Federal Office
for Information Security

Guidelines for Developer Documentation

according to Common Criteria Version 3.1

Version 1.0

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Phone: +49 (0)3018 9582- 111
Fax: +49 (0)3018 9582-5400
E-Mail: zerti@bsi.bund.de
Internet: http://www.bsi.bund.de

Table of Contents

List of tables

1 Introduction to this Guidance

This chapter provides an overview about the purpose, the content, and the structure of this guidance.

1.1 Target Audience

This guidance aims at developer personnel or consultants who prepare for a security evaluation according to the Common Criteria (CC) or are for other reasons interested in learning about the corresponding requirements.

1.2 Purpose of this Guidance

The Common Criteria are internationally well recognised standards for the evaluation of products incorporating security functionality. The broad acceptance is strongly supported by the fact that the Common Criteria have been developed with the claim that security evaluations can be performed according to the same principles based on the same requirements under the control of various National Schemes.

To allow to achieve the goal of mutual recognition of evaluation results, the requirements on the evaluation process and the documentation and other evidence to be provided as input for the evaluation process have to be clear, technically sound and detailed. The criteria, specifically the Common Evaluation Methodology (CEM), thus contain detailed information which is directed primarily to evaluators. The structure of CEM has been optimised to serve as an evaluation directive for evaluators. However, since CEM contains detailed requirements regarding the evaluation evidence to be provided, it has also to be consulted by developers who intend to get involved in an evaluation.

The purpose of this guidance is to offer assistance to less experienced developers by extracting the information regarding the evidence to be provided from the criteria, by explaining the documentation requirements regarding contents and evidence to be provided, and by providing examples. It addresses the requirements for evaluation assurance level (a term explained in the next chapter) EAL1 to EAL5. Last but not least it supports the developers work in proposing samples for structuring the documentation.

1.3 Structure of this Guidance

After this introductory chapter an introduction to the Common Criteria (CC) and important concepts (like class, family, and component) is given in chapter 2. Chapter 3 gives an overview about the assurance classes of the CC and introduces the evaluation assurance level (EAL) concept. Chapter 4 identifies reference documents on which this guidance bases and identifies important websites. Chapter 5 contains a list of the abbreviations used in this guidance. Chapters 6 to 10 are dedicated to the explanations of the requirements of the various CC assurances classes and Appendixes A to E propose structure and content of the corresponding developer information to be provided.

2 Introduction to the Common Criteria (CC)

The CC permits comparability between the results of independent security evaluations. The CC does so by providing a common set of requirements for the security functionality of (collections of) IT products and for assurance measures applied to these IT products during a security evaluation. The evaluation process establishes a level of confidence that the security functionality of these products and the assurance measures applied to these IT products meet these requirements. The evaluation results may help consumers to determine whether these IT products fulfil their security needs.

The CC addresses protection of information from unauthorised disclosure, modification, or loss of use. The categories of protection relating to these three types of failure of security are commonly called confidentiality, integrity, and availability, respectively. The CC may also be applicable to aspects of IT security outside of these three. Typical additional aspects to be considered are authenticity of information and communication partners, and non-repudiation of origin or receipt of information The CC is primarily applicable to risks arising from human activities (malicious or otherwise) and in some respect also to risks arising from non-human activities.

2.1 General Structure of the CC and CEM

The CC is presented as a set of distinct but related parts as identified below:

Part 1, Introduction and general model is the introduction to the CC. It defines the general concepts and principles of IT security evaluation and presents a general model of evaluation. It also contains directives for the specification of Security Targets and Protection Profiles.

Part 2, Security functional components establishes a set of functional components that serve as standard templates upon which to base functional requirements for TOEs. CC Part 2 catalogues the set of functional components and organises them in families and classes.

Part 3, Security assurance components establishes a set of assurance components that serve as standard templates upon which to base assurance requirements for TOEs. CC Part 3 catalogues the set of assurance components and organises them into families and classes. CC Part 3 also defines evaluation criteria for PPs and STs and presents seven pre-defined assurance packages which are called the Evaluation Assurance Levels (EALs).

In support of the three parts of the CC listed above, other documents have been published, most notably the Common Evaluation Methodology (CEM). The CEM is a normative document for the evaluator. It breaks down the necessary evaluator actions (identified in Part 3) into Evaluator Work Units. The CEM is a cornerstone for the mutual recognition of evaluation results.

Part 2 and Part 3 of the CC are catalogues which can be used by a developer to select pre-defined requirements as claims in a Security Target or Protection Profile. Part 2 specifies security functional requirements (SFRs), while Part 3 specifies security assurance requirements (SARs). This guidance only deals with assurance requirements taken from Part 3.

The “objects” of the catalogues are assembled into major groups (“classes”), subgroups (“families”), and elementary objects (“components”):

> The **assurance classes** specified in Part 3 contain requirements addressing a common topic (like “ADV – Development” or “AGD – Guidance documents”). The assurance classes are introduced in chapter 3 and are discussed in more detail in the remaining portions of this guidance.

The assurance classes are structured into **assurance families**. As an example assurance class ADV contains assurance families “ADV_FSP – Functional Specification”, “ADV_TDS – TOE design”, and other families. The assurance families are introduced in chapter 3 and are discussed in more detail in the remaining portions of this guidance.

The assurance families are hierarchically structured into **assurance components**. These are the objects which can be selected in a Security Target (ST) to specify the rigour and depth of an evaluation. As an example assurance family ADV_FSP contains assurance components “ADV_FSP.1 – Basic functional specification”, “ADV_FSP.2 – Security enforcing functional specification”, “ADV_FSP.3 – Functional specification with complete summary”, and other assurance components. Hierarchically means, that ADV_FSP.2 extends the requirements of ADV_FSP.1, and that ADV_FSP.3 extends the requirements of ADV_FSP.2. Requirements derived from the various assurance components are discussed in detail in the remaining portions of this guidance without referring to specific components directly.

Assurance components are selected for specification in a Security Target (ST) or Protection Profile (PP) document, which is the central document for a security evaluation according to the Common Criteria. Guidance for Security Targets are addressed in [STG].

2.2 Field of Application of the CC and CEM

The CC is useful as a guide for the development, evaluation and/or procurement of (collections of) products with IT security functionality.

The CC is applicable to IT security functionality implemented in hardware, firmware or software.

3 Overview of Assurance Classes

This chapter gives an overview about the assurance classes of the CC and introduces the evaluation assurance level (EAL) concept.

3.1 Assurance Classes

This chapter gives an overview about the various assurance classes introduced and described in Part 3 of the CC.

3.1.1 Class ACO - Composition

This class 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. This class is **not** further addressed by this guidance.

3.1.2 Class ADV - Development

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.

The Development class encompasses six families of requirements for structuring and representing the **TOE Security Functions (TSF)** at various levels and varying forms of abstraction. The TSF are a set of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the Security Functional Requirements (SFRs). The objective of this class is to demonstrate that design and implementation of the TOE have been performed correctly (following the specification in the Security Target) covering all levels of abstraction of the TSF. The six 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),
- 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 the 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 minimisation 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; that is, 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 that the security functionality cannot be corrupted or bypassed. It should be noted that both properties need to be realised: the more confidence one has that the properties are satisfied, the more trustworthy the TOE is. 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. Each level of decomposition must be a complete and accurate instantiation of the higher level of decomposition. 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.

3.1.3 Class AGD – Guidance documents

The guidance documents class provides the requirements for guidance documentation for all user roles. For the secure preparation

> *(i.e. the product life-cycle phase comprising the customer's acceptance of the delivered TOE and its installation which may include such things as booting, initialisation, start-up, progressing the TOE to a state ready for operation)*

and operation

> *(i.e. the usage phase of the TOE; this includes "normal usage", administration and maintenance of the TOE)*

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.

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, family AGD_PRE)*

and with the operational user guidance

> *(what has to be done during the operation of the TOE in its evaluated configuration, family AGD_OPE)*.

3.1.4 Class ALC – Life-cycle support

Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during its development and maintenance. 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 and maintenance activities.

In the product life-cycle it is distinguished whether the TOE is under the responsibility of the developer or the user rather than whether it is located in the development or user environment. The point of transition is the moment where the TOE is handed over to the user. This is also the point of transition from the ALC to the AGD class.

