify that the correct version of the dependent component under evaluation has been installed onto or embedded into the evaluated version of the related base component of the composite product.

The composite product evaluation sponsor shall ensure that appropriate evidence generated by the composite product integrator is available for the composite product evaluator. This evidence can include elements such as the configuration list of the base component developer (e.g. provided within the acknowledgement statement).

The composite product evaluator shall verify that the delivery procedures of the base component developer and the dependent component developer are compatible with the acceptance procedures used by the composite product integrator.

The composite product evaluator shall verify that all configuration parameters prescribed by the base component developer and the dependent component developer (e.g. pre-personalization data, pre-personalisation scripts) are used by the composite product integrator.

The composite product evaluation sponsor shall ensure that appropriate evidence generated by the composite product integrator is available for the composite product evaluator. This evidence can include elements such as the element of evidence for the dependent component reception, acceptance and parameterisation by the base component developer (e.g. in form of their acknowledgement statement).

12.10.4 Integration of the dependent component into the related base component and Consistency check for delivery and acceptance procedures (ALC_COMP.1)

[Table 232](#) specifies the developer action elements for ALC_COMP.1, with [Table 233](#) indicating their related content and presentation requirements, and [Table 234](#) the corresponding evaluation actions.

Table 232 — Developer action elements for ALC_COMP.1

ALC_COMP.1.1D	The developer shall provide components' configuration evidence.
---------------	---

Table 233 — Content and presentation elements for ALC_COMP.1

ALC_COMP.1.1C	The components' configuration evidence shall show that the evaluated version of the dependent component has been installed onto or embedded into the evaluated version of the related base component.
ALC_COMP.1.2C	The components' configuration evidence shall show that: <ul style="list-style-type: none"> — the evidence for delivery and acceptance compatibility shall show that the delivery procedures of the base component developer and the dependent component developer are compatible with the acceptance procedures of the composite product integrator; — the evidence shall show that preparative guidance procedures prescribed by the base component developer and the dependent component developer are either actually being used by the composite product integrator or compatible with the composite product integrator guidance and do not contradict each other.

Table 234 — Evaluator action elements for ALC_COMP.1

ALC_COMP.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ALC_COMP.1.2E	The evaluator <i>shall confirm</i> that the evidence for delivery compatibility is complete, coherent, and internally consistent.

13 Class ATE Tests

13.1 General

The class "Tests" encompasses five families:

- ATE_COV (Coverage) ([13.2](#)),
- ATE_DPT (Depth) ([13.3](#)),
- ATE_IND (Independent testing) ([13.5](#)) (i.e. functional testing performed by evaluators),
- ATE_FUN (Functional tests) ([13.4](#)) and
- ATE_COMP (Composite functional testing) ([13.6](#)).

Testing provides assurance that the TSF behaves as described (in the functional specification, TOE design, implementation representation, and allows straightforward traceability of SFR in test scenario).

The emphasis in this class is on confirmation that the TSF operates according to its design descriptions. This class does not address penetration testing, which is based upon an analysis of the TSF that specifically seeks to identify vulnerabilities in the design and implementation of the TSF. Penetration testing is addressed separately as an aspect of vulnerability assessment in class AVA (Vulnerability assessment) ([Clause 14](#)).

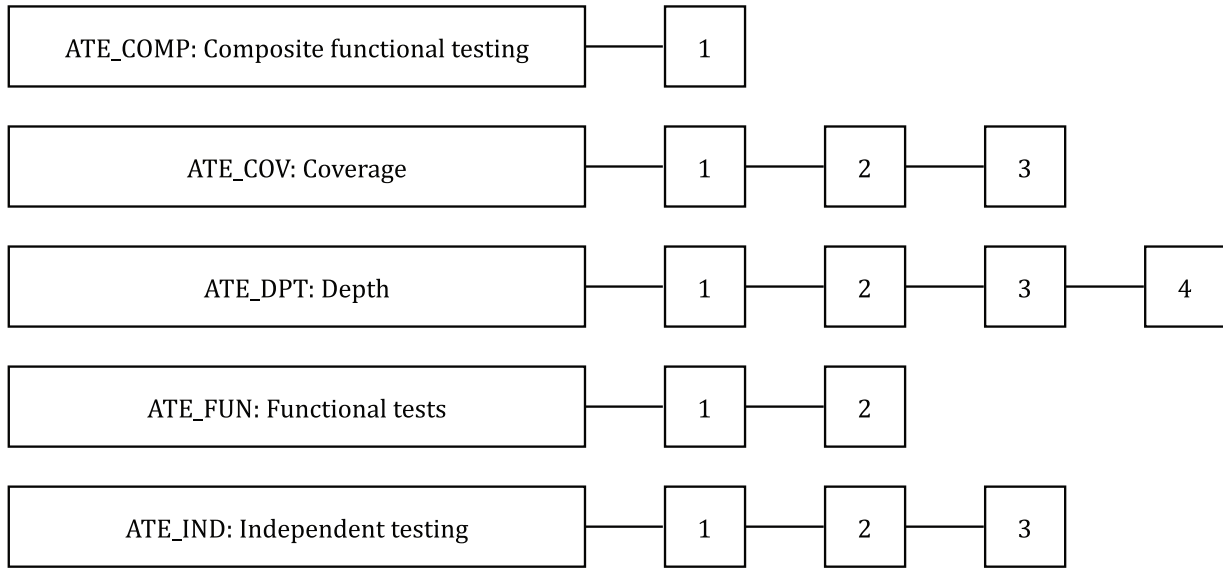
ATE (Tests) separates testing into developer testing and evaluator testing. The ATE_COV (Coverage) ([13.2](#)) and ATE_DPT (Depth) ([13.3](#)) families address the completeness of developer testing. ATE_COV (Coverage) ([13.2](#)) addresses the rigour with which the functional specification is tested. ATE_DPT (Depth) ([13.3](#)) addresses whether testing against other design descriptions (security architecture, TOE design, and implementation representation) is required.

ATE_FUN (Functional tests) ([13.4](#)) addresses the performing of the tests by the developer and how this testing should be documented. Finally, ATE_IND (Independent testing) ([13.5](#)) then addresses evaluator testing: whether the evaluator should repeat part or all of the developer testing and how much independent testing the evaluator shall do.

ATE_COMP (Composite functional testing) ([13.6](#)) determines whether the composite product as a whole exhibits the properties necessary to satisfy the functional requirements of its ST.

ISO/IEC 15408-3:2026(en)

Figure 12 describes the family hierarchy of class ATE.



Key
 1,2,3,4 components

Figure 12 — ATE: Tests, class decomposition

Direct, indirect and optional dependencies for the components of class ATE are shown in Table 235.

Table 235 — Dependency table for class ATE: Tests

	ADV_ARC.1 (10.2.4)	ADV_FSP.1 (10.3.4)	ADV_FSP.2 (10.3.5)	ADV_FSP.4 (10.3.7)	ADV_IMP.1 (10.4.4)	ADV_TDS.2 (10.7.5)	ADV_TDS.3 (10.7.6)	ADV_TDS.4 (10.7.7)	AGD_OPE.1 (11.2.4)	AGD_PRE.1 (11.3.4)	ATE_COV.1 (13.2.3)	ATE_FUN.1 (13.4.4)
ATE_COMP.1 (13.6.4)												
ATE_COV.1 (13.2.3)			X								-	X
ATE_COV.2 (13.2.4)			X								-	X
ATE_COV.3 (13.2.5)			X								-	X
ATE_DPT.1 (13.3.4)	X		-			X					-	X
ATE_DPT.2 (13.3.5)	X		-				X				-	X
ATE_DPT.3 (13.3.6)	X		-					X			-	X
ATE_DPT.4 (13.3.7)	X		-		X			X			-	X
ATE_FUN.1 (13.4.4)			-								X	-
ATE_FUN.2 (13.4.5)			-								X	-
ATE_IND.1 (13.5.4)		X							X	X		
ATE_IND.2 (13.5.5)			X						X	X	X	X
ATE_IND.3 (13.5.6)			-	X					X	X	X	X

13.2 Coverage (ATE_COV)

13.2.1 Objectives

This family establishes that the TSF has been tested against its functional specification. This is achieved through an examination of developer evidence of correspondence.

ISO/IEC 15408-3:2026(en)

13.2.2 Component levelling

The components in this family are levelled on the basis of specification.

13.2.3 Evidence of coverage (ATE_COV.1)

13.2.3.1 Component relationships

Dependencies: ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.2.3.2 Objectives

The objective of this component is to establish that some of the TSFIs have been tested.

13.2.3.3 Application notes

In this component the developer shows how tests in the test documentation correspond to TSFIs in the functional specification. This can be achieved by a statement of correspondence, for example by using a table.

13.2.3.4 Component elements

[Table 236](#) specifies the developer action elements for ATE_COV.1, with [Table 237](#) indicating their related content and presentation requirements, and [Table 238](#) the corresponding evaluation actions.

Table 236 — Developer action elements for ATE_COV.1

ATE_COV.1.1D	The developer shall provide evidence of the test coverage.
--------------	--

Table 237 — Content and presentation elements for ATE_COV.1

ATE_COV.1.1C	The evidence of the test coverage shall show the correspondence between the tests in the test documentation and the TSFIs in the functional specification.
--------------	--

Table 238 — Evaluator action elements for ATE_COV.1

ATE_COV.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	---

13.2.4 Analysis of coverage (ATE_COV.2)

13.2.4.1 Component relationships

Dependencies: ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.2.4.2 Objectives

The objective of this component is to confirm that all the TSFIs have been tested.

13.2.4.3 Application notes

In this component the developer confirms that tests in the test documentation correspond to all the TSFIs in the functional specification. This can be achieved by a statement of correspondence, for example by using a table, but the developer is also required to provide an analysis of the test coverage.

13.2.4.4 Component elements

[Table 239](#) specifies the developer action elements for ATE_COV.2, with [Table 240](#) indicating their related content and presentation requirements, and [Table 241](#) the corresponding evaluation actions.

Table 239 — Developer action elements for ATE_COV.2

ATE_COV.2.1D	The developer shall provide an analysis of the test coverage.
--------------	--

Table 240 — Content and presentation elements for ATE_COV.2

ATE_COV.2.1C	The analysis of the test coverage shall demonstrate the correspondence between the tests in the test documentation and the TSFIs in the functional specification.
ATE_COV.2.2C	The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification have been tested.

Table 241 — Evaluator action elements for ATE_COV.2

ATE_COV.2.1E	The evaluator shall confirm that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

13.2.5 Rigorous analysis of coverage (ATE_COV.3)

13.2.5.1 Component relationships

Dependencies: ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.2.5.2 Objectives

In this component, the objective is to confirm that the developer performed exhaustive tests of all interfaces in the functional specification.

The objective of this component is to confirm that all parameters of all the TSFIs have been tested.

13.2.5.3 Application notes

In this component the developer is required to show how tests in the test documentation correspond to all the TSFIs in the functional specification. This can be achieved by a statement of correspondence, for example by using a table, but in addition the developer is required to demonstrate that the tests exercise all of the parameters of all TSFIs. This additional requirement includes bounds testing (i.e. verifying that errors are generated when stated limits are exceeded) and negative testing (e.g. when access is given to User A, verifying not only that User A now has access, but also that User B did not suddenly gain access). This kind of testing is not, strictly speaking, exhaustive because not every possible value of the parameters is expected to be checked.

13.2.5.4 Component elements

[Table 242](#) specifies the developer action elements for ATE_COV.3, with [Table 243](#) indicating their related content and presentation requirements, and [Table 244](#) the corresponding evaluation actions.

Table 242 — Developer action elements for ATE_COV.3

ATE_COV.3.1D	The developer shall provide an analysis of the test coverage.
--------------	---

ISO/IEC 15408-3:2026(en)

Table 243 — Content and presentation elements for ATE_COV.3

ATE_COV.3.1C	The analysis of the test coverage shall demonstrate the correspondence between the tests in the test documentation and the TSFIs in the functional specification.
ATE_COV.3.2C	The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification have been completely tested.

Table 244 — Evaluator action elements for ATE_COV.3

ATE_COV.3.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

13.3 Depth (ATE_DPT)

13.3.1 Objectives

The components in this family deal with the level of detail to which the TSF is tested by the developer. Testing of the TSF is based upon increasing depth of information derived from additional design representations and descriptions (TOE design, implementation representation, and security architecture description).

The objective is to counter the risk of missing an error in the development of the TOE. Testing that exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired external security behaviour, but also that this behaviour stems from correctly operating internal functionality.

13.3.2 Component levelling

The components in this family are levelled on the basis of increasing detail provided in the TSF representations, from the TOE design to the implementation representation. This levelling reflects the TSF representations presented in class ADV (Development) ([Clause 10](#)).

13.3.3 Application notes

The TOE design describes the internal components (e.g. subsystems) and, perhaps, modules of the TSF, together with a description of the interfaces among these components and modules. Evidence of testing of this TOE design shall show that the internal interfaces have been exercised and seen to behave as described. This can be achieved through testing via the external interfaces of the TSF, or by testing of the TOE subsystem or module interfaces in isolation, perhaps employing a test harness. In cases where some aspects of an internal interface cannot be tested via the external interfaces, there shall either be a justification that these aspects are excluded from testing, or the internal interface shall be tested directly. In the latter case, the TOE design shall be sufficiently detailed in order to facilitate direct testing.

In cases where the description of the TSF's architectural soundness (in ADV_ARC (Security architecture) ([10.2](#))) cites specific mechanisms, the tests performed by the developer are required to show that the mechanisms have been exercised and seen to behave as described.

At the highest component of this family, the testing is performed not only against the TOE design, but also against the implementation representation.

13.3.4 Testing: basic design (ATE_DPT.1)

13.3.4.1 Component relationships

Dependencies: ADV_ARC.1 (Security architecture description) ([10.2.4](#))
 ADV_TDS.2 (Architectural design) ([10.7.5](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

ISO/IEC 15408-3:2026(en)

13.3.4.2 Objectives

The subsystem descriptions of the TSF provide a high-level description of the internal workings of the TSF. Testing at the level of the TOE subsystems provides assurance that the TSF subsystems behave and interact as described in the TOE design and the security architecture description.

13.3.4.3 Component elements

[Table 245](#) specifies the developer action elements for ATE_DPT.1, with [Table 246](#) indicating their related content and presentation requirements, and [Table 247](#) the corresponding evaluation actions.

Table 245 — Developer action elements for ATE_DPT.1

ATE_DPT.1.1D	The developer shall provide the analysis of the depth of testing.
--------------	---

Table 246 — Content and presentation elements for ATE_DPT.1

ATE_DPT.1.1C	The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the TSF subsystems in the TOE design.
ATE_DPT.1.2C	The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have been tested.

Table 247 — Evaluator action elements for ATE_DPT.1

ATE_DPT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

13.3.5 Testing: security enforcing modules (ATE_DPT.2)

13.3.5.1 Component relationships

Dependencies: ADV_ARC.1 (Security architecture description) ([10.2.4](#))
 ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.3.5.2 Objectives

The subsystem and module descriptions of the TSF provide a high-level description of the internal workings, and a description of the interfaces of the SFR-enforcing modules, of the TSF. Testing at this level of TOE description provides assurance that the TSF subsystems and SFR-enforcing modules behave and interact as described in the TOE design and the security architecture description.

13.3.5.3 Component elements

[Table 248](#) specifies the developer action elements for ATE_DPT.2, with [Table 249](#) indicating their related content and presentation requirements, and [Table 250](#) the corresponding evaluation actions.

Table 248 — Developer action elements for ATE_DPT.2

ATE_DPT.2.1D	The developer shall provide the analysis of the depth of testing.
--------------	---

ISO/IEC 15408-3:2026(en)

Table 249 — Content and presentation elements for ATE_DPT.2

ATE_DPT.2.1C	The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the TSF subsystems and SFR-enforcing modules in the TOE design.
ATE_DPT.2.2C	The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have been tested.
ATE_DPT.2.3C	The analysis of the depth of testing shall demonstrate that the SFR-enforcing modules in the TOE design have been tested.

Table 250 — Evaluator action elements for ATE_DPT.2

ATE_DPT.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

13.3.6 Testing: modular design (ATE_DPT.3)

13.3.6.1 Component relationships

Dependencies: ADV_ARC.1 (Security architecture description) ([10.2.4](#))
 ADV_TDS.4 (Semi-Formal modular design) ([10.7.7](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.3.6.2 Objectives

The subsystem and module descriptions of the TSF provide a high-level description of the internal workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE description provides assurance that the TSF subsystems and modules behave and interact as described in the TOE design and the security architecture description.

13.3.6.3 Component elements

[Table 251](#) specifies the developer action elements for ATE_DPT.3, with [Table 252](#) indicating their related content and presentation requirements, and [Table 253](#) the corresponding evaluation actions.

Table 251 — Developer action elements for ATE_DPT.3

ATE_DPT.3.1D	The developer shall provide the analysis of the depth of testing.
--------------	---

Table 252 — Content and presentation elements for ATE_DPT.3

ATE_DPT.3.1C	The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the TSF subsystems and modules in the TOE design.
ATE_DPT.3.2C	The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have been tested.
ATE_DPT.3.3C	The analysis of the depth of testing shall demonstrate that all TSF modules in the TOE design have been tested.

Table 253 — Evaluator action elements for ATE_DPT.3

ATE_DPT.3.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

ISO/IEC 15408-3:2026(en)

13.3.7 Testing: implementation representation (ATE_DPT.4)

13.3.7.1 Component relationships

Dependencies: ADV_ARC.1 (Security architecture description) ([10.2.4](#))
 ADV_TDS.4 (Semi-Formal modular design) ([10.7.7](#))
 ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.3.7.2 Objectives

The subsystem and module descriptions of the TSF provide a high-level description of the internal workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE description provides assurance that the TSF subsystems and modules behave and interact as described in the TOE design and the security architecture description, and in accordance with the implementation representation.

13.3.7.3 Component elements

[Table 254](#) specifies the developer action elements for ATE_DPT.4, with [Table 255](#) indicating their related content and presentation requirements, and [Table 256](#) the corresponding evaluation actions.

Table 254 — Developer action elements for ATE_DPT.4

ATE_DPT.4.1D	The developer shall provide the analysis of the depth of testing.
--------------	---

Table 255 — Content and presentation elements for ATE_DPT.4

ATE_DPT.4.1C	The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the TSF subsystems and modules in the TOE design.
ATE_DPT.4.2C	The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have been tested.
ATE_DPT.4.3C	The analysis of the depth of testing shall demonstrate that all modules in the TOE design have been tested.
ATE_DPT.4.4C	The analysis of the depth of testing shall demonstrate that the TSF operates in accordance with its implementation representation.

Table 256 — Evaluator action elements for ATE_DPT.4

ATE_DPT.4.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

13.4 Functional tests (ATE_FUN)

13.4.1 Objectives

Functional testing performed by the developer provides assurance that the tests in the test documentation are performed and documented correctly. The correspondence of these tests to the design descriptions of the TSF is achieved through the ATE_COV (Coverage) ([13.2](#)) and ATE_DPT (Depth) ([13.3](#)) families.

This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

The families ATE_COV (Coverage) ([13.2](#)), ATE_DPT (Depth) ([13.3](#)) and ATE_FUN (Functional tests) are used in combination to define the evidence of testing to be supplied by a developer. Independent functional testing by the evaluator is specified by ATE_IND (Independent testing) ([13.5](#)).

ISO/IEC 15408-3:2026(en)

13.4.2 Component levelling

This family contains two components, the higher requiring that ordering dependencies are analysed.

13.4.3 Application notes

Procedures for performing tests are expected to provide instructions for using test programs and test suites, including the test environment, test conditions, test data parameters and values. The test procedures shall also show how the test results are derived from the test inputs.

Ordering dependencies are relevant when the successful execution of a particular test depends upon the existence of a particular state. For example, this can require that test A be executed immediately before test B, since the state resulting from the successful execution of test A is a prerequisite for the successful execution of test B. Thus, failure of test B can be related to a problem with the ordering dependencies. In the above example, test B can fail because test C (rather than test A) was executed immediately before it, or the failure of test B can be related to a failure of test A.

13.4.4 Functional testing (ATE_FUN.1)

13.4.4.1 Component relationships

Dependencies: ATE_COV.1 (Evidence of coverage) ([13.2.3](#))

13.4.4.2 Objectives

The objective is for the developer to demonstrate that the tests in the test documentation are performed and documented correctly.

13.4.4.3 Component elements

[Table 257](#) specifies the developer action elements for ATE_FUN.1, with [Table 258](#) indicating their related content and presentation requirements, and [Table 259](#) the corresponding evaluation actions.

Table 257 — Developer action elements for ATE_FUN.1

ATE_FUN.1.1D	The developer shall test the TSF and document the results.
ATE_FUN.1.2D	The developer shall provide test documentation.

Table 258 — Content and presentation elements for ATE_FUN.1

ATE_FUN.1.1C	The test documentation shall consist of test plans, expected test results and actual test results.
ATE_FUN.1.2C	The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.
ATE_FUN.1.3C	The expected test results shall show the anticipated outputs from a successful execution of the tests.
ATE_FUN.1.4C	The actual test results shall be consistent with the expected test results.

