INTERNATIONAL STANDARD

ISO/IEC 15408-3

Fourth edition
2022-08

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

Reference number
ISO/IEC 15408-3:2022(E)

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

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).

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

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*.

This fourth edition cancels and replaces the third edition (ISO/IEC 15408-3:2008), which has been technically revised.

The main changes are as follows:

— the terminology has been reviewed and updated;

— the exact conformance type has been incorporated;

— low assurance PPs have been removed and direct rationale PPs have been incorporated;

— PP-Modules and PP-Configurations for modular evaluations have been incorporated;

— multi-assurance evaluation has been incorporated.

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

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.

Legal notice

The governmental organizations listed below 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.

Australia	The Australian Signals Directorate
Canada	Communications Security Establishment
France	Agence Nationale de la Sécurité des Systèmes d'Information
Germany	Bundesamt für Sicherheit in der Informationstechnik
Japan	Information-technology Promotion Agency
Netherlands	Netherlands National Communications Security Agency
New Zealand	Government Communications Security Bureau
Republic of Korea	National Security Research Institute
Spain	Ministerio de Asuntos Económicos y Transformación Digital
Sweden	FMV, Swedish Defence Materiel Administration
United Kingdom	National Cyber Security Centre
United States	The National Security Agency

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, evaluators of secure IT products and others. ISO/IEC 15408-1:2022, Clause 5 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:

a) 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.

b) 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.

c) 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.

NOTE This document uses bold and italic type in some cases to distinguish terms from the rest of the text. The relationship between components within a family is highlighted using a bolding convention. This convention calls for the use of bold type for all new requirements. For hierarchical components, requirements are presented in bold type when they are enhanced or modified beyond the requirements of the previous component. In addition, any new or enhanced permitted operations beyond the previous component are also highlighted using bold type.

The use of italics indicates text that has a precise meaning. For security assurance requirements the convention is for special verbs relating to evaluation.

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

Part 3: Security assurance components

1 Scope

This document defines the 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 15408-1:2022, *Information security — Evaluation criteria for IT security — Part 1: Introduction and general model*

ISO/IEC 15408-2, *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, *Information security — Evaluation criteria for IT security — Part 5: Pre-defined packages of security requirements*

ISO/IEC 18045:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Methodology for IT security evaluation*

ISO/IEC IEEE 24765, *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 15408-4, ISO/IEC 15408-5, ISO/IEC 18045 and 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/

3.1
acceptance procedure
procedure followed in order to accept newly created or modified *configuration items* (3.3) 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:

a) acceptance of an item into the configuration management system for the first time, in particular as part of an integration process;

b) progression of configuration items to the next life-cycle phase at each stage of the construction of the TOE

EXAMPLE 1 Module, subsystem, quality control of the finished TOE;

c) subsequent to transport of configuration items

EXAMPLE 2 Parts of the TOE or preliminary products between different *development* (3.15) sites;

d) subsequent to the *delivery* (3.14) of the TOE to the consumer;

e) subsequent to the integration of the TOE

EXAMPLE 3 Inclusion of software, firmware and hardware components from other sources into the TOE.

3.2
action
evaluator or developer action element of ISO/IEC 15408-3

Note 1 to entry: These actions are either explicitly stated as evaluator actions or implicitly derived from developer actions (implied evaluator actions) within ISO/IEC 15408-3 assurance components.

3.3
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* (3.15)

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 (for example, files) or by reference (for example, hardware parts) together with their version.

3.4
configuration list
configuration management output (3.8) document listing all *configuration items* (3.3) 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. (Of course, the list can be an electronic document inside of a *configuration management tool* (3.12). In that 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.

3.5
configuration management
CM
discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a *configuration item* (3.3), 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 1]

3.6
configuration management documentation
CM documentation
documentation including *configuration management output* (3.8), configuration management list(s), *configuration management system records* (3.11), *configuration management plan* (3.9) and *configuration management usage documentation* (3.13)

3.7
configuration management evidence
everything that may be used to establish confidence in the correct operation of the configuration management system

EXAMPLE *Configuration management output* (3.8), rationales provided by the developer, observations, experiments, or interviews made by the evaluator during a site visit.

3.8
configuration management output
results, related to configuration management, produced, or enforced by the configuration management system

Note 1 to entry: These configuration management related results can occur as documents (e.g. filled paper forms, *configuration management system records* (3.11), logging data, hard-copies, and electronic output data) as well as actions (e.g. manual measures to fulfil configuration management instructions). Examples of such configuration management outputs are *configuration lists* (3.4), *configuration management plans* (3.9) and/or behaviours during the product life-cycle.

3.9
configuration management plan
description of how the configuration management system 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* (3.15).

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

3.10
configuration management 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.

3.11
configuration management system record
output produced during the operation of the configuration management system documenting important configuration management activities

EXAMPLE Configuration management item change control forms and configuration management item access approval forms.

3.12
configuration management tool
manually operated or automated tool realizing or supporting a configuration management system

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

3.13
configuration management usage documentation
part of the configuration management system, which describes how the configuration management system is defined and applied by using, e.g. handbooks, regulations and/or documentation of tools and procedures

3.14
delivery
transmission of the finished target of evaluation (TOE) from the *production* (3.24) environment into the hands of the customer

Note 1 to entry: This product life-cycle phase can include packaging and storage at the *development* (3.15) site, but does not include transportation of the unfinished TOE or parts of the TOE between different developers or different development sites.

3.15
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 ALC: Life-cycle support requirements, development, and related terms (developer, develop) are meant in the more general sense to comprise development and *production* (3.24).

3.16
encountered potential vulnerability
potential weakness in the target of evaluation (TOE) identified by the evaluator while performing evaluation activities that can be used to violate the security functional requirements (SFRs)

3.17
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

3.18
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

3.19
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 have to be performed by the developer, they are denoted as “generation” throughout the class ALC: Life-cycle support. If the TOE requires an initial start-up that does not need to be repeated regularly, this process would be classified as installation.

3.20
life-cycle model
framework containing the processes, activities, and tasks involved in the *development* (3.15), operation, 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 2]

3.21
operation
<TOE life-cycle> usage phase of the target of evaluation (TOE) including normal usage, administration, and maintenance of the TOE after *delivery* (3.14) and *preparation* (3.23)

3.22
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).

3.23
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* (3.19)

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

3.24
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* (3.14) to the customer

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

3.25
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

3.26
sub-activity
application of an assurance component of ISO/IEC 15408-3

Note 1 to entry: Assurance families are not explicitly addressed in the ISO/IEC 15408 series because evaluations are conducted on a single assurance component from an assurance family.

3.27
time period to exposure
time interval when an element is participating in an IT system and can be attacked

3.28
vulnerability
weakness in the TOE that can be used to violate the security functional requirements (SFRs) in some environment

3.29
window of opportunity
period of time that an attacker has access to the target of evaluation (TOE)

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, evaluation assurance levels along with their relationships, and the structure of the composed assurance packages (CAPs). It also characterizes the assurance classes and families found in Clauses 7 through 15.

Clauses 7 through 15 provide the detailed definitions of this document 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.

Annex C provides a summary of the dependencies between the assurance components.

5 Assurance paradigm

5.1 General

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

5.2 ISO/IEC 15408 series 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/or 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 is to 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 of 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 will be considered for inclusion in the ISO/IEC 15408 series, which is so structured as to allow their future introduction.

5.3.2 Significance of vulnerabilities

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

a) eliminated: active steps should be taken to expose, and remove or neutralize, all exercisable vulnerabilities;

b) minimised: active steps should be taken to reduce, to an acceptable residual level, the potential impact of any exercise of a vulnerability;

c) 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:

a) 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;

b) 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 may be face in its intended operational environment is highly dependent on an architecture that prohibits those attacks or, if they cannot be prohibited, allows for detection of such attacks and/or limitation of the damage such an attack can cause;

c) development: an IT product does not meet its specifications and/or vulnerabilities have been introduced as a result of poor development standards or incorrect design choices;

d) delivery, installation and configuration: an IT product has vulnerabilities introduced during the delivery, installation and configuration of the product;

e) 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.

f) maintenance: an IT product is maintained in such a way that new vulnerabilities are introduced.

5.3.4 ISO/IEC 15408 series 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:

a) analysis and checking of process(es) and procedure(s);

b) checking that process(es) and procedure(s) are being applied;

c) analysis of the correspondence between TOE design representations;

d) analysis of the TOE design representation against the requirements;

e) verification of proofs;

f) analysis of guidance documents;

g) analysis of functional tests developed and the results provided;

h) independent functional testing;

i) analysis for vulnerabilities (including flaw hypotheses);

j) penetration testing;

k) analysis of the delivery process;

l) analysis of the maintenance process.

5.4 ISO/IEC 15408 series 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:

a) scope: the effort is greater because a larger portion of the IT product is included;

b) depth: the effort is greater because it is deployed to a finer level of design and implementation detail;

c) rigour: the effort is greater because it is applied in a more structured, formal manner.

6 Security assurance components

6.1 General

Subclauses 6.2 to 6.6 describe the constructs used in representing the assurance classes, families, and components.

Figure 1 illustrates the security assurance requirements (SARs) defined in this document. Note that 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/ST.

6.2 Assurance class structure

6.2.1 General

Figure 1 illustrates the 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.

6.2.4 Assurance families

Each assurance class contains at least one assurance family. The structure of the assurance families is described in the following subclause.

Figure 1 illustrates the assurance family structure.

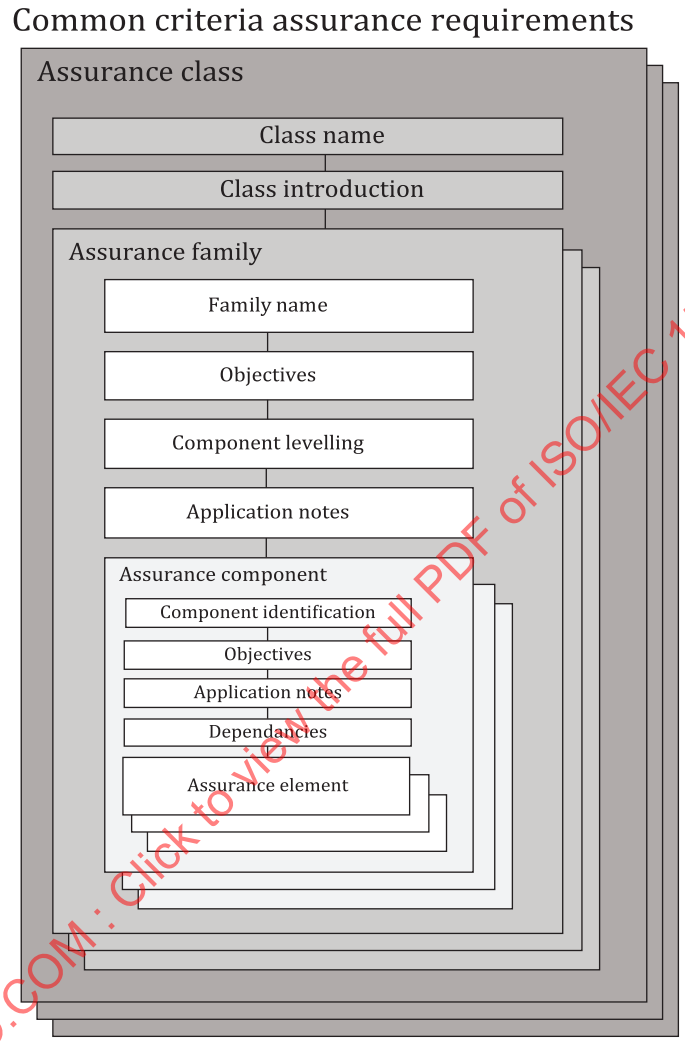


Figure 1 — Assurance class/family/component/element hierarchy

6.3 Assurance family structure

6.3.1 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.

6.3.2 Objectives

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

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.

6.3.3 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, and/or rigour.

6.3.4 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.

6.3.5 Assurance components

Each assurance family has at least one assurance component. The structure of the assurance components is provided in the following subclause.

6.4 Assurance component structure

6.4.1 General

Figure 2 illustrates the assurance component structure.