The ALC class consists of seven families. Life-cycle definition (ALC_LCD) is the high-level description of the TOE life-cycle; CM capabilities (ALC_CMC) a more detailed description of the management of the configuration items. CM scope (ALC_CMS) requires a minimum set of configuration items to be managed in the defined way. Development security (ALC_DVS) is concerned with the developer's security measures to protect the TOE and its associated design information from interference or disclosure; Tools and techniques (ALC_TAT) with the development tools and implementation standards used by the developer; Flaw remediation (ALC_FLR) with the handling of security flaws. Delivery (ALC_DEL) defines the procedures used for the delivery of the TOE to the consumer. Delivery processes occurring during the development of the TOE are denoted rather as transports, and are handled in the context of integration and acceptance procedures in other families of this class.

3.1.5 Class ASE – Security Target Evaluation

This important class is outside of the scope of this guidance. It is addressed in detail in [STG].

3.1.6 Class ATE – Tests

The emphasis in this class is on confirmation that the TSF operates according to its design descriptions. Testing provides such assurance that the TSF behaves as described in the functional specification, TOE design, and implementation representation.

The class “Tests” encompasses four families: Coverage (ATE_COV), Depth (ATE_DPT), Independent testing (ATE_IND), and Functional tests (ATE_FUN). It separates testing into developer testing and evaluator testing. Functional tests (ATE_FUN) addresses the performing of the tests by the developer and how this testing should be documented. 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, implementation representation) is required. Independent testing (ATE_IND) addresses evaluator testing: whether the evaluator should repeat part or all of the developer testing and how much independent testing the evaluator should do.

3.1.7 Class AVA – Vulnerability assessment

This class addresses the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE. It 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) could allow attackers to violate the security functional requirements.

This class contains only the family Vulnerability analysis (AVA_VAN). It deals with the threat 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 authorised capabilities of other users.

3.2 Evaluation Assurance Levels

Seven hierarchically ordered evaluation assurance levels are predefined in the CC for the rating of a TOE's assurance. They are hierarchically ordered inasmuch as each EAL represents more assurance than all lower EALs. The increase in assurance from EAL to EAL is accomplished by substitution of a hierarchically higher assurance component from the same assurance family (i.e. increasing rigour, scope, and/or depth) and from the addition of assurance components from other assurance families (i.e. adding new requirements). These EALs consist of an appropriate combination of assurance components taken from CC Part 3.

Table 1 represents a summary of the EALs. The columns represent a hierarchically ordered set of EALs, while the rows represent assurance families. Each number in the resulting matrix identifies a specific assurance component where applicable. Note, that the cells highlighted in magenta (like 1) indicate those assurance components (such as ADV_FSP.1) which are addressed in this guidance. Some cells are hatched (like). Such cells identify cases in which an EAL does not comprise a requirement from the specific assurance family (like EAL1 and ADV_ARC). The requirements for this family are thus implicitly met. The assurance components of Family ALC_FLR are not part of any EAL. They are also covered by this guidance and are highlighted in magenta. The components of family ASE (Security Target) are not covered by this guidance but are addressed in [STG].

Assurance Class	Assurance Family	Assurance Components by Evaluation Assurance Level						
		EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7
Development	ADV_ARC		1	1	1	1	1	1
	ADV_FSP	1	2	3	4	5	5	6
	ADV_IMP				1	1	2	2
	ADV_INT					2	3	3
	ADV_SPM						1	1
	ADV_TDS		1	2	3	4	5	6
Guidance documents	AGD_OPE	1	1	1	1	1	1	1
	AGD_PRE	1	1	1	1	1	1	1
Life-cycle support	ALC_CMC	1	2	3	4	4	5	5
	ALC_CMS	1	2	3	4	5	5	5
	ALC_DEL		1	1	1	1	1	1
	ALC_DVS			1	1	1	2	2
	ALC_FLR							
	ALC_LCD			1	1	1	1	2
	ALC_TAT				1	2	3	3
Security Target Evaluation	ASE_CCL	1	1	1	1	1	1	1
	ASE_ECD	1	1	1	1	1	1	1
	ASE_INT	1	1	1	1	1	1	1
	ASE_OBJ	1	2	2	2	2	2	2
	ASE_REQ	1	2	2	2	2	2	2
	ASE_SPD		1	1	1	1	1	1
	ASE_TSS	1	1	1	1	1	1	1
Tests	ATE_COV		1	2	2	2	3	3
	ATE_DPT			1	2	3	3	4
	ATE_FUN		1	1	1	1	2	2
	ATE_IND	1	2	2	2	2	2	3
Vulnerability Assessment	AVA_VAN	1	2	2	3	4	5	5

Table 1: Evaluation assurance level summary

While the EALs are defined in the CC, it is possible to represent other combinations of assurance. Specifically, the notion of “augmentation” allows the addition of assurance components (from assurance families not already included in the EAL) or the substitution of assurance components (with another hierarchically higher assurance component in the same assurance family) to an EAL. Of the assurance constructs defined in the CC, only EALs may be augmented. An augmented EAL is often indicated by a ”+”-sign (like EAL4+). The notion of an “EAL minus a constituent assurance component” is not recognised by the standard as a valid claim. Augmentation carries with it the obligation on the part of the claimant to justify the utility and added value of the added assurance component to the EAL.

What does a higher evaluation assurance level mean for a developer?

- Additional evidence have to be provided (addressing additional assurance aspects),
- more detailed evidence has to be provided,
- more requirements on the development environment,
- an increased need to support the evaluation process,
- the evaluation process will take more time,
- an increased amount of internal and external resources (i.e. costs),
- a more effective independent review,
- increased marketing benefits.

What does a higher evaluation assurance level mean for an evaluator?

- A broader analysis has to be performed,
- a deeper analysis has to be performed,
- more testing has to be performed,
- a deeper understanding of the product is gained.

What does a higher evaluation assurance level mean for a consumer?

- A higher confidence in the security properties of the product is gained.

What are the requirements for EAL1?

- ***for class ADV (Development)***

  A functional specification shall be provided which describes (on a high-level) the interfaces which enforce or support the security functional requirements.

- ***for class AGD (Guidance documents)***

  Guidance documentation for all user roles shall be provided.

- ***for class ALC (Life-cycle support)***

  The TOE shall be uniquely referenced. A configuration list shall be provided which contains the TOE itself and the evaluation evidence.

- ***for class ATE (Tests)***

  The evaluator has to perform testing and has to provide evidence of the results

  .

- ***for class AVA (Vulnerability assessment)***

  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.

What are the increased requirements for EAL2?

- ***for class ADV (Development)***

  The TSF shall be designed and implemented so that it is able to protect itself from tampering by untrusted active entities. A description of the security architecture shall be provided. Design documentation of the TOE shall be provided. The functional specification shall describe the interfaces which enforce the security functional requirements in terms of actions and responses and shall provide a high level description of the other interfaces to the TOE Security Functionality (TSF). The design documentation shall describe the structure of the TOE in terms of subsystems.

- ***for class AGD (Guidance documents)***

  none

- ***for class ALC (Life-cycle support)***

  The developer shall use a configuration management (CM) system and shall provide CM documentation. The configuration list shall also contain the parts (software modules, hardware components) of the TOE. Delivery procedures shall be documented.

- ***for class ATE (Tests)***

  The developer has to perform functional tests covering of the interfaces to the TSF described in the Functional Specification. The developer shall provide to the evaluator an equivalent set of resources to those that were used in the developer's functional testing. The evaluator has to repeat a sample of the developer’s tests.

- ***for class AVA (Vulnerability assessment)***

  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. 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.

What are the increased requirements for EAL3?

- ***for class ADV (Development)***

  The functional specification shall be more detailed regarding the description of the TSF iinterfaces (TSFI). The design documentation shall additionally describe the interactions among the subsystems.

- ***for class AGD (Guidance documents)***

  none

- ***for class ALC (Life-cycle support)***

  The CM system shall provide authorisation controls for configuration items. The CM documentation shall include a CM plan. The configuration list shall also contain the implementation representation (software source code, hardware drawings) of the TOE. A description of the security of the developer site shall be provided. A life-cycle model for development and maintenance of the TOE shall be established.

- ***for class ATE (Tests)***

  The developer has to provide an analysis which demonstrates, that his functional tests cover all interfaces to the TSF described in the Functional Specification. The developer has to provide an analysis which demonstrates, that the results of his test are consistent with the behaviour of the subsystems defined in the TOE design.