Table 259 — Evaluator action elements for ATE_FUN.1

ATE_FUN.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	---

ISO/IEC 15408-3:2026(en)

13.4.5 Ordered functional testing (ATE_FUN.2)

13.4.5.1 Component relationships

Dependencies: ATE_COV.1 (Evidence of coverage) ([13.2.3](#))

13.4.5.2 Objectives

The objectives are for the developer to demonstrate that the tests in the test documentation are performed and documented correctly, and to ensure that testing is structured such as to avoid circular arguments about the correctness of the interfaces being tested.

13.4.5.3 Application notes

Although the test procedures can state pre-requisite initial test conditions in terms of ordering of tests, they will not necessarily provide a rationale for the ordering. An analysis of test ordering is an important factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the ordering of tests.

13.4.5.4 Component elements

[Table 260](#) specifies the developer action elements for ATE_FUN.2, with [Table 261](#) indicating their related content and presentation requirements, and [Table 262](#) the corresponding evaluation actions.

Table 260 — Developer action elements for ATE_FUN.2

ATE_FUN.2.1D	The developer shall test the TSF and document the results.
ATE_FUN.2.2D	The developer shall provide test documentation.

Table 261 — Content and presentation elements for ATE_FUN.2

ATE_FUN.2.1C	The test documentation shall consist of test plans, expected test results and actual test results.
ATE_FUN.2.2C	The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.
ATE_FUN.2.3C	The expected test results shall show the anticipated outputs from a successful execution of the tests.
ATE_FUN.2.4C	The actual test results shall be consistent with the expected test results.
ATE_FUN.2.5C	The test documentation shall include an analysis of the test procedure ordering dependencies.

Table 262 — Evaluator action elements for ATE_FUN.2

ATE_FUN.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
--------------	--

13.5 Independent testing (ATE_IND)

13.5.1 Objectives

The objectives of this family are built upon the assurances achieved in the following families:

- ATE_FUN (Functional tests) ([13.4](#)),
- ATE_COV (Coverage) ([13.2](#)),

— ATE_DPT (Depth) ([13.3](#)).

In ATE_IND (Independent testing) the developer testing is verified by additional tests performed by the evaluator.

13.5.2 Component levelling

Levelling is based upon the amount of developer test documentation and test support and the amount of evaluator testing.

13.5.3 Application notes

This family deals with the degree to which there is independent functional testing of the TSF. Independent functional testing may take the form of repeating the developer's functional tests (in whole or in part) or of extending the scope or the depth of the developer's tests. These activities are complementary, and an appropriate mix shall be planned for each TOE, which takes into account the availability and coverage of test results, and the functional complexity of the TSF.

Sampling of developer tests is intended to provide confirmation that the developer has carried out their planned test programme on the TSF and has correctly recorded the results. The size of sample selected will be influenced by the detail and quality of the developer's functional test results. The evaluator shall also consider the scope for devising additional tests, and the relative benefit that can be gained from effort in these two areas. It is recognized that repetition of all developer tests can be feasible and desirable in some cases, but can be very arduous and less productive in others. The highest component in this family should therefore be used with caution. Sampling shall address the whole range of test results available, including those supplied to meet the requirements of both ATE_COV (Coverage) ([13.2](#)) and ATE_DPT (Depth) ([13.3](#)).

There is also a requirement to consider the different configurations of the TOE that shall be included within the evaluation. The evaluator shall assess the applicability of the results provided, and plan the testing accordingly.

The suitability of the TOE for testing is based on the access to the TOE, and the supporting documentation and information required (including any test software or tools) to run tests. The need for such support is addressed by the dependencies to other assurance families.

Additionally, suitability of the TOE for testing can be based on other considerations. For example, it is possible that the version of the TOE submitted by the developer is not the final version.

The term interfaces refers to interfaces described in the functional specification and TOE design, and parameters passed through invocations identified in the implementation representation. The exact set of interfaces to be used is selected through ATE_COV (Coverage) ([13.2](#)) and the ATE_DPT (Depth) ([13.3](#)) components.

References to a subset of the interfaces are intended to allow the evaluator to design an appropriate set of tests which is consistent with the objectives of the evaluation being conducted.

13.5.4 Independent testing - conformance (ATE_IND.1)

13.5.4.1 Component relationships

Dependencies: ADV_FSP.1 (Basic functional specification) ([10.3.4](#))
 AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
 AGD_PRE.1 (Preparative procedures) ([11.3.4](#))

13.5.4.2 Objectives

In this component, the objective is to demonstrate that the TOE operates in accordance with its design representations and guidance documents.

ISO/IEC 15408-3:2026(en)

13.5.4.3 Application notes

This component does not address the use of developer test results. It is applicable where such results are not available, and where the developer's testing is accepted without validation. The evaluator is required to devise and conduct tests with the objective of confirming that the TOE operates in accordance with its design representations, including but not limited to the functional specification. The approach is to gain confidence in correct operation through representative testing, rather than to conduct every possible test. The extent of testing to be planned for this purpose is a methodology issue, and shall be considered in the context of a particular TOE and the balance of other evaluation activities.

13.5.4.4 Component elements

[Table 263](#) specifies the developer action elements for ATE_IND.1, with [Table 264](#) indicating their related content and presentation requirements, and [Table 265](#) the corresponding evaluation actions.

Table 263 — Developer action elements for ATE_IND.1

ATE_IND.1.1D	The developer shall provide the TOE for testing.
--------------	--

Table 264 — Content and presentation elements for ATE_IND.1

ATE_IND.1.1C	The TOE shall be suitable for testing.
--------------	--

Table 265 — Evaluator action elements for ATE_IND.1

ATE_IND.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
ATE_IND.1.2E	The evaluator <i>shall test</i> a subset of the TSF to confirm that the TSF operates as specified.

13.5.5 Independent testing - sample (ATE_IND.2)

13.5.5.1 Component relationships

Dependencies: ADV_FSP.2 (Security-enforcing functional specification) ([10.3.5](#))
 AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
 AGD_PRE.1 (Preparative procedures) ([11.3.4](#))
 ATE_COV.1 (Evidence of coverage) ([13.2.3](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.5.5.2 Objectives

In this component, the objective is to demonstrate that the TOE operates in accordance with its design representations and guidance documents. Evaluator testing confirms that the developer performed some tests of some interfaces in the functional specification.

13.5.5.3 Application notes

The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This can include such things as machine-readable test documentation and test programs.

This component contains a requirement that the evaluator has available test results from the developer to supplement the programme of testing. The evaluator shall repeat a sample of the developer's tests to gain confidence in the results obtained. Having established such confidence the evaluator shall build upon the developer's testing by conducting additional tests that exercise the TOE in a different manner. By using a platform of validated developer test results, the evaluator can gain confidence that the TOE operates correctly in a wider range of conditions than would be possible purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that the developer has tested the TOE, the evaluator

ISO/IEC 15408-3:2026(en)

will also have more freedom, where appropriate, to concentrate testing in areas where examination of documentation or specialist knowledge has raised concerns.

13.5.5.4 Component elements

[Table 266](#) specifies the developer action elements for ATE_IND.2, with [Table 267](#) indicating their related content and presentation requirements, and [Table 268](#) the corresponding evaluation actions.

Table 266 — Developer action elements for ATE_IND.2

ATE_IND.2.1D	The developer shall provide the TOE for testing.
--------------	--

Table 267 — Content and presentation elements for ATE_IND.2

ATE_IND.2.1C	The TOE shall be suitable for testing.
ATE_IND.2.2C	The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Table 268 — Evaluator action elements for ATE_IND.2

ATE_IND.2.1E	The evaluator shall confirm that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
ATE_IND.2.2E	The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.
ATE_IND.2.3E	The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

13.5.6 Independent testing - complete (ATE_IND.3)

13.5.6.1 Component relationships

Dependencies: ADV_FSP.4 (Complete functional specification) ([10.3.7](#))
 AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
 AGD_PRE.1 (Preparative procedures) ([11.3.4](#))
 ATE_COV.1 (Evidence of coverage) ([13.2.3](#))
 ATE_FUN.1 (Functional testing) ([13.4.4](#))

13.5.6.2 Objectives

In this component, the objective is to demonstrate that the TOE operates in accordance with its design representations and guidance documents. Evaluator testing includes repeating all developer tests.

13.5.6.3 Application notes

The intent is that the developer should provide the evaluator with materials necessary for the efficient reproduction of developer tests. This may include machine-readable test documentation and test programs.

In this component the evaluator shall repeat all the developer's tests as part of the programme of testing. As in the previous component, the evaluator shall also conduct tests that aim to exercise the TSF in a different manner from that achieved by the developer. In cases where developer testing has been exhaustive, there can remain little scope for this.

13.5.6.4 Component elements

[Table 269](#) specifies the developer action elements for ATE_IND.3, with [Table 270](#) indicating their related content and presentation requirements, and [Table 271](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 269 — Developer action elements for ATE_IND.3

ATE_IND.3.1D	The developer shall provide the TOE for testing.
--------------	--

Table 270 — Content and presentation elements for ATE_IND.3

ATE_IND.3.1C	The TOE shall be suitable for testing.
ATE_IND.3.2C	The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Table 271 — Evaluator action elements for ATE_IND.3

ATE_IND.3.1E	The evaluator shall confirm that the information provided meets all ISO/IEC 18045 and ISO/IEC 15408 series requirements for content and presentation of evidence.
ATE_IND.3.2E	The evaluator shall execute all tests in the test documentation to verify the developer test results.
ATE_IND.3.3E	The evaluator shall test the TSF to confirm that the entire TSF operates as specified.

13.6 Composite functional testing (ATE_COMP)

13.6.1 Objectives

The aim of this family is to determine whether the composite product as a whole exhibits the properties necessary to satisfy the functional requirements of its composite product ST.

13.6.2 Component levelling

This family contains only one component.

13.6.3 Application notes

A composite product can be tested by testing its components separately and by testing the integrated product. Separate testing means that the base component and dependent component are tested independently of each other. A lot of tests of the base component can have been performed within the scope of its accomplished evaluation. The dependent component can be tested on a simulator or an emulator, which represent a virtual machine.

Integration testing means that the composite product is being tested as it is, i.e. the dependent component is running together with the related base component.

Some dependent component functionality testing can only be performed on emulators, before its embedding/integration onto the base component, as it is possible that the effectiveness of this testing is not visible using the interfaces of the composite product. Nevertheless, functional testing of the composite product shall be performed also on composite product samples according to the description of the security functions of the composite product and using the standard approach as required by the relevant ATE (Tests) assurance class. No additional developer's action is required here.

Since the amount, the coverage and the depth of the functional tests of the base component have already been validated by the base component evaluation, it is not necessary to re-perform these tasks in the composite evaluation. The ETR for composite evaluation does not provide any information on functional testing for the base component.

The behaviour of implementation of some SFRs can depend on properties of the base component as well as on the dependent component (e.g. correctness of the measures of the composite product to withstand a side channel attack or correctness of the implementation of tamper resistance against physical attacks). In such case the SFR implementation shall be tested on the final composite product, but not on a simulator or an emulator.

ISO/IEC 15408-3:2026(en)

This family focuses exclusively on testing of the composite product as a whole and represents merely partial efforts within the general test approach being covered by the assurance ATE (Tests). These integration tests shall be specified and performed, whereby the approach of the standard assurance families of ATE (Tests) shall be applied.

The composite product evaluation sponsor shall ensure that the composite product samples suitable for testing are available for the composite product evaluator.

13.6.4 Composite product functional testing (ATE_COMP.1)

[Table 272](#) specifies the developer action elements for ATE_COMP.1, with [Table 273](#) indicating their related content and presentation requirements, and [Table 274](#) the corresponding evaluation actions.

Table 272 — Developer action elements for ATE_COMP.1

ATE_COMP.1.1D	The developer shall provide a set of tests as required by the assurance package chosen.
ATE_COMP.1.2D	The developer shall provide the composite product for testing.

Table 273 — Content and presentation elements for ATE_COMP.1

ATE_COMP.1.1C	Content and presentation of the specification and documentation of the integration tests shall correspond to the requirements of the assurance families ATE_FUN (Functional tests) (13.4) and ATE_COV (Coverage) (13.2).
ATE_COMP.1.2C	The composite product provided shall be suitable for testing.

Table 274 — Evaluator action elements for ATE_COMP.1

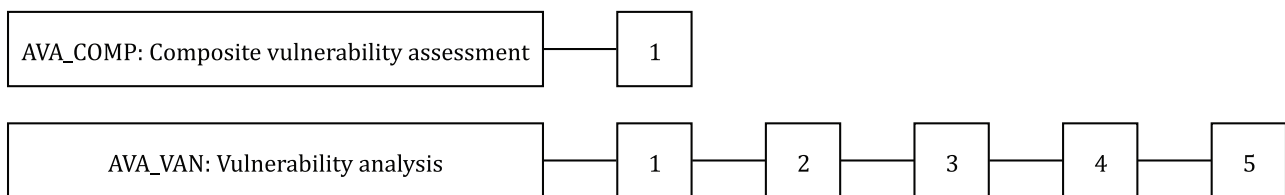
ATE_COMP.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all ISO/IEC 15408 series and ISO/IEC 15408 series requirements for content and presentation of evidence.
---------------	--

14 Class AVA Vulnerability assessment

14.1 General

AVA (Vulnerability assessment) addresses the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE.

[Figure 13](#) describes the family hierarchy of class AVA.



Key

1,2,3,4,5 components

Figure 13 — AVA: Vulnerability assessment, class decomposition

Direct, indirect and optional dependencies for the components of class AVA are shown in [Table 275](#).

ISO/IEC 15408-3:2026(en)

Table 275 — Dependency table for class AVA: Vulnerability assessment

	ADV_ARC.1 (10.2.4)	ADV_FSP.1 (10.3.4)	ADV_FSP.2 (10.3.5)	ADV_FSP.4 (10.3.7)	ADV_IMP.1 (10.4.4)	ADV_TDS.1 (10.7.4)	ADV_TDS.3 (10.7.6)	AGD_OPE.1 (11.2.4)	AGD_PRE.1 (11.3.4)	ATE_DPT.1 (13.3.4)
AVA_COMP.1 (14.4.4)										
AVA_VAN.1 (14.3.3)		X						X	X	
AVA_VAN.2 (14.3.4)	X		X			X		X	X	
AVA_VAN.3 (14.3.5)	X			X	X		X	X	X	X
AVA_VAN.4 (14.3.6)	X			X	X		X	X	X	X
AVA_VAN.5 (14.3.7)	X			X	X		X	X	X	X

14.2 Application notes

Generally, the vulnerability assessment activity covers various vulnerabilities in the development and operation of the TOE. Development vulnerabilities take advantage of some property of the TOE, or the product where the TOE resides, which was introduced during its development, e.g. defeating the TSF self-protection through tampering, direct attack or monitoring of the TSF, defeating the TSF domain separation through monitoring or direct attack the TSF, or defeating non-bypassability through circumventing (bypassing) the TSF. Explicit dependencies of the TOE on IT systems in the environment shall also be considered. Operational vulnerabilities take advantage of weaknesses in non-technical countermeasures to violate the TOE SFRs, e.g. misuse or incorrect configuration. Misuse investigates whether the TOE can be configured or used in a manner that is insecure, but that an administrator or user of the TOE would reasonably believe to be secure.

Assessment of development vulnerabilities is covered by the assurance family AVA_VAN (Vulnerability analysis) (14.3). All development vulnerabilities can be considered in the context of AVA_VAN (Vulnerability analysis) (14.3) because this family allows the application of a wide range of assessment methodologies which are unspecific to the kind of attack scenario. These unspecific assessment methodologies include the specific methodologies for those TSF where covert channels shall be considered (a channel capacity estimation can be done using informal engineering measurements, as well as actual test measurements) or can be overcome by the use of sufficient resources in the form of a direct attack.

NOTE The underlying technical concept of those TSF is based on probabilistic or permutational mechanisms; a qualification of their security behaviour and the effort required to overcome them can be made using a quantitative or statistical analysis.

If there are security objectives specified in the ST to either to prevent one user of the TOE from observing activity associated with another user of the TOE, or to ensure that information flows cannot be used to achieve enforced illicit data signals, covert channel analysis shall be considered during the conduct of the vulnerability analysis. This is often reflected by the inclusion of FPR_UNO (Unobservability) (see ISO/IEC 15408-2:2026, 14.6) and multilevel access control policies specified through either FDP_ACC (Access control policy) (see ISO/IEC 15408-2:2026, 11.3) or FDP_IFC (Information flow control policy) (see ISO/IEC 15408-2:2026, 11.7), or both, requirements in the ST.

14.3 Vulnerability analysis (AVA_VAN)

14.3.1 Objectives

Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified, during the evaluation of the development and anticipated operation of the TOE or by other methods (e.g. by flaw hypotheses or quantitative or statistical analysis of the security behaviour of the underlying security mechanisms), can allow attackers to violate the SFRs.

Vulnerability analysis deals with the threats that an attacker will be able to discover flaws that will allow unauthorized access to data and functionality, allow the ability to interfere with or alter the TSF, or interfere with the authorized capabilities of other users.

ISO/IEC 15408-3:2026(en)

In case of a multi-assurance evaluation the vulnerability analysis shall assess the defined sub-TSF as well as the TOE as a whole.

NOTE This Family uses attack potentials defined in ISO/IEC 18045.

14.3.2 Component levelling

Levelling is based on an increasing rigour of vulnerability analysis by the evaluator and increased levels of attack potential required by an attacker to identify and exploit the potential vulnerabilities.

14.3.3 Vulnerability survey (AVA_VAN.1)

14.3.3.1 Component relationships

Dependencies: ADV_FSP.1 (Basic functional specification) ([10.3.4](#))
 AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
 AGD_PRE.1 (Preparative procedures) ([11.3.4](#))

14.3.3.2 Objectives

A vulnerability survey of information available in the public domain is performed by the evaluator to ascertain potential vulnerabilities that can be easily found by an attacker.

The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE. Penetration testing is performed by the evaluator assuming an attack potential of Basic.

14.3.3.3 Component elements

[Table 276](#) specifies the developer action elements for AVA_VAN.1, with [Table 277](#) indicating their related content and presentation requirements, and [Table 278](#) the corresponding evaluation actions.

Table 276 — Developer action elements for AVA_VAN.1

AVA_VAN.1.1D	The developer shall provide the TOE for testing.
--------------	--

Table 277 — Content and presentation elements for AVA_VAN.1

AVA_VAN.1.1C	The TOE shall be suitable for testing.
--------------	--

Table 278 — Evaluator action elements for AVA_VAN.1

AVA_VAN.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.1.2E	The evaluator <i>shall perform</i> a search of public domain sources to identify potential vulnerabilities in the TOE.
AVA_VAN.1.3E	The evaluator <i>shall conduct</i> penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

14.3.4 Vulnerability analysis (AVA_VAN.2)

14.3.4.1 Component relationships

Dependencies: ADV_ARC.1 (Security architecture description) (10.2.4)
 ADV_FSP.2 (Security-enforcing functional specification) (10.3.5)
 ADV_TDS.1 (Basic design) (10.7.4)
 AGD_OPE.1 (Operational user guidance) (11.2.4)
 AGD_PRE.1 (Preparative procedures) (11.3.4)

14.3.4.2 Objectives

A vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities.

The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE. Penetration testing is performed by the evaluator assuming an attack potential of Basic.

14.3.4.3 Component elements

Table 279 specifies the developer action elements for AVA_VAN.2, with Table 280 indicating their related content and presentation requirements, and Table 281 the corresponding evaluation actions.

Table 279 — Developer action elements for AVA_VAN.2

AVA_VAN.2.1D	The developer shall provide the TOE for testing.
AVA_VAN.2.2D	The developer shall provide a list of third-party components included in the TOE and the TOE delivery.

Table 280 — Content and presentation elements for AVA_VAN.2

AVA_VAN.2.1C	The TOE shall be suitable for testing.
AVA_VAN.2.2C	The list of third-party components shall include components provided by third parties, and that are part of the TOE or otherwise part of the TOE delivery.

Table 281 — Evaluator action elements for AVA_VAN.2

AVA_VAN.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.2.2E	The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE, the components in the list of third-party components, and specific IT products in the environment on which the TOE depends.
AVA_VAN.2.3E	The evaluator <i>shall perform an independent vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design and security architecture description</i> to identify potential vulnerabilities in the TOE.
AVA_VAN.2.4E	The evaluator <i>shall conduct</i> penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

ISO/IEC 15408-3:2026(en)

14.3.5 Focused vulnerability analysis (AVA_VAN.3)

14.3.5.1 Component relationships

Dependencies:

- ADV_ARC.1 (Security architecture description) ([10.2.4](#))
- ADV_FSP.4 (Complete functional specification) ([10.3.7](#))
- ADV_TDS.3 (Basic modular design) ([10.7.6](#))
- ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
- AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
- AGD_PRE.1 (Preparative procedures) ([11.3.4](#))
- ATE_DPT.1 (Testing: basic design) ([13.3.4](#))

14.3.5.2 Objectives

A vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities.

The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE. Penetration testing is performed by the evaluator assuming an attack potential of Enhanced-Basic.

14.3.5.3 Component elements

[Table 282](#) specifies the developer action elements for AVA_VAN.3, with [Table 283](#) indicating their related content and presentation requirements, and [Table 284](#) the corresponding evaluation actions.

Table 282 — Developer action elements for AVA_VAN.3

AVA_VAN.3.1D	The developer shall provide the TOE for testing.
AVA_VAN.3.2D	The developer shall provide a list of third-party components included in the TOE and the TOE delivery.