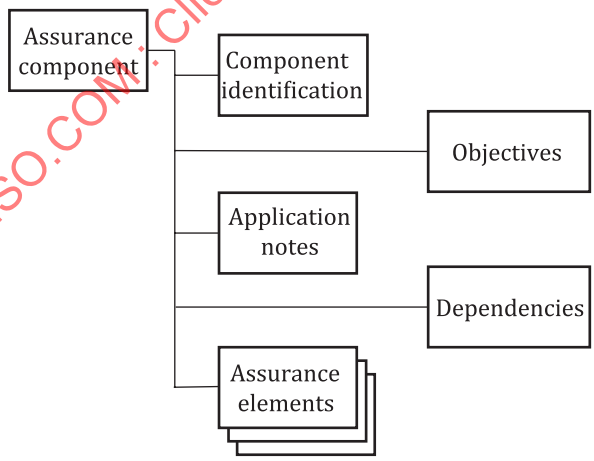


Figure 2 — Assurance component structure

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 identification

The component identification subclause 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.

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 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 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 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. 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.

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:

a) 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.

b) 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.

c) 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 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 are also to 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, in accordance with the classes ACE, APE and ASE in Clauses, ACE: ACE: Protection Profile Configuration evaluation, APE: Protection Profile evaluation and ASE: Security Target evaluation. 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.

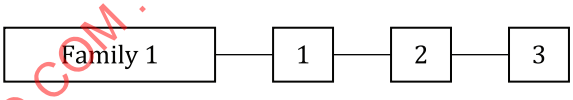


Figure 3 — Sample class decomposition diagram

In Figure 3 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 mandatory criterion for assurance families that may 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 other 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.

Clause 7 should be used in conjunction with ISO/IEC 15408-1:2022, Annexes B and D, as these annexes clarify the concepts here and provide many examples.

Figure 4 shows the families within this class, and the hierarchy of components within the families.

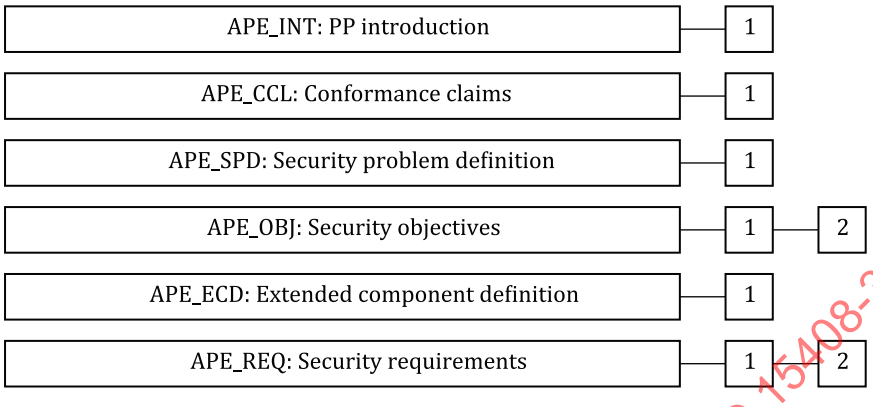


Figure 4 — APE: Protection Profile (PP) evaluation class decomposition

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 APE_INT.1 PP introduction

Dependencies: No dependencies.

Developer action elements

APE_INT.1.1D

The developer shall provide a PP introduction.

Content and presentation elements

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.

Evaluator action elements

APE_INT.1.1E

The evaluator *shall confirm* 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. In addition, this family specifies how STs and other PPs are to claim conformance with the PP.

7.3.2 APE_CCL.1 Conformance claims

Dependencies: APE_INT.1 PP introduction

APE_ECD.1 Extended components definition

APE_REQ.1 Direct rationale PP-Module security requirements

Developer action elements

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.

Content and presentation elements

APE_CCL.1.1C

The conformance claim shall identify the ISO/IEC 15408 edition 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 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/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:

- **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.**

Evaluator action elements

APE_CCL.1.1E

The evaluator *shall confirm* 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 defines 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 APE_SPD.1 Security problem definition

Dependencies: No dependencies.

Developer action elements

APE_SPD.1.1D

The developer shall provide a security problem definition.

Content and presentation elements

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.

Evaluator action elements

APE_SPD.1.1E

The evaluator *shall confirm* 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 Security problem definition (APE_SPD) 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 APE_OBJ.1 Security objectives for the operational environment

Dependencies: No dependencies.

Developer action elements

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 objectives for the operational environment.

Content and presentation elements

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.

Evaluator action elements

APE_OBJ.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

7.5.4 APE_OBJ.2 Security objectives

Dependencies: APE_SPD.1 Security problem definition.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

APE_OBJ.2.1E

The evaluator *shall confirm* 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: 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. they may not be clearly expressed using existing ISO/IEC 15408-2 or this document components.

7.6.2 APE_ECD.1 Extended components definition

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

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 ISO/IEC 15408 series components, families, and classes.

APE_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.

APE_ECD.1.5C

The extended components shall consist of measurable and objective elements such that conformance or nonconformance to these elements may be demonstrated.

Evaluator action elements

APE_ECD.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

APE_ECD.1.2E

The evaluator *shall confirm* that no extended component may be clearly expressed 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 APE_REQ.1 Direct rationale PP-Module security requirements

Dependencies: APE_ECD.1 Extended components definition

APE_OBJ.1 Security objectives for the operational environment

Developer action elements

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.

Content and presentation elements

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 environment) counter all threats for the TOE.

APE_REQ.1.8C

The security requirements rationale shall demonstrate that the SFRs (in conjunction with the security objectives for the 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.

Evaluator action elements

APE_REQ.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

7.7.4 APE_REQ.2 Derived security requirements

Dependencies: APE_OBJ.2 Security objectives

APE_ECD.1 Extended components definition

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

APE_REQ.2.1E

The evaluator *shall confirm* 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.

The class ACE is defined for the evaluation of a PP-Configuration composed of at least one PP and one other component (PPs and/or PP-Modules). The evaluation of PPs is addressed in Class APE. The class ACE defines the requirements for:

— Evaluating the PP-Modules in the framework of their PP-Modules Base(s) (components ACE_INT.1, ACE_CCL.1, ACE_SPD.1, ACE_OBJ.1 or ACE_OBJ.2, ACE_REQ.1 or ACE_REQ.2, and ACE_MCO.1).

— Evaluating the consistency of the combination of all the PPs and PP-Modules that belong to the PP-Configuration (see ACE_CCO.1).

Clause 8 should be used in conjunction with ISO/IEC 15408-1:2022, Annex C.

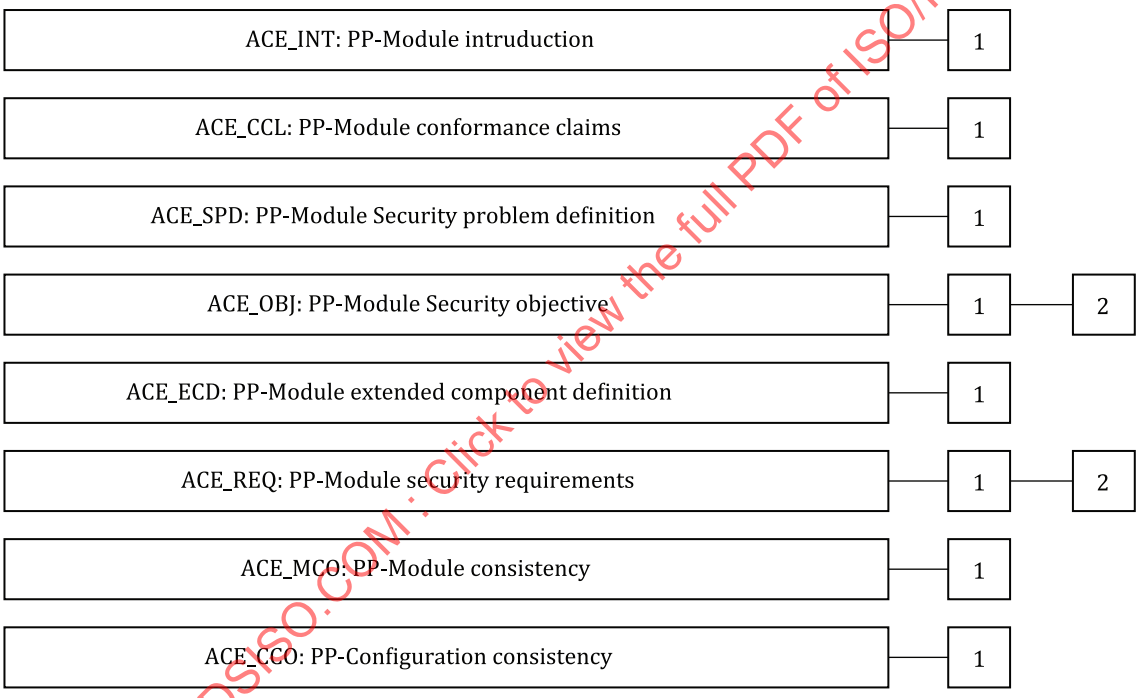


Figure 5 — ACE: Protection Profile Configuration evaluation class decomposition

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 ACE_INT.1 PP-Module introduction

Dependencies: No dependencies.

Developer action elements

ACE_INT.1.1D

The developer shall provide a PP-Module introduction.

Content and presentation elements

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).

Evaluator action elements

ACE_INT.1.1E

The evaluator *shall confirm* 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. A PP-Module cannot claim conformance to any PP, PP-Configuration, or another PP-Module.

8.3.2 ACE_CCL.1 PP-Module conformance claims

Dependencies: ACE_INT.1 PP-Module introduction

ACE_ECD.1 PP-Module extended components definition

ACE_REQ.1 PP-Module stated security requirements or ACE_REQ.2 PP-Module derived security requirements

Developer action elements

ACE_CCL.1.1D

The developer shall provide a conformance claim.

ACE_CCL.1.2D

The developer shall provide a conformance statement.

Content and presentation elements

ACE_CCL.1.1C

The conformance claim shall identify the ISO/IEC 15408 edition 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 this document 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 the PP-Module Base(s) and/or in the packages (if any) for which conformance is being claimed by the PP-Module under evaluation.

Evaluator action elements

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 defines 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 ACE_SPD.1 PP-Module security problem definition

Dependencies: No dependencies.

Developer action elements

ACE_SPD.1.1D

The developer shall provide a security problem definition.

Content and presentation elements

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.

Evaluator action elements

ACE_SPD.1.1E

The evaluator *shall confirm* 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 Security problem definition (APE_SPD) 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), or also security objectives for the TOE (see ACE_OBJ.2).

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

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ACE_OBJ.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

8.5.4 ACE_OBJ.2 PP-Module security objectives

Dependencies: ACE_SPD.1 PP-Module security problem definition.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

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: 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. they may not be clearly expressed using existing ISO/IEC 15408-2 or this document components.

8.6.2 ACE_ECD.1 PP-Module extended components definition

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

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 may be demonstrated

Evaluator action elements

ACE_ECD.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ACE_ECD.1.2E

The evaluator *shall confirm* that no extended component may be clearly expressed 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), or whether the SFRs are derived from the security objectives for the TOE (see ACE_REQ.2.).

8.7.3 ACE_REQ.1 PP-Module stated security requirements

Dependencies: APE_ECD.1 Extended components definition

ACE_SPD.1 PP-Module security problem definition

Developer action elements

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.

Content and presentation elements

ACE_REQ.1.1C

The statement of security requirements shall describe the SFRs and SARs (the SARs that apply to the PP-Module may be explicitly stated, or inherited from the PP-Module Base(s)).

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.

ACE_REQ.1.7C

The security requirements rationale shall demonstrate that the SFRs (in conjunction with the security objectives for the 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 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.

Evaluator action elements

ACE_REQ.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

8.7.4 ACE_REQ.2 PP-Module derived security requirements

Dependencies: ACE_ECD.1 PP-Module extended components definition

ACE_OBJ.2 PP-Module security objectives

Developer action elements

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 requirement rationale for the PP-Module.

Content and presentation elements

ACE_REQ.2.1C

The statement of security requirements shall describe the SFRs and SARs (the SARs that apply to the PP-Module may 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.

Evaluator action elements

ACE_REQ.2.1E

The evaluator *shall confirm* 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 ACE_MCO.1 PP-Module consistency

Dependencies: ACE_INT.1 PP-Module introduction

ACE_SPD.1 PP-Module Security problem definition

ACE_OBJ.1 Direct Rationale PP-Module Security objectives for the environment or ACE_OBJ.2 PP-Module Security objectives

ACE_REQ.1 Direct Rationale PP-Module security requirements or ACE_REQ.2 PP-Module derived security requirements

Developer action elements

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.

Content and presentation elements

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 some of its PP-Module Bases and amongst them those for which the PP-Module and the PP-Module Base define different security problems.