- ***for class AVA (Vulnerability assessment)***

  none

What are the increased requirements for EAL4?

- ***for class ADV (Development)***

  The implementation representation (i.e. software source code, hardware drawings, etc.) of the TSF shall be provided. A mapping between the TOE design description and a sample of the implementation representation shall be provided. The functional specification shall describe all kinds of TSFI to the same high level of detail. The design documentation shall additionally describe the TSF in terms of modules.

- ***for class AGD (Guidance documents)***

  none

- ***for class ALC (Life-cycle support)***

  The CM system shall provide automation, production support, and acceptance procedures. The configuration list shall also contain security flaw reports and resolution status. A description of the development tools shall be supplied.

- ***for class ATE (Tests)***

  The developer has to provide an analysis which demonstrates, that the results of his tests are consistent with the behaviour of the subsystems and modules defined in the TOE design.

- ***for class AVA (Vulnerability assessment)***

  The evaluator shall also use the implementation representation to perform an independent vulnerability analysis of the TOE and to identify potential vulnerabilities in the TOE. The TOE must be resistant to attacks performed by an attacker possessing Enhanced-Basic attack potential.

What are the increased requirements for EAL5?

- ***for class ADV (Development)***

  The TSF shall be designed and implemented such that it has well-structured internals. An internals description and justification shall be provided. The functional specification shall describe the TSFI using a semi-formal style and shall contain additional error information. The design documentation shall be described in a semi-formal style and shall include additionally details.

- ***for class AGD (Guidance documents)***

  none

- ***for class ALC (Life-cycle support)***

  The configuration list shall also contain development tools. The developer shall apply implementation standards.

- ***for class ATE (Tests)***

  The developer has to provide an analysis which demonstrates, that all modules have been tested.

- ***for class AVA (Vulnerability assessment)***

  The evaluator's vulnerability analysis shall be systematic. The TOE must be resistant to attacks performed by an attacker possessing Moderate attack potential.

What is an adequate evaluation assurance level for a specific product?

The aspects identified above demonstrate that the selection of an evaluation assurance level will be dominated by factors such as personnel and monetary resources, complexity of the product, experience, expertise and market expectations.

Market expectations do often correspond to certificates and evaluation assurance levels gained for competitive products.

Some rules of thumb may also support the selection process:

EAL1 is appropriate for products meeting specific customer needs requesting low assurance.

EAL2 is a level which is often used for applications with security functionality. It is also an "entry level" for developers who are aiming at a higher level but want to get familiar with the evaluation and certification process, before.

EAL3 is a level which is typically selected for complex products (like operating systems) in case a higher level is considered to be too costly.

EAL4 is a level which is adequate for products and customers mandating requiring a high assurance level (firewalls, smart card components).

EAL5 is a level which is selected in case customers request extra assurance (smart card components).

EAL6 and EAL7 are beyond the scope of this guidance. They are selected in extraordinary cases, only.

4 References

The following documents are of special interest as background and supplementary information for this guidance:

[CC1] Common Criteria for Information Technology Security Evaluation – Part 1: Introduction and general model – September 2006, Version 3.1, Revision 1, CCMB-2006-09-001

[CC2] Common Criteria for Information Technology Security Evaluation – Part 2: Security functional components – September 2006, Version 3.1, Revision 1, CCMB-2006-09-002

[CC3] Common Criteria for Information Technology Security Evaluation – Part 3: Security assurance components – September 2006, Version 3.1, Revision 1, CCMB-2006-09-003

[CEM] Common Criteria for Information Technology Security Evaluation – Evaluation Methodology – September 2006, Version 3.1, Revision 17, CCMB-2006-09-004

[STG] PP/ST-Guide

The following websites are recommended for readers which are interested in actual information about the Common Criteria and certified products.

- www.commoncriteriaportal.org
- www.bsi.bund.de

5 Abbreviations

CAD Computer Aided Design

CC Common Criteria

CD Compact Disk

CEM Common Evaluation Methodology

CM Configuration Management

EAL Evaluation Assurance Level

GUI Graphical User Interface

IC Integrated Circuit

IT Information Technology

PP Protection Profile

SAR Security Assurance Requirement

SFR Security Functional Requirement

ST Security Target

TOE Target of Evaluation

TSF TOE Security Functionality

TSFI TSF Interface

TSS TOE Summery Specification

URL Uniform/Universal Resource Locator

6 Glossary

For the purpose of this document, the following terms and definitions apply. They are mainly reproduced from CC Part 1.

acceptance criteria the criteria to be applied when performing the acceptance procedures (e.g. successful document review, or successful testing in the case of software, firmware or hardware).

acceptance procedures the procedures followed in order to accept newly created or modified configuration items as part of the TOE, or to move them to the next step of the life-cycle. These procedures identify the roles or individuals responsible for the acceptance and the criteria to be applied to decide on the acceptance.

administrator an entity that has complete trust with respect to all policies implemented by the TSF.

assurance grounds for confidence that a TOE meets the SFRs.

attack potential a measure of the effort to be expended in attacking a TOE, expressed in terms of an attacker's expertise, resources and motivation.

augmentation the addition of one or more requirement(s) to a package.

call tree a diagram that identifies the modules in a system and shows which modules call one another. All the modules named in a call tree that originates with (i.e., is rooted by) a specific module are the modules that directly or indirectly implement the functions of the originating module.

class a grouping of CC families that share a common focus.

CM documentation (documentation of the CM system) overall term for the following:

- CM output
- CM list (configuration list)
- CM system records
- CM plan
- CM usage documentation

CM evidence everything that may be used to establish confidence in the correct operation of the CM system, e.g., CM output, rationales provided by the developer, observations, experiments or interviews made by the evaluator during a site visit.

CM item (configuration item) object managed by the CM system during the TOE development. These may be either parts of the TOE or objects related to the development of the TOE like evaluation documents or development tools. CM items may be stored in the CM system directly (for example files) or by reference (for example hardware parts) together with their version.

CM list (configuration list) a CM output document listing all configuration items for a specific product together with the exact version of each CM item relevant for a specific version of the complete product. 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 CM 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 CM tool. 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 CM requirements of ALC_CMC.

CM output CM related results produced or enforced by the CM system. These CM related results could occur as documents (for example filled paper forms, CM system records, logging data, hard-copies and electronic output data) as well as actions (for example manual measures to fulfil CM instructions). Examples of such CM outputs are configuration lists, CM plans and/or behaviours during the product life-cycle.

CM plan part of the CM documentation describing how the CM system is used for the TOE. The objective of issuing a CM plan is that staff members can see clearly what they have to do. From the point of view of the overall CM system this can be seen as an output document (because it may be produced as part of the application of the CM 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 CM plan defines the usage of the system for the specific product; the same system may be used to a different extent for other products. That means the CM plan defines and describes the output of the CM system of a company which is used during the TOE development.

CM system overall term for the set of procedures and tools (including their documentation) used by a developer to develop and maintain configurations of his products during their life-cycles. CM systems may have varying degrees of rigour and function. At higher levels, CM systems may be automated, with flaw remediation, change controls, and other tracking mechanisms.

CM system records those CM output documents which are produced during the operation of the CM system documenting important activities. Examples of CM system records are CM item change control forms or CM item access approval forms.

CM tools tools realising or supporting a CM system, for example tools for the version management of the parts of the TOE. They may require manual operation or may be automated.