Table 283 — Content and presentation elements for AVA_VAN.3

AVA_VAN.3.1C	The TOE shall be suitable for testing.
AVA_VAN.3.2C	The list of third-party components shall include components provided by third parties, and that are part of the TOE or otherwise part of the TOE delivery.

Table 284 — Evaluator action elements for AVA_VAN.3

AVA_VAN.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.3.2E	The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE, the components in the list of third-party components, and specific IT products in the environment that the TOE depends on .
AVA_VAN.3.3E	The evaluator shall perform an independent, focused vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design, security architecture description and implementation representation to identify potential vulnerabilities in the TOE.
AVA_VAN.3.4E	The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Enhanced-Basic attack potential.

ISO/IEC 15408-3:2026(en)

14.3.6 Methodical vulnerability analysis (AVA_VAN.4)

14.3.6.1 Component relationships

Dependencies:

- ADV_ARC.1 (Security architecture description) ([10.2.4](#))
- ADV_FSP.4 (Complete functional specification) ([10.3.7](#))
- ADV_TDS.3 (Basic modular design) ([10.7.6](#))
- ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
- AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
- AGD_PRE.1 (Preparative procedures) ([11.3.4](#))
- ATE_DPT.1 (Testing: basic design) ([13.3.4](#))

14.3.6.2 Objectives

A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities.

The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE. Penetration testing is performed by the evaluator assuming an attack potential of Moderate.

14.3.6.3 Component elements

[Table 285](#) specifies the developer action elements for AVA_VAN.4, with [Table 286](#) indicating their related content and presentation requirements, and [Table 287](#) the corresponding evaluation actions.

Table 285 — Developer action elements for AVA_VAN.4

AVA_VAN.4.1D	The developer shall provide the TOE for testing.
AVA_VAN.4.2D	The developer shall provide a list of third-party components included in the TOE and the TOE delivery.

Table 286 — Content and presentation elements for AVA_VAN.4

AVA_VAN.4.1C	The TOE shall be suitable for testing.
AVA_VAN.4.2C	The list of third-party components shall include components provided by third parties, and that are part of the TOE or otherwise part of the TOE delivery.

Table 287 — Evaluator action elements for AVA_VAN.4

AVA_VAN.4.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.4.2E	The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE, the components in the list of third-party components, and specific IT products in the environment that the TOE depends on.
AVA_VAN.4.3E	The evaluator shall perform an independent, methodical vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design, security architecture description and implementation representation to identify potential vulnerabilities in the TOE.
AVA_VAN.4.4E	The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to determine that the TOE is resistant to attacks performed by an attacker possessing Moderate attack potential.

ISO/IEC 15408-3:2026(en)

14.3.7 Advanced methodical vulnerability analysis (AVA_VAN.5)

14.3.7.1 Component relationships

Dependencies: ADV_ARC.1 (Security architecture description) ([10.2.4](#))
 ADV_FSP.4 (Complete functional specification) ([10.3.7](#))
 ADV_TDS.3 (Basic modular design) ([10.7.6](#))
 ADV_IMP.1 (Implementation representation of the TSF) ([10.4.4](#))
 AGD_OPE.1 (Operational user guidance) ([11.2.4](#))
 AGD_PRE.1 (Preparative procedures) ([11.3.4](#))
 ATE_DPT.1 (Testing: basic design) ([13.3.4](#))

14.3.7.2 Objectives

A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities.

The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE. Penetration testing is performed by the evaluator assuming an attack potential of High.

14.3.7.3 Component elements

[Table 288](#) specifies the developer action elements for AVA_VAN.5, with [Table 289](#) indicating their related content and presentation requirements, and [Table 290](#) the corresponding evaluation actions.

Table 288 — Developer action elements for AVA_VAN.5

AVA_VAN.5.1D	The developer shall provide the TOE for testing.
AVA_VAN.5.2D	The developer shall provide a list of third-party components included in the TOE and the TOE delivery.

Table 289 — Content and presentation elements for AVA_VAN.5

AVA_VAN.5.1C	The TOE shall be suitable for testing.
AVA_VAN.5.2C	The list of third-party components shall include components provided by third parties, and that are part of the TOE or otherwise part of the TOE delivery.

Table 290 — Evaluator action elements for AVA_VAN.5

AVA_VAN.5.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
AVA_VAN.5.2E	The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE, the components in the list of third-party components, and specific IT products in the environment that the TOE depends on.
AVA_VAN.5.3E	The evaluator shall perform an independent, methodical vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design, security architecture description and implementation representation to identify potential vulnerabilities in the TOE.
AVA_VAN.5.4E	The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to determine that the TOE is resistant to attacks performed by an attacker possessing High attack potential.

14.4 Composite vulnerability assessment (AVA_COMP)

14.4.1 Objectives

The aim of this family is to determine the exploitability of flaws or weaknesses in the composite product as a whole in the intended environment.

14.4.2 Component levelling

This family contains only one component.

14.4.3 Application notes

This family focuses exclusively on the vulnerability assessment of the composite product as a whole and represents merely partial efforts within the general approach being covered by the standard assurance family AVA_VAN (Vulnerability analysis) ([14.3](#)).

The composite product evaluator shall perform a vulnerability analysis for the composite product using elements such as the results of the base component evaluation. This vulnerability analysis shall be confirmed by penetration testing.

The composite product evaluator shall check that the confidentiality protection of the dependent component embedded into, or installed onto, the base component is consistent with the confidentiality level claimed by the dependent component developer for ALC_DVS (Developer environment security) ([12.5](#)).

In special cases, the vulnerability analysis and the definition of attacks can be difficult, need considerable time and require extensive pre-testing, if only documentation is available. The base component can also be used in a way that was not foreseen by the base component developer and the base component evaluator, or it is possible that the dependent component developer has not followed the stipulations provided with the base component. Different possibilities exist to shorten composite product vulnerability analysis in such cases: e.g. the composite product evaluator can consult the base component evaluator and draw on their experience gained during the base component evaluation. Alternatively, an approach based upon the separation of vulnerabilities of the dependent component and the base component by using specific test samples of the base component on which the composite product evaluator can load test dependent components. The intention is to use test dependent components without countermeasures and without deactivating any base component inherent countermeasure.

The results of the vulnerability assessment for the base component of the composite product represented in the ETR for composite evaluation can be re-used if they are up-to-date and all composite activities for correctness are finalized with the verdict PASS.

The composite activities for correctness are:

- ASE_COMP.1 (Consistency of Security Target (ST)) ([9.9.4](#)),
- ALC_COMP.1 (Integration of the dependent component into the related base component and Consistency check for delivery and acceptance procedures) ([12.10.4](#)),
- ADV_COMP.1 (Design compliance with the base component-related user guidance, ETR for composite evaluation and report of the base component evaluation authority) ([10.8.4](#)), and
- ATE_COMP.1 (Composite product functional testing) ([13.6.4](#)).

Due to the composing of the base component and the dependent component a new quality arises, which can cause additional vulnerabilities of the base component which is possibly not mentioned in the ETR for composite evaluation. In these circumstances, the composite product evaluation authority can require a re-assessment or re-evaluation of the base component focusing on the new vulnerabilities' issues.

ISO/IEC 15408-3:2026(en)

The composite product evaluation sponsor shall ensure that the following is made available for the composite product evaluator:

- the base component-related user guidance,
- the base component-related ETR for composite evaluation prepared by the base component evaluator,
- the report of the base component evaluation authority.

14.4.4 Composite product vulnerability assessment (AVA_COMP.1)

[Table 291](#) specifies the developer action elements for AVA_COMP.1, with [Table 292](#) indicating their related content and presentation requirements, and [Table 293](#) the corresponding evaluation actions.

Table 291 — Developer action elements for AVA_COMP.1

AVA_COMP.1.1D	The developer shall provide the composite product for penetration testing.
---------------	--

Table 292 — Content and presentation elements for AVA_COMP.1

AVA_COMP.1.1C	The composite product provided shall be suitable for testing as a whole.
---------------	--

Table 293 — Evaluator action elements for AVA_COMP.1

AVA_COMP.1.1E	The evaluator <i>shall conduct</i> penetration testing of the composite product as a whole building on the evaluator's own vulnerability analysis to ensure that the vulnerabilities being relevant for the composite product Security Target are not exploitable.
---------------	--

15 Class ACO Composition

15.1 General

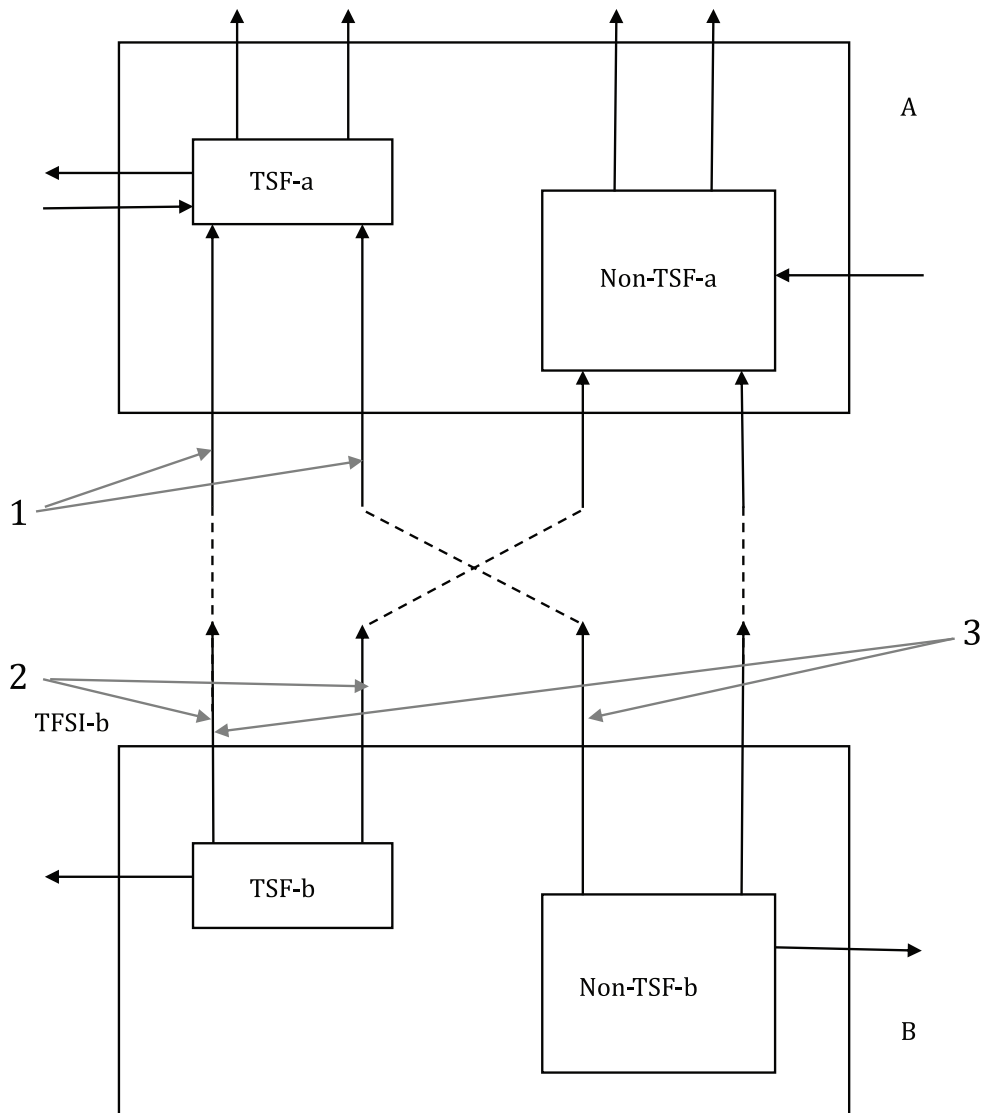
ACO (Composition) encompasses five families. These families specify assurance requirements that are designed to provide confidence that a composed TOE shall operate securely when relying upon security functionality provided by previously evaluated software, firmware or hardware components.

Composition involves taking two or more IT entities successfully evaluated against the security assurance requirements packages specified in the ISO/IEC 15408 series (base components and dependent components, see [Annex B](#)) and combining them for use, with no further development of either IT entity. The development of additional IT entities is not included (entities that have not previously been the subject of a component evaluation). The composed TOE forms a new product that can be installed and integrated into any specific environment instance that meets the objectives for the operational environment.

This approach does not provide an alternative approach for the evaluation of components. under ACO (Composition) provides a composed TOE integrator a method, which can be used as an alternative to other assurance levels specified in the ISO/IEC 15408 series, to gain confidence in a TOE that is the combination of two or more successfully evaluated components without having to re-evaluate the composite TSF. The composed TOE integrator is referred to as "developer" throughout ACO (Composition), with any references to the developer of the base or dependent components clarified as such.

CAPs, as defined in ISO/IEC 15408-5, provide an assurance scale for composed TOEs. This assurance scale is required in addition to other assurance packages, for example the EALs, because to combine components evaluated against another assurance package and gain equivalent assurance in the resulting composed TOE, all SARs shall be applied to the composed TOE. Although reuse can be made of the component TOE evaluation results, there are often additional aspects of the components that shall be considered in the composed TOE, as described in [B.3](#). Due to the different parties involved in a composed TOE evaluation activity, it is generally not possible to gain all necessary evidence about these additional aspects of the components to apply the appropriate EAL. Hence, CAPs have been defined to address the issue of combining evaluated components and gaining a meaningful result. This is discussed further in [Annex B](#).

ISO/IEC 15408-3:2026(en)



Key

- A dependent component-a
- B base component-b
- 1 ACO_REL (Reliance of dependent component) (15.4) (component-a)
- 2 ADV_FSP (Functional specification) (10.3) (component-b)
- 3 ACO_DEV (Development evidence) (15.3) (component-b)

Figure 14 — Relationship between ACO families and interactions between components

In a composed TOE it is generally the case that one component relies on the services provided by another component as shown in Figure 14. The component requiring services is termed the dependent component and the component providing the services is termed the base component. This interaction and distinct is discussed further in Annex B. It is assumed to be the case that the developer of the dependent component is supporting the composed TOE evaluation in some manner (as developer, sponsor, or just cooperating and providing the necessary evaluation evidence from the dependent component evaluation). The ACO (Composition) components included in the CAP assurance packages should not be used as augmentations for component TOE evaluations, as this would provide no meaningful assurance for the component.

The families within ACO (Composition) interact in a similar manner to class ADV (Development) (Clause 10), class ATE (Tests) (Clause 13) and class AVA (Vulnerability assessment) (Clause 14) in a component TOE evaluation and hence leverage from the specification of requirements from those classes where applicable. There are however a few items specific to composed TOE evaluations. To determine how the components interact and identify any deviations from the evaluations of the components, the dependencies that the

ISO/IEC 15408-3:2026(en)

dependent component has upon the underlying base component are identified in ACO_REL (Reliance of dependent component) (15.4). This reliance on the base component is specified in terms of the interfaces through which the dependent component makes calls for services in support of the dependent component SFRs. The interfaces, and at higher levels the supporting behaviour, provided by the base component in response to those service requests are analysed in ACO_DEV (Development evidence) (15.3). The ACO_DEV (Development evidence) (15.3) family is based on the ADV_TDS (TOE design) (10.7) family, as at the simplest level the TSF of each component can be viewed as a subsystem of the composed TOE, with additional portions of each component seen as additional subsystems. Therefore, the interfaces between the components are seen as interactions between subsystems in a component TOE evaluation.

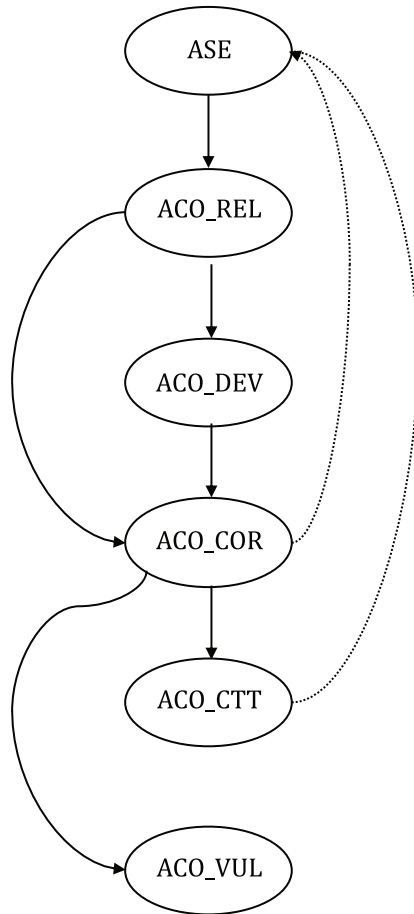
It is possible that the interfaces and supporting behaviour descriptions provided for ACO_DEV (Development evidence) (15.3) are incomplete. This is determined during the conduct of ACO_COR (Composition rationale) (15.2). The ACO_COR (Composition rationale) (15.2) family takes the outputs of ACO_REL (Reliance of dependent component) (15.4) and ACO_DEV (Development evidence) (15.3) and determines whether the components are being used in their evaluated configuration and identifies where any specifications are incomplete, which are then identified as inputs into testing (ACO_CTT (Composed TOE testing) (15.5)) and vulnerability analysis (ACO_VUL (Composition vulnerability analysis) (15.6)) activities of the composed TOE.

Testing of the composed TOE is performed to determine that the composed TOE exhibits the expected behaviour as determined by the composed TOE SFRs. At higher levels, this testing demonstrates the compatibility of the interfaces between the components of the composed TOE.

The vulnerability analysis of the composed TOE leverages from the outputs of the vulnerability analysis of the component evaluations. The composed TOE vulnerability analysis considers any residual vulnerabilities from the component evaluations to determine that the residual vulnerabilities are not applicable to the composed TOE. A search of publicly available information relating to the components is also performed to identify any issues reported in the components since the completion of the respective evaluations.

The interaction between the ACO (Composition) families is depicted in [Figure 15](#). Solid arrowed lines indicate where the evidence and understanding gained in one family feeds into the next activity and the dashed arrows show where an activity explicitly traces back to the composed TOE SFRs, as described above.

ISO/IEC 15408-3:2026(en)



Key

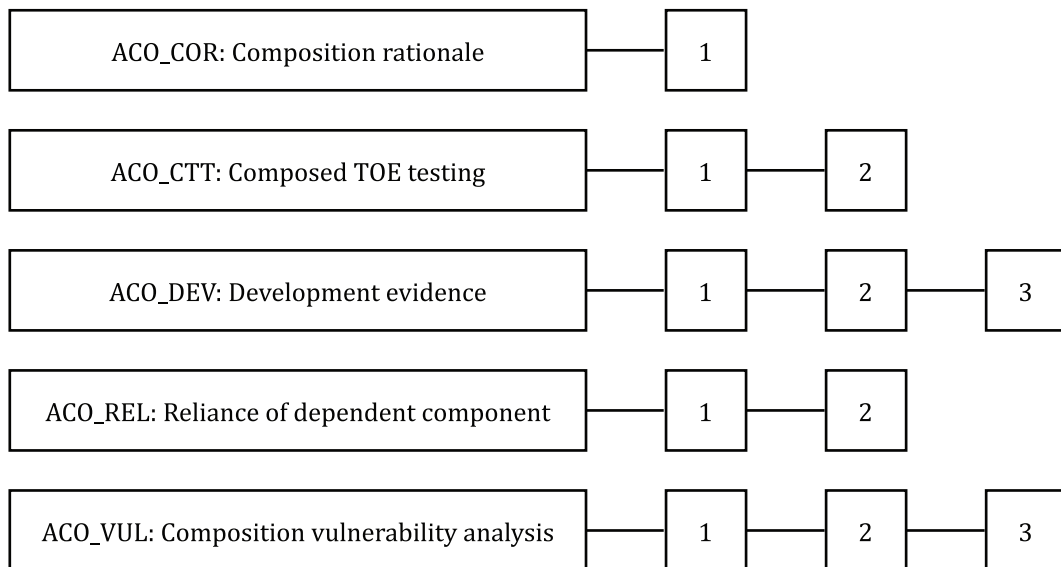
- solid line evidence and understanding
- dashed line activity traces back to the composed TOE SFRs

Figure 15 — Relationship between ACO families

Further discussion of the definition and interactions within composed TOEs is provided in [Annex B](#).

[Figure 16](#) describes the family hierarchy of class ACO.

ISO/IEC 15408-3:2026(en)



Key
 1,2,3 components

Figure 16 — ACO: Composition, class decomposition

Direct, indirect and optional dependencies for the components of class ACO are shown in [Table 294](#).

Table 294 — Dependency table for class ACO: Composition

	ACO_DEV.1 (15.3.4)	ACO_DEV.2 (15.3.5)	ACO_DEV.3 (15.3.6)	ACO_REL.1 (15.4.4)	ACO_REL.2 (15.4.5)	ALC_CMC.1 (12.2.4)
ACO_COR.1 (15.2.3)	X			X		X
ACO_CTT.1 (15.5.4)	X			X		
ACO_CTT.2 (15.5.5)		X		-	X	
ACO_DEV.1 (15.3.4)				X		
ACO_DEV.2 (15.3.5)				X		
ACO_DEV.3 (15.3.6)					X	
ACO_REL.1 (15.4.4)						
ACO_REL.2 (15.4.5)						
ACO_VUL.1 (15.6.4)	X			-		
ACO_VUL.2 (15.6.5)		X		-		
ACO_VUL.3 (15.6.6)			X		-	

15.2 Composition rationale (ACO_COR)

15.2.1 Objectives

This family addresses the requirement to demonstrate that the base component can provide an appropriate level of assurance for use in composition.