ACE_MCO.1.3C

The consistency rationale shall demonstrate that:

- **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:

- **the security objectives definition is consistent with the security objectives of its PP-Module Base(s);**
- **the security objectives definition is 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:

- **the security functional requirements definition is consistent with the security functional requirements of its PP-Modules Base(s);**
- **the security functional requirements definition is 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).

Evaluator action elements

ACE_MCO.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence. If the PP-Module specifies alternative PP-Module Bases, the evaluator *shall perform* 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 ACE_CCO.1 PP-Configuration consistency

Dependencies: ACE_INT.1 PP-Module introduction

ACE_CCL.1 PP-Module conformance claims

ACE_SPD.1 PP-Module security problem definition

ACE_OBJ.1 Direct Rationale PP-Module security objectives for the environment or ACE_OBJ.2 PP-Module security objectives

ACE_ECD.1 PP-Module extended component definition

ACE_REQ.1 Direct Rational PP-Module security requirements or ACE_REQ.2 PP-Module derived security requirements

ACE_MCO.1 PP-Module consistency

APE_* (all APE components)

Developer action elements

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.

ACE_CCO.1.6D

The developer shall provide a consistency rationale.

ACE_CCO.1.7D

The developer shall provide a SAR statement.

ACE_CCO.1.8D

The developer shall provide the set of evaluation methods and/or activities that are applicable to the PP-Configuration.

Content and presentation elements

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 hardware/software/firmware available to the TOE.

ACE_CCO.1.8C

The conformance claim shall identify the ISO/IEC 15408 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 this document 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.

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:

- **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.**

Evaluator action elements

ACE_CCO.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ACE_CCO.1.2E

The evaluator *shall check* 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

Evaluating an ST is required to demonstrate that the ST is sound and internally consistent, and, if the ST is based on a PP-Configuration, or one or more PPs or packages, 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.

Clause 9 should be used in conjunction with ISO/IEC 15408-1:2022, Annexes B, C and D as these annexes clarify the concepts here and provide many examples.

Figure 6 shows the families within this class, and the hierarchy of components within the families.

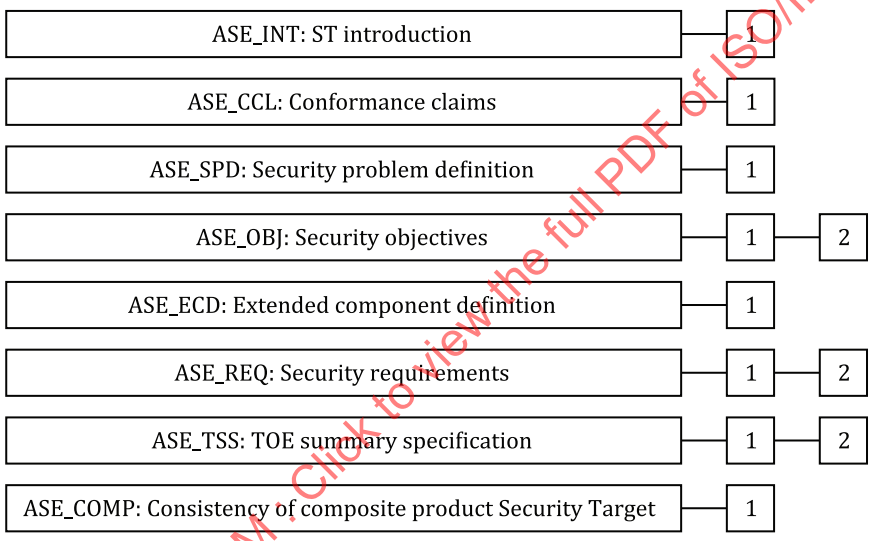


Figure 6 — ASE: Security Target (ST) evaluation class decomposition

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 ASE_INT.1 ST introduction

Dependencies: No dependencies.

Developer action elements

ASE_INT.1.1D

The developer shall provide an ST introduction.

Content and presentation elements

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/software/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.

Evaluator action elements

ASE_INT.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ASE_INT.1.2E

The evaluator *shall confirm* 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. In addition, this family specifies how STs are to claim conformance with the PP or PP-Configuration.

9.3.2 ASE_CCL.1 Conformance claims

Dependencies: ASE_INT.1 ST introduction

ASE_ECD.1 Extended components definition

ASE_REQ.1 Direct rationale stated security requirements

Developer action elements

ASE_CCL.1.1D

The developer shall provide a conformance claim.

ASE_CCL.1.2D

The developer shall provide a conformance claim rationale.

Content and presentation elements

ASE_CCL.1.1C

The conformance claim shall identify the edition of ISO/IEC 15408 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 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[1], PPs and any functional packages for which conformance is being claimed.

1) In practice, this refers to the union of SPDs defined in the PP-Configuration components.

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[2], 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[3], 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.

Evaluator action elements

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 defines 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 ASE_SPD.1 Security problem definition

Dependencies: No dependencies.

Developer action elements

ASE_SPD.1.1D

The developer shall provide a security problem definition.

Content and presentation elements

ASE_SPD.1.1C

The security problem definition shall describe the threats.

ASE_SPD.1.2C

2) In practice, this refers to the union of security objectives defined in the PP-Configuration components.

3) In practice, this refers to the union of SFRs defined in the PP-Configuration components.

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.

Evaluator action elements

ASE_SPD.1.1E

The evaluator *shall confirm* 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 Security problem definition (ASE_SPD) 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), or also security objectives for the TOE (ASE_OBJ.2).

9.5.3 ASE_OBJ.1 Security objectives for the operational environment

Dependencies: No dependencies

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ASE_OBJ.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

9.5.4 ASE_OBJ.2 Security objectives

Dependencies: ASE_SPD.1 Security problem definition

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

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: 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. they may not be clearly expressed using existing ISO/IEC 15408-2 or this document components.

9.6.2 ASE_ECD.1 Extended components definition

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

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 may be demonstrated.

Evaluator action elements

ASE_ECD.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ASE_ECD.1.2E

The evaluator *shall confirm* that no extended component may be clearly expressed 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), or whether the SFRs are derived from security objectives for the TOE (see ASE_REQ.2.).

9.7.3 ASE_REQ.1 Direct rationale security requirements

Dependencies: ASE_ECD.1 Extended components definition

Developer action elements

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.

Content and presentation elements

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.

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 demonstrate that the SFRs (in conjunction with the security objectives for the 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 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 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

Evaluator action elements

ASE_REQ.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

9.7.4 ASE_REQ.2 Derived security requirements

Dependencies: ASE_OBJ.2 Security objectives

ASE_ECD.1 Extended components definition

Developer action elements

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.

Content and presentation elements

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.

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 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 defines 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.

Evaluator action elements

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 only needs to describe how the TOE meets the SFRs, or whether the TOE summary specification also needs to describe how the TOE protects itself against logical tampering and bypass. This additional description may be used in special circumstances where there can be a specific concern regarding the TOE security architecture.

9.8.3 ASE_TSS.1 TOE summary specification

Dependencies: ASE_INT.1 ST introduction

ASE_REQ.1 Direct rationale stated security requirements

ADV_FSP.1 Basic functional specification

Developer action elements

ASE_TSS.1.1D

The developer shall provide a TOE summary specification.

Content and presentation elements

ASE_TSS.1.1C

The TOE summary specification shall describe how the TOE meets each SFR.

Evaluator action elements

ASE_TSS.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ASE_TSS.1.2E

The evaluator *shall confirm* that the TOE summary specification is consistent with the TOE overview and the TOE description.

9.8.4 ASE_TSS.2 TOE summary specification with architectural design summary

Dependencies: ASE_INT.1 ST introduction

ASE_REQ.1 Direct rationale stated security requirements

ADV_ARC.1 Security architecture description

Developer action elements

ASE_TSS.2.1D

The developer shall provide a TOE summary specification.

Content and presentation elements

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.

Evaluator action elements

ASE_TSS.2.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ASE_TSS.2.2E

The evaluator *shall confirm* 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[4] does not contradict the ST of the related base component[5],[6].

9.9.2 Component levelling

This family contains only one component.

9.9.3 Application notes

A 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. The information available in the public version of the base component ST may not be 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:

4) Denoted by composite product Security Target or composite-ST in the following.

5) Denoted by base component Security Target or base-ST in the following.

6) Generally, a Security Target expresses a security policy for the TOE defined.

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)[7) 8)].

Step 3: The developer determines under which conditions he 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 mandatory that the composite product and its related base component are being evaluated according to the same edition of ISO/IEC 15408. It is due to the fact that the dependent component of the composite product can rely on some security services of the base component, if (i) the assurance level of the base component covers the intended assurance level of the composite product and (ii) 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 series editions shall be established / 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":

composite-ST → PP 1 → PP 2 ← base-ST

PP 1 may require any kind of conformance[9)], but this does not affect the 'additional elements' that the base-ST may 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 consistency may be no direct matching: Objectives for the base component's environment may become objectives for the composite TOE.

9.9.4 ASE_COMP.1 Consistency of Security Target (ST)

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

ASE_COMP.1.1C

7) Because the TSF enforce the Security Target (together with the organisational measures enforcing the security objectives for the operational environment of the TOE).

8) The comparison shall be performed on the abstraction level of SFRs. If the developer defined security functionality groups (TSF-groups) in the TSS part of his Security Target, the evaluator should also consider them in order to get a better understanding for the context of the security services offered by the TOE.

9) e.g. "strict", "exact" or "demonstrable" according to the ISO/IEC 15408 series.

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, i.e. 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 may be provided by indicating the concerned elements directly in the composite product Security Target followed by explanatory text, if necessary.

Evaluator action elements

ASE_COMP.1.1E

The evaluator *shall confirm* 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 the AVA and ATE classes.

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, ADV_TDS, ADV_IMP and ADV_COMP).

— requirements for the description of the architecture-oriented features of domain separation, TSF self-protection and non-bypassability of the security functionality (ADV_ARC).

— requirements for a security policy model and for correspondence mappings between security policy model and the functional specification (ADV_SPM).

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

When documenting the security functionality of a TOE, there are two properties that need to 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 are structured to support these different approaches. The families Functional specification (ADV_FSP), TOE design (ADV_TDS), Implementation representation (ADV_IMP), and Security policy modelling (ADV_SPM) deal with the first property: the specification of the security functionality. The families Security Architecture (ADV_ARC) and TSF internals (ADV_INT) 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 need 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 the development class ADV. The family Composite design compliance (ADV_COMP) determines whether the requirements on the dependent component, imposed by the related base componen, are fulfilled in a composite product. Due to the distribution of the TSF of a composite product to various levels in the families of the class ADV, this family is not represented in

Figure 7. 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), and showing the implementation of the TSF. A formal model of the security behaviour also may 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/ST author to specify which TSF representations are required. The level chosen will dictate the assurance desired/gained.

Figure 7 indicates the relationships among the various TSF representations of the ADV class, as well as their relationships with other classes. As the figure indicates, the APE and ASE classes define the requirements for the correspondence between the SFRs and the security objectives for the TOE. Class ASE also defines 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 include the verification that the TSF that is tested under the ATE and AVA classes is in fact the one described by all of the ADV decomposition levels.