CM usage documentation that part of the CM system, which describes, how the CM system is defined and applied by using for example handbooks, regulations and/or documentation of tools and procedures.

cohesion (also called module strength) the manner and degree to which the tasks performed by a single software module are related to one another; types of cohesion include coincidental, communicational, functional, logical, sequential, and temporal.

coincidental cohesion a module with this characteristic performs unrelated, or loosely related, activities.

communicational cohesion a module with this characteristic contains functions that produce output for, or use output from, other functions within the module. An example of a communicationally cohesive module is an *access check* module that includes mandatory, discretionary, and capability checks.

complexity this is a measure of how difficult software is to understand, and thus to analyse, test, and maintain. Reducing complexity is the ultimate goal for using modular decomposition, layering and minimisation. Controlling coupling and cohesion contributes significantly to this goal.

component the smallest selectable set of elements on which requirements may be based.

coupling the manner and degree of interdependence between software modules; types of coupling include call, common and content coupling. These types of coupling are characterised below, listed in the order of decreasing desirability

- *call:* two modules are call coupled if they communicate strictly through the use of their documented function calls; examples of call coupling are data, stamp, and control, which are defined below.
  1. *data:* two modules are data coupled if they communicate strictly through the use of call parameters that represent single data items.
  2. *stamp:* two modules are stamp coupled if they communicate through the use of call parameters that comprise multiple fields or that have meaningful internal structures.
  3. *control:* two modules are control coupled if one passes information that is intended to influence the internal logic of the other.
- *common:* two modules are common coupled if they share a common data area or a common system resource. Global variables indicate that modules using those global variables are common coupled. Common coupling through global variables is generally allowed, but only to a limited degree. For example, variables that are placed into a global area, but are used by only a single module, are inappropriately placed, and should be removed. Other factors that need to be considered in assessing the suitability of global variables are:
  1. The number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification must be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them.
  2. The number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.
- *content:* two modules are content coupled if one can make direct reference to the internals of the other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that some or all of the content of one module are effectively included in the other. Content coupling can be thought of as using unadvertised module interfaces; this is in contrast to call coupling, which uses only advertised module interfaces.

delivery the product life-cycle phase which is concerned with the transmission of the finished TOE from the production environment into the hands of the customer. This may include packaging and storage at the development site, but does not include transportations of the unfinished TOE or parts of the TOE between different developers or different development sites.

developer the organisation responsible for the development of the TOE.

development the product life-cycle phase which is concerned with generating the implementation representation of the TOE. Throughout the ALC requirements, development and related terms (developer, develop) are meant in the more general sense to comprise development *and production*.

development environment the environment in which the TOE is developed.

development tools tools (including test software, if applicable) supporting the development and production of the TOE. E.g., for a software TOE, development tools are usually programming languages, compilers, linkers and generating tools.

domain separation the security architecture property whereby the TSF defines separate security domains for each user and for the TSF and ensures that no user process can affect the contents of a security domain of another user or of the TSF.

error message description an identification of the condition that generated the error, and a description what the message is, and what the meaning of any error codes are. 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 that can be tied to 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 might 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.

evaluation assessment of a PP, an ST or a TOE, against defined criteria.

evaluation assurance level (EAL) an assurance package, consisting of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale.

exploitable vulnerability a weakness in the TOE that can be used to violate the SFRs in the operational environment for the TOE.

family a grouping of components that share a similar goal but may differ in emphasis or rigour.

functional cohesion a module with this characteristic performs activities related to a single purpose. A functionally cohesive module transforms a single type of input into a single type of output, such as a stack manager or a queue manager.

guidance documentation documentation that describes the delivery, preparation, operation, management and/or use of the TOE.

implementation representation the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement. Source code that is then compiled or a hardware drawing that is used to build the actual hardware are examples of parts of an implementation representation.

installation the procedures that the user has to perform normally only once after receipt and acceptance of the TOE to progress it to the secure configuration as described in the ST including the embedding of the TOE in its operational environment. If similar processes have to be performed by the developer they are denoted as “generation” throughout 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 here.

interaction general communication-based relationship between entities.

interface a means of interaction with a component or module.

layering the design of software such that separate groups of modules (the *layers*) are hierarchically organised to have separate responsibilities such that one layer depends only on

layers below it in the hierarchy for services, and provides its services only to the layers above it. Strict layering adds the constraint that each layer receives services only from the layer immediately beneath it, and provides services only to the layer immediately above it.

life-cycle the sequence of stages of existence of an object (for example a product or a system) in time.

life-cycle definition the definition of the life-cycle model.

life-cycle model description of the stages and their relations to each other that are used in the management of the life-cycle of a certain object, how the sequence of stages looks like and which high level characteristics the stages have.

logical (or procedural) cohesion a module with this characteristic performs similar activities on different data structures. A module exhibits logical cohesion if its functions perform related, but different, operations on different inputs.

method of use of an interface a description of 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).

modular decomposition the process of breaking a system into components to facilitate design and development.

non-bypassability (of the TSF) the security architecture property whereby all SFR-related actions are mediated by the TSF.

operation (of the TOE) usage of the TOE after delivery and preparation. This includes “normal usage”, administration and maintenance of the TOE.

operational environment the environment in which the TOE is operated.

package a named set of either functional or assurance requirements (e.g. EAL 3).

parameters 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.

preparation the product life-cycle phase comprising the customer's acceptance of the delivered TOE and its installation which may include such things as booting, initialisation, start-up, progressing the TOE to a state ready for operation.

potential vulnerability a weakness the existence of which is suspected (by virtue of a postulated attack path), but not confirmed, to violate the SFRs.

production the production life-cycle phase follows the development phase and consists of transforming the implementation representation into the implementation of the TOE, i.e. into a state acceptable for delivery to the customer. This phase may comprise manufacturing, integration, generation, internal transports, storage, and labelling of the TOE.

Protection Profile (PP) an implementation-independent statement of security needs for a TOE type.

purpose of an interface a high-level description of the general goal of the interface (e.g. process GUI commands, receive network packets, provide printer output, etc.). 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 might 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 *summarised*, meaning the description merely makes clear that it is indeed not SFR-related.

role a predefined set of rules establishing the allowed interactions between a user and the TOE.

secure state a state in which the TSF data are consistent and the TSF continues correct enforcement of the SFRs.

security domain the collection of resources to which an active entity has access.

security function policy (SFP) a set of rules describing specific security behaviour enforced by the TSF and expressible as a set of SFRs.

Security Target (ST) an implementation-dependent statement of security needs for a specific identified TOE.

sequential cohesion a module with this characteristic contains functions each of whose output is input for the following function in the module. An example of a sequentially cohesive module is one that contains the functions to write audit records and to maintain a running count of the accumulated number of audit violations of a specified type.

target of evaluation (TOE) a set of software, firmware and/or hardware possibly accompanied by guidance.

temporal cohesion a module with this characteristic contains functions that need to be executed at about the same time. Examples of temporally cohesive modules include *initialisation*, *recovery*, and *shutdown* modules.

TOE Security Functionality (TSF) a set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs.

TSF interface (TSFI) a 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 and invoke services from the TSF.

TSF self-protection the security architecture property whereby the TSF cannot be corrupted by non-TSF code or entities.

vulnerability a weakness in the TOE that can be used to violate the SFRs in some environment.

7 Class ADV: Development

Assurance class ADV class defines requirements to provide information about the design of the TOE, its structure, and its interfaces. 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).

All portions of the TSF are security relevant, meaning that they must be relied upon to preserve the security of the TOE as expressed by the SFRs and additional 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. **SFR-enforcing** portions play a direct role in implementing any SFR on the TOE. Other components are not so much enforcing the security policy but rather play a supporting role; such components are termed **SFR-supporting**. A third category of components – **SFR-non-interfering** – includes portions of the TSF which do not play an enforcing or contributing role, but which nevertheless have to be relied upon.

7.1 Security Architecture (ADV_ARC)

The security architecture family provides 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. 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.

Requirement: EAL2 EAL3 EAL4 EAL5

a) The developer shall design and implement the TOE so that the security features of the TSF can not be bypassed. He shall design and implement the TSF so that it is able to protect itself from tampering by untrusted active entities. The developer shall provide a security architecture description of the TSF.

Background: A *security architecture* is set of properties that the TSF exhibits; these properties include self-protection, domain isolation, and non-bypassability.

Role in the evaluation process: The purpose of this requirement is to justify the approach to limit the security analysis of the TOE to the examination the TSF. Without a sound architecture, the entire TOE functionality would have to be examined.

Related Evaluator Actions: The evaluator will check whether the description of the architectural design, coupled with other evidence delivered for the TOE and the TSF, demonstrates that design and implementation are appropriate to achieve such protection.

Requirement: EAL2 EAL3 EAL4 EAL5

b) The security architecture description shall be at a level of detail commensurate with the description of the SFR-enforcing abstractions in the TOE design document.

Related Evaluator Actions: The evaluator will ensure that the self-protection functionality described covers those effects that are evident at the TSFI. He will also ensure that the security architecture description contains information on how any subsystems that contribute to the TSF domain separation work. For EAL4 and above the evaluator will ensure that the security architecture description also contains implementation dependent information.

Example

- To provide implementation dependent information, such a description might contain information pertaining to coding conventions for parameter checking that would prevent TSF compromises (e.g. buffer overflows), and information on stack management for call and return operations.