15.2.2 Component levelling

There is only a single component in this family.

ISO/IEC 15408-3:2026(en)

15.2.3 Composition rationale (ACO_COR.1)

15.2.3.1 Component relationships

Dependencies: ACO_DEV.1 (Functional description) ([15.3.4](#))
ALC_CMC.1 (Labelling of the TOE) ([12.2.4](#))
ACO_REL.1 (Basic reliance information) ([15.4.4](#))

15.2.3.2 Component elements

[Table 295](#) specifies the developer action elements for ACO_COR.1, with [Table 296](#) indicating their related content and presentation requirements, and [Table 297](#) the corresponding evaluation actions.

Table 295 — Developer action elements for ACO_COR.1

ACO_COR.1.1D	The developer shall provide composition rationale for the base component.
--------------	---

Table 296 — Content and presentation elements for ACO_COR.1

ACO_COR.1.1C	The composition rationale shall demonstrate that a level of assurance at least as high as that of the dependent component has been obtained for the support functionality of the base component, when the base component is configured as required to support the TSF of the dependent component.
--------------	---

Table 297 — Evaluator action elements for ACO_COR.1

ACO_COR.1.1E	The evaluator <i>shall confirm</i> that the information meets all requirements for content and presentation of evidence.
--------------	--

15.3 Development evidence (ACO_DEV)

15.3.1 Objectives

This family sets out requirements for a specification of the base component in increasing levels of detail. Such information is required to gain confidence that the appropriate security functionality is provided to support the requirements of the dependent component (as identified in the reliance information).

15.3.2 Component levelling

The components are levelled on the basis of increasing amounts of detail about the interfaces provided, and how they are implemented.

15.3.3 Application notes

The TSF of the base component is often defined without knowledge of the dependencies of the possible applications with which it can be composed. The TSF of this base component is defined to include all parts of the base component that enforcement of the base component SFRs relies upon. This shall include all parts of the base component required to implement the base component SFRs.

The functional specification of the base component shall describe the TSFI in terms of the interfaces the base component provides to allow an external entity to invoke operations of the TSF. This includes interfaces to the human user to permit interaction with the operation of the TSF invoking SFRs and also interfaces allowing an external IT entity to make calls into the TSF.

The functional specification only provides a description of what the TSF provides at its interface and the means by which that TSF functionality are invoked. Therefore, the functional specification does not necessarily provide a complete interface specification of all possible interfaces available between an external entity and the base component. It does not include what the TSF expects/requires from the operational

ISO/IEC 15408-3:2026(en)

environment. The description of what a dependent component TSF relies upon of a base component is considered in ACO_REL (Reliance of dependent component) (15.4) and the development information evidence provides a response to the interfaces specified.

The development information evidence includes a specification of the base component. This can be the evidence used during evaluation of the base component to satisfy the class ADV (Development) (Clause 10) requirements, or can be another form of evidence produced by either the base component developer or the composed TOE developer. This specification of the base component is used during ACO_DEV (Development evidence) to gain confidence that the appropriate security functionality is provided to support the requirements of the dependent component. The level of detail required of this evidence increases to reflect the level of required assurance in the composed TOE. This is expected to broadly reflect the increasing confidence gained from the application of the assurance packages to the components. The evaluator determines that this description of the base component is consistent with the reliance information provided for the dependent component.

15.3.4 Functional description (ACO_DEV.1)

15.3.4.1 Component relationships

Dependencies: ACO_REL.1 (Basic reliance information) (15.4.4)

15.3.4.2 Objectives

A description of the interfaces in the base component, on which the dependent component relies, is required. This is examined to determine whether it is consistent with the description of interfaces on which the dependent component relies, as provided in the reliance information.

15.3.4.3 Component elements

Table 298 specifies the developer action elements for ACO_DEV.1, with Table 299 indicating their related content and presentation requirements, and Table 300 the corresponding evaluation actions.

Table 298 — Developer action elements for ACO_DEV.1

ACO_DEV.1.1D	The developer shall provide development information for the base component.
--------------	--

Table 299 — Content and presentation elements for ACO_DEV.1

ACO_DEV.1.1C	The development information shall describe the purpose of each interface of the base component used in the composed TOE.
ACO_DEV.1.2C	The development information shall show correspondence between the interfaces, used in the composed TOE, of the base component and the dependent component to support the TSF of the dependent component.

Table 300 — Evaluator action elements for ACO_DEV.1

ACO_DEV.1.1E	The evaluator <i>shall confirm</i> that the information meets all requirements for content and presentation of evidence.
ACO_DEV.1.2E	The evaluator <i>shall determine</i> that the interface description provided is consistent with the reliance information provided for the dependent component.

15.3.5 Basic evidence of design (ACO_DEV.2)

15.3.5.1 Component relationships

Dependencies: ACO_REL.1 (Basic reliance information) (15.4.4)

15.3.5.2 Objectives

A description of the interfaces in the base component, on which the dependent component relies, is required. This is examined to determine whether it is consistent with the description of interfaces on which the dependent component relies, as provided in the reliance information.

In addition, the security behaviour of the base component that supports the dependent component TSF is described.

15.3.5.3 Component elements

[Table 301](#) specifies the developer action elements for ACO_DEV.2, with [Table 302](#) indicating their related content and presentation requirements, and [Table 303](#) the corresponding evaluation actions.

Table 301 — Developer action elements for ACO_DEV.2

ACO_DEV.2.1D	The developer shall provide development information for the base component.
--------------	---

Table 302 — Content and presentation elements for ACO_DEV.2

ACO_DEV.2.1C	The development information shall describe the purpose and method of use of each interface of the base component used in the composed TOE.
ACO_DEV.2.2C	The development information shall provide a high-level description of the behaviour of the base component, which supports the enforcement of the dependent component SFRs.
ACO_DEV.2.3C	The development information shall show correspondence between the interfaces, used in the composed TOE, of the base component and the dependent component to support the TSF of the dependent component.

Table 303 — Evaluator action elements for ACO_DEV.2

ACO_DEV.2.1E	The evaluator shall confirm that the information meets all requirements for content and presentation of evidence.
ACO_DEV.2.2E	The evaluator shall determine that the interface description provided is consistent with the reliance information provided for the dependent component.

15.3.6 Detailed evidence of design (ACO_DEV.3)

15.3.6.1 Component relationships

Dependencies: ACO_REL.2 (Reliance information) ([15.4.5](#))

15.3.6.2 Objectives

A description of the interfaces in the base component, on which the dependent component relies, is required. This is examined to determine whether it is consistent with the description of interfaces on which the dependent component relies, as provided in the reliance information.

The interface description of the architecture of the base component is provided to enable the evaluator to determine whether that interface formed part of the TSF of the base component.

15.3.6.3 Component elements

[Table 304](#) specifies the developer action elements for ACO_DEV.3, with [Table 305](#) indicating their related content and presentation requirements, and [Table 306](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 304 — Developer action elements for ACO_DEV.3

ACO_DEV.3.1D	The developer shall provide development information for the base component.
--------------	---

Table 305 — Content and presentation elements for ACO_DEV.3

ACO_DEV.3.1C	The development information shall describe the purpose and method of use of each interface of the base component used in the composed TOE.
ACO_DEV.3.2C	The development information shall identify the subsystems of the base component that provide interfaces of the base component used in the composed TOE.
ACO_DEV.3.3C	The development information shall provide a high-level description of the behaviour of the base component subsystems , which support the enforcement of the dependent component SFRs.
ACO_DEV.3.4C	The development information shall provide a mapping from the interfaces to the subsystems of the base component.
ACO_DEV.3.5C	The development information shall show correspondence between the interfaces, used in the composed TOE, of the base component and the dependent component to support the TSF of the dependent component.

Table 306 — Evaluator action elements for ACO_DEV.3

ACO_DEV.3.1E	The evaluator shall confirm that the information meets all requirements for content and presentation of evidence.
ACO_DEV.3.2E	The evaluator shall determine that the interface description provided is consistent with the reliance information provided for the dependent component.

15.4 Reliance of dependent component (ACO_REL)

15.4.1 Objectives

The purpose of this family is to provide evidence that describes the reliance that a dependent component has upon the base component. This information is useful to persons responsible for integrating the component with other evaluated IT components to form the composed TOE, and for providing insight into the security properties of the resulting composition.

This provides a description of the interface between the dependent and base components of the composed TOE that possibly were not analysed during evaluation of the individual components, because the interfaces were not TSFIs of the individual component TOEs.

15.4.2 Component levelling

The components in this family are levelled according to the amount of detail provided in the description of the reliance by the dependent component upon the base component.

15.4.3 Application notes

The ACO_REL (Reliance of dependent component) family considers the interactions between the components where the dependent component relies upon a service from the base component to support the operation of security functionality of the dependent component. It is possible that the interfaces into these services of the base component have not been considered during evaluation of the base component because the service in the base component was not considered security-relevant in the component evaluation. This could have happened either because of the inherent purpose of the service (e.g. adjust type font) or because associated ISO/IEC 15408-2 SFRs are not being claimed in the base component's ST (e.g. the login interface when no class FIA (Identification and authentication) (see ISO/IEC 15408-2:2026, Clause 12) SFRs are claimed). These interfaces into the base component are often viewed as functional interfaces in the evaluation of the base component, and are in addition to the security interfaces (TSFI) considered in the functional specification.

ISO/IEC 15408-3:2026(en)

In summary, the TSFIs described in the functional specification only include the calls made into a TSF by external entities and responses to those calls. Calls made by a TSF, which were not explicitly considered during evaluation of the components, are described by the reliance information provided to satisfy ACO_REL (Reliance of dependent component).

15.4.4 Basic reliance information (ACO_REL.1)

[Table 307](#) specifies the developer action elements for ACO_REL.1, with [Table 308](#) indicating their related content and presentation requirements, and [Table 309](#) the corresponding evaluation actions.

Table 307 — Developer action elements for ACO_REL.1

ACO_REL.1.1D	The developer shall provide reliance information of the dependent component.
--------------	---

Table 308 — Content and presentation elements for ACO_REL.1

ACO_REL.1.1C	The reliance information shall describe the functionality of the base component hardware, firmware, software that is relied upon by the dependent component TSF.
ACO_REL.1.2C	The reliance information shall describe all interactions through which the dependent component TSF requests services from the base component.
ACO_REL.1.3C	The reliance information shall describe how the dependent TSF protects itself from interference and tampering by the base component.

Table 309 — Evaluator action elements for ACO_REL.1

ACO_REL.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	--

15.4.5 Reliance information (ACO_REL.2)

[Table 310](#) specifies the developer action elements for ACO_REL.2, with [Table 311](#) indicating their related content and presentation requirements, and [Table 312](#) the corresponding evaluation actions.

Table 310 — Developer action elements for ACO_REL.2

ACO_REL.2.1D	The developer shall provide reliance information of the dependent component.
--------------	--

Table 311 — Content and presentation elements for ACO_REL.2

ACO_REL.2.1C	The reliance information shall describe the functionality of the base component hardware, firmware, software that is relied upon by the dependent component TSF.
ACO_REL.2.2C	The reliance information shall describe all interactions through which the dependent component TSF requests services from the base component.
ACO_REL.2.3C	The reliance information shall describe each interaction in terms of the interface used and the return values from those interfaces.
ACO_REL.2.4C	The reliance information shall describe how the dependent TSF protects itself from interference and tampering by the base component.

Table 312 — Evaluator action elements for ACO_REL.2

ACO_REL.2.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
--------------	---

ISO/IEC 15408-3:2026(en)**15.5 Composed TOE testing (ACO_CTT)****15.5.1 Objectives**

This family requires that testing of composed TOE and testing of the base component, as used in the composed TOE, is performed.

15.5.2 Component levelling

The components in this family are levelled on the basis of increasing rigour of interface testing and increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the composed TSF operates in accordance with the reliance information and the composed TOE SFRs.

15.5.3 Application notes

There are two distinct aspects of testing associated with this family:

- testing of the interfaces between the base component and the dependent component, which the dependent component rely upon for enforcement of security functionality, to demonstrate their compatibility;
- testing of the composed TOE to demonstrate that the TOE behaves in accordance with the SFRs for the composed TOE.

If the test configurations used during evaluation of the dependent component included use of the base component as a "platform" and the test analysis sufficiently demonstrates that the TSF behaves in accordance with the SFRs, the developer is not required to perform further testing of the composed TOE functionality. However, if the base component was not used in the testing of the dependent component, or the configuration of either component varied, then the developer shall perform testing of the composed TOE. This may take the form of repeating the dependent component developer testing of the dependent component, provided this adequately demonstrates the composed TOE TSF behaves in accordance with the SFRs.

The developer shall provide evidence of testing the base component interfaces used in the composition. The operation of base component TSFIs would have been tested as part of the class ATE (Tests) ([Clause 13](#)) activities during evaluation of the base component. Therefore, the evaluator action ACO_CTT.1.1E ([15.5.4](#)) may be met through reuse of the base component class ATE (Tests) ([Clause 13](#)) verdicts, provided that:

- the appropriate interfaces were included within the test sample of the base component evaluation
- it was determined in ACO_COR (Composition rationale) ([15.2](#)) that the base component is operating in accordance with the base component evaluated configuration, with all security functionality required by the dependent component included in the TSF.

If this is not the case, the base component interfaces used relevant to the composition that are affected by any variations to the evaluated configuration and any additional security functionality shall be tested to ensure they demonstrate the expected behaviour. The expected behaviour to be tested is that described in the ACO_REL (Reliance of dependent component) ([15.4](#)) evidence.

15.5.4 Interface testing (ACO_CTT.1)**15.5.4.1 Component relationships**

Dependencies: ACO_REL.1 (Basic reliance information) ([15.4.4](#))
 ACO_DEV.1 (Functional description) ([15.3.4](#))

15.5.4.2 Objectives

The objective of this component is to ensure that each interface of the base component, on which the dependent component relies, is tested.

15.5.4.3 Component elements

[Table 313](#) specifies the developer action elements for ACO_CTT.1, with [Table 314](#) indicating their related content and presentation requirements, and [Table 315](#) the corresponding evaluation actions.

Table 313 — Developer action elements for ACO_CTT.1

ACO_CTT.1.1D	The developer shall provide composed TOE test documentation.
ACO_CTT.1.2D	The developer shall provide base component interface test documentation.
ACO_CTT.1.3D	The developer shall provide the composed TOE for testing.
ACO_CTT.1.4D	The developer shall provide an equivalent set of resources to those that were used in the base component developer's functional testing of the base component.

Table 314 — Content and presentation elements for ACO_CTT.1

ACO_CTT.1.1C	The composed TOE and base component interface test documentation shall consist of test plans, expected test results and actual test results.
ACO_CTT.1.2C	The test documentation from the developer execution of the composed TOE tests shall demonstrate that the TSF behaves as specified.
ACO_CTT.1.3C	The test documentation from the developer execution of the base component interface tests shall demonstrate that the base component interface relied upon by the dependent component behaves as specified.
ACO_CTT.1.4C	The base component shall be suitable for testing.

Table 315 — Evaluator action elements for ACO_CTT.1

ACO_CTT.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ACO_CTT.1.2E	The evaluator <i>shall execute</i> a sample of test in the test documentation to verify the developer test results.
ACO_CTT.1.3E	The evaluator <i>shall test</i> a subset of the TSF interfaces of the composed TOE to confirm that the composed TSF operates as specified.

15.5.5 Rigorous interface testing (ACO_CTT.2)

15.5.5.1 Component relationships

Dependencies: ACO_REL.2 (Reliance information) ([15.4.5](#))
 ACO_DEV.2 (Basic evidence of design) ([15.3.5](#))

15.5.5.2 Objectives

The objective of this component is to ensure that each interface of the base component, on which the dependent component relies, is tested.

15.5.5.3 Component elements

[Table 316](#) specifies the developer action elements for ACO_CTT.2, with [Table 317](#) indicating their related content and presentation requirements, and [Table 318](#) the corresponding evaluation actions.

ISO/IEC 15408-3:2026(en)

Table 316 — Developer action elements for ACO_CTT.2

ACO_CTT.2.1D	The developer shall provide composed TOE test documentation.
ACO_CTT.2.2D	The developer shall provide base component interface test documentation.
ACO_CTT.2.3D	The developer shall provide the composed TOE for testing.
ACO_CTT.2.4D	The developer shall provide an equivalent set of resources to those that were used in the base component developer's functional testing of the base component.

Table 317 — Content and presentation elements for ACO_CTT.2

ACO_CTT.2.1C	The composed TOE and base component interface test documentation shall consist of test plans, expected test results and actual test results.
ACO_CTT.2.2C	The test documentation from the developer execution of the composed TOE tests shall demonstrate that the TSF behaves as specified and is complete .
ACO_CTT.2.3C	The test documentation from the developer execution of the base component interface tests shall demonstrate that the base component interface relied upon by the dependent component behaves as specified and is complete .
ACO_CTT.2.4C	The base component shall be suitable for testing.

Table 318 — Evaluator action elements for ACO_CTT.2

ACO_CTT.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ACO_CTT.2.2E	The evaluator shall execute a sample of test in the test documentation to verify the developer test results.
ACO_CTT.2.3E	The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the composed TSF operates as specified.

15.6 Composition vulnerability analysis (ACO_VUL)

15.6.1 Objectives

This family calls for an analysis of vulnerability information available in the public domain and of vulnerabilities that can be introduced as a result of the composition.

15.6.2 Component levelling

The components in this family are levelled on the basis of increasing scrutiny of vulnerability information from the public domain and independent vulnerability analysis.

15.6.3 Application notes

The developer shall provide details of any residual vulnerabilities reported during evaluation of the components. These can be gained from the component developers or evaluation reports for the components. These shall be used as inputs into the evaluator's vulnerability analysis of the composed TOE in the operational environment.

The operational environment of the composed TOE is examined to ensure that the assumptions and objectives for the component operational environment (specified in each component ST) are satisfied in the composed TOE. An initial analysis of the consistency of assumptions and objectives between the components and the composed TOE STs shall have been performed during the conduct of the class ASE (Security Target (ST) evaluation) (Clause 9) activities for the composed TOE. However, this analysis is revisited with the knowledge acquired during the ACO_REL (Reliance of dependent component) (15.4), ACO_DEV (Development evidence) (15.3) and the ACO_COR (Composition rationale) (15.2) activities to ensure that, for example, assumptions of the dependent component that were addressed by the environment in the dependent component ST are not reintroduced as a result of composition (i.e. that the base component adequately addresses the assumptions of the dependent component ST in the composed TOE).

ISO/IEC 15408-3:2026(en)

A search by the evaluator for issues in each component shall identify potential vulnerabilities reported in the public domain since completion of the evaluation of the components. Any potential vulnerabilities shall then be subject to testing.

If the base component used in the composed TOE has been the subject of assurance continuity activities since certification, the evaluator shall consider during the composed TOE vulnerability analysis activities the changes made in base component.

15.6.4 Composition vulnerability review (ACO_VUL.1)

15.6.4.1 Component relationships

Dependencies: ACO_DEV.1 (Functional description) ([15.3.4](#))

15.6.4.2 Component elements

[Table 319](#) specifies the developer action elements for ACO_VUL.1, with [Table 320](#) indicating their related content and presentation requirements, and [Table 321](#) the corresponding evaluation actions.

Table 319 — Developer action elements for ACO_VUL.1

ACO_VUL.1.1D	The developer shall provide the composed TOE for testing.
--------------	---

Table 320 — Content and presentation elements for ACO_VUL.1

ACO_VUL.1.1C	The composed TOE shall be suitable for testing.
--------------	---

Table 321 — Evaluator action elements for ACO_VUL.1

ACO_VUL.1.1E	The evaluator <i>shall confirm</i> that the information provided meets all requirements for content and presentation of evidence.
ACO_VUL.1.2E	The evaluator <i>shall perform</i> an analysis to determine that any residual vulnerabilities identified for the base and dependent components are not exploitable in the composed TOE in its operational environment.
ACO_VUL.1.3E	The evaluator <i>shall perform</i> a search of public domain sources to identify possible vulnerabilities arising from use of the base and dependent components in the composed TOE operational environment.
ACO_VUL.1.4E	The evaluator <i>shall conduct</i> penetration testing, based on the identified vulnerabilities, to demonstrate that the composed TOE is resistant to attacks by an attacker with Basic attack potential.

15.6.5 Composition vulnerability analysis (ACO_VUL.2)

15.6.5.1 Component relationships

Dependencies: ACO_DEV.2 (Basic evidence of design) ([15.3.5](#))

15.6.5.2 Component elements

[Table 322](#) specifies the developer action elements for ACO_VUL.2, with [Table 323](#) indicating their related content and presentation requirements, and [Table 324](#) the corresponding evaluation actions.

Table 322 — Developer action elements for ACO_VUL.2

ACO_VUL.2.1D	The developer shall provide the composed TOE for testing.
--------------	---

ISO/IEC 15408-3:2026(en)

Table 323 — Content and presentation elements for ACO_VUL.2

ACO_VUL.2.1C	The composed TOE shall be suitable for testing.
--------------	---

Table 324 — Evaluator action elements for ACO_VUL.2

ACO_VUL.2.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ACO_VUL.2.2E	The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for the base and dependent components are not exploitable in the composed TOE in its operational environment.
ACO_VUL.2.3E	The evaluator shall perform a search of public domain sources to identify possible vulnerabilities arising from use of the base and dependent components in the composed TOE operational environment.
ACO_VUL.2.4E	The evaluator shall perform an independent vulnerability analysis of the composed TOE, using the guidance documentation, reliance information and composition rationale to identify potential vulnerabilities in the composed TOE.
ACO_VUL.2.5E	The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate that the composed TOE is resistant to attacks by an attacker with Basic attack potential.