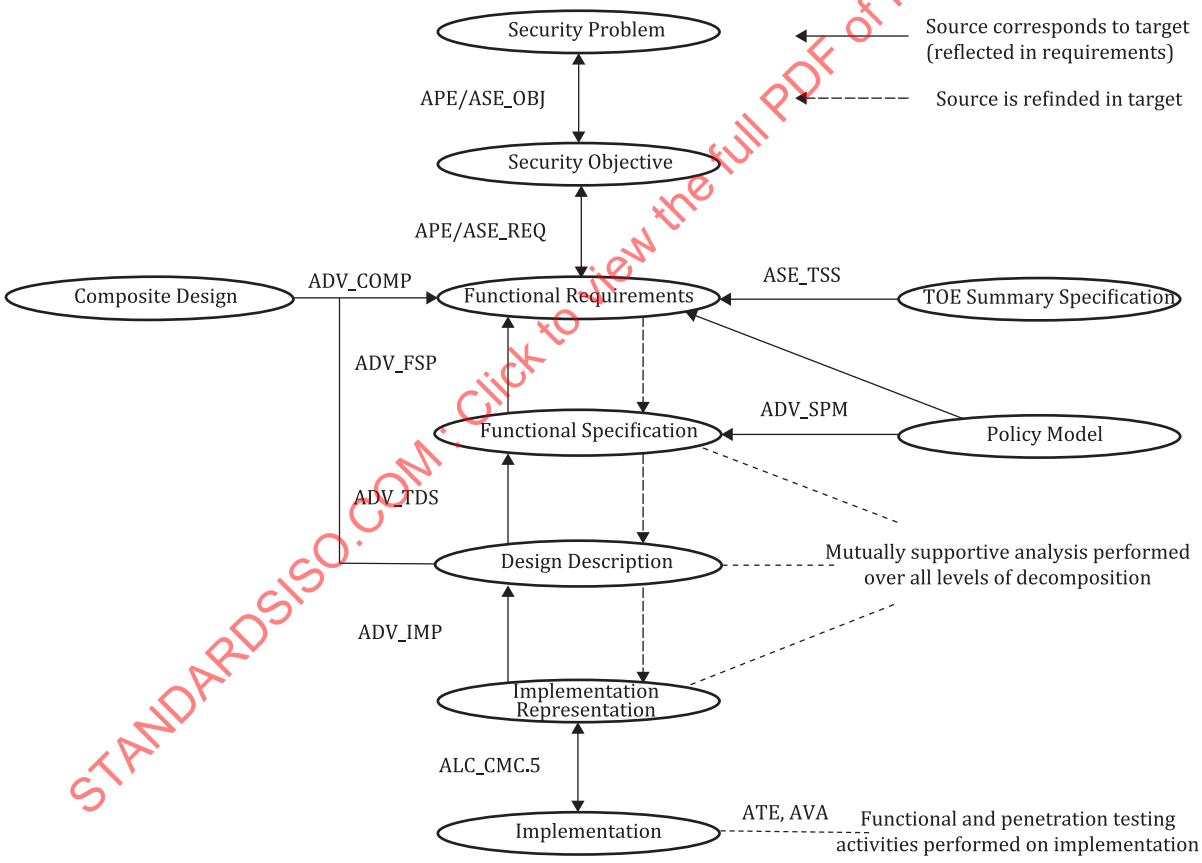


Figure 7 — Relationships of ADV constructs to one another and to other families

The requirements for all other correspondence shown in Figure 7 are defined in the ADV class for the TOE. The Security policy modelling (ADV_SPM) family defines 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. Functional specification (ADV_FSP), TOE design (ADV_TDS) and Implementation representation (ADV_IMP)) defines requirements relating that TSF representation to the SFRs. All decompositions must 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 family is not represented in this figure, 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 family is not represented in the figure because it relates to the architectural soundness, rather than representation, of the TSF. Both ADV_INT and ADV_ARC 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, 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 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 a) is correctly implemented; b) cannot be corrupted; and c) 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 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 need to be examined in detail 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 Functional specification (ADV_FSP), TOE design (ADV_TDS), and Implementation representation (ADV_IMP) 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 must 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 sort 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. Such functionality may be depended on by SFR-enforcing functionality, but the dependence is generally at a functional level; for example, memory management, buffer management, etc. 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; for example, any code running in a privileged hardware mode on an operating system. It needs to 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 architecture family [Security Architecture (ADV_ARC)] 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 functionality. Further, the depth of analysis for such properties will vary depending on the design of the TOE. The ADV families are constructed to address this by a separate family [Security Architecture (ADV_ARC)] 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 may be the case 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, semiformal and formal. The functional specification and TOE design documentation are always written in either informal or semiformal style. A semiformal style reduces the ambiguity in these documents over an informal presentation. A formal specification may 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 semiformal and informal documents is only a matter of formatting or presentation: a semiformal notation includes, e.g. an explicit glossary of terms, a standardised presentation format. A semiformal specification is written to a standard presentation template. The presentation should use terms consistently if written in a natural language. The presentation may 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 must be used in the presentation. The

glossary explicitly identifies the words that are being used in a precise and constant manner; similarly, the standardised format implies that extreme care has been taken in methodically preparing the document in a manner that maximises clarity. It should be noted that fundamentally different portions of the TSF may have different semiformal notation conventions and presentation styles (as long as the number of different "semiformal notations" is small); this still conforms to the concept of a *semiformal 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 needs to be evidence that it is impossible to derive contradictions, and all rules supporting the notation need to be defined or referenced.

Figure 8 shows the families within this class, and the hierarchy of components within the families.

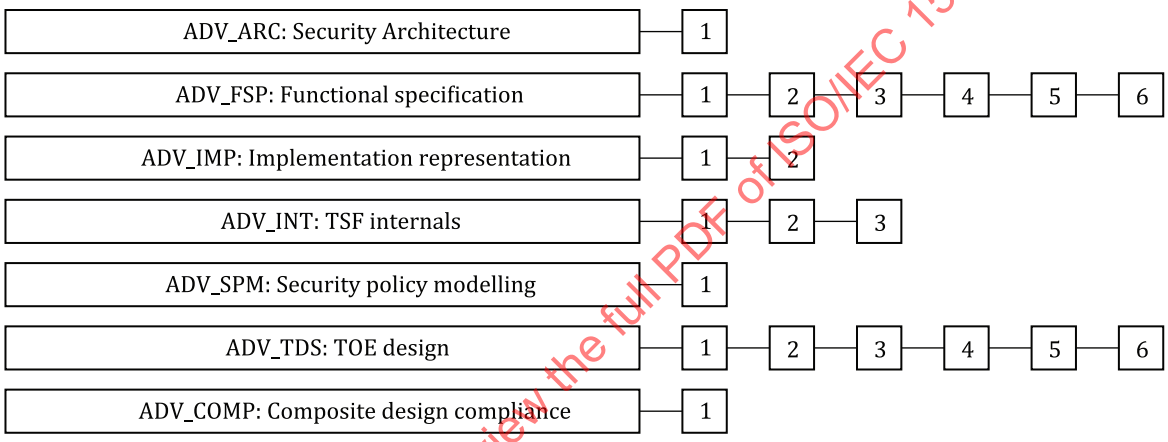


Figure 8 — ADV: Development class decomposition

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, ADV_TDS, ADV_IMP and ADV_COMP) will be presented for the **sub-TSF of** the TOE. The architecture family (Security Architecture (ADV_ARC)) provides for requirements and analysis of the TOE based on properties of domain separation, self-protection, and non-bypassability which also may hold for boundaries between the **sub-TSF**.

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 will allow analysis of the information that, when coupled with the other evidence presented for the TSF, will confirm 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 would have to 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** 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 above-mentioned properties, 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 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 the Functional specification (ADV_FSP) and TOE design (ADV_TDS)] 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 and ADV_TDS. 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 Depth (ATE_DPT) requirements; where the soundness is achieved solely through the architecture, the behaviour will be tested as part of the AVA: Vulnerability assessment 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 may 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, ADV_ARC: Supplementary material on security architectures.

10.2.4 ADV_ARC.1 Security architecture description

Dependencies: ADV_FSP.1 Basic functional specification

ADV_TDS.1 Basic design

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ADV_ARC.1.1E

The evaluator *shall confirm* 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 TOE design (ADV_TDS) and Reliance of dependent component (ACO_REL) 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, where the description of the TSFIs may be used to gain better understanding of how the TSF is protected against corruption (i.e. subversion of self-protection or domain separation) and/or bypass;

— ATE, where the description of the TSFIs is an important input for both developer and evaluator testing;

— AVA, 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.2.2 for guidance and examples of determining TSFI), they are described. At lower-level components, developers focus their documentation (and evaluators focus their analysis) on the more security-relevant aspects of the TOE. Three categories of TSFIs are defined, based upon the relevance the services available through them have to 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*. Note that it is possible that an interface may have various services and results, some of which may be SFR-enforcing and some of which may 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 must have *no* SFR-enforcing services or results. In contrast, an SFR-enforcing interface may have SFR-supporting services (for example, the ability to set the system clock may be an SFR-enforcing service of an interface, but if that same interface is used to display the system date that service may 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 categorised/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 of 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 may not manifest itself at the TSFIs, the functional specification and/or the tracing 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 TSFI 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 command shell, *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, the Windows users changing a setting in the registry, etc.) as well as behaviour at other interfaces (e.g. generating an audit record).

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 the Windows Registry; the signals across a set of pins on a chip; the flags that can be set for the *ls*, 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 an 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 one might want to use it, 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 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 should never 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, 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, 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, the developer must now, in addition to the information required at ADV_FSP.2, provide enough information about the SFR-supporting and SFR-non-interfering actions to show that they are not SFR-enforcing. Further, the developer must now document all of the direct error messages resulting from the invocation of SFR-enforcing TSFIs.

At ADV_FSP.4 Complete functional specification, all TSFIs, whether SFR-enforcing, SFR-supporting or SFR-non-interfering, must be described to the same degree, including all of the direct error messages.

At ADV_FSP.5 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, in addition to the information required by ADV_FSP.5, all remaining error messages are included. The developer must also provide a formal description of the TSFI. This provides an alternative view of the TSFI that may expose inconsistencies or incomplete specification.

10.3.4 ADV_FSP.1 Basic functional specification

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

ADV_FSP.1.1C

The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supportingTSFI.

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.

Evaluator action elements

ADV_FSP.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.1.2E

The evaluator *shall determine* that the functional specification is an accurate and complete instantiation of the SFRs.

10.3.5 ADV_FSP.2 Security-enforcing functional specification

Dependencies: ADV_TDS.1 Basic design

Developer action elements

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.

Content and presentation elements

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** TSFI.

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.

Evaluator action elements

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 ADV_FSP.3 Functional specification with complete summary

Dependencies: ADV_TDS.1 Basic design

Developer action elements

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.

Content and presentation elements

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 TSFI.

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.

Evaluator action elements

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 ADV_FSP.4 Complete functional specification

Dependencies: ADV_TDS.1 Basic design

Developer action elements

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.

Content and presentation elements

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 TSFI.

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 may 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.

Evaluator action elements

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

Dependencies: ADV_TDS.1 Basic design

ADV_IMP.1 Implementation representation of the TSF

Developer action elements

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.

Content and presentation elements

ADV_FSP.5.1C

The functional specification shall completely represent the TSF.

ADV_FSP.5.2C

The functional specification shall describe the TSFI using a semi-formal style.

ADV_FSP.5.3C

The functional specification shall describe the purpose and method of use for all TSFI.

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 may 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.

Evaluator action elements

ADV_FSP.5.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.5.2E

The evaluator *shall determine* that the functional specification is an accurate and complete instantiation of the SFRs.

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

Dependencies: ADV_TDS.1 Basic design

ADV_IMP.1 Implementation representation of the TSF

Developer action elements

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.

Content and presentation elements

ADV_FSP.6.1C

The functional specification shall completely represent the TSF.

ADV_FSP.6.2C

The functional specification shall describe the TSFI using a **formal** style.

ADV_FSP.6.3C

The functional specification shall describe the purpose and method of use for all TSFI.

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 may 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 TSFI 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.

Evaluator action elements

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 Implementation representation (ADV_IMP) 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 may be software source code, firmware source code, hardware diagrams and/or IC hardware design language code or layout data.

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 or hardware diagrams and/or IC hardware design language code or 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 must be made available to the evaluator, this does not imply that the evaluator needs to possess that representation. For instance, the developer may 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 of 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 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 may 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, etc.) 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 may 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. Because of the uncertainty of which portions of the implementation representation will need 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 may 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 may 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 may 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 may 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 understanding and/or analysis. 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 may be closer to the implementation (in terms of structure) than the original, un-shrouded representation, supplying such obfuscated code may 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/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 ADV_IMP.1 Implementation representation of the TSF

Dependencies: ADV_TDS.3 Basic modular design

ALC_TAT.1 Well-defined development tools

Developer action elements

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.

Content and presentation elements

ADV_IMP.1.1C

The implementation representation shall define the TSF to a level of detail such that the TSF may 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.

Evaluator action elements

ADV_IMP.1.1E

The evaluator *shall confirm* that, for the selected sample of the implementation representation, the information provided meets all requirements for content and presentation of evidence.

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

Dependencies: ADV_TDS.3 Basic modular design

ALC_TAT.1 Well-defined development tools

ALC_CMC.5 Advanced support

Developer action elements

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.

Content and presentation elements

ADV_IMP.2.1C

The implementation representation shall define the TSF to a level of detail such that the TSF may 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.

Evaluator action elements

ADV_IMP.2.1E

The evaluator *shall confirm* 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 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 should assist the developer in simplifying its maintainability.