Requirement: EAL2 EAL3 EAL4 EAL5

c) The security architecture description shall describe the security domains maintained by the TSF.

Background: Security domains refer to environments supplied by the TSF for use by potentially-harmful entities. For some TOEs such domains do not exist because all of the interactions available to users are severely constrained by the TSF.

Related Evaluator Actions: The evaluator will check whether the architecture description addresses all security domains that can be identified by other evaluation evidence. The evaluator will check whether the architecture description takes into account all SFRs claimed by the TOE.

Examples

- A secure operating system provides a security domain by supplying a set of resources (address space, per-process environment variables) for use by processes with limited access rights and security properties.
- A packet-filter firewall is an example of a TOE which does not provide a security domain. Users on the LAN or WAN do not interact with the TOE, so there is no need for security domains; there are only data structures maintained by the TSF to keep the users' packets separated.

Requirement: EAL2 EAL3 EAL4 EAL5

d) The security architecture description shall demonstrate that the TSF initialisation process preserves security. This portion of the security architecture description should list the system initialisation components and describe the processing that occurs in transitioning from the "down" state to the initial secure stage (i.e. when all parts of the TSF are operational) when power-on or a reset is applied.

Related Evaluator Actions: The evaluator will check that the security architecture description demonstrates how the TSF initialisation process preserves security.

Examples

- The TSF can preserve security by ensuring that the initialisation components are not accessible to untrusted entities after the secure stage is reached.

- The TSF can preserve security by ensuring that the interfaces provided by the initialisation components to untrusted users can not be used to tamper with the TSF.

Requirement: EAL2 EAL3 EAL4 EAL5

e) The security architecture description shall demonstrate that the TSF protects itself from tampering.

Background:
"Self-protection" refers to the ability of the TSF to protect itself from manipulation from external entities that may result in changes to the TSF. For TOEs that have dependencies on other IT entities, it is often the case that the TOE uses services supplied by the other IT entities in order to perform its functions. In such cases, the TSF alone does not protect itself because it depends on the other IT entities to provide some of the protection. For the purposes of the security architecture description, the notion of *self-protection* applies only to the services provided by the TSF through its TSFI, and not to services provided by underlying IT entities that it uses.
The CC admits TOE with incomplete self protection capabilities but requires that such limitations are clearly stated in the security architecture documentation. The following cases are admitted:

- In cases that a TOE uses services or resources supplied by other IT entities in order to perform its functions (e.g. an application that relies upon its underlying operating system), the TSF does not protect itself entirely on its own, because it depends on the other IT entities to protect the services it uses.
- The TSF might also have reliance in its non-IT environment to provide some support to the protection of the TSF.

Related Evaluator Actions: The evaluator will check whether the security architecture description demonstrated such property, that no user input which is handled by the TSF can corrupt the TSF. He will also check whether the description is complete and accurate with respect to the other evaluation evidence (such as functional specification, system design description, TOE design, TSF Internals documentation, implementation representation).

Examples

- Self-protection can be supported by processor-based separation mechanisms such as privilege levels or rings.
- Self-protection can also be supported by software-based means. The security architecture description might contain information pertaining to coding conventions for parameter checking that would prevent TSF compromises (e.g. buffer overflows), and information on stack management for call and return operations.
- Self-protection can also be supported by physical and logical restrictions on access to the TOE.
- The description might include protection placed upon the executable images of the TSF, and protection placed on objects (e.g. files used by the TSF).
- The TSF might have reliance on its environment to provide some support to the protection of the TSF.

Requirement: EAL2 EAL3 EAL4 EAL5

f) The security architecture description shall demonstrate that the TSF prevents bypass of the SFR-enforcing functionality. This means that the TSF protection mechanisms are always

invoked to prevent that a TSFI can be used to access protected data or resources in an unauthorised way.

Related Evaluator Actions: The evaluator will check that the security architecture description analyses each TSFI and demonstrates that either no access to protected data or resources is possible or that the TSF is invoked to enforce the corresponding protection policies. The evaluator will search the security architecture description for systematic arguments based on the security functional requirements, the underlying policies and the TSFI. The evaluator will at least analyse the TOE summary specification (TSS) of the Security Target, the functional specification, and the TOE design documentation to establish the background for understanding protected data and functions.

Examples:

- Suppose the TSF provides file protection. Further suppose that although the “traditional” system call TSFIs for open, read, and write invoke the file protection mechanism described in the TOE design, there exists a TSFI that allows access to a batch job facility (creating batch jobs, deleting jobs, modifying unprocessed jobs). The evaluator should be able to determine from the vendor-provided description that this TSFI invokes the same protection mechanisms as do the “traditional” interfaces. This could be done, for example, by referencing the appropriate sections of the TOE design that discuss *how* the batch job facility TSFI achieves its security objectives.
- Using this same example, suppose there is a TSFI whose sole purpose is to display the time of day. The description should adequately argue that this TSFI is not capable of manipulating any protected resources and should not invoke any security functionality.
- Another example of bypass is when the TSF is supposed to maintain confidentiality of a cryptographic key (one is allowed to use it for cryptographic operations, but is not allowed to read/write it). If an attacker has direct physical access to the device, he might be able to examine side-channels such as the power usage of the device, the exact timing of the device, or even any electromagnetic emanations of the device and, from this, infer the key. If such side-channels may be present, the demonstration should address the mechanisms that prevent these side-channels from occurring, such as random internal clocks, dual-line technology etc.

7.2 Functional specification (ADV_FSP)

This family contains requirements upon the description of the TSF interfaces (TSFI). The TSFI consist of all means for users to invoke a service from the TSF (by supplying data that is processed by the TSF) and the corresponding responses to those service invocations. 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).

If an action available through an interface can be traced to any of the SFRs contained in the ST, then that interface is ***SFR-enforcing***.

Interfaces to actions that SFR-enforcing functionality depends on, but need only to function correctly in order for the security policies of the TOE to be preserved, are termed ***SFR-supporting***.

Interfaces to actions on which SFR-enforcing functionality has no dependence are termed ***SFR-non-interfering***.

For an interface to be SFR-supporting or SFR non-interfering it must have *no* SFR-enforcing actions or results. In contrast, an SFR-enforcing interface may have also SFR-supporting actions.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

a) The developer shall provide a functional specification. This functional specification shall describe the purpose and method of use for the category of TSF Interfaces (TSFI) as mandated by the following requirements.

Role in the evaluation process: The purpose in specifying the TSFI is to provide the necessary information to conduct testing. Without knowing the possible means to interact with the TSF, one cannot adequately test the behaviour of the TSF.

Background: The *purpose* of a TSFI is a general statement summarising the functionality provided by the interface. It is not intended to be a complete statement of the actions and results related to the interface, but rather a statement to help the reader to understand in general what the interface is intended to be used for.

The *method of use* for a TSFI summarises how the interface is manipulated in order to invoke the actions and obtain the results associated with the TSFI.

In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must first be identified. This identification is actually a part of the TOE design (ADV_TDS) analysis, but is also performed implicitly (through identification and description of the TSFI) by the developer for EAL1 where TOE design (ADV_TDS) is not included in the assurance package. In this analysis, a portion of the TOE must be considered to be in the TSF if it contributes to the satisfaction of an SFR in the ST (in whole or in part). Once the TSF has been defined, the TSFIs are identified. The TSFIs consist of all means for users to invoke a service from the TSF (by supplying data that is processed by the TSF) and the corresponding responses to those service invocations. These service invocations and responses are the means of crossing the TSF boundary. While many of these are readily apparent, others might not be as obvious. The question that should be asked when determining the TSFIs is: “How can a potential attacker interact with the TSF in an attempt to subvert any of the the SFRs?”

Related Evaluator Actions: The evaluator will check that a functional specification has been provided which possesses the characteristics specified by the following requirements.

Example:

- If an operating system presents the user with a means of self-identification, of creating files, of modifying or deleting files, of setting permissions defining what other users may access files, and of communicating with remote machines, its functional specification would contain descriptions of these and how they are realised through interactions with externally-visible interfaces to the TSF. If there is also audit functionality that detects and records the occurrences of such events, description of this audit functionality would also be expected to be part of the functional specification; while this functionality is technically not directly invoked by the user at the external interface, it certainly is affected by what occurs at the user’s external interface.
- In TOEs such as smart cards, where the adversary has not only logical access to the TOE, but also complete physical access to the TOE, the TSF boundary is the physical boundary. Therefore, the exposed electrical interfaces are considered TSFI because their manipulation could affect the behaviour of the TSF. As such, all these interfaces (electrical contacts) need to be described: various voltages that might be applied, etc.
  The TSFIs of a TOE that performs protocol processing would be those protocol layers to which a potential attacker has direct access. This need not be the entire protocol stack.