15.6.6 Enhanced-Basic composition vulnerability analysis (ACO_VUL.3)

15.6.6.1 Component relationships

Dependencies: ACO_DEV.3 (Detailed evidence of design) ([15.3.6](#))

15.6.6.2 Component elements

[Table 325](#) specifies the developer action elements for ACO_VUL.3, with [Table 326](#) indicating their related content and presentation requirements, and [Table 327](#) the corresponding evaluation actions.

Table 325 — Developer action elements for ACO_VUL.3

ACO_VUL.3.1D	The developer shall provide the composed TOE for testing.
--------------	---

Table 326 — Content and presentation elements for ACO_VUL.3

ACO_VUL.3.1C	The composed TOE shall be suitable for testing.
--------------	---

Table 327 — Evaluator action elements for ACO_VUL.3

ACO_VUL.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
ACO_VUL.3.2E	The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for the base and dependent components are not exploitable in the composed TOE in its operational environment.
ACO_VUL.3.3E	The evaluator shall perform a search of public domain sources to identify possible vulnerabilities arising from use of the base and dependent components in the composed TOE operational environment.
ACO_VUL.3.4E	The evaluator shall perform an independent vulnerability analysis of the composed TOE, using the guidance documentation, reliance information and composition rationale to identify potential vulnerabilities in the composed TOE.
ACO_VUL.3.5E	The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate that the composed TOE is resistant to attacks by an attacker with Enhanced-Basic attack potential.

Annex A

(informative)

Development (ADV)

A.1 ADV_ARC: Supplementary material on security architectures

A.1.1 General

This annex contains ancillary material to further explain and provide additional examples for the topics brought up in families of the class ADV (Development) ([Clause 10](#)).

A security architecture is a set of properties that the TSF exhibits. These properties include self-protection, domain separation, and non-bypassability. Having these properties provides a basis of confidence that the TSF is providing its security services. This annex provides additional material on these properties, as well as discussion on contents of a security architecture description.

[A.1.2](#) and [A.1.3](#) explain these properties and discuss the kinds of information that are needed to describe how the TSF exhibits those properties.

A.1.2 Security architecture properties

Self-protection refers to the ability of the TSF to protect itself from manipulation from external entities that may result in changes to the TSF. Without these properties, the TSF can be disabled from performing its security services.

It is oftentimes the case that a TOE uses services or resources supplied by other IT entities in order to perform its functions (e.g. an application that relies upon its underlying operating system). In these cases, the TSF does not protect itself entirely on its own, because it depends on the other IT entities to protect the services it uses.

Domain separation is a property whereby the TSF creates separate security domains for each untrusted active entity to operate on its resources, and then keeps those domains separated from one another so that no entity can run in the domain of any other. For example, an operating system TOE supplies a domain (address space, per-process environment variables) for each process associated with untrusted entities.

For some TOEs such domains do not exist because all the actions of the untrusted entities are brokered by the TSF. A packet-filter firewall is an example of such a TOE, where there are no untrusted entity domains; there are only data structures maintained by the TSF. The existence of domains, then, is dependant upon 1) the type of TOE and 2) the SFRs levied on the TOE. In the cases where the TOE does provide domains for untrusted entities, this family requires that those domains are isolated from one another such that untrusted entities in one domain are prevented from tampering (affecting without brokering by the TSF) from another untrusted entity's domain.

Non-bypassability is a property whereby the security functionality of the TSF (as specified by the SFRs) is always invoked and cannot be circumvented when appropriate for that specific mechanism. For example, if access control to files is specified as a capability of the TSF via an SFR, there shall be no interfaces through which files can be accessed without invoking the TSF's access control mechanism. An interface through which a raw disk access takes place can be an example of such an interface.

As is the case with self-protection, the very nature of some TOEs can depend upon their environments to play a role in non-bypassability of the TSF. For example, a security application TOE requires that it be invoked by the underlying operating system. Similarly, a firewall depends upon the fact that there are no direct connections between the internal and external networks and that all traffic between them shall go through the firewall.

A.1.3 Security architecture descriptions

A.1.3.1 General

The security architecture description:

- explains how the properties described above are exhibited by the TSF;
- describes how domains are defined;
- describes how the TSF keeps them separate;
- describes what prevents untrusted processes from getting to the TSF and modifying it;
- describes what ensures that all resources under the TSF's control are adequately protected and that all actions related to the SFRs are mediated by the TSF;
- explains any role the environment plays in any of these (e.g. presuming it gets correctly invoked by its underlying environment, how are its security functions invoked?).

The security architecture description presents the TSF's properties of self-protection, domain separation, and non-bypassability in terms of the decomposition descriptions. The level of this description is commensurate with the TSF description required by the ADV_FSP (Functional specification) (10.3), ADV_TDS (TOE design) (10.7) and ADV_IMP (Implementation representation) (10.4) requirements that are being claimed. For example, if ADV_FSP (Functional specification) (10.3) is the only TSF description available, it would be difficult to provide any meaningful security architecture description because none of the details of any internal workings of the TSF would be available.

However, if the TOE design were also available, even at the most basic level (ADV_TDS.1 (Basic design) (10.7.4)), there would be some information available concerning the subsystems that make up the TSF, and there would be a description of how they work to implement self-protection, domain separation, and non-bypassability. For example, perhaps all user interaction with the TOE is constrained through a process that acts on that user's behalf, adopting all of the user's security attributes; the security architecture description would describe how such a process comes into being, how the process's behaviour is constrained by the TSF (so it cannot corrupt the TSF), how all actions of that process are mediated by the TSF (thereby explaining why the TSF cannot be bypassed), etc.

If the available TOE design is more detailed (e.g. at the modular level), or the implementation representation is also available, then the security architecture description is expected to be correspondingly more detailed, explaining how the user's process communicate with the TSF processes, how different requests are processed by the TSF, what parameters are passed, what programmatic protections (buffer overflow prevention, parameter bounds checking, time of check/time of use checking, etc.) are in place. Similarly, a TOE whose ST claimed the ADV_IMP (Implementation representation) (10.4) component would go into implementation-specific detail.

The explanations provided in the security architecture description are expected to be of sufficient detail that one would be able to test their accuracy, i.e. simple assertions (e.g. "The TSF keeps domains separate") provide no useful information to convince the reader that the TSF does indeed create and separate domains.

A.1.3.2 Domain separation

In cases where the TOE exhibits domain separation entirely on its own, there would be a straightforward description of how this is attained. The security architecture description would explain the different kinds of domains that are defined by the TSF, how they are defined (i.e. what resources are allocated to each domain), how no resources are left unprotected, and how the domains are kept separated so that active entities in one domain cannot tamper with resources in another domain.

For cases where the TOE depends upon other IT entities to play a role in domain separation, that sharing of roles shall be made clear. For example, a TOE that is solely application software relies upon the underlying operating system to correctly instantiate the domains that the TOE specifies; if the TOE specifies separate processing space, memory space, etc, for each domain, it depends upon the underlying operating system to

ISO/IEC 15408-3:2026(en)

operate correctly and benignly (e.g. allow the process to execute only in the execution space that is requested by the TOE software).

For example, mechanisms that implement domain separation (e.g. memory management, protected processing modes provided by the hardware) would be identified and described or, the TSF can implement software protection constructs or coding conventions that contribute to implementing separation of software domains, for example by delineating user address space from system address space.

The vulnerability analysis and testing (see AVA_VAN (Vulnerability analysis) (14.3)) activities will likely include attempts to defeat the described TSF domain separation through the use of monitoring or direct attack the TSF.

A.1.3.3 TSF self-protection

In cases where the TOE exhibits self-protection entirely on its own, there would be a straightforward description of how this self-protection is attained. Mechanisms that provide domain separation to define a TSF domain that is protected from other (user) domains would be identified and described.

For cases where the TOE depends upon other IT entities to play a role in protecting itself, that sharing of roles shall be made clear. For example, a TOE that is solely application software relies upon the underlying operating system to operate correctly and benignly; the application cannot protect itself against a malicious operating system that subverts it (for example, by overwriting its executable code or TSF data).

The security architecture description also covers how user input is handled by the TSF in such a way that the TSF does not subject itself to being corrupted by that user input. For example, the TSF can implement the notion of privilege and protect itself by using privileged-mode routines to handle user data. The TSF can make use of processor-based separation mechanisms (e.g. privilege levels or rings) to separate TSF code and data from user code and data. The TSF can implement software protection constructs or coding conventions that contribute to implementing separation of software, for example by delineating user address space from system address space.

For TOEs that start up in a low-function mode (e.g. a single-user mode accessible only to installers or administrators) and then transition to the evaluated secure configuration (e.g. a mode whereby untrusted users are able to login and use the services and resources of the TOE), the security architecture description also includes an explanation of how the TSF is protected against this initialisation code that does not run in the evaluated configuration. For such TOEs, the security architecture description would explain what prevents those services that should be available only during initialisation (e.g. direct access to resources) from being accessible in the evaluated configuration. It would also explain what prevents initialisation code from running while the TOE is in the evaluated configuration.

There shall also be an explanation of how the trusted initialisation code will maintain the integrity of the TSF (and of its initialisation process) such that the initialisation process is able to detect any modification that would result in the TSF being spoofed into believe it was in an initial secure state.

The vulnerability analysis and testing (see AVA_VAN (Vulnerability analysis) (14.3)) activities will likely include attempts to defeat the described TSF self-protection through the use of tampering, direct attack, or monitoring of the TSF.

A.1.3.4 TSF non-bypassability

The property of non-bypassability is concerned with interfaces that permit the bypass of the enforcement mechanisms. In most cases this is a consequence of the implementation, where if a programmer is writing an interface that accesses or manipulates an object, it is that programmer's responsibility to use interfaces that are part of the SFR enforcement mechanism for the object and not to try to circumvent those interfaces. For the description pertaining to non-bypassability, then, there are two broad areas that shall be covered.

The first consists of those interfaces to the SFR-enforcement. The property for these interfaces is that they contain no operations or modes that allow them to be used to bypass the TSF. It is likely that the evidence for ADV_FSP (Functional specification) (10.3) and ADV_TDS (TOE design) (10.7) can be used in large part to make this determination. Since non-bypassability is the concern, if only certain operations available through these

ISO/IEC 15408-3:2026(en)

TSFIs are documented (because they are SFR-enforcing) and others are not, the developer should consider the necessity of additional information (to that presented in ADV_FSP (Functional specification) (10.3) and ADV_TDS (TOE design) (10.7)). It can be necessary to make a determination that the SFR-supporting and SFR-non-interfering operations of the TSFI do not afford an untrusted entity the ability to bypass the policy being enforced. If such information is necessary, it is included in the security architecture description.

The second area of non-bypassability is concerned with those interfaces whose interactions are not associated with SFR-enforcement. Depending on the ADV_FSP (Functional specification) (10.3) and ADV_TDS (TOE design) (10.7) components claimed, it is possible that some information about these interfaces exists, or not, in the functional specification and TOE design documentation. The information presented for such interfaces (or groups of interfaces) should be sufficient so that a reader can make a determination (at the level of detail commensurate with the rest of the evidence supplied in class ADV (Development) (Clause 10)) that the enforcement mechanisms cannot be bypassed.

The property that the security functionality cannot be bypassed applies to all security functionality equally, i.e. the design description should cover objects that are protected under the SFRs (e.g. components of class FDP (User data protection) (see ISO/IEC 15408-2:2026, Clause 11)) and functionality (e.g. audit) that is provided by the TSF. The description should also identify the interfaces that are associated with security functionality; this can make use of the information in the functional specification. This description should also describe any design constructs, such as object managers, and their method of use. For instance, if routines are intended to use a standard macro to produce an audit record, this convention is a part of the design that contributes to the non-bypassability of the audit mechanism. It is important to note that non-bypassability in this context is not an attempt to answer the question whether a part of the TSF implementation could, if malicious, bypass the security functionality, but rather to document how the implementation does not bypass the security functionality.

The vulnerability analysis and testing (see AVA_VAN (Vulnerability analysis) (14.3)) activities will likely include attempts to defeat the described non-bypassability by circumventing the TSF.

A.2 ADV_FSP: Supplementary material on functional specification

A.2.1 General

The purpose in specifying the TSFIs is to provide the necessary information to conduct testing. Without knowing the possible means to interact with the TSF, it is not possible to adequately test the behaviour of the TSF.

There are two parts to specifying the TSFIs: identifying them and describing them. Owing to the diversity of possible TOEs, and of different TSFs therein, there is no standard set of interfaces that constitute TSFIs. A.2 provides guidance on the factors that determine which interfaces are TSFIs.

A.2.2 Non-TSF part of the TOE

The TSF comprises all parts of the TOE the user is required to rely on in order to trust the security functionality.

In other words, those parts of the TOE that do not belong to the TSF can be modified by an attacker without any impact on the TOE security functionality. If this is not the case, these parts of the TOE shall be included in the TSF.

If the TSF and the TSF implementation are defined, then it is clear whether there exist further parts of the TOE which can be classified as non-TSF parts of the TOE. Such parts are not required to be part of the TSF but they are still part of the TOE.

The relationship between TSF and non-TSF parts of TOE is given by their definitions and the ARC properties as follows:

- non-TSF parts do not bypass the TSF;
- parts of the TSF protects themselves against tampering.

ISO/IEC 15408-3:2026(en)

A subsystem of the TOE which is not part of the TSF shall fulfil the following condition (described as a rule of thumb): The subsystem shall not have any security impact of the TOE even if it were substituted by an attacker.

NOTE 1 This rule is only valid to some extent because the actual requirement "The non-TSF part shall not bypass the TSF." is not that strong as the given rule of thumb.

Therefore, between the non-TSF parts and the TSF parts it seems that some kind of separation mechanism is recommended because such a separation mechanism may build the basis for the assessment that there is no impact on the TSF parts from the non-TSF parts possible.

NOTE 2 The separation mechanism is only a proposal here. The developer is free to provide evidence using other kind of security implementation as long as the requirement showing the non-bypassability for the TSF part of the TOE from the non-TSF part of the TOE is fulfilled.

Such a "separation mechanism" can be implemented by the security architecture or by an explicitly realized part of the implementation (e.g. a firewall between TSF and non-TSF parts of the TOE).

The analysis of the "separation mechanism" is then subject of the vulnerability assessment because it shall withstand attacks by an attacker of the respective strength according to the VAN level of the evaluation.

The developer shall provide evidence for non-bypassability and self-protection in its security architecture description and the evaluator shall analyse this evidence in subactivity for ADV_ARC.1 (Security architecture description) ([10.2.4](#)) and assess the effectiveness in the vulnerability assessment.

The goal of TOE design documentation is to provide sufficient information to determine the TSF boundary, and to describe how the TSF implements the SFR. Further attention is needed by the fact that the family ADV_TDS (TOE design) ([10.7](#)) requires only identification of the non-TSF subsystems of the TOE. No interface description is provided for these subsystems in ADV_FSP (Functional specification) ([10.3](#)) or ADV_TDS (TOE design) ([10.7](#)). SFR non-interference of these subsystems is assumed but not demonstrated by the developer and not examined in details by the evaluator. However from the TOE design point of view this is not that important as long as the above mentioned separation mechanism is in place and the vulnerability assessment confirms that it is strong enough. Therefore this "separation mechanism" implements the TSF or enforces ARC properties as security feature. But non-bypassability may be enforced by "pure architecture properties" as well.

Parts of the TOE classified as non-TSF must not provide means to bypass the TSF (no matter whether a valid user or even an attacker makes uses of those parts) and must not contribute to the TSF. It is important that the developer provides clear evidence and demonstrate how this requirement is fulfilled.

Therefore the developer shall demonstrate and the evaluator shall examine that the TOE identification of subsystems as non-TSF (see ADV_TDS.*.1) is correct and consequently no detailed description of these subsystems is necessary.

NOTE 3 The asterisk in "ADV_TDS.*.1" is used in the common text matching sense as a wildcard taking any appropriate value (1,2..).

The evaluator examination shall include the ARC properties non-bypassability and self-protection being described in the ADV_ARC (Security architecture) ([10.2](#)) documentation provided by the developer (see the paragraphs above).

A.2.3 Determining the TSFI

A.2.3.1 General

In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF shall first be identified. This identification is actually a part of the ADV_TDS (TOE design) ([10.7](#)) analysis, but is also performed implicitly (through identification and description of the TSFI) by the developer in cases where ADV_TDS (TOE design) ([10.7](#)) is not included in the assurance package. In this analysis, a portion of the TOE shall be considered to be in the TSF if it contributes to the satisfaction of an SFR in the ST (in whole or in part). This includes, for example, everything in the TOE that contributes to TSF run-time initialisation,

ISO/IEC 15408-3:2026(en)

such as software that runs prior to the TSF being able to protect itself because enforcement of the SFRs has not yet begun (e.g. while booting up). Also included in the TSF are all parts of the TOE that contribute to the architectural principles of TSF self-protection, domain separation, and non-bypassability (see ADV_ARC (Security architecture) ([10.2](#))).

Once the TSF has been defined, the TSFI are identified. The TSFI consists of all means by which external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data from the TSF or invoke services from the TSF. These service invocations and responses are the means of crossing the TSF boundary. While many of these are readily apparent, it is possible that others are not as obvious. The question that should be asked when determining the TSFIs is how a potential attacker can interact with the TSF in an attempt to subvert the SFRs.

Therefore, from the evaluation point of view it is also important whether the interface can be misused by an attacker to get access to the security functionality in order to compromise the assets protected by TSF.

Any interface of the TSF which can be potentially used by an attacker belongs to the TSFI (regardless of the further classification as SFR-enforcing, SFR-supporting or SFR-non-interfering).

It is not important whether the TSF will be accessed from outside or whether the TSF accesses the external resources (e.g. TSF calls platform or user). The only criterion is whether there is a potential interference with the TSF from outside.

The following discussions illustrate the application of the TSFI definition in different contexts.

A.2.3.2 Electrical interfaces

In TOEs such as smart cards, where the adversary has not only logical access to the TOE, but also complete physical access to the TOE, the TSF boundary is the physical boundary. Therefore, the exposed electrical interfaces are considered TSFI because their manipulation can affect the behaviour of the TSF. As such, all these interfaces (electrical contacts) shall be described, e.g. various voltages that can be applied.

A.2.3.3 Network protocol stack

The TSFIs of a TOE that performs protocol processing would be those protocol layers to which a potential attacker has direct access. This need not be the entire protocol stack, but it can be.

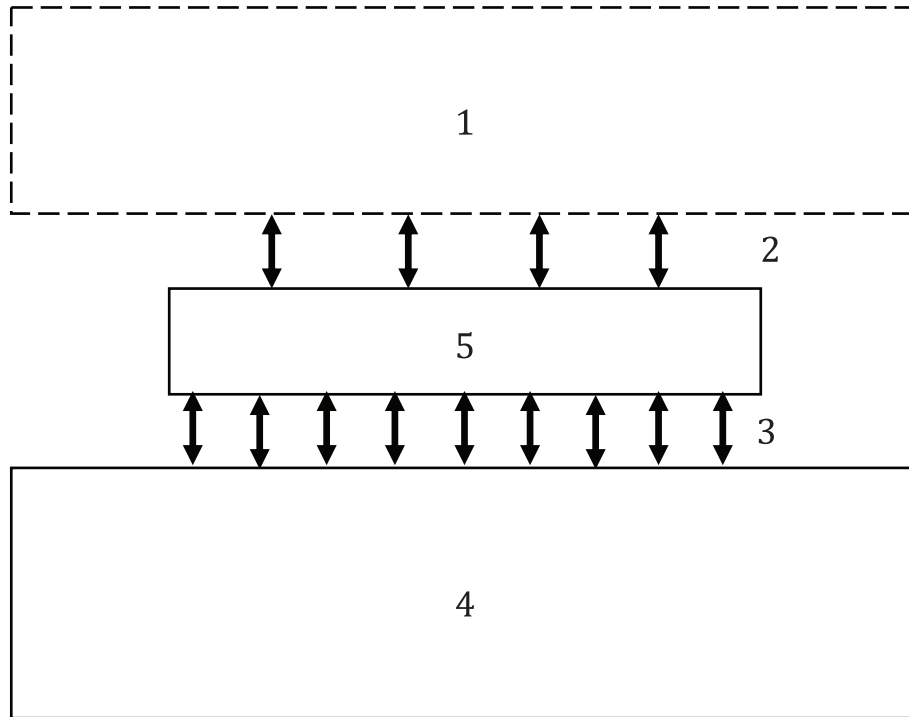
For example, if the TOE was some sort of a network appliance that allowed potential attackers to affect every level of the protocol stack (i.e. to send arbitrary signals, arbitrary voltages, arbitrary packets, arbitrary datagrams), then the TSF boundary exists at each layer of the stack. Therefore, the functional specification would have to address every protocol at every layer of the stack.