The requirements in this family are presented at a fairly 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 ADV_INT.1 Well-structured subset of TSF internals

Dependencies: ADV_IMP.1 Implementation representation of the TSF

ADV_TDS.3 Basic modular design

ALC_TAT.1 Well-defined development tools

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.

Application notes

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

a) 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.");

b) 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.").

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* 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).

Developer action elements

ADV_INT.1.1D

The developer shall design and implement [assignment: *subset of the TSF*] such that it has well-structured internals.

ADV_INT.1.2D

The developer shall provide an internals description and justification.

Content and presentation elements

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.

Evaluator action elements

ADV_INT.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ADV_INT.1.2E

The evaluator *shall perform* an internals analysis on the assigned subset of the TSF.

10.5.5 ADV_INT.2 Well-structured internals

Dependencies: ADV_IMP.1 Implementation representation of the TSF

ADV_TDS.3 Basic modular design

ALC_TAT.1 Well-defined development tools

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.

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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

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 ADV_INT.3 Minimally complex internals

Dependencies: ADV_IMP.1 Implementation representation of the TSF

ADV_TDS.3 Basic modular design

ALC_TAT.1 Well-defined development tools

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.

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.

Developer action elements

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.

Content and presentation elements

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.**

Evaluator action elements

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 Security policy modelling (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 and its properties, as defined by the SFRs and the security objectives of the ST, further referred to as the formal model and the formal properties, respectively. It is expected to establish by means of a formal proof that these formal properties hold in the formal model and to establish by means of a correspondence rationale that the TOE functional specification preserves the formal properties proven for the formal model. A formal proof or semiformal demonstration of preservation of the formal properties in the formal or semiformal specification is expected if the latter exists (ADV_FSP.5 or ADV_FSP.6, 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 may be problematic because the definition must 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 may 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 must 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 must 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). The correspondence rationale consists of a formal proof when mapping to formal aspects of the TOE functional specification and semiformal demonstration otherwise. A combination of different formal systems (modelling languages, tools, proof systems) can be used for different parts of the ST (SFRs & Security Objectives) and correspondence rationales.

10.6.4 ADV_SPM.1 Formal TOE security policy model

Dependencies: ASE_OBJ.2 Security Objectives

ASE_REQ.2 Derived security requirements

ADV_FSP.4 Complete functional specification

Developer action elements

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.

Content and presentation elements

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 modeling 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.

Evaluator action elements

ADV_SPM.1.1E

The evaluator *shall confirm* 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 should be able to 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 may 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. Very simple TOEs, in contrast, might 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 (ATE: Tests).

In the requirements for this family, the term *interface* is used as the means of communication (between two subsystems or modules). It describes how the communication is invoked; this is similar to the details of TSFI [see Functional specification (ADV_FSP)]. 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 define collections of details about subsystems and modules to be provided:

a) The subsystems and modules are *identified* with a simple list of what they are.

b) Subsystems and modules may be *categorised* (either implicitly or explicitly) as "SFR-enforcing", "SFR-supporting", or "SFR-non-interfering"; these terms are used the same as they are used in Functional specification (ADV_FSP).

c) A subsystem's *behaviour* is what it does. The behaviour may also be categorised as SFR-enforcing, SFR-supporting, or SFR-non-interfering. The behaviour of the subsystem is never categorised 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.

d) 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").

e) A *behaviour description* of a subsystem is an explanation of everything it does. This description should be at a level of detail that one can readily determine whether the behaviour has any relevance to the enforcement of the SFRs.

f) 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 need not 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.

g) 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.

h) 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.

i) 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, ADV_TDS: Subsystems and Modules.

10.7.4 ADV_TDS.1 Basic design

Dependencies: ADV_FSP.2 Security-enforcing functional specification.

Developer action elements

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 TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

Content and presentation elements

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.

Evaluator action elements

ADV_TDS.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ADV_TDS.1.2E

The evaluator *shall determine* that the design is an accurate and complete instantiation of all security functional requirements.

10.7.5 ADV_TDS.2 Architectural design

Dependencies: ADV_FSP.3 Functional specification with complete summary.

Developer action elements

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 TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

Content and presentation elements

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.

Evaluator action elements

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 ADV_TDS.3 Basic modular design

Dependencies: ADV_FSP.4 Complete functional specification.

Developer action elements

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 TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

Content and presentation elements

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.

Evaluator action elements

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.

10.7.7 ADV_TDS.4 Semiformal modular design

Dependencies: ADV_FSP.5 Complete semi-formal functional specification with additional error information

Developer action elements

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 TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

Content and presentation elements

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 **semiformal** 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.

Evaluator action elements

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 ADV_TDS.5 Complete semiformal modular design

Dependencies: ADV_FSP.5 Complete semi-formal functional specification with additional error information.

Developer action elements

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 TSFI of the functional specification to the lowest level of decomposition available in the TOE design.

Content and presentation elements

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 semiformal 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 semiformal 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.

Evaluator action elements

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 ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation

Dependencies: ADV_FSP.6 Complete semi-formal functional specification with additional formal specification

Developer action elements

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 TSFI 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.

Content and presentation elements

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 semiformal 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 **semiformal 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.

Evaluator action elements

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,

— 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 the development class ADV. 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), security architecture (ADV_ARC) and the implementation (ADV_IMP). In case that, these design representation levels are not available (e.g. due to the assurance package chosen is EAL1), 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 functional specification (ADV_FSP) as representation level is not appropriate for analysing the design compliance.

Security architecture ADV_ARC as assurance family is dedicated to ensure 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, the composite product evaluator shall:

i. determine whether the dependent component uses services of the related base document 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; if yes, then

ii. the evaluator shall determine, whether the dependent component uses these services of the base component in an appropriate/secure way (please refer to the base component user guidance).

As consistency of the composite product security policy has already been considered in the context of the ST in the assurance family ASE_COMP, there is no necessity to consider non-contradictoriness of the security policy model (ADV_SPM) of the composite product and the security policy model of its related base component.

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

Dependencies: No dependencies.

Developer action elements

ADV_COMP.1.1D

The developer shall provide a design compliance justification.

Content and presentation elements

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.

Evaluator action elements

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 has to 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 has to be done during the operation of the TOE in its evaluated configuration).

Figure 9 shows the families within this class, and the hierarchy of components within the families.

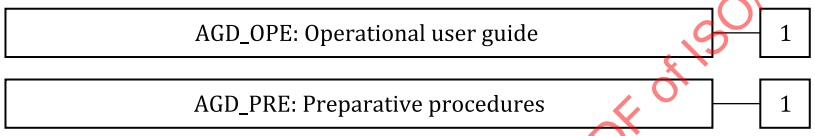


Figure 9 — AGD: Guidance documents class decomposition

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: end-users, persons responsible for maintaining and administering the TOE in a correct manner for maximum security, and by others (e.g. programmers) using the TOE's external interfaces. Operational user guidance describes the security functionality provided by the TSF, provides instructions and guidelines (including warnings), helps to understand the TSF and 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 should 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 may 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 should 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 can be a single one.

The requirement AGD_OPE.1.1C 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/ST are appropriately covered in the user guidance.

The concept of secure values, as employed in AGD_OPE.1.3C, has relevance where a user has control over security parameters. Guidance needs to be provided on secure and insecure settings for such parameters.

AGD_OPE.1.4C 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 need not always be the case (e.g. the audit log fills up, an intrusion is detected). Furthermore, a security-relevant event may happen as a result of a specific chain of functions or, conversely, several security-relevant events may be triggered by one function.

AGD_OPE.1.7C requires that the user guidance is clear and reasonable. Misleading or unreasonable guidance may 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 may 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 AGD_OPE.1 Operational user guidance

Dependencies: ADV_FSP.1 Basic functional specification.

Developer action elements

AGD_OPE.1.1D

The developer shall provide operational user guidance.

Content and presentation elements

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 need to 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.

Evaluator action elements

AGD_OPE.1.1E

The evaluator *shall confirm* 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 has to 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 has to be ensured otherwise.

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 may be inappropriate to require and analyse installation procedures.

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

11.3.4 AGD_PRE.1 Preparative procedures

Dependencies: No dependencies.

Developer action elements

AGD_PRE.1.1D

The developer shall provide the TOE including its preparative procedures.

Content and presentation elements

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.

Evaluator action elements

AGD_PRE.1.1E

The evaluator *shall confirm* 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.

The ALC class consists of nine families:

— Development Life-cycle definition (ALC_LCD) 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;

— CM capabilities (ALC_CMC) provides requirements for the management of the configuration items;

— CM scope (ALC_CMS) requires a minimum set of configuration items to be managed in the defined way;

— Developer environment security (ALC_DVS) is concerned with the developer's physical, logical, procedural, personnel, and other security controls;

— Tools and techniques (ALC_TAT) provides requirements for the development tools and implementation standards used by the developer;

— Flaw remediation (ALC_FLR) provides requirements for the handling of security flaws;

— Delivery (ALC_DEL) provides requirements for the procedures used for the delivery of the TOE to the downstream user. Delivery processes occurring during the development of the TOE are denoted rather as transfers, and are handled in the context of integration and acceptance procedures in other families of this class;

— ALC_TDA is concerned with the generation of certain artefacts during the development process;

— ALC_COMP 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 10 shows the families within this class, and the hierarchy of components within the families.

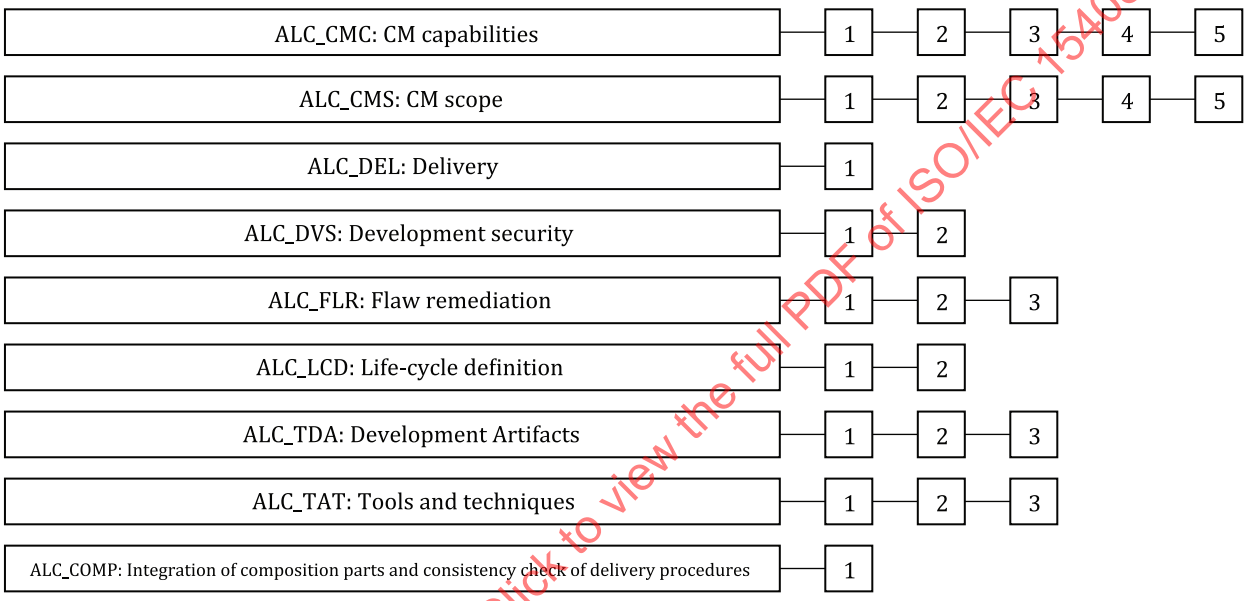


Figure 10 — ALC: Life-cycle support class decomposition

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 intented to reduce the likelihood that accidental or unauthorised modifications of the configuration items will occur. The CM system should 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 unauthorised modification, automated systems are less susceptible to human error or negligence.

The objectives of this family include the following:

a) ensuring that the TOE is identifiable and complete before it is sent to the downstream user;

b) ensuring that no configuration items are missed during evaluation;

c) preventing unauthorised 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 product as a whole.

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. CM scope (ALC_CMS) can be used to narrow this discretion by identifying specific items that must be included in the configuration list, and hence within the scope of the overall CM system.

ALC_CMC.2.3C 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 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 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 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 still needs to provide an automated means to ascertain the changes between the TOE and a future version of the TOE.

12.2.4 ALC_CMC.1 Labelling of the TOE

Dependencies: ALC_CMS.1 TOE CM coverage

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.