For a given TOE, not all of the interfaces may be *accessible*. That is, the security objectives for the operational environment (in the Security Target) may prevent access to these interfaces or

limit access in such a way that they are practically inaccessible. Such interfaces would **not** be considered TSFIs. Some examples:

- If the security objectives for the operational environment of the stand-alone firewall state that "the firewall will be operational in a server room environment to which only trusted and trained personnel will have access, and which will be equipped with an interruptible power supply (against power failure)", physical and power interfaces will not be accessible, since trusted and trained personnel will not attempt to dismantle the firewall and/or disable its power supply.
- If the security objectives for the operational environment of a software firewall (application) state that "the OS and the hardware will provide a security domain for the application free from tampering by other programs", the interfaces through which the firewall can be accessed by other applications on the OS (e.g. deleting or modifying the firewall executable, direct reading or writing to the memory space of the firewall) will not be accessible, since the OS/hardware part of the operational environment makes this interface inaccessible.
- If the security objectives for the operational environment of a software firewall additionally state that the OS and hardware will faithfully execute the commands of the TOE, and will not tamper with the TOE in any manner, interfaces through which the firewall obtains primitive functionality from the OS and hardware (executing machine code instructions, OS APIs, such as creating, reading, writing or deleting files, graphical APIs etc.) will not be accessible, since the OS/hardware are the only entities that can access that interface, and they are completely trusted.

Requirement: EAL1

b) The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSF Interface (TSFI). The developer does **not** have to categorise the TSFI as SFR-enforcing, SFR-supporting, and SFR-non-interfering. If administrative interfaces are claimed to be inaccessible to untrusted users, the developer must provide a rationale why these functions are inaccessible. Inaccessibility should be addressed by the developer in their test suite.

Related Evaluator Actions: The evaluator will check that the functional specification contains the requested information for each SFR-enforcing and SFR-supporting TSF Interface (TSFI). The evaluator will attempt to categorise the TSFI as SFR-enforcing, SFR-supporting, and SFR-non-interfering.

Requirement: EAL2 EAL3 EAL4 EAL5

c) The functional specification shall completely represent the TSF. This functional specification shall describe the purpose and method of use for all TSF Interfaces (TSFI). If administrative interfaces are claimed to be inaccessible to untrusted users, the developer must provide a rationale why these functions are inaccessible. Such inaccessibility should be addressed by the developer in their test suite.

Related Evaluator Actions: The evaluator will check that the functional specification contains the requested information for all TSF Interfaces.

Requirement: EAL5

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

Background: A semiformal specification is written to a standard presentation template. The presentation may also use more structured languages/diagrams (e.g. data-flow diagrams, state transition diagrams, etc.). Whether based on diagrams or natural language, a set of conventions must be used in the presentation. A glossary has to explicitly identify the words that are being used in a precise and constant manner. A semiformal specification style reduces the ambiguity in the functional specification.

Related Evaluator Actions: The evaluator will check that the TSFI description uses a semiformal style.

Requirement: EAL1

e) The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

Background: *Parameters* are explicit inputs or outputs to an interface that control the behaviour of that interface. For examples,;;; the signals across a set of pins on a chip; etc.

Related Evaluator Actions: The evaluator will check that the requested information has been provided.

Examples:
- Parameters can be the arguments supplied to an API,
- parameters can be the various fields in the packet for a given network protocol,
- parameters can be the individual key values in the Windows Registry,
- parameters can be the signals across a set of pins on a chip.

Requirement: EAL2 EAL3 EAL4 EAL5

f) The functional specification shall identify and describe all parameters associated with each TSFI. The description of the parameters shall be accurate and complete.

Background: *Parameters* are explicit inputs or outputs to an interface that control the behaviour of that interface.

Related Evaluator Actions: The evaluator will check that the requested information has been provided. He will also make use of other evaluation deliverables to determine whether the provided information is complete. “Interface xyz has a parameter i which is an integer" is not an acceptable parameter description. A description such as “parameter i is an integer that indicates the number of users currently logged into the system” is much more acceptable.

Requirement: EAL1

g) The functional specification shall provide rationale for the implicit categorisation of interfaces as SFR-non-interfering.

Background: For SFR-non-interfering interfaces no information (addressed by other requirements) has to be provided. However, a high level rationale is needed why the interfaces **not** addressed in the TSFI description are neither SFR-enforcing nor SFR-supporting.

Related Evaluator Actions: The evaluator will check whether the arguments provided in the rationale are valid.

Requirement: EAL2 EAL3

h) For SFR-enforcing TSFIs, the functional specification shall describe the SFR-enforcing actions associated with the TSFI. The developer does **not** have to label the corresponding TSFI or actions as SFR-enforcing. However, he has to support the analysis of the evaluator (see below) by providing consistent, accurate and complete interface descriptions.

Background: If an action available through an interface plays a role in enforcing any security policy on the TOE (that is, if one of the actions of the interface can be traced to one of the SFRs levied on the TSF), then that interface is *SFR-enforcing*. Such policies are not limited to the access control policies, but also refer to any functionality specified by one of the SFRs contained in the ST. Note that it is possible that an interface may have various actions and results, some of which may be SFR-enforcing and some of which may not.

The SFR-enforcing actions are those that are visible at any external interface and that provide for the enforcement of the SFRs being claimed. For example, if audit requirements are included in the ST, then audit-related actions would be SFR-enforcing and therefore must be described, even if the result of that action is generally not visible through the invoked interface (as is often the case with audit, where a user action at one interface would produce an audit record visible at another interface).

Related Evaluator Actions: The evaluator will use the interface descriptions to support the generation and assessment of test cases against that interface. If the description is unclear or lacking detail such that meaningful testing (which incorporated the stipulation of security functions and validation of the test results) cannot be conducted against the TSFI, it is likely that the description is inadequate. He will thus check that the consistency, accurateness and completeness of the interface descriptions.

Requirement: EAL3

i) The functional specification shall summarise the non-SFR-enforcing actions associated with each TSFI. The developer does **not** have to label the SFR-enforcing actions. However, he has to support the analysis of the evaluator (see below) by providing consistent, and accurate interface descriptions addressing all possible actions in which the TSFIs are involved.

Related Evaluator Actions: The evaluator will use this portions of the interface descriptions to gain extended confidence in his own analysis about categorisation of TSF interfaces and actions. He thus checks the consistency, accurateness and completeness of the interface descriptions.

Requirement: EAL4 EAL5

j) The functional specification shall describe all actions associated with each TSFI.

Related Evaluator Actions: The evaluator will use all evaluation documents to check that the specifications of the TSFIs provided with the functional specification are complete.

Requirement: EAL2

k) For SFR-enforcing TSFIs, the functional specification shall describe the direct error messages resulting from processing associated with the SFR-enforcing actions.

Related Evaluator Actions: The evaluator will check that the functional specification accurately describes all values and messages which the interface itself returns across the TSF boundary as response to SFR-enforcing actions. In order to determine accuracy, the evaluator must be able to understand meaning of the error. For example, if an interface returns a numeric code of 0, 1, or 2, the evaluator would not be able to understand the error if the functional specification only listed: “possible errors resulting from invocation of the *foo()* interface are 0, 1, or 2”. Instead the evaluator checks to ensure that the errors are described such as: “possible errors resulting from invocation of the *foo()* interface are 0 (processing successful), 1 (file not found), or 2 (incorrect filename specification)”.

Examples:

- For an API, the interface itself may return an error code, set a global error condition, or set a certain parameter with an error code.
- For a configuration file, an incorrectly configured parameter may cause an error message to be written to a log file.
- For a hardware PCI card, an error condition may raise a signal on the bus, or trigger an exception condition to the CPU.

Requirement: EAL3

m) For SFR-enforcing TSFIs, the functional specification shall describe the direct error messages resulting from invocation of the TSFI.