If, however, the TOE was a firewall that protects an internal network from the Internet, a potential attacker would have no means of directly manipulating the voltages that enter the TOE; any extreme voltages would simply not be passed through the Internet, i.e. the attacker would have access only to those protocols at the Internet layer or above. The TSF boundary exists at each layer of the stack. Therefore, the functional specification would have to address only those protocols at or above the Internet layer: it would describe each of the different communication layers at which the firewall is exposed in terms of what constitutes well-formed input for what can appear on the line, and the result of both well-formed and malformed inputs. For example, the description of the Internet protocol layer would describe what constitutes a well-formed IP packet and what happens when both correctly-formed and malformed packets are received. Likewise, the description of the TCP layer would describe a successful TCP connection and what happens both when successful connections are established and when connections cannot be established or are inadvertently dropped. Presuming the firewall's purpose is to filter application-level commands (e.g. SSH or telnet), the description of the application layer would describe the application-level commands that are recognized and filtered by the firewall, as well as the results of encountering unknown commands.

The descriptions of these layers would likely reference published communication standards (e.g. telnet, SSH, TCP) that are used, noting which user-defined options are chosen.

A.2.3.4 Wrappers

Wrappers translate complex series of interactions into simplified common services, such as when operating systems create APIs for use by applications (as shown in [Figure A.1](#)). Whether the TSFIs would be the system calls or the APIs depends upon what is available to the application. If the application can use the system calls directly, then the system calls are the TSFIs. If, however, there were something that prohibits their direct use and requires all communication through the APIs, then the APIs would be the TSFIs.



Key

1	application
2	APIs
3	system
4	kernel (TSF)
5	wrappers

Figure A.1 — Wrappers

A graphical user interface is similar: it translates between machine-understandable commands and user-friendly graphics. Similarly, the TSFIs would be the commands if users have access to them, or the graphics (pull-down menus, check-boxes, text fields) if the users are constrained to using them.

It is worth noting that, in both of these examples, if the user is prohibited from using the more primitive interfaces (i.e. the system calls or the commands), the description of this restriction and of its enforcement would be included in the security architecture description (see [A.1](#)). The wrapper would also be part of the TSF.

A.2.3.5 Inaccessible interfaces

For a given TOE, not all the interfaces may be accessible, i.e. the security objectives for the operational environment (in the ST) may prevent access to these interfaces or limit access in such a way that they are practically inaccessible. Such interfaces would not be considered TSFIs. Some examples include:

- the security objectives for the operational environment for the stand-alone firewall may state that the firewall will be operational in a server room environment to which only trusted and trained personnel will have access, and which will be equipped with an interruptible power supply (against power failure).

ISO/IEC 15408-3:2026(en)

The IT environment consists of the operating system itself (represented by the box labelled OS), as well as an external server (labelled SRV). This external server, like the OS, provides a service that the TSF depends on, and thus is required to be in the IT environment. Interfaces in [Figure A.2](#) are labelled A1, A2 and A3 for TSFI, and B1, B2 and B3 for other interfaces that would be documented in class ACO (Composition) ([Clause 15](#)). Each of these groups of interfaces is discussed below.

Interface group A1 represents the most obvious set of TSFI. These are interfaces used by users to directly access the database and its security functionality and resources.

Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g. Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and shall be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

The interfaces labelled B1, B2 and B3 represent interfaces to functionality in the IT Environment. These interfaces are not TSFI and need only be discussed and analysed when the TOE is being used in a composite evaluation as part of the activities associated with class ACO (Composition) ([Clause 15](#)).

A.2.5 Example functional specification

A.2.5.1 General

The example firewall is used between an internal network and an external network. It verifies the source address of data received (to ensure that external data are not attempting to masquerade as originating from the internal data). If it detects any such attempts, it saves the offending attempt to the audit log. The administrator connects to the firewall by establishing a telnet connection to the firewall from the internal network. Administrator actions consist of authenticating, changing passwords, reviewing the audit log, and setting or changing the addresses of the internal and external networks.

The example firewall presents the following interfaces to the internal network:

- IP datagrams;
- administrator commands;

and the following interfaces to the external network:

- IP datagrams.

A.2.5.2 Interfaces Descriptions: IP Datagrams

The datagrams are in the format specified by RFC 791:

- purpose: to transmit blocks of data ("datagrams") from source hosts to destination hosts identified by fixed length addresses; also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through small-packet networks;
- method of use: they arrive from the lower-level (e.g. data link) protocol;
- parameters: the following fields of the IP datagram header: source address, destination address, don't-fragment flag;
- parameter description: [As defined by RFC 791, subclause 3.1 ("Internet Header Format")];
- actions: Transmits datagrams that are not masquerading; fragments large datagrams if necessary; reassembles fragments into datagrams;

ISO/IEC 15408-3:2026(en)

- error messages: (none). No reliability guaranteed (reliability to be provided by upper-level protocols) Undeliverable datagrams (e.g. shall be fragmented for transmission, but don't-fragment flag is set) dropped.

A.2.5.3 Interfaces Descriptions: Administrator Commands

The administrator commands provide a means for the administrator to interact with the firewall. These commands and responses ride atop a telnet (RFC 854) connection established from any host on the internal network. Available commands are:

- **Passwd**
 - purpose: sets administrator password;
 - method of use: `Passwd < password >`;
 - parameters: password;
 - parameter description: value of new password;
 - actions: changes password to new value supplied. There are no restrictions;
 - error messages: none.
- **Readaudit**
 - purpose: presents the audit log to the administrator;
 - method of use: `Readaudit`;
 - parameters: none;
 - parameter description: none;
 - actions: provides the text of the audit log;
 - error messages: none.
- **Setintaddr**
 - purpose: sets the address of the internal address;
 - method of use: `Setintaddr < address >`;
 - parameters: address;
 - parameter description: first three fields of an IP address (as defined in RFC 791). For example: 123.123.123;
 - actions: changes the internal value of the variable defining the internal network, the value of which is used to judge attempted masquerades;
 - error messages: "address in use": indicates the identified internal network is the same as the external network.
- **Setextaddr**
 - purpose: sets the address of the external address;
 - method of use: `Setextaddr < address >`;
 - parameters: address;
 - parameter description: first three fields of an IP address (as defined in RFC 791). For example: 123.123.123;

- actions: changes the internal value of the variable defining the external network;
- error messages: "address in use": indicates the identified external network is the same as the internal network.

A.3 ADV_INT: Supplementary material on TSF internals

A.3.1 General

The wide variety of TOEs makes it impossible to codify anything more specific than "well-structured" or "minimum complexity". Judgements on structure and complexity are expected to be derived from the specific technologies used in the TOE. For example, software is likely to be considered well-structured if it exhibits the characteristics cited in the software engineering disciplines.

This clause provides supplementary material on assessing the structure and complexity of procedure-based software portions of the TSF. This material is based on information readily available in software engineering literature. For other kinds of internals (e.g. hardware, non-procedural software such as object-oriented code), corresponding literature on good practices should be consulted.

A.3.2 Structure of procedural software

A.3.2.1 General

The structure of procedural software is traditionally assessed according to its modularity. Software written with a modular design aids in achieving understandability by clarifying what dependencies a module has on other modules (coupling) and by including in a module only tasks that are strongly related to each other (cohesion). The use of modular design reduces the interdependence between elements of the TSF and thus reduces the risk that a change or error in one module will have effects throughout the TOE. Its use enhances clarity of design and provides for increased assurance that unexpected effects do not occur. Additional desirable properties of modular decomposition are a reduction in the amount of redundant or unneeded code.

Minimizing the amount of functionality in the TSF allows the evaluator as well as the developer to focus only on that functionality which is necessary for SFR enforcement, contributing further to understandability and further lowering the likelihood of design or implementation errors.

The incorporation of modular decomposition, layering and minimization into the design and implementation process shall be accompanied by sound software engineering considerations. A practical, useful software system will usually entail some undesirable coupling among modules, some modules that include loosely-related functions, and some subtlety or complexity in a module's design. These deviations from the ideals of modular decomposition are often deemed necessary to achieve some goal or constraint, be it related to performance, compatibility, future planned functionality, or some other factors, and may be acceptable, based on the developer's justification for them. In applying the requirements of this class, due consideration shall be given to sound software engineering principles; however, the overall objective of achieving understandability shall be achieved.

A.3.2.2 Cohesion

Cohesion is the manner and degree to which the tasks performed by a single software module are related to one another; types of cohesion include coincidental, communicational, functional, logical, sequential, and temporal. These types of cohesion are characterized below, listed in the order of decreasing desirability.

- Functional cohesion: a module with functional cohesion performs activities related to a single purpose. A functionally cohesive module transforms a single type of input into a single type of output, such as a stack manager or a queue manager.
- Sequential cohesion: a module with sequential cohesion contains functions each of whose output is input for the following function in the module. An example of a sequentially cohesive module is one

ISO/IEC 15408-3:2026(en)

that contains the functions to write audit records and to maintain a running count of the accumulated number of audit violations of a specified type.

- Communicational cohesion: a module with communicational cohesion contains functions that produce output for, or use output from, other functions within the module. An example of a communicationally cohesive module is an access check module that includes mandatory, discretionary, and capability checks.
- Temporal cohesion: a module with temporal cohesion contains functions that shall be executed at about the same time. Examples of temporally cohesive modules include initialisation, recovery, and shutdown modules.
- Logical (or procedural) cohesion: a module with logical cohesion performs similar activities on different data structures. A module exhibits logical cohesion if its functions perform related, but different, operations on different inputs.
- Coincidental cohesion: a module with coincidental cohesion performs unrelated, or loosely related, activities.

A.3.2.3 Coupling

Coupling is the manner and degree of interdependence between software modules; types of coupling include call, common and content coupling. These types of coupling are characterized below, listed in the order of decreasing desirability:

- call: two modules are call coupled if they communicate strictly through the use of their documented function calls; examples of call coupling are data, stamp, and control, which are defined below:
 - data: two modules are data coupled if they communicate strictly through the use of call parameters that represent single data items;
 - stamp: two modules are stamp coupled if they communicate through the use of call parameters that comprise multiple fields or that have meaningful internal structures;
 - control: two modules are control coupled if one passes information that is intended to influence the internal logic of the other.
- common: two modules are common coupled if they share a common data area or a common system resource. Global variables indicate that modules using those global variables are common coupled. Common coupling through global variables is generally allowed, but only to a limited degree. For example, variables that are placed into a global area, but are used by only a single module, are inappropriately placed, and should be removed. Other factors that shall be considered in assessing the suitability of global variables are:
 - the number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification shall be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them;
 - the number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.
- content: two modules are content coupled if one can make direct reference to the internals of the other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that some or all the content of one module are effectively included in the other. Content coupling can be thought of

as using unadvertised module interfaces; this is in contrast to call coupling, which uses only advertised module interfaces.

A.3.3 Complexity of procedural software

Complexity is the measure of the decision points and logical paths of execution that code takes. Software engineering literature cites complexity as a negative characteristic of software because it impedes understanding of the logic and flow of the code. Another impediment to the understanding of code is the presence of code that is unnecessary, in that it is unused or redundant.

The use of layering to separate levels of abstraction and minimize circular dependencies further enables a better understanding of the TSF, providing more assurance that the TOE SFRs are accurately and completely instantiated in the implementation.

Reducing complexity also includes reducing or eliminating mutual dependencies, which pertains both to modules in a single layer and to those in separate layers. Modules that are mutually dependent may rely on one another to formulate a single result, which can result in a deadlock condition, or worse yet, a race condition (e.g. time of check vs. time of use concern), where the ultimate conclusion can be indeterminate and subject to the computing environment at the given instant in time.

Design complexity minimization is a key characteristic of a reference validation mechanism, the purpose of which is to arrive at a TSF that is easily understood so that it can be completely analysed. There are other important characteristics of a reference validation mechanism, such as TSF self-protection and non-bypassability; these other characteristics are covered by requirements in the ADV_ARC (Security architecture) ([10.2](#)) family.

A.4 ADV_TDS: Subsystems and Modules

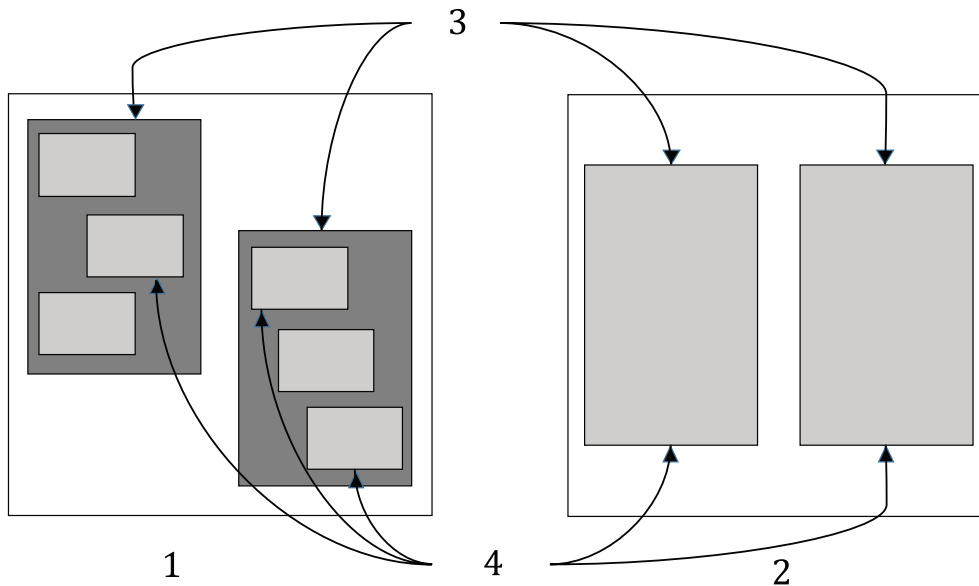
A.4.1 General

This clause provides additional guidance on the TDS family, and its use of the terms subsystem and module. This is followed by a discussion of how, as more detailed information about the TOE becomes available, the requirement for the less detailed is reduced.

A.4.2 Subsystems

[Figure A.3](#) shows that, depending on the complexity of the TSF, the design may be described in terms of subsystems and modules (where subsystems are at a higher level of abstraction than modules); or it may just be described in terms of one level of abstraction (e.g. subsystems at lower assurance levels, modules at higher levels). In cases where a lower level of abstraction (modules) is presented, requirements levied on higher-level abstractions (subsystems) are essentially met by default. This concept is further elaborated in the discussion on subsystems and modules in the following text.

ISO/IEC 15408-3:2026(en)



Key

- 1 TOE 1 (complex)
- 2 TOE 2 (simple)
- 3 subsystems
- 4 modules

Figure A.3 — Subsystems and modules

The developer is expected to describe the design of the TOE in terms of subsystems. The term subsystem was chosen to be specifically vague so that it can refer to units appropriate to the TOE (e.g. subsystems, modules). Subsystems can even be uneven in scope, as long as the requirements for description of subsystems are met.

The first use of subsystems is to distinguish the TSF boundary, i.e. the portions of the TOE that comprise the TSF. In general, a subsystem is part of the TSF if it has the capability (whether by design or implementation) to affect the correct operation of any of the SFRs. For example, for software that depends on different hardware execution modes to provide domain separation (see [A.1](#)) where SFR-enforcing code is executed in one domain, then all subsystems that execute in that domain would be considered part of the TSF. Likewise, if a server outside that domain implemented an SFR (e.g. enforced an access control policy over objects it managed), then it too would be considered part of the TSF.

The second use of subsystems is to provide a structure for describing the TSF at a level of description that, while describing how the TSF works, does not necessarily contain low-level implementation detail found in module descriptions. Subsystems are described at either a high level (lacking an abundance of implementation detail) or a detailed level (providing more insight into the implementation). The level of description provided for a subsystem is determined by the degree to which that subsystem is responsible for implementing an SFR.

An SFR-enforcing subsystem is a subsystem that provides mechanisms for enforcing an element of any SFR, or directly supports a subsystem that is responsible for enforcing an SFR. If a subsystem provides (implements) an SFR-enforcing TSFI, then the subsystem is SFR-enforcing.

Subsystems can also be identified as SFR-supporting and SFR-non-interfering. An SFR-supporting subsystem is one that is depended on by an SFR-enforcing subsystem in order to implement an SFR, but does not play as direct a role as an SFR-enforcing subsystem. An SFR-non-interfering subsystem is one that is not depended upon, in either a supporting or enforcing role, implementing an SFR.

A.4.3 Modules

A module is generally a relatively small architectural unit that can be characterized in terms of the properties discussed in ADV_INT (TSF internals) ([10.5](#)). When both ADV_TDS.3 (Basic modular design) ([10.7.6](#)) (or

ISO/IEC 15408-3:2026(en)

above) requirements and ADV_INT (TSF internals) (10.5) requirements are present in a PP or ST, a module in terms of the ADV_TDS (TOE design) (10.7) requirements refers to the same entity as a module for the ADV_INT (TSF internals) (10.5) requirements. Unlike subsystems, modules describe the implementation in a level of detail that can serve as a guide to reviewing the implementation representation.

It is important to note that, depending on the TOE, modules and subsystems may refer to the same abstraction. For ADV_TDS.1 (Basic design) (10.7.4) and ADV_TDS.2 (Architectural design) (10.7.5) (which do not require description at the module level) the subsystem description provides the lowest level detail available about the TSF. For ADV_TDS.3 (Basic modular design) (10.7.6) (which require module descriptions), these descriptions provide the lowest level of detail, while the subsystem descriptions (if they exist as separate entities) merely serve to put to the module descriptions in context. That is, it is not necessary to provide detailed subsystem descriptions if module descriptions exist. In TOEs that are sufficiently simple, a separate subsystem description is not necessary; the requirements can be met through documentation provided by modules. For complex TOEs, the purpose of the subsystem description (with respect to the TSF) is to provide the reader context so they can focus their analysis appropriately. This difference is illustrated in [Figure A.3](#).

An SFR-enforcing module completely or partially implements an SFR in the ST. Such modules can implement an SFR-enforcing TSFI, but it is possible that some functionality expressed in an SFR (e.g. audit and object re-use functionality) is directly tied to a single TSFI. As was the case with subsystems, SFR-supporting modules are those modules that are depended upon by an SFR-enforcing module, but are not responsible for directly implementing an SFR. SFR-non-interfering modules are those modules that do not deal, directly or indirectly, with the enforcement of SFRs.

It is important to note that the determination of what "directly implements" means is somewhat subjective. In the narrowest sense of the term, it can be interpreted to mean the one or two lines of code that actually perform a comparison, zeroing operation, etc. that implements a requirement. A broader interpretation can be that it includes the module that is invoked in response to an SFR-enforcing TSFI, and all modules that may be invoked in turn by that module (and so on until the completion of the call). Neither of these interpretations is particularly satisfying, since the narrowness of the first interpretation may lead to important modules being incorrectly categorized as SFR supporting, while the second leads to modules that are actually not SFR-enforcing being classified as such.

A description of a module should be such that one can create an implementation of the module from the description, and the resulting implementation would be:

- identical to the actual TSF implementation in terms of the interfaces presented,
- identical in the use of interfaces that are mentioned in the design, and
- functionally equivalent to the description of the purpose of the TSF module.

For instance, RFC 793 provides a high-level description of the TCP protocol. It is necessarily implementation independent. While it provides a wealth of detail, it is not a suitable design description because it is not specific to an implementation. An actual implementation can add to the protocol specified in the RFC, and implementation choices (e.g. the use of global data vs. local data in various parts of the implementation) may have an impact on the analysis that is performed. The design description of the TCP module would list the interfaces presented by the implementation (rather than just those defined in RFC 793), as well as an algorithm description of the processing associated with the modules implementing TCP (assuming they were part of the TSF).

In the design, modules are described in detail in terms of:

- the function they provide (the purpose);
- the interfaces they present (when required by the criteria);
- the return values from such interfaces;
- the interfaces (presented by other modules) they use (provided those interfaces are required to be also described); and

ISO/IEC 15408-3:2026(en)

- a description of how they provide their functionality using a technique appropriate to the method used to implement the module.

The purpose of a module should be described indicating what function the module is providing. It should be sufficient so that the reader can get a general idea of what the module's function is in the architecture.

The interfaces presented by a module are those interfaces used by other modules to invoke the functionality provided. Interfaces include both explicit interfaces (e.g. a calling sequence invoked by other modules) as well as implicit interfaces (e.g. global data manipulated by the module). Interfaces are described in terms of how they are invoked, and any values that are returned. This description would include a list of parameters, and descriptions of these parameters. If a parameter were expected to take on a set of values (e.g. a "flag" parameter), the complete set of values the parameter can take on that would have an effect on module processing would be specified. Likewise, parameters representing data structures are described such that each field of the data structure is identified and described. Global data should be described to the extent required to understand their purpose. The level of description required for a global data structure is identical to the one for module interfaces, where the input parameter and return values correspond to the individual fields and their possible values in the data structure. Global data structures may be described separate from the modules that manipulate or read them as long as the design of the modules contain sufficient information about the global data structures updated or the information extracted from global data structures.

Different programming languages may have additional "interfaces" that would be non-obvious; an example would be operator/function overloading in C++. This "implicit interface" in the class description would also be described as part of the module design. Although a module can present only one interface, it is more common that a module presents a small set of related interfaces.

When it is required to describe the interfaces used by a module, it shall be clear from either the design description of the module or the purpose of the module called, what service is expected from the module called. For example, if Module A is being described, and it uses Module B's bubble sort routine, the description of the interaction between modules shall allow to identify why Module B's bubble sort routine is called and what this call contributes to the implementation of the SFRs. The interface and purpose of Module B's bubble sort routine shall be described as part of the interfaces of Module B (provided the level of ADV_TDS (TOE design) (10.7) and the classification of Module B require a description its interfaces) and so Module A just shall identify what data it requires to have sorted using this routine. An adequate description would be: "Module A invokes Module B's interface `double_bubble()` to sort the usernames in alphabetical order".