Developer action elements

ALC_CMC.1.1D

The developer shall provide the TOE and a unique reference for the TOE.

Content and presentation elements

ALC_CMC.1.1C

The TOE shall be labelled with its unique reference.

Evaluator action elements

ALC_CMC.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.2.5 ALC_CMC.2 Use of the CM system

Dependencies: ALC_CMS.1 TOE CM coverage

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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_CMC.2.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.2.6 ALC_CMC.3 Authorization controls

Dependencies: ALC_CMS.1 TOE CM coverage

ALC_DVS.1 Identification of security measures

ALC_LCD.1 Developer defined life-cycle processes

A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE. Aspects of the process that may 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,

— Development Life-cycle definition (ALC_LCD),

— acceptance subsequent to internal transportations in Developer environment security (ALC_DVS),

— acceptance of parts into the CM system in CM capabilities (ALC_CMC), and

— acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

The first three types may 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.

A life-cycle model provides for the necessary control over the development and maintenance of the TOE, if the model enables sufficient minimisation 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 and/or metrics) 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, which are used to increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the security of the TOE.

One should take into account that there exist standardised life-cycle models on the one hand (like the waterfall model) and standardised metrics on the other hand (like error density), which may be combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard defining both aspects.

ALC_LCD.1 Developer defined life-cycle processes

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 unauthorised 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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_CMC.3.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.2.7 ALC_CMC.4 Production support, acceptance procedures and automation

Dependencies: ALC_CMS.1 TOE CM coverage

ALC_DVS.1 Identification of security measures

ALC_LCD.1 Developer defined life-cycle processes

A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE. Aspects of the process that may 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,

— Development Life-cycle definition (ALC_LCD),

— acceptance subsequent to internal transportations in Developer environment security (ALC_DVS),

— acceptance of parts into the CM system in CM capabilities (ALC_CMC), and

— acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

The first three types may 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.

A life-cycle model provides for the necessary control over the development and maintenance of the TOE, if the model enables sufficient minimisation 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 and/or metrics) 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, which are used to increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the security of the TOE.

One should take into account that there exist standardised life-cycle models on the one hand (like the waterfall model) and standardised metrics on the other hand (like error density), which may be combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard defining both aspects.

ALC_LCD.1 Developer defined life-cycle processes

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 access controls to help ensure that unauthorised 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 a CM system where the quantity and organization of configuration items is complex, it is difficult to control changes without the support of automated tools. In particular, these automated tools need to 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. 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 have to be carried out.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_CMC.4.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.2.8 ALC_CMC.5 Advanced support

Dependencies: ALC_CMS.1 TOE CM coverage

ALC_DVS.2 Sufficiency of security measures

ALC_LCD.1 Developer defined life-cycle processes

A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the TOE. Aspects of the process that may 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,

— Development Life-cycle definition (ALC_LCD),

— acceptance subsequent to internal transportations in Developer environment security (ALC_DVS),

— acceptance of parts into the CM system in CM capabilities (ALC_CMC), and

— acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

The first three types may 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.

A life-cycle model provides for the necessary control over the development and maintenance of the TOE, if the model enables sufficient minimisation 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 and/or metrics) 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, which are used to increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the security of the TOE.

One should take into account that there exist standardised life-cycle models on the one hand (like the waterfall model) and standardised metrics on the other hand (like error density), which may be combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard defining both aspects.

ALC_LCD.1 Developer defined life-cycle processes

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 unauthorised 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 meet defined criteria in regard to the integrity of the TOE. Criteria for acceptance procedures may include code review, checking for vulnerabilities, authenticity checking, and functional testing 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 need to 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 has to take place, the use of automatic tools is adequate.

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 have to be carried out.

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

Providing an automated means of ascertaining changes between versions of the TOE and identifying which configuration items are affected by modifications to other configuration items assists 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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_CMC.5.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ALC_CMC.5.2E

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

12.3.4 ALC_CMS.1 TOE CM coverage

Dependencies: No dependencies.

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 and the evaluation evidence required by the other SARs in the ST under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

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 CM capabilities (ALC_CMC).

Developer action elements

ALC_CMS.1.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements

ALC_CMS.1.1C

The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.1.2C

The configuration list shall uniquely identify the configuration items.

Evaluator action elements

ALC_CMS.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.3.5 ALC_CMS.2 Parts of the TOE CM coverage

Dependencies: No dependencies.

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 authorisations.

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 CM capabilities (ALC_CMC).

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

Developer action elements

ALC_CMS.2.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements

ALC_CMS.2.1C

The configuration list shall include the following: the TOE itself; the evaluation evidence required by the SARs; **and 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.

Evaluator action elements

ALC_CMS.2.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.3.6 ALC_CMS.3 Implementation representation CM coverage

Dependencies: No dependencies.

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 authorisations.

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 CM capabilities (ALC_CMC).

Developer action elements

ALC_CMS.3.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements

ALC_CMS.3.1C

The configuration list shall include the following: the TOE itself; the evaluation evidence required by the SARs; the parts that comprise the TOE; **and 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.

Evaluator action elements

ALC_CMS.3.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.3.7 ALC_CMS.4 Problem tracking CM coverage

Dependencies: No dependencies.

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 authorisations.

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

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 CM capabilities (ALC_CMC). This requires that information regarding previously identified security flaw reports and their resolution be maintained.

Developer action elements

ALC_CMS.4.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements

ALC_CMS.4.1C

The configuration list shall include the following: the TOE itself; the evaluation evidence required by the SARs; the parts that comprise the TOE; the implementation representation; **and 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.

Evaluator action elements

ALC_CMS.4.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.3.8 ALC_CMS.5 Development tools CM coverage

Dependencies: No dependencies.

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 authorisations.

Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that access to them is managed, further, it may 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.

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 CM capabilities (ALC_CMC). 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.

Developer action elements

ALC_CMS.5.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements

ALC_CMS.5.1C

The configuration list shall include the following: 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; **and 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.

Evaluator action elements

ALC_CMS.5.1E

The evaluator *shall confirm* 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/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 Vulnerability analysis (AVA_VAN) 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 Developer environment security (ALC_DVS).

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

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

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

a) ensuring that the TOE received by the consumer corresponds precisely to the evaluated version of the TOE;

b) avoiding or detecting any tampering with the actual version of the TOE;

c) preventing submission of a counterfeit version of the TOE;

d) avoiding unwanted knowledge of distribution of the TOE to the consumer: there can be cases where potential attackers should not know when and how it is delivered;

e) avoiding or detecting the TOE being intercepted during delivery; and

f) 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 Preparative procedures (AGD_PRE) family, if present.

12.4.4 ALC_DEL.1 Delivery procedures

Dependencies: No dependencies.

Developer action elements

ALC_DEL.1.1D

The developer shall document and provide procedures for delivery of the TOE or parts of it to the consumer.

ALC_DEL.1.2D

The developer shall use the delivery procedures.

Content and presentation elements

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 consumer.

Evaluator action elements

ALC_DEL.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.5 Developer environment security (ALC_DVS)

12.5.1 Objectives

Development 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.

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 development security. This may include sites of subcontractors involved in the TOE development and production. Any decision not to visit shall be agreed with the evaluation authority.

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

It is recognized that confidentiality may not always be 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 ALC_DVS.1 Identification of security controls

Dependencies: No dependencies.

Developer action elements

ALC_DVS.1.1D

The developer shall produce and provide development security documentation.

Content and presentation elements

ALC_DVS.1.1C

The development 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.

Evaluator action elements

ALC_DVS.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.1.2E

The evaluator *shall confirm* that the security controls are being applied.

12.5.5 ALC_DVS.2 Sufficiency of security controls

Dependencies: No dependencies.

Developer action elements

ALC_DVS.2.1D

The developer shall produce and provide development security documentation.

Content and presentation elements

ALC_DVS.2.1C

The development security documentation shall describe all the physical, 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 development security documentation shall justify that the security controls provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

Evaluator action elements

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 may be an organisational 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 may 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 may be reported by the developer, by users of the TOE, or by other parties with familiarity with the TOE. Some flaws may not be reparable immediately. There may be some occasions where a flaw cannot be fixed and other (e.g. procedural) controls must 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 may 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 ALC_FLR.1 Basic flaw remediation

Dependencies: No dependencies.

Developer action elements

ALC_FLR.1.1D

The developer shall document and provide flaw remediation procedures addressed to TOE developers.

Content and presentation elements

ALC_FLR.1.1C

The flaw remediation procedures documentation 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 flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

Evaluator action elements

ALC_FLR.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.6.5 ALC_FLR.2 Flaw reporting procedures

Dependencies: No dependencies.

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, TOE users need 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.

Developer action elements

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.

Content and presentation elements

ALC_FLR.2.1C

The flaw remediation procedures documentation 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 flaw remediation procedures documentation 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.

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.

Evaluator action elements

ALC_FLR.2.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.6.6 ALC_FLR.3 Systematic flaw remediation

Dependencies: No dependencies.

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, TOE users need to understand how to submit security flaw reports to the developer, and how to register themselves with the developer so that they may receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures that TOE users are aware of this important information.

Developer action elements

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.

Content and presentation elements

ALC_FLR.3.1C

The flaw remediation procedures documentation 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 flaw remediation procedures documentation 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 might 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.

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 may 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.

Evaluator action elements

ALC_FLR.3.1E

The evaluator *shall confirm* 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.

Adopting a life-cycle model, or models that meets the needs of the developer's organization will improve the 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 may 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,

— Life-cycle definition (ALC_LCD),

— acceptance subsequent to internal transportations in Development security (ALC_DVS),

— acceptance of parts into the CM system in CM capabilities (ALC_CMC), and

— acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

The first three types may 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.

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 and/or metrics) 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, which are used to increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the security of the TOE.

One should take into account that there exist standardised life-cycle models on the one hand (like the waterfall model) and standardised metrics on the other hand (like error density), which may be combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard defining both aspects.

12.7.4 ALC_LCD.1 Developer defined life-cycle processes

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_LCD.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.7.5 ALC_LCD.2 Measurable life-cycle model

Dependencies: No dependencies.

Developer action elements

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.

Content and presentation elements

ALC_LCD.2.1C

The life-cycle definition documentation shall describe the model used to develop and maintain the TOE **including the details of its arithmetic parameters and/or metrics used to measure the quality of the TOE and/or its development.**

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.

Evaluator action elements

ALC_LCD.2.1E

The evaluator *shall confirm* 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 in the development processes and/or the TOE itself.

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 family. As ALC_TAT 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 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 question "is that really the source code for this software" or "is that really the register-transfer level (RTL) design or description for this integrated circuit hardware"" or "is that really the set of implementation representation for this TOE", which is potentially relevant in an evaluation. Such degree of trust is built on

a) the timing of when the set of implementation representation identifiers is recorded or logged,

b) the integrity and authenticity of the record of implementation representation identifiers, and

c) 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, the requirements in ALC_TDA.2 make sure that these implementation representation elements actually are identifiable through the use of the implementation representation identifiers of ALC_TDA.1.

With the accurate recording or logging of the actual implementation representation being used by the development tools under the scope of ALC_TAT, 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 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 ALC_TDA.1 Uniquely identifying implementation representation

Dependencies: No dependencies.

Developer action elements

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

a) **the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;**

b) **the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;**

c) **the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;**

d) **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;**

e) **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.**

Content and presentation elements

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 may be the name of an affiliate of an organization.

Evaluator action elements

ALC_TDA.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ALC_TDA.1.2E

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.1.3E

The evaluator *shall confirm* that 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 *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.1.5E

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.1.6E

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.

12.8.5 ALC_TDA.2 Matching CMS scope of implementation representation

Dependencies: ALC_CMS.3 Implementation representation CM coverage

Developer action elements

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

a) the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;

b) the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;

c) the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;

d) 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;

e) 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 are identified by the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

Content and presentation elements

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 may 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 shall match with the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

Evaluator action elements

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 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 matches with the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

12.8.6 ALC_TDA.3 Regenerate TOE with well-defined development tools

Dependencies: ALC_CMS.3 Implementation representation CM coverage

ALC_TAT.1 Well-defined development tools and

ADV_IMP.1 Implementation representation of the TSF

Developer action elements

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

a) the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;

b) the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;

c) the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;

d) 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;

e) 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 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 the functional differences, if any, between the TOE copy and the original TOE.