Related Evaluator Actions: The evaluator will check that the functional specification accurately describes all values and messages which an SFR-enforcing TSFI itself returns across the TSF boundary. In order to determine accuracy, the evaluator must be able to understand meaning of the error. For example, if an interface returns a numeric code of 0, 1, or 2, the evaluator would not be able to understand the error if the functional specification only listed: “possible errors resulting from invocation of the *foo()* interface are 0, 1, or 2”. Instead the evaluator checks to ensure that the errors are described such as: “possible errors resulting from invocation of the *foo()* interface are 0 (processing successful), 1 (file not found), or 2 (incorrect filename specification)”.

Examples:

- For an API, the interface itself may return an error code, set a global error condition, or set a certain parameter with an error code.
- For a configuration file, an incorrectly configured parameter may cause an error message to be written to a log file.
- For a hardware PCI card, an error condition may raise a signal on the bus, or trigger an exception condition to the CPU.

Requirement: EAL4 EAL5

n) The functional specification shall describe all direct error messages resulting from invocation of the TSFI.

Related Evaluator Actions: The evaluator will check that the functional specification accurately describes all values and messages which a TSFI returns across the TSF boundary in reaction to invocation of the TSFI. In order to determine accuracy, the evaluator must be able to understand meaning of the error. For example, if an interface returns a numeric code of 0, 1, or 2, the evaluator would not be able to understand the error if the functional specification only listed: “possible errors resulting from invocation of the *foo()* interface are 0, 1, or 2”. Instead the evaluator checks to ensure that the errors are described such as: “possible errors resulting from invocation of the *foo()* interface are 0 (processing successful), 1 (file not found), or 2 (incorrect filename specification)”.

Examples:

- For an API, the interface itself may return an error code, set a global error condition, or set a certain parameter with an error code.
- For a configuration file, an incorrectly configured parameter may cause an error message to be written to a log file.
- For a hardware PCI card, an error condition may raise a signal on the bus, or trigger an exception condition to the CPU.

Requirement: EAL5

o) The functional specification shall describe all error messages that do not result from an invocation of a TSFI. The developer shall provide a rationale for each error message contained in the TSF implementation which does not result from an invocation of a TSFI.

Related Evaluator Actions: The evaluator will check that the functional specification accurately and completely describes all values and messages which the design and implementation of all TSFIs comprise. He will also check that the developer has provided a rationale for each for each error message contained in the TSF implementation which does not result from an invocation of a TSFI.

Example: The error messages related to this requirement are typically those that are not expected to be generated, but are constructed as a matter of good programming practices. For example, a *case* statement that defines actions resulting from each of a list of cases may end with a final *else* statement to apply to anything that might not be expected; this practice ensures the TSF does not get into an undefined state. However, it is not expected that the path of execution would ever get to this *else* statement; therefore, any error message generation within this *else* statement would never be generated. Although it would not get generated, it must still be included in the functional specification.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

p) The developer shall provide a tracing from the functional specification to the SFRs. The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Role in the evaluation process: The purpose of this requirement is that the developer supports the evaluator in its analysis whether the functional specification is an accurate and complete instantiation of the SFRs.

Related Evaluator Actions: The evaluator shall check that the tracing links the SFRs to the corresponding TSFIs. The evaluator shall examine the functional specification to determine that it is a complete instantiation of the SFRs.

Example: The requested tracing can be as simple as a table which the evaluator can use as input for his analysis.

7.3 Implementation representation (ADV_IMP)

The function of this family is for the developer to make available the implementation representation 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. The implementation representation is expected to be in a form that captures the detailed internal workings of the TSF.

Requirement: EAL4 EAL5

a) The developer shall make available the implementation representation for the entire TSF. It is not mandated that the developer provides a copy of the implementation representation to the evaluator. When setting up a contract between developer and evaluator it can also be agreed that the evaluator gains access to the implementation representation within the development environment. The implementation representation shall be in the form used by the development personnel.

Related Evaluator Actions: The evaluator will check that the provided implementation representation covers the entire TSF and has been provided in ac acceptable form. The evaluator will use a sample of this implementation representation of his own choice as appropriate in the evaluation process.

Examples: Depending on the nature of the TOE the implementation representation may be software source code, firmware source code, hardware diagrams, IC and/or hardware design language code or layout data.

The implementation representation is manipulated by the developer in form that it 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 they use, 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 maximise the evaluator's analysis efforts.

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.

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 by taking a version of the implementation representation that is used by the TOE developer and running a shrouding or obfuscation program on it. While the shrouded representation is what is compiled and may be closer to the implementation 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 mechanisms.

Requirement: EAL4 EAL5

b) The implementation representation shall define the TSF to a level of detail such that the TSF can be generated without further design decisions.

Related Evaluator Actions: The evaluator samples the implementation representation to gain confidence that it is at the appropriate level and not, for instance, a pseudo-code level which requires additional design decisions to be made.

Requirement: EAL4 EAL5

c) The developer shall provide a mapping between the TOE design description and an adequate sample of the implementation representation. The mapping between the TOE design description and the implementation representation shall demonstrate their correspondence.

Note: It is worth pointing out the developer must choose whether to perform the mapping for the entire implementation representation, thereby guaranteeing that the chosen sample will be covered, or waiting for the sample to be chosen before performing the mapping. The first option is likely more work, but may be completed before the evaluation begins. The second option is less work, but will produce a suspension of evaluation activity while the necessary evidence is being produced.

Related Evaluator Actions: The evaluator will verify the accuracy of a portion of the implementation representation and the TOE design description. For parts of the TOE design description that are interesting, the evaluator would verify that the implementation representation accurately reflects the description provided in the TOE design description.

Example: The TOE design description might identify a login module that is used to identify and authenticate users. If user authentication is sufficiently significant, the evaluator would verify that the corresponding code in fact implements that service as described in the TOE design description. It might also be worthwhile to verify that the code accepts the parameters as described in the functional specification.

7.4 TSF internals (ADV_INT)

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 could lead to vulnerabilities; it is also easier to maintain without the introduction of flaws.

Requirement: EAL5

a) The developer shall provide an TSF internals description and justification. The justification shall explain the characteristics used to judge the meaning of "well-structured".

Role in the evaluation process: The purpose of this requirement is to establish the basis for determining whether the TSF is well-structured.

Related Evaluator Actions: The evaluator will verify that the criteria for determining the characteristic of being well-structured are clearly defined in the justification.

Example: For example, procedural software that executes linearly is traditionally viewed as well-structured if it adheres to software engineering programming practises, such as those defined in IEEE Std 610.12-1990. For example, it would identify the criteria for the procedural software portions of the TSF:

- the process used for modular decomposition
- coding standards used in the development of the implementation
- a description of the maximum acceptable level of intermodule coupling exhibited by the TSF
- a description of the minimum acceptable level of cohesion exhibited the modules of the TSF

For other types of technologies used in the TOE - such as non-procedural software (e.g. object-oriented programming), widespread commodity hardware (e.g. PC microprocessors), and special-purpose hardware (e.g. smart-card processors) - the evaluation authority should be consulted for determining the adequacy of criteria for being "well-structured".

Requirement: EAL5

b) The developer shall design and implement the entire TSF such that it has well-structured internals. The TSF internals description shall demonstrate that the entire TSF is well-structured.

Related Evaluator Actions: The evaluator will analyse the TSF internals description to determine whether it provides a sound explanation of how the TSF meets the criteria laid down in the justification. For example, it would explain how the procedural software portions of the TSF meet the following:

- that there is a one-to-one correspondence between the modules identified in the TSF and the modules described in the TOE design
- how the TSF design is a reflection of the modular decomposition process
- a justification for all instances where the coding standards were not used or met
- a justification for any coupling or cohesion outside the acceptable bounds

7.5 Security policy modelling (ADV_SPM)

It is the objective of this family to provide additional assurance from the development of a formal *security policy model* of the TSF, and establishing a correspondence between the functional specification and this security policy model. Preserving internal consistency the security policy model is expected to formally establish the security principles from its characteristics by means of a mathematical proof.

There are no requirements of this family for EAL1 to EAL5.

7.6 TOE design (ADV_TDS)

The requirements of the TOE design family are intended to provide information so that a determination can be made that the security functional requirements are realised. The goal of design documentation is to provide sufficient information to determine the TSF boundary, and to describe *how* the TSF implements the Security Functional Requirements. As assurance needs increase, the level of detail provided in the description also increases. The amount and structure of the design documentation will depend on the complexity of the TOE and the number of SFRs. The assurance provided for very complex TOEs will benefit 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.

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

An ***SFR-enforcing*** subsystem is a subsystem that provides mechanisms for enforcing an element of a security functional requirement, or directly supports a subsystem that is responsible for enforcing a security functional requirement. If a subsystem provides (implements) an SFR-enforcing TSFI, then the subsystem is SFR-enforcing.

Subsystems can also be identified as ***SFR-supporting*** and ***SFR-non-interfering***. An SFR-supporting subsystem is one that is depended on by an SFR-enforcing subsystem in order to implement an SFR, but does not play as direct a role as an SFR-supporting requirement. An SFR-non-interfering subsystem is one that is not depended upon, in either a supporting or enforcing role, to implement an SFR.

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 non-SFR-enforcing behaviour.

A ***behaviour summary*** of a subsystem is an overview of the actions it performs (e.g. “The TCP subsystem assembles IP datagrams into reliable byte streams”).

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

The ***purpose*** of a module provides sufficient detail that no further design decisions are needed. The correspondence between the source code that implements the module, and the purpose of the module should be readily apparent.

Requirement: EAL2 EAL3 EAL4 EAL5

a) The developer shall provide the design of the TOE.

Role in the evaluation process: The purpose of this requirement is to provide 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.

Related Evaluator Actions: The evaluator will check that the design provides sufficient information to determine the TSF boundary, and that the description shows how the TSF implement the Security Functional Requirements.

Requirement: EAL2 EAL3 EAL4 EAL5

b) The design shall describe the structure of the TOE in terms of subsystems.

Background: The term “subsystem” was chosen to be specifically vague so that it could refer to units appropriate to the TOE (e.g., subsystems, modules). subsystems can even be uneven in scope, as long as the requirements for description of subsystems are met.

Related Evaluator Actions: The evaluator will check the design to ensure that all subsystems of the TOE are identified.

Example: A firewall might be composed of subsystems that deal with packet filtering, with remote administration, with auditing, and with connection-level filtering. The design description of the firewall would describe the actions that are taken by each affected subsystem when a packet arrives at the firewall.

Requirement: EAL4 EAL5

c) The design shall describe the structure of the TOE in terms of modules.

Background: A module is generally a relatively small architectural unit which, unlike subsystems, describe the implementation in a level of detail that can serve as a guide to reviewing the implementation representation. A software module consists of one or more source code files that cannot be decomposed into smaller compilable units. A description of a module should be such that one could create an implementation of the module from the description, and the resulting implementation would be 1) identical to the actual TSF implementation in terms of the interfaces presented and used by the module, and 2) algorithmically identical to the TSF module.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that the entire TOE is described in terms of modules.

Example: For instance, RFC 793 provides a high-level description of the TCP protocol. It is necessarily implementation independent. While it provides a wealth of detail, it is ***not*** a suitable design description because it is not specific to an implementation. An actual implementation can add to the protocol specified in the RFC, and implementation choices (for instance, the use of global data vs. local data in various parts of the implementation) may have an impact on the analysis that is performed. The design description of the TCP module would list the interfaces presented by the implementation (rather than just those defined in RFC 793), as well as an algorithm description of the processing associated with the modules implementing TCP (assuming it was part of the TSF).

Requirement: EAL2 EAL3 EAL4 EAL5

d) The developer shall identify all subsystems of the TSF. This means that he has to identify the subsystems of the TOE which make up the TSF.

Role in the evaluation process: The purpose of this requirement is to assure that the boundary of the TSF is clearly identified.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that the entire TSF is described in terms of subsystems.

Example: For simple TOEs the TSF might cover the whole TOE. For more complex TOE it is expected that the TSF will form a real subset of the TOE.

Requirement: EAL4 EAL5

e) The developer shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that the mapping between the subsystems of the TSF to the modules of the TSF is complete.

Example: For TOEs that are complex enough to warrant a subsystem-level description of the TSF in addition to the modular description, the developer provides a simple mapping showing how the modules of the TSF are allocated to the subsystems.

Requirement: EAL5

f) The developer shall designate each TSF module as either SFR-enforcing, SFR-supporting, or SFR-non-interfering.

Note: The accuracy of these designations is continuously reviewed as the evaluation progresses. The concern is the mis-designation of modules as being less important (and hence, having less information) than is really the case.

Role in the evaluation process: The purpose of designating each module (according to the role a particular module plays in the enforcement of the SFRs) is to allow developers to provide less information about the parts of the TSF that have little role in security.

Related Evaluator Actions: The evaluator will check the TOE design to determine that the TSF modules are identified as either SFR-enforcing, SFR-supporting, or SFR-non-interfering.

Requirement: EAL2

g) The design shall describe the behaviour of each SFR-supporting or SFR-non-interfering TSF subsystem in detail sufficient to determine that it is not SFR-enforcing. The developer does **not** have to categorise the subsystems as SFR-enforcing, SFR-supporting, and SFR-non-interfering.

Role in the evaluation process: The point of categorising the subsystems is to allow the developer to provide less information for non-SFR-enforcing subsystems than for SFR-enforcing subsystems. The evaluator makes a determination to categorise the subsystems based on the evidence provided by the developer, that the subsystems that do not have high-level descriptions are non-SFR-enforcing (that is, either SFR-supporting or SFR-non-interfering).

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that each non-SFR-enforcing subsystem of the TSF is described such that the evaluator can determine that the subsystem is non-SFR-enforcing.

Requirement: EAL3

h) The design shall describe the behaviour of each SFR-non-interfering TSF subsystem in detail sufficient to determine that it is SFR-non-interfering.

Role in the evaluation process: The point of categorising the subsystems is to allow the developer to provide less information for non-SFR-enforcing subsystems than for SFR-non-interfering subsystems. The evaluator makes a determination to categorise the subsystems based on the evidence provided by the developer, that the subsystems that do not have detailed descriptions are SFR-non-interfering.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that each SFR-non-interfering subsystem of the TSF is described such that the evaluator can determine that the subsystem is SFR-non-interfering.

Requirement: EAL4 EAL5

i) The design shall provide a description of each subsystem of the TSF. The subsystem-level description has to contain a description of how the security functional requirements are achieved in the design, but at a level of abstraction above the modular description.

Background: When having a subsystem-level description of the TSF in addition to the modular description, the goal of the subsystem-level description is to give the evaluator context for the modular description that follows.

Role in the evaluation process: The purpose of this requirement is to assure that the role of each subsystem of the TSF in the enforcement of SFRs is described.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that the TOE design describes each subsystem of the TSF with its role in the enforcement of SFRs.

Requirement: EAL2

j) The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.

Background: SFR-enforcing behaviour refers to *how* a subsystem provides the functionality that implements an SFR. A high-level description need not refer to specific data structures (although

it may), but instead talks about more general data flow, message flow, and control relationships within a subsystem. The goal of these descriptions is to give the evaluator enough information to understand *how* the SFR-enforcing behaviour is achieved.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that it provides a complete, accurate, and high-level description of the SFR-enforcing behaviour of the SFR-enforcing subsystems.

Requirement: EAL3

k) The design shall describe the SFR-enforcing behaviour of the SFR-enforcing subsystems. SFR-enforcing behaviour refers to *how* a subsystem provides the functionality that implements an SFR. While not at the level of an algorithmic description, a detailed description of behaviour typically discusses how the functionality is provided in terms of what key data and data structures are, what control relationships exist within a subsystem, and how these elements work together to provide the SFR-enforcing behaviour. The developer does **not** have to categorise the subsystems as SFR-enforcing, SFR-supporting, and SFR-non-interfering.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that it provides a complete, accurate, and detailed description of the SFR-enforcing behaviour of the SFR-enforcing subsystems.

Requirement: EAL4 EAL5

l) The design shall describe each SFR-enforcing module in terms of its purpose, SFR-related interfaces, return values from those interfaces, and called interfaces to other modules. For EAL4 the developer does **not** have to categorise the modules as SFR-enforcing, SFR-supporting, and SFR-non-interfering. The description of the interfaces presented by each SFR-enforcing module shall contain an accurate and complete description of the SFR-related parameters, the invocation conventions for each interface, and any values returned directly by the interface.

Background: The SFR-related interfaces of a module are those interfaces used by other modules as a means to invoke the SFR-related operations provided, and to provide inputs to or receive outputs from the module. The purpose in the specification of these interfaces is to permit the exercise of them during testing. Inter-module interfaces that