If this sorting of the user names is not important for the enforcement of any SFR (e.g. it is just done to speed up things and an algorithmically identical implementation of Module A can also avoid to have the usernames sorted), the use of Module B's bubble sort routine is not SFR-enforcing and it is sufficient to explain in the description of Module A that the usernames are sorted in alphabetical order to enhance performance. Module B may be classified as "SFR-supporting" only and the level of ADV_TDS (TOE design) (10.7) chosen indicates if the interfaces of SFR-supporting modules shall be described or if it is sufficient to just describe the purpose of Module B.

As discussed previously, the algorithmic description of the module should describe in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or (at ADV_TDS.3 (Basic modular design) (10.7.6)) informal text. It discusses how the module inputs and called functions are used to accomplish the module's function. It notes changes to global data, system state, and return values produced by the module. It is at the level of detail that an implementation can be derived that would be very similar to the actual implementation of the TOE.

It should be noted that source code does not meet the module documentation requirements. Although the module design describes the implementation, it is not the implementation. The comments surrounding the source code can be sufficient documentation if they provide an explanation of the intent of the source code. In-line comments that merely state what each line of code is doing are useless because they provide no explanation of what the module is meant to accomplish.

In A.4.4, the labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) discussed for subsystems and modules are used to describe the amount and type of information that shall be made available by the developer. The elements have been structured so that there is no expectation that the developer provide

ISO/IEC 15408-3:2026(en)

only the information specified, i.e. if the developer's documentation of the TSF provides the information in the requirements in [A.4.4](#), there is no expectation that the developer update their documentation and label subsystems and modules as SFR-enforcing, SFR-supporting or SFR-non-interfering. The primary purpose of this labelling is to allow developers with less mature development methodologies (and associated artefacts, such as detailed interface and design documentation) to provide the necessary evidence without undue cost.

A.4.4 Levelling approach

Since there is subjectivity in determining what is SFR-enforcing vs. SFR-supporting (and in some cases, even determining what is SFR-non-interfering) the following paradigm has been adopted in this family. In early components of the family, the developer makes a determination about the classification of the subsystems into SFR-enforcing, SFR-supporting and SFR-non-interfering supplying the appropriate information, and little additional evidence is available for the evaluator to examine to support this claim. As the level of desired assurance increases while the developer still makes a classification determination, the evaluator obtains more and more evidence that is used to confirm the developer's classification.

In order to focus the evaluator's analysis on the SFR-related portions of the TOE, especially at lower levels of assurance, the components of the family are levelled such that initially detailed information is required only for SFR-enforcing architectural entities. As the level of assurance increases, more information is required for SFR-supporting and (eventually) SFR-non-interfering entities. It should be noted that even when complete information is required, it is not necessary that all of this information be analysed in the same level of detail. The focus should be in all cases on whether the necessary information has been provided and analysed.

[Table A.1](#) summarizes the information required at each of the family components for the architectural entities to be described.

ISO/IEC 15408-3:2026(en)

Table A.1 — Description detail levelling

	TSF subsystem			TSF module		
	SFR enforce	SFR support	SFR NI	SFR enforce	SFR support	SFR NI
ADV_TDS.1 (Basic design) (10.7.4) (informal presentation)	structure, summary of SFR-Enf. behaviour, interactions	designation support ^a	designation support			
ADV_TDS.2 (Architectural design) (10.7.5) (informal presentation)	structure, detailed description of SFR-Enf. behaviour, summary of other behaviour, interactions	structure, summary of other behaviour, interactions	designation support, interactions			
ADV_TDS.3 (Basic modular design) (10.7.6) (informal presentation)	description, interactions	description, interactions	description, interactions	purpose, SFR interfaces ^b	interaction, purpose	interaction, purpose
ADV_TDS.4 (Semi-Formal modular design) (10.7.7) (semi-formal presentation)	description, interactions	description, interactions	description, interactions	purpose, SFR interfaces	purpose, SFR interfaces	interaction, purpose
ADV_TDS.5 (Complete semi-formal modular design) (10.7.8) (semi-formal presentation)	description, interactions	description, interactions	description, interactions	purpose, all interfaces ^c	purpose, all interfaces	purpose, all interfaces
ADV_TDS.6 (Complete semi-formal modular design with formal high-level design presentation) (10.7.9) (semi-formal presentation; additional formal presentation)	description, interactions	description, interactions	description, interactions	purpose, all interfaces	purpose, all interfaces	purpose, all interfaces
^a Designation support means that only documentation sufficient to support the classification of the subsystem/module is needed. ^b SFR interfaces means that the module description contains, for each SFR-related interface, the returned values and the called interfaces to other modules. ^c All interfaces means that the module description contains, for each interface, the returned values and the called interfaces to other modules.						

A.4.5 Security relevance

The ISO/IEC 15408 series concentrates the description, the evidence and the analysis on the security functionality of the TOE. This requires characterization of security relevance of functional and physical parts of the TOE. Interfaces, subsystems and modules may be categorized (either implicitly or explicitly) as "SFR-enforcing", "SFR-supporting", or "SFR-noninterfering".

The developer evidence and the evaluation analysis relate to the TOE and focus on the TSF and its SFR-enforcing and SFR-supporting implementation. The security architecture description shall demonstrate that the identified non-TSF subsystems of the TOE are not bypassing the TSF and the TSF protects themselves against corruption by non-TSF code or entities. The developer shall describe the SFR-noninterfering interfaces, subsystems and modules in the TOE design and demonstrate that they do not interfere with the TSF because of their purposes, interactions or separation of resources.

An interface, subsystem or module is:

- SFR-enforcing, if it directly implements an SFR;

ISO/IEC 15408-3:2026(en)

- SFR-supporting if it has to operate functionally correctly in order to support the proper function of the SFRs;
- SFR-non-interfering if it is not related to the implementation of the SFRs.

The focus on security enforcing and security supporting functionality requires evidence of non-interference of the other functionality. Even correct implemented security enforcing functions and security mechanisms may be bypassed, circumvented, deactivated, corrupted, or directly attacked. Non-interference implies that the TSF cannot be misused and unauthorized access to the resources of the TSF implementation is prevented or impossible. Therefore, the security architecture aspects of non-bypassability and self-protection are critical if security relevance of interfaces, subsystems and modules is categorized and this categorization is used in the vulnerability analysis.

TSF self-protection is the security architecture property whereby the TSF cannot be corrupted by non-TSF code or entities. This includes non-TSF subsystems of TOE and non-TOE parts of the IT product. It is similar to the evidence for SFR-non-interfering subsystems/modules.

The security domains are environments provided by the TSF for the use by untrusted entities in such a way that these environments are isolated and protected from each other.

Therefore the analysis of non-interference during evaluation requires examination of the security architecture of the TOE (ADV_ARC (Security architecture) (10.2)) and may need more information on non-TSF subsystems than only the TOE structure in terms of subsystems as provided for ADV_TDS.*.1.

NOTE The asterisk in "ADV_TDS.*.1" is used in the common text matching sense as a wildcard taking any appropriate value (1,2..).

The developers shall provide a rationale that TSF is correctly defined and the analysis of SFR-non-interfering module in terms of its purpose and interaction with other modules:

- purpose: how a module provides their functionality, no further design decisions are needed;
- interaction: reason that subsystems or modules communicate, and characterizes the information that is passed (less details than for interfaces).

During evaluation non-interference shall be analysed as part of the examination of functional specification and TOE design, and the vulnerability analysis. The categorization of interfaces, subsystems and modules as SFR-enforcing, SFR-supporting and SFR-noninterfering implies specific examination of the functional specification, design and testing. An interpretation of TSFI as all accessible external interfaces of the TSF would help this analysis. The functional tests of all TSF subsystems (beginning with ATE_DPT.1 (Testing: basic design) (13.3.4)) and all TSF modules (ATE_DPT.3 (Testing: modular design) (13.3.6) and higher) should provide evidence for the correctness of their security categorization.

A.5 Supplementary material on formal methods

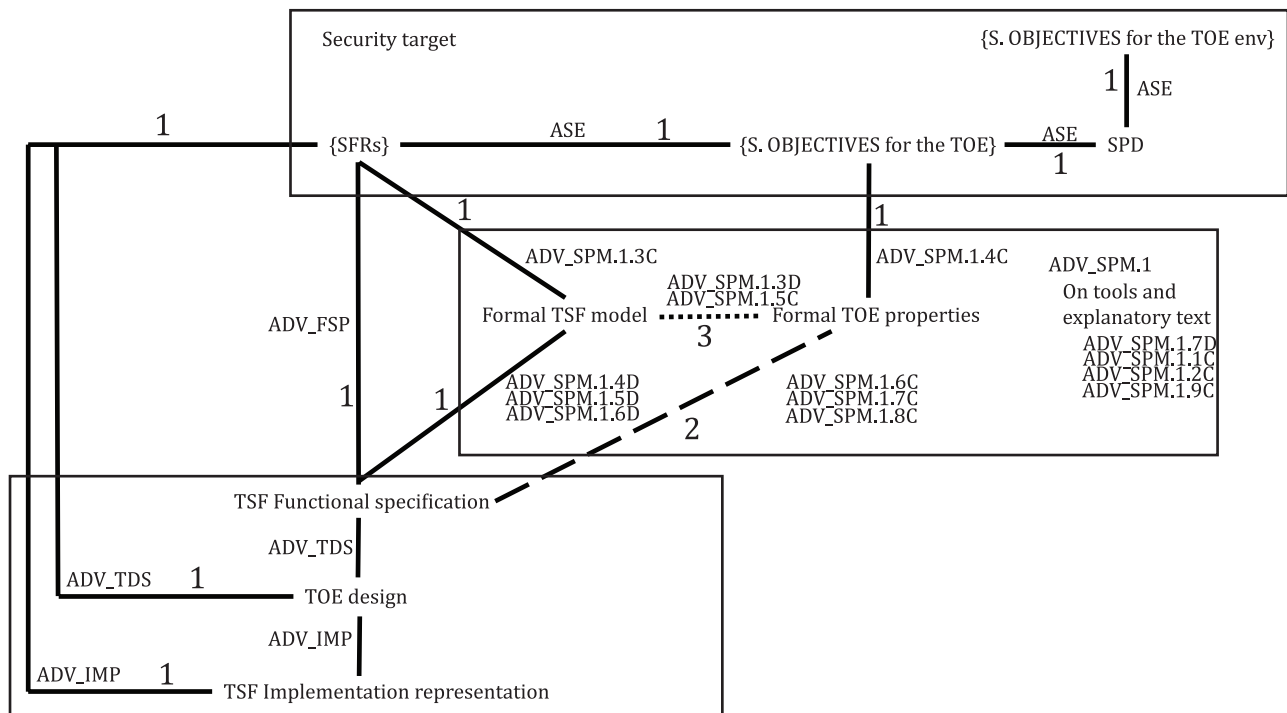
Formal methods provide a mathematical representation of the TSF and its behaviour and are required by:

- ADV_SPM.1 (Formal TSF model) (10.6.4) and also ADV_FSP.6 (Complete semi-formal functional specification with additional formal specification) (10.3.9),
- ADV_TDS.6 (Complete semi-formal modular design with formal high-level design presentation) (10.7.9) components.

In ISO/IEC 18045:2026, Annex C, supplementary material on formal methods is provided.

Figure A.4 illustrates the relationships between the SPM as specified in ADV_SPM.1 (Formal TSF model) (10.6.4) and the representations of the TSF provided by the ST and the functional specification.

ISO/IEC 15408-3:2026(en)



Key

- 1 relation of correspondence
- 2 presentation of properties
- 3 formal proof

Figure A.4 — Relationship of ADV_SPM to other families and constructs

Class ASE (Security Target (ST) evaluation) (Clause 9) specifies the requirements for the correspondence between SFRs and the security objectives for the TOE, as well as the requirements for the correspondence between the SPD elements and the security objectives for the TOE and those for the operational environment, respectively.

Each assurance family specific to a TSF representation, i.e. ADV_FSP (Functional specification) (10.3), ADV_TDS (TOE design) (10.7), and ADV_IMP (Implementation representation) (10.4) specifies both:

- requirements pertaining to developer actions for the correspondence of that specific TSF representation and the one directly above it;
- requirements for the evaluation of the correspondence of that specific TSF representation to the set of SFRs.

The ADV_SPM (Formal TSF model) (10.6) family focuses on a formal security model which is a formal representation of the essential aspects of security (i.e. the TSF) and their relationship to the behaviour of the TOE. Specifically, the formal TSF model is a formal description of the system being evaluated as defined by the entire set of SFRs described in the ST. The set of formal TOE properties defined for this formal model covers all the security objectives for the TOE. To this end, the ADV_SPM (Formal TSF model) (10.6) family specifies:

- requirements pertaining to developer actions for formally modelling the TSF (ADV_SPM.1.1D (10.6.4)) and the set of formal TOE properties (ADV_SPM.1.2D (10.6.4));
- requirements regarding the content and presentation of the correspondence between:
 - the formal TSF model and the complete set of SFRs (ADV_SPM.1.3C (10.6.4));
 - the formal TOE properties and the security objectives for the TOE (ADV_SPM.1.4C (10.6.4)).

ISO/IEC 15408-3:2026(en)

Assurance is provided by formally proving that the formal TSF model satisfies the formal TOE properties. To this end, the ADV_SPM (Formal TSF model) (10.6) family specifies requirements for this formal proof (ADV_SPM.1.3D (10.6.4) and ADV_SPM.1.5C (10.6.4)). The confidence gained by formally proving the properties of the formal model is accompanied by confidence gained by defining a correspondence rationale between the formal model and the TSF functional specification (ADV_SPM.1.4D (10.6.4)). The correspondence rationale consists of a formal proof when mapping to formal aspects of the TSF functional specification (ADV_SPM.1.6D (10.6.4)). It consists of a semi-formal demonstration when the functional specification is described using semi-formal style (ADV_SPM.1.5D (10.6.4)). The ADV_SPM (Formal TSF model) (10.6) family specifies content requirements for the correspondence rationale concerning the preservation of the formal TOE properties by the TSF functional specification (ADV_SPM.1.6C (10.6.4)/7C/8C).

The ADV_SPM (Formal TSF model) (10.6) family includes:

- requirements concerning the underlying mathematical theory (ADV_SPM.1.1C (10.6.4));
- the tools used for the formal modelling and proof (ADV_SPM.1.9C (10.6.4));
- requirements on the explanatory text supporting and documenting each element (ADV_SPM.1.2C (10.6.4)).

ADV_FSP (Functional specification) (10.3) requires that the developer establishes the correspondence between the TSF functional specification and the SFRs. Although this requirement is independent of the SPM, when ADV_SPM.1 (Formal TSF model) (10.6.4) is used this correspondence is a by-product of the correspondence between SFRs and formal TSF model on one hand and between the model and the functional specification on the other hand.

Figure A.4 shows the role of the formal TSF model in the relationship between the functional specification and the ST (SFRs and TOE properties), which is afterwards propagated across the design and implementation representations by means of ADV_TDS (TOE design) (10.7) and ADV_IMP (Implementation representation) (10.4) families of requirements.

Annex B **(informative)**

Composition (ACO)

B.1 General

The goal of this annex is to explain the concepts behind composition evaluations and the class ACO (Composition) ([Clause 15](#)) criteria. This annex does not define the class ASE (Security Target (ST) evaluation) ([Clause 9](#)) criteria; this definition can be found in class ASE (Security Target (ST) evaluation) ([Clause 9](#)).

B.2 Necessity for composed TOE evaluations

The IT market is made up of vendors offering a particular type of product/technology. Although there is some overlap, where a PC hardware vendor may also offer application software or operating systems or both, or a chip manufacturer may also develop a dedicated operating system for their own chipset, it is often the case that an IT solution is implemented by a variety of vendors.

There is sometimes a need for assurance in the combination (composition) of components in addition to the assurance of the individual components. Although there is cooperation between these vendors, in the dissemination of certain material required for the technical integration of the components, the agreements rarely stretch to the extent of providing detailed design information and development process/procedure evidence. This lack of information from the developer of a component on which another component relies means that the dependent component developer does not have access to the type of information necessary to perform an evaluation of both the dependent and base components at EAL2 (see ISO/IEC 15408-5:2026, 4.5) or higher EAL-levels. Therefore, while an evaluation of the dependent component can still be performed at any assurance level, to compose components with assurance at EAL2 (see ISO/IEC 15408-5:2026, 4.5) or higher EAL-levels, it is necessary to reuse the evaluation evidence and results of evaluations performed for the component developer.

It is intended that the class ACO (Composition) ([Clause 15](#)) criteria are applicable in the situation where one IT entity is dependent on another for the provision of security services. The entity providing the services is termed the base component, and that receiving the services is termed the dependent component. This relationship may exist in a number of contexts. For example, an application (dependent component) may use services provided by an operating system (base component). Alternatively, the relationship may be peer-to-peer, in the sense of two linked applications, either running in a common operating system environment, or on separate hardware platforms. If there is a dominant peer providing the services to the minor peer, the dominant peer is considered to be the base component and the minor peer the dependent component. If the peers provide services to each other in a mutual manner, each peer will be considered to be the base component for the services offered and dependent component for the services required. This will require iterations of the class ACO (Composition) ([Clause 15](#)) components applying all requirements to each type of component peer.

The criteria are also intended to be more broadly applicable, stepwise (where a composed TOE comprised of a dependent component and a base component itself becomes the base component of another composed TOE), in more complex relationships, but this may require further interpretation.

It is still required for composed TOE evaluations that the individual components are evaluated independently, as the composition evaluation builds on the results of the individual component evaluations. The evaluation of the dependent component may still be in progress when the composed TOE evaluation commences. However, the dependent component evaluation must complete before the composed TOE evaluation completes.

ISO/IEC 15408-3:2026(en)

The composed evaluation activities may take place at the same time as the dependent component evaluation. This is due to two factors:

- economic/business drivers: the dependent component developer will either be sponsoring the composition evaluation activities or supporting these activities as the evaluation deliverables from the dependent component evaluation are required for composed evaluation activities;
- technical drivers: the components consider whether the requisite assurance is provided by the base component (e.g. considering the changes to the base component since completion of the component evaluation) with the understanding that the dependent component has recently undergone (or is still undergoing) component evaluation and all evaluation deliverables associated with the evaluation are available. Therefore, there are no activities during composition requesting the dependent component evaluation activities to be re-verified. Also, it is verified that the base component forms at least one of the test configurations for the testing of the dependent component during the dependent component evaluation, leaving ACO_CTT (Composed TOE testing) (15.5) to consider the base component in this configuration.

The evaluation evidence from the evaluation of the dependent component is required input into the composed TOE evaluation activities. The only evaluation material from the evaluation of the base component that is required as input into the composed TOE evaluation activities are the Residual vulnerabilities in the base component, as reported during the base component evaluation. This is required for the ACO_VUL (Composition vulnerability analysis) (15.6) activities.

No other evaluation evidence from the base component activities should be required for the composed TOE evaluation, as the evaluation results from the component evaluation of the base component should be reused. Additional information about the base component may be required if the composed TOE TSF includes more of the base component than was considered to be TSF during component evaluation of the base component.

The component evaluation of the base and dependent components is assumed to be complete by the time final verdicts are assigned for the class ACO (Composition) (Clause 15) components.

The ACO_VUL (Composition vulnerability analysis) (15.6) components only consider resistance against an attacker with an attack potential up to Enhanced-Basic. This is due to the level of design information that can be provided of how the base component provides the services on which the dependent component relies through application of the ACO_DEV (Development evidence) (15.3) activities. Therefore, the confidence arising from composed TOE evaluations using CAPs is limited to a level similar to that obtained from EAL4 (see ISO/IEC 15408-5:2026, 4.7) component TOE evaluations. Although assurance in the components that comprise the composed TOE may be higher than EAL4 (see ISO/IEC 15408-5:2026, 4.7).

B.3 Performing Security Target (ST) evaluation for a composed TOE

An ST will be submitted by the developer for the evaluation of the composed TOE (base component and dependent component). This ST will identify the assurance package to be applied to the composed TOE, providing assurance in the composed entity by drawing upon the assurance gained in the component evaluations.

The purpose of considering the composition of components within an ST is to validate the compatibility of the components from the point of view of both the environment and the requirements, and also to assess that the composed TOE ST is consistent with the component STs and the security policies expressed within them. This includes determining that the component STs and the security policies expressed within them are compatible.

The composed TOE ST may refer out to the content of the component STs, or the ST author may choose to reiterate the material of the component STs within the composed TOE ST providing a rationale of how the component STs are represented in the composed TOE ST.

During the conduct of the ASE_CCL (Conformance claims) (9.3) evaluation activities for a composed TOE ST, the evaluator determines that the component STs are accurately represented in the composed TOE ST. This is achieved through determining that the composed TOE ST demonstrably conforms to the component

ISO/IEC 15408-3:2026(en)

TOE STs. Also, the evaluator shall determine that the dependencies of the dependent component on the operational environment are adequately fulfilled in the composed TOE.

The composed TOE description will describe the composed solution. The logical and physical scope and boundary of the composed solution will be described, and the logical boundary(ies) between the components will also be identified. The description will identify the security functionality to be provided by each component.