ALC_TDA.3.9D

The developer shall produce and provide documentation explaining the functional differences, if any, between the regenerated TOE copy and the original TOE.

Content and presentation elements

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 may 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 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 the functional differences, if any, between the regenerated TOE copy and the original TOE shall take into account all visible differences, if any, between the regenerated TOE copy and the original TOE

Evaluator action elements

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 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 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 the functional differences, if any, between the regenerated TOE copy and the original TOE takes into account all visible differences, if any, 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 standards bodies are considered 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 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 and ALC_TAT.3, implementation guidelines may 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 practise in a specific industry, but developer-specific implementation guidelines may 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) and the implementation standards for "all parts of the TOE" (ALC_TAT.3.3D) which include third party software, hardware, or firmware. The configuration list introduced in CM scope (ALC_CMS) 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 ALC_TAT.1 Well-defined development tools

Dependencies: ADV_IMP.1 Implementation representation of the TSF.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_TAT.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

12.9.5 ALC_TAT.2 Compliance with implementation standards

Dependencies: ADV_IMP.1 Implementation representation of the TSF.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

ALC_TAT.2.1E

The evaluator *shall confirm* 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 ALC_TAT.3 Compliance with implementation standards - all parts

Dependencies: ADV_IMP.1 Implementation representation of the TSF

Developer action elements

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.**

Content and presentation elements

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.

Evaluator action elements

ALC_TAT.3.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.3.2E

The evaluator *shall confirm* that the implementation standards have been applied.

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 / 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 / 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 may include, amongst other, the configuration list of the base component developer (e.g. provided within his 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 may include, amongst other, the element of evidence for the dependent component reception, acceptance and parameterisation by the base component developer (e.g. in form of his acknowledgement statement).

12.10.4 ALC_COMP.1 Integration of the dependent component into the related base component and Consistency check for delivery and acceptance procedures

Dependencies: No dependencies

Developer action elements

ALC_COMP.1.1D

The developer shall provide components configuration evidence.

Content and presentation elements

ALC_COMP.1.1C

The components configuration evidence shall show that the evaluated version of the dependent component has been installed onto / embedded into the evaluated version of the related base component.

ALC_COMP.1.2C

The components configuration evidence shall show that:

a) **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.**

b) **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.**

Evaluator action elements

ALC_COMP.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ALC_COMP.1.2E

The evaluator *shall confirm* 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: Coverage (ATE_COV), Depth (ATE_DPT), Independent testing (ATE_IND) (i.e. functional testing performed by evaluators), Functional tests (ATE_FUN) and Composite functional testing (ATE_COMP). 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 the AVA: Vulnerability assessment class.

The ATE: Tests class separates testing into developer testing and evaluator testing. The Coverage (ATE_COV), and Depth (ATE_DPT) families address the completeness of developer testing. Coverage (ATE_COV) addresses the rigour with which the functional specification is tested; Depth (ATE_DPT) addresses whether testing against other design descriptions (security architecture, TOE design, and implementation representation) is required.

Functional tests (ATE_FUN) addresses the performing of the tests by the developer and how this testing should be documented. Finally, Independent testing (ATE_IND) then addresses evaluator testing: whether the evaluator should repeat part or all of the developer testing and how much independent testing the evaluator should do.

Composite functional testing (ATE_COMP) determines whether the composite product as a whole exhibits the properties necessary to satisfy the functional requirements of its ST.

Figure 11 shows the families within this class, and the hierarchy of components within the families.

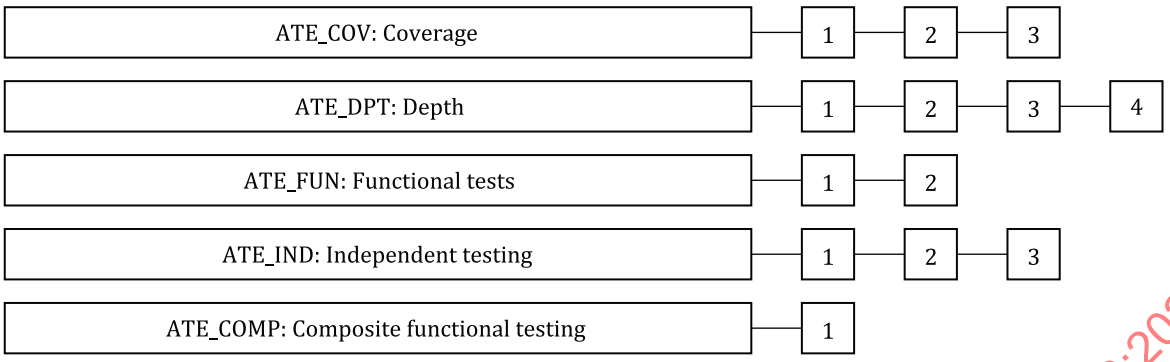


Figure 11 — ATE: Tests class decomposition

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.

13.2.2 Component levelling

The components in this family are levelled on the basis of specification.

13.2.3 Application notes

13.2.4 ATE_COV.1 Evidence of coverage

Dependencies: ADV_FSP.2 Security-enforcing functional specification

ATE_FUN.1 Functional testing

Objectives

The objective of this component is to establish that some of the TSFIs have been tested.

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, perhaps using a table.

Developer action elements

ATE_COV.1.1D

The developer shall provide evidence of the test coverage.

Content and presentation elements

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.

Evaluator action elements

ATE_COV.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

13.2.5 ATE_COV.2 Analysis of coverage

Dependencies: ADV_FSP.2 Security-enforcing functional specification

ATE_FUN.1 Functional testing

Objectives

The objective of this component is to confirm that all of the TSFIs have been tested.

Application notes

In this component the developer confirms that tests in the test documentation correspond to all of the TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps using a table, but the developer also provides an analysis of the test coverage.

Developer action elements

ATE_COV.2.1D

The developer shall provide **an analysis** of the test coverage.

Content and presentation elements

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.

Evaluator action elements

ATE_COV.2.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

13.2.6 ATE_COV.3 Rigorous analysis of coverage

Dependencies: ADV_FSP.2 Security-enforcing functional specification

ATE_FUN.1 Functional testin

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 of the TSFIs have been tested.

Application notes

In this component the developer is required to show how tests in the test documentation correspond to all of the TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps 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.

Developer action elements

ATE_COV.3.1D

The developer shall provide an analysis of the test coverage.

Content and presentation elements

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.

Evaluator action elements

ATE_COV.3.1E

The evaluator *shall confirm* that the information provided meets all 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 the ADV class.

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 must show that the internal interfaces have been exercised and seen to behave as described. This may 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 should either be justification that these aspects need not be tested, or the internal interface needs to be tested directly. In the latter case the TOE design needs to be sufficiently detailed in order to facilitate direct testing.

In cases where the description of the TSF's architectural soundness [in Security Architecture (ADV_ARC)] cites specific mechanisms, the tests performed by the developer must 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 ATE_DPT.1 Testing: basic design

Dependencies: ADV_ARC.1 Security architecture description

ADV_TDS.2 Architectural design

ATE_FUN.1 Functional testing

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.

Developer action elements

ATE_DPT.1.1D

The developer shall provide the analysis of the depth of testing.

Content and presentation elements

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.

Evaluator action elements

ATE_DPT.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

13.3.5 ATE_DPT.2 Testing: security enforcing modules

Dependencies: ADV_ARC.1 Security architecture description

ADV_TDS.3 Basic modular design

ATE_FUN.1 Functional testing

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.

Developer action elements

ATE_DPT.2.1D

The developer shall provide the analysis of the depth of testing.

Content and presentation elements

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.

Evaluator action elements

ATE_DPT.2.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

13.3.6 ATE_DPT.3 Testing: modular design

Dependencies: ADV_ARC.1 Security architecture description

ADV_TDS.4 Semiformal modular design

ATE_FUN.1 Functional testing

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.

Developer action elements

ATE_DPT.3.1D

The developer shall provide the analysis of the depth of testing.

Content and presentation elements

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.

Evaluator action elements

ATE_DPT.3.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

13.3.7 ATE_DPT.4 Testing: implementation representation

Dependencies: ADV_ARC.1 Security architecture description

ADV_TDS.4 Semiformal modular design

ADV_IMP.1 Implementation representation of the TSF

ATE_FUN.1 Functional testing

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.

Developer action elements

ATE_DPT.4.1D

The developer shall provide the analysis of the depth of testing.

Content and presentation elements

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.

Evaluator action elements

ATE_DPT.4.1E

The evaluator *shall confirm* that the information provided meets all 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 Coverage (ATE_COV) and Depth (ATE_DPT) families.

This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively small.

The families Coverage (ATE_COV), Depth (ATE_DPT) and Functional tests (ATE_FUN) 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 Independent testing (ATE_IND).

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 should 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 ATE_FUN.1 Functional testing

Dependencies: ATE_COV.1 Evidence of coverage.

Objectives

The objective is for the developer to demonstrate that the tests in the test documentation are performed and documented correctly.

Developer action elements

ATE_FUN.1.1D

The developer *shall test* the TSF and document the results.

ATE_FUN.1.2D

The developer shall provide test documentation.

Content and presentation elements

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.

Evaluator action elements

ATE_FUN.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

13.4.5 ATE_FUN.2 Ordered functional testing

Dependencies: ATE_COV.1 Evidence of coverage.

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.

Application notes

Although the test procedures may state pre-requisite initial test conditions in terms of ordering of tests, they may not 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.

Developer action elements

ATE_FUN.2.1D

The developer *shall test* the TSF and document the results.

ATE_FUN.2.2D

The developer shall provide test documentation.

Content and presentation elements

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.

Evaluator action elements

ATE_FUN.2.1E

The evaluator *shall confirm* that the information provided meets all 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 ATE_FUN, ATE_COV, and ATE_DPT families by verifying the developer testing and performing additional tests 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 must 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 his 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 will also need to consider the scope for devising additional tests, and the relative benefit that may be gained from effort in these two areas. It is recognized that repetition of all developer tests may be feasible and desirable in some cases, but may be very arduous and less productive in others. The highest component in this family should therefore be used with caution. Sampling will address the whole range of test results available, including those supplied to meet the requirements of both Coverage (ATE_COV) and Depth (ATE_DPT).

There is also a need to consider the different configurations of the TOE that are included within the evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his own 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 may be based on other considerations. For example, the version of the TOE submitted by the developer may not be 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 Coverage (ATE_COV) and the Depth (ATE_DPT) 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 ATE_IND.1 Independent testing - conformance

Dependencies: ADV_FSP.1 Basic functional specification

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

Objectives

In this component, the objective is to demonstrate that the TOE operates in accordance with its design representations and guidance documents.

Application notes

This component does not address the use of developer test results. It is applicable where such results are not available, and also in cases 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 needs to be considered in the context of a particular TOE and the balance of other evaluation activities.

Developer action elements

ATE_IND.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements

ATE_IND.1.1C

The TOE shall be suitable for testing.

Evaluator action elements

ATE_IND.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.1.2E

The evaluator *shall test* a subset of the TSF to confirm that the TSF operates as specified.

13.5.5 ATE_IND.2 Independent testing - sample

Dependencies: ADV_FSP.2 Security-enforcing functional specification

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_COV.1 Evidence of coverage

ATE_FUN.1 Functional testing

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.

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 such things as machine-readable test documentation, test programs, etc.

This component contains a requirement that the evaluator has available test results from the developer to supplement the programme of testing. The evaluator will repeat a sample of the developer's tests to gain confidence in the results obtained. Having established such confidence the evaluator will 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 is able to 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 will also have more freedom, where appropriate, to concentrate testing in areas where examination of documentation or specialist knowledge has raised particular concerns.

Developer action elements

ATE_IND.2.1D

The developer shall provide the TOE for testing.

Content and presentation elements

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.

Evaluator action elements

ATE_IND.2.1E

The evaluator *shall confirm* that the information provided meets all 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 ATE_IND.3 Independent testing - complete

Dependencies: ADV_FSP.4 Complete functional specification

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_COV.1 Evidence of coverage

ATE_FUN.1 Functional testing

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 of the developer tests.

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, e.g. machine-readable test documentation, test programs.

In this component the evaluator must repeat all of the developer's tests as part of the programme of testing. As in the previous component the evaluator will 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 may remain little scope for this.

Developer action elements

ATE_IND.3.1D

The developer shall provide the TOE for testing.

Content and presentation elements

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.

Evaluator action elements

ATE_IND.3.1E

The evaluator *shall confirm* that the information provided meets all 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 its base component and its dependent component are being tested independently of each other. A lot of tests of the base component may have been performed within the scope of its accomplished evaluation. The dependent component may 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: 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 effectiveness of this testing may not be 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 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. Please note that 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.

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 class ATE. These integration tests shall be specified and performed, whereby the approach of the standard assurance families of the class ATE shall be applied.

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

— composite product samples suitable for testing.

13.6.4 ATE_COMP.1 Composite product functional testing

Dependencies: No dependencies

Developer action elements

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.

Content and presentation elements

ATE_COMP.1.1C

Content and presentation of the specification and documentation of the *integration* tests shall correspond to the standard[10] requirements of the assurance families ATE_FUN and ATE_COV.

ATE_COMP.1.2C

The composite product provided shall be suitable for testing.

Evaluator action elements

ATE_COMP.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

14 Class AVA: Vulnerability assessment

14.1 General

The AVA: Vulnerability assessment class addresses the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE.

Figure 12 shows the families within this class, and the hierarchy of components within the families.

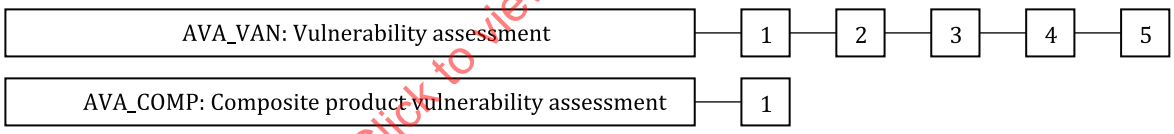


Figure 12 — AVA: Vulnerability assessment class decomposition

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 must 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. Basically, all development vulnerabilities can be considered in the context of AVA_VAN due to the fact, that this family allows application of a wide range of assessment methodologies being unspecific to the kind of an attack scenario. These unspecific assessment methodologies comprise, among other, also the specific methodologies for those TSF where covert channels are to be considered (a channel capacity estimation can be done using informal engineering measurements, as well as actual test measurements) or can be

10) i.e. as defined by ISO/IEC 18045.

overcome by the use of sufficient resources in the form of a direct attack (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 should be considered during the conduct of the vulnerability analysis. This is often reflected by the inclusion of Unobservability (FPR_UNO) and multilevel access control policies specified through Access control policy (FDP_ACC) and/or Information flow control policy (FDP_IFC) 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 unauthorised access to data and functionality, allow the ability to interfere with or alter the TSF, or interfere with the authorized capabilities of other users.

In case of a **multi-assurance evaluation** the vulnerability analysis shall assess the defined **sub-TSF** as well as the TOE as a whole.

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 AVA_VAN.1 Vulnerability survey

Dependencies: ADV_FSP.1 Basic functional specification

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

Objectives

A vulnerability survey of information available in the public domain is performed by the evaluator to ascertain potential vulnerabilities that may 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.

Developer action elements

AVA_VAN.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements

AVA_VAN.1.1C

The TOE shall be suitable for testing.

Evaluator action elements

AVA_VAN.1.1E

The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2E

The evaluator *shall perform* a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3E

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 Basic attack potential.

14.3.4 AVA_VAN.2 Vulnerability analysis

Dependencies: ADV_ARC.1 Security architecture description

ADV_FSP.2 Security-enforcing functional specification

ADV_TDS.1 Basic design

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

AVA_VAN.2.1E

The evaluator *shall confirm* 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 that the TOE depends on.**

AVA_VAN.2.3E

The evaluator *shall perform* an independent vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design and security architecture description to identify potential vulnerabilities in the TOE.

AVA_VAN.2.4E

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 Basic attack potential.

14.3.5 AVA_VAN.3 Focused vulnerability analysis

Dependencies: ADV_ARC.1 Security architecture description

ADV_FSP.4 Complete functional specification

ADV_TDS.3 Basic modular design

ADV_IMP.1 Implementation representation of the TSF

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_DPT.1 Testing: basic design

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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

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.

14.3.6 AVA_VAN.4 Methodical vulnerability analysis

Dependencies: ADV_ARC.1 Security architecture description

ADV_FSP.4 Complete functional specification

ADV_TDS.3 Basic modular design

ADV_IMP.1 Implementation representation of the TSF

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_DPT.1 Testing: basic design

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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

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.

14.3.7 AVA_VAN.5 Advanced methodical vulnerability analysis

Dependencies: ADV_ARC.1 Security architecture description

ADV_FSP.4 Complete functional specification

ADV_TDS.3 Basic modular design

ADV_IMP.1 Implementation representation of the TSF

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_DPT.1 Testing: basic design

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.

Developer action elements

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.

Content and presentation elements

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.

Evaluator action elements

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[11] assurance family of the class AVA: AVA_VAN.

The composite product evaluator shall perform a vulnerability analysis for the composite product using, amongst other, 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/installed onto the base component is consistent with the confidentiality level claimed by the dependent component developer for ALC_DVS.

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 may also be used in a way that was not foreseen by the base component developer and the base component evaluator, or the dependent component developer may not have 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 may consult the base component evaluator and draw on his experience gained during the base component evaluation. Alternatively, an approach aiming on 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 may load test dependent components on his own discretion. The intention hereby 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 under the following conditions: they are up-to-date and all composite activities for correctness – ASE_COMP.1, ALC_COMP.1, ADV_COMP.1 and ATE_COMP.1 – are finalised with the verdict PASS.

Due to composing of the base component and the dependent component a new quality arises, which may cause additional vulnerabilities of the base component which might be not mentioned in the *ETR for composite evaluation*. In these circumstances the composite product evaluation authority may require a re-assessment or re-evaluation of the base component focusing on the new vulnerabilities' issues.

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 AVA_COMP.1 Composite product vulnerability assessment

Dependencies: No dependencies

Developer action elements

AVA_COMP.1.1D

The developer shall provide the composite product for penetration testing.

Content and presentation elements

AVA_COMP.1.1C

The composite product provided shall be suitable for testing as a whole.

11) i.e. as defined by ISO/IEC 18045.

Evaluator action elements

AVA_COMP.1.1E

The evaluator *shall conduct* 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

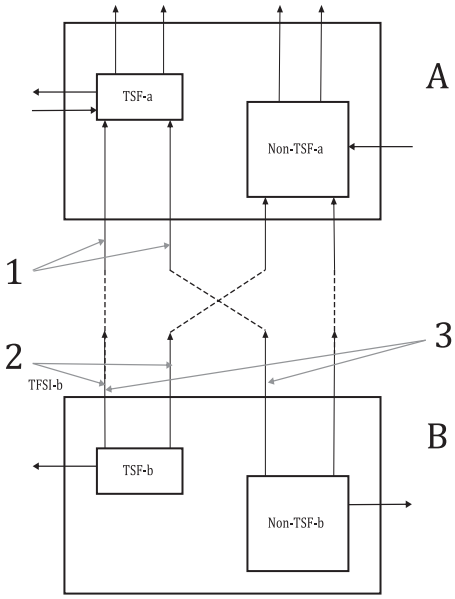
15.1 General

The class ACO: Composition encompasses five families. These families specify assurance requirements that are designed to provide confidence that a composed TOE will 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 ISO/IEC 15408 series security assurance requirements packages (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 environment.

This approach does not provide an alternative approach for the evaluation of components. Composition under ACO 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 the ACO class, 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 have to 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.



Key

A dependent component-a
B base component-b
1 ACO_REL (component-a)
2 ADV_FSP (component-b)
3 ACO_DEV (component-b)

Figure 13 — 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. 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 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 the ACO class interact in a similar manner to the ADV, ATE and AVA classes 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 dependent component has upon the underlying base component are identified (ACO_REL). 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. The ACO_DEV family is based on the ADV_TDS 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 are incomplete. This is determined during the conduct of ACO_COR. The ACO_COR family takes the outputs of ACO_REL and ACO_DEV 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) and vulnerability analysis (ACO_VUL) 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, and at higher levels 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 families is depicted in Figure 14 below. This shows by solid arrowed lines where the evidence and understanding gained in one family feeds into the next activity and the dashed arrows identify where an activity explicitly traces back to the composed TOE SFRs, as described above.

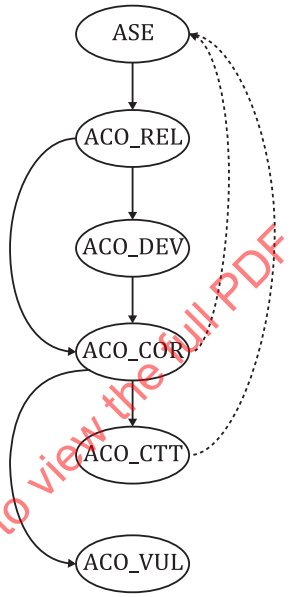


Figure 14 — Relationship between ACO families

Further discussion of the definition and interactions within composed TOEs is provided in Annex B.

Figure 15 shows the families within this class, and the hierarchy of components within the families.

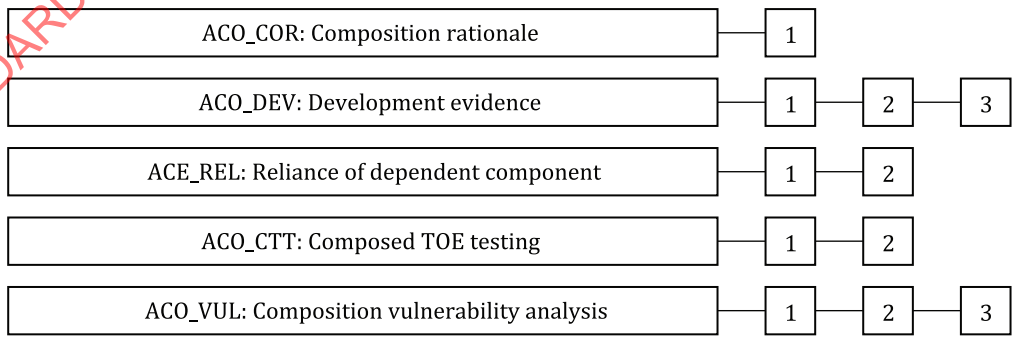


Figure 15 — ACO: Composition class decomposition

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.

15.2.3 ACO_COR.1 Composition rationale

Dependencies: ACO_DEV.1 Functional Description

ALC_CMC.1 Labelling of the TOE

ACO_REL.1 Basic reliance information

Developer action elements

ACO_COR.1.1D

The developer shall provide composition rationale for the base component.

Content and presentation elements

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.

Evaluator action elements

ACO_COR.1.1E

The evaluator *shall confirm* 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 may by composed. The TSF of this base component is defined to include all

parts of the base component that have 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 functional specification of the base component will 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 environment. The description of what a dependent component TSF relies upon of a base component is considered in Reliance of dependent component (ACO_REL) and the development information evidence provides a response to the interfaces specified.

The development information evidence includes a specification of the base component. This may be the evidence used during evaluation of the base component to satisfy the ADV requirements, or may 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 Development evidence (ACO_DEV) 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 ACO_DEV.1 Functional Description

Dependencies: ACO_REL.1 Basic reliance information

Objectives

A description of the interfaces in the base component, on which the dependent component relies, is required. This is examined to determine whether or not it is consistent with the description of interfaces on which the dependent component relies, as provided in the reliance information.

Developer action elements

ACO_DEV.1.1D

The developer shall provide development information for the base component.

Content and presentation elements

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.

Evaluator action elements

ACO_DEV.1.1E

The evaluator *shall confirm* that the information meets all requirements for content and presentation of evidence.

ACO_DEV.1.2E

The evaluator *shall determine* that the interface description provided is consistent with the reliance information provided for the dependent component.

15.3.5 ACO_DEV.2 Basic evidence of design

Dependencies: ACO_REL.1 Basic reliance information

Objectives

A description of the interfaces in the base component, on which the dependent component relies, is required. This is examined to determine whether or not 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.

Developer action elements

ACO_DEV.2.1D

The developer shall provide development information for the base component.

Content and presentation elements

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.

Evaluator action elements

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 ACO_DEV.3 Detailed evidence of design

Dependencies: ACO_REL.2 Reliance information.

Objectives

A description of the interfaces in the base component, on which the dependent component relies, is required. This is examined to determine whether or not 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 or not that interface formed part of the TSF of the base component.

Developer action elements

ACO_DEV.3.1D

The developer shall provide development information for the base component.

Content and presentation elements

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.

Evaluator action elements

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 may not have been analysed during evaluation of the individual components, as the interfaces were not TSFIs of the individual component TOEs.