The statement of SFRs for the composed TOE will identify which component is to satisfy an SFR. If an SFR is met by both components, then the statement will identify which component meets the different aspects of the SFR. Similarly, the composed TOE Summary Specification will identify which component provides the security functionality described.

The package of class ASE (Security Target (ST) evaluation) ([Clause 9](#)) requirements applied to the composed TOE ST should be consistent with the package of class ASE (Security Target (ST) evaluation) ([Clause 9](#)) requirements used in the component evaluations.

Reuse of evaluation results from the evaluation of component STs can be made in the instances that the composed TOE ST directly refers to the component STs, e.g. if the composed TOE ST refers to a component ST for part of its statement of SFRs, the evaluator can understand that the requirement for the completion of all assignment and selection operations (as stated in ASE_REQ.*.3C) has been satisfied in the component evaluations.

NOTE The asterisk in "ASE_REQ.*.3C" is used in the common text matching sense as a wildcard taking any appropriate value (1,2..).

B.4 Interactions between composed IT entities

The TSF of the base component is often defined without knowledge of the dependencies of the possible applications with which it may be composed. The TSF of this base component is defined to include all parts of the base component that are expected to be relied upon for enforcement of the base component SFRs. This will include all parts of the base component required to implement the base component SFRs.

The TSFI of this base component represents the interfaces provided by the TSF to the external entities defined in the statement of SFRs to invoke a service of the TSF. This includes interfaces to the human user and interfaces to external IT entities. However, the TSFI only includes those interfaces to the TSF, and therefore is not necessarily an exhaustive interface specification of all possible interfaces available between an external entity and the base component. The base component may present interfaces to services that were not considered security-relevant, either because of the inherent purpose of the service (e.g. adjust type font) or because associated ISO/IEC 15408-2:2026 SFRs are not being claimed in the base component's ST [e.g. the login interface when no class FIA (Identification and authentication) (see ISO/IEC 15408-2:2026, Clause 12) SFRs are claimed].

The functional interfaces provided by the base component are in addition to the security interfaces (TSFIs) and are not required to be considered during the base component evaluation. These often include interfaces that are used by a dependent component to invoke a service provided by the base component.

The base component may include some indirect interfaces through which TSFIs may be called, e.g. APIs that can be used to invoke a service of the TSF, which were not considered during the evaluation of the base component.

ISO/IEC 15408-3:2026(en)

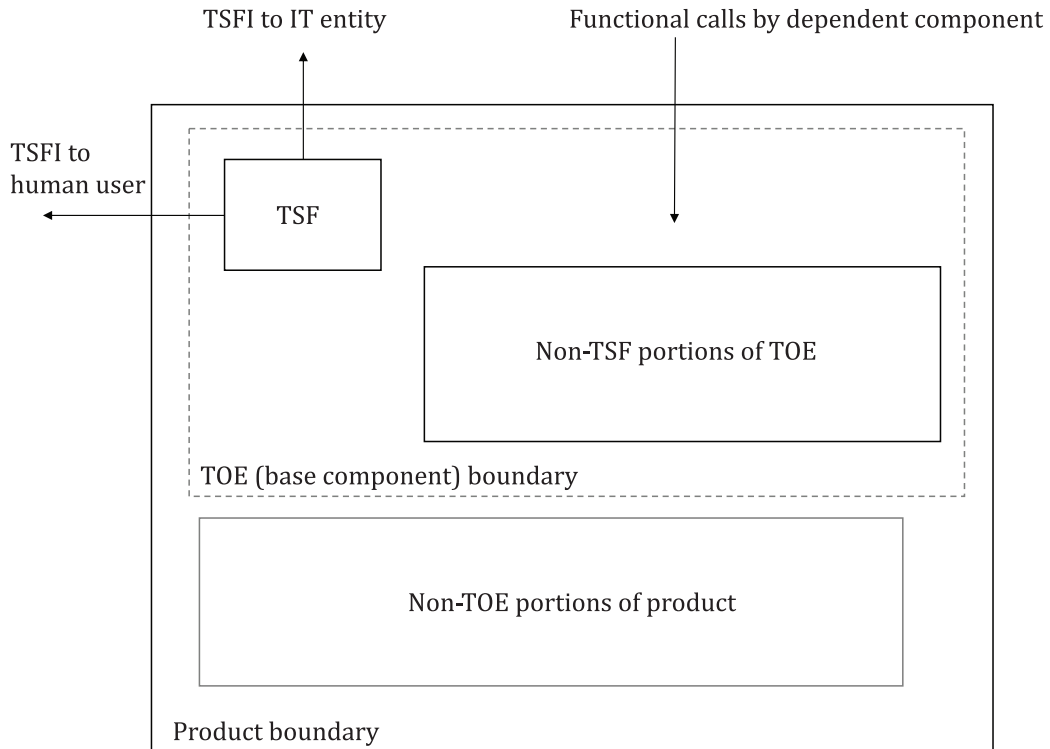


Figure B.1 — Base component abstraction

The dependent component, which relies on the base component, is similarly defined: interfaces to external entities defined in the SFRs of the component ST are categorized as TSFI and are examined in ADV_FSP (Functional specification) (10.3). This is illustrated in [Figure B.1](#).

Any call out from the dependent TSF to the environment in support of an SFR will indicate that the dependent TSF requires some service from the environment in order to satisfy the enforcement of the stated dependent component SFRs. Such a service is outside the dependent component boundary and the base component is unlikely to be defined in the dependent ST as an external entity. Hence, the calls for services made out by the dependent TSF to its underlying platform (the base component) will not be analysed as part of the ADV_FSP (Functional specification) (10.3) activities. These dependencies on the base component are expressed in the dependent component ST as security objectives for the environment.

This abstraction of the dependent component and the interfaces is shown in [Figure B.2](#) below.

ISO/IEC 15408-3:2026(en)

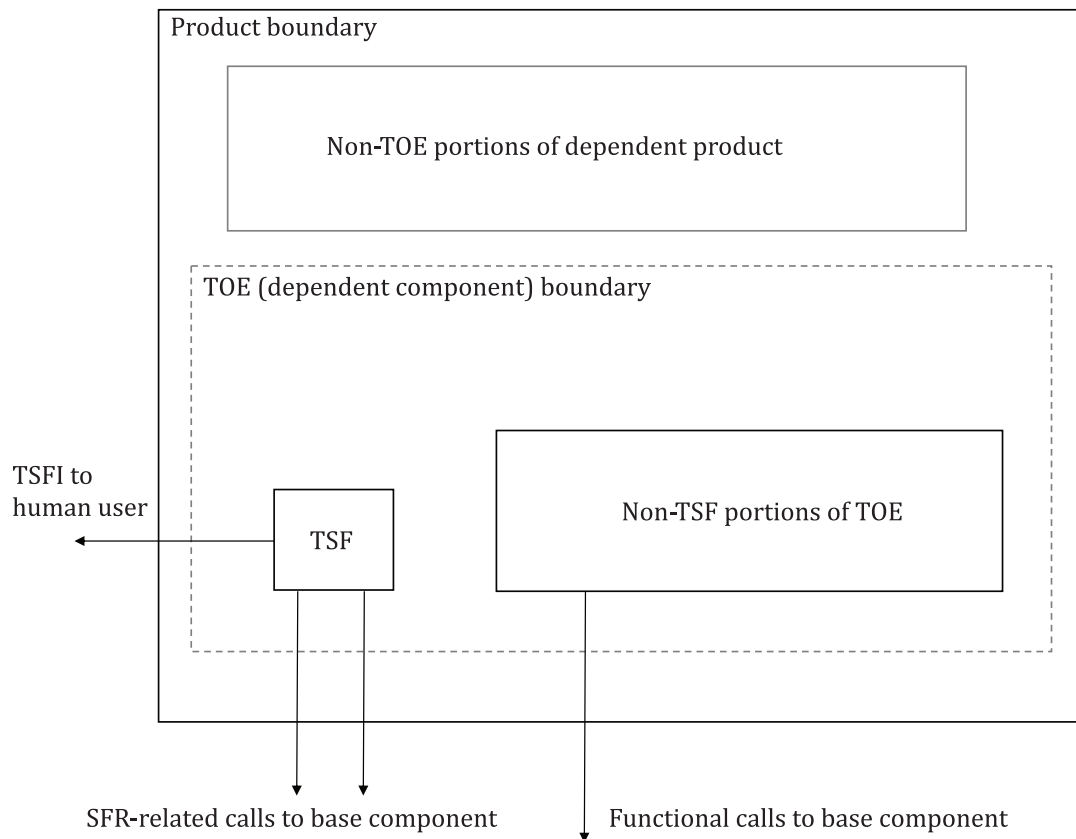


Figure B.2 — Dependent component abstraction

When considering the composition of the base component and the dependent component, if the dependent component's TSF requires services from the base component to support the implementation of the SFR, the interface to the service shall be defined. If that service is provided by the base component's TSF, then that interface should be a TSFI of the base component and will therefore already be defined within the functional specification of the base component.

If, however, the service called by the dependent component's TSF is not provided by the TSF of the base component [i.e. it is implemented in the non-TSF portion of the base component or possibly even in the non-TOE portion of the base component (not illustrated in [Figure B.3](#))], there is unlikely to be a TSFI of the base component relating to the service. This is unless the service is mediated by the TSF of the base component. The interfaces to these services from the dependent component to the operational environment are considered in the family ACO_REL (Reliance of dependent component) ([15.4](#)).

The non-TSF portion of the base component is drawn into the TSF of the composed TOE due to the dependencies that the dependent component has on the base component to support the SFRs of the dependent component. Therefore, in such cases, the TSF of the composed TOE would be larger than simply the sum of the components' TSFs.

ISO/IEC 15408-3:2026(en)

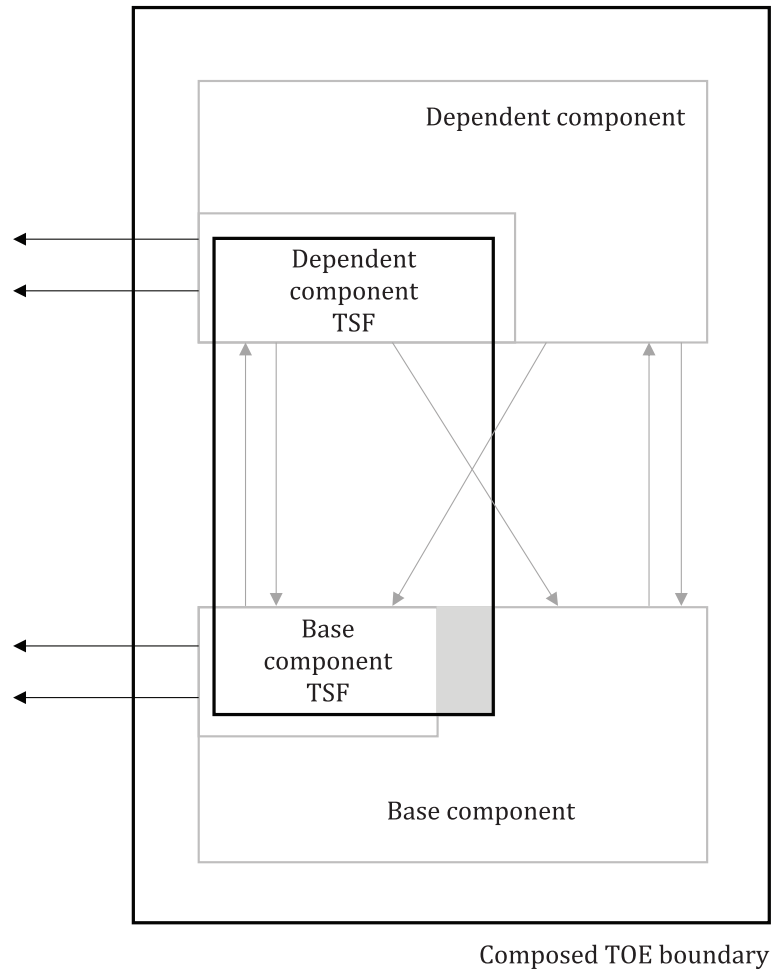
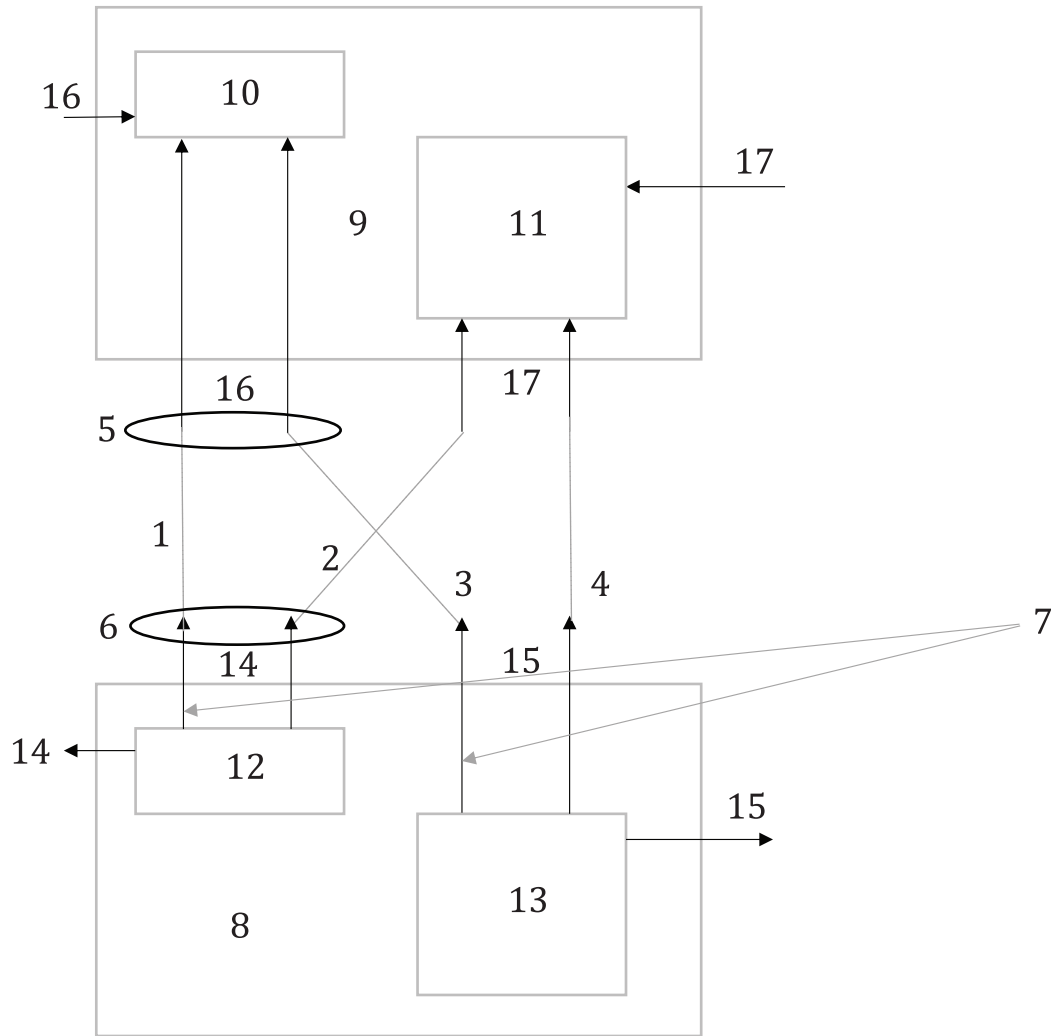


Figure B.3 — Composed TOE abstraction

It is possible that the base component TSFI is called in a manner that was unforeseen in the base component evaluation. In this case there would be a requirement for further testing of the base component TSFI.

The possible interfaces are further illustrated in [Figure B.4](#) and supporting text.

ISO/IEC 15408-3:2026(en)



Key

- 1 interface type 1
- 2 interface type 2
- 3 interface type 3
- 4 interface type 4
- 5 ACO_REL (reliance of dependent component) (15.4) (component-a)
- 6 ADV_FSP (functional specification) (10.3) (component-b)
- 7 ACO_DEV (development evidence) (15.3) (component-b)
- 8 base component-b
- 9 dependent component-a
- 10 TSF-a
- 11 non-TSF-a
- 12 TSF-b
- 13 non-TSF-b
- 14 services provided by the base component (TSF-a)
- 15 services provided by the base component (non-TSF-a)
- 16 dependent component expects environment to respond to a service request (TSF-a)
- 17 dependent component expects environment to respond to a service request (non-TSF-a)

Figure B.4 — Composed component interfaces

The following considerations apply.

ISO/IEC 15408-3:2026(en)

When there are dependent components and base components: the arrows coming out of TSF-a are services provided by TSF-a and are therefore TSFIs(a); likewise, the arrows coming out of TSF-b ("14") are TSFIs(b). These are each detailed in their respective functional specification. A dependent component is such that it requires services from its environment: those needed by the TSF(a) are labelled "16"; the other (not related to TSF-a) services are labelled "17".

When a dependent component and a base component are combined, there are four possible combinations of {services needed by the dependent component} and {services provided by the base component}, shown as broken lines (types of communication between pairs of interfaces). Any set of these can exist for a particular composition:

- TSF-a requires those services that are provided by TSF-b ("16" is connected to "14"): this is straightforward: the details about "14" are in the FSP for the base component. In this instance the interfaces should all be defined in the functional specifications for the base component;
- non-TSF-a requires those services that are provided by TSF-b ("17" is connected to "14"): this is straightforward (again, the details about "14" are in the FSP for the base component), but unimportant security wise;
- non-TSF-a requires those services that are provided by non-TSF-b ("17" is connected to "15"): there are no details available about 15, but there are no security implications about the use of these interfaces, so they must not be considered in the evaluation, although they are likely to be an integration issue for the developer;
- TSF-a requires those services that are provided by non-TSF-b ("16" is connected to "15"): this would arise when the dependent component and the base component have different senses of what a "security service" is. It is possible that the base component makes no claims about I&A [it has no class FIA (Identification and authentication) (see ISO/IEC 15408-2:2026, Clause 12) SFRs in its ST], but the dependent component requires authentication provided by its environment. There are no details about the "15" interfaces available, they are not TSFI (b), so they are not in the base component's FSP.

NOTE 1 If the kind of interaction described in case d above exists, then the TSF of the composed TOE would be TSF-a + TSF-b + non-TSF-b. Otherwise, the TSF of the composed TOE would be TSF-a + TSF-b.

Interfaces types 2 and 4 of [Figure B.4](#) are not directly relevant to the evaluation of the composed TOE. Interfaces 1 and 3 will be considered during the application of different families:

- ADV_FSP (Functional specification) ([10.3](#)) (for component-b) will describe the "14" interfaces.
- ACO_REL (Reliance of dependent component) ([15.4](#)) will describe the "16" interfaces.
- ACO_DEV (Development evidence) ([15.3](#)) will describe 14 for connection type 1 and 15 for connection type 3.

A typical example where composition can be applied is a database management system (DBMS) that relies upon its underlying operating system (OS). During the evaluation of the DBMS component, there will be an assessment made of the security properties of that DBMS (to whatever degree of rigour is dictated by the assurance components used in the evaluation). This assessment involves the following:

- its TSF boundary will be identified,
- its functional specification will be assessed to determine whether it describes the interfaces to the security services provided by the TSF,
- it is possible that additional information about the TSF (its design, architecture, internal structure) will be provided,
- the TSF will be tested,
- aspects of its life cycle and its guidance documentation will be assessed.

However, the DBMS evaluation will not call for any evidence concerning the dependency the DBMS has on the OS. The ST of the DBMS will most likely state assumptions about the OS in its Assumptions subclause

ISO/IEC 15408-3:2026(en)

and state security objectives for the OS in its Environment subclause. The DBMS ST may even instantiate those objectives for the environment in terms of SFRs for the OS. However, there will be no specification for the OS that mirrors the detail in the functional specification, architecture description, or other class ADV (Development) ([Clause 10](#)) evidence as for the DBMS. ACO_REL (Reliance of dependent component) ([15.4](#)) will fulfil that need.

ACO_REL (Reliance of dependent component) ([15.4](#)) describes the interfaces of the dependent TOE that make the calls to the base component for the provision of services. These are the interfaces to which the base component is to respond. The interface descriptions are provided from the dependent component's viewpoint.

ACO_DEV (Development evidence) ([15.3](#)) describes the interfaces provided by the base component, which respond to the dependent component service requests. These interfaces are mapped to the relevant dependent component interfaces that are identified in the reliance information.

NOTE 2 The completeness of this mapping, whether the base component interfaces described represent all dependent component interfaces, is not verified here, but in ACO_COR (Composition rationale) ([15.2](#)).

At the higher levels of ACO_DEV (Development evidence) ([15.3](#)) the subsystems providing the interfaces are described.

Any interfaces required by the dependent component that have not been described for the base component are reported in the rationale for ACO_COR (Composition rationale) ([15.2](#)). The rationale also reports whether the interfaces of the base component on which the dependent component relies were considered within the base component evaluation. For any interfaces that were not considered in the base component evaluation, a rationale is provided of the impact of using the interface on the base component TSF.

Bibliography

- [1] ISO 10007:2017, *Quality management — Guidelines for configuration management*
- [2] RFC 791, *INTERNET PROTOCOL - DARPA INTERNET PROGRAM - PROTOCOL SPECIFICATION*



ICS 35.030

Price based on 176 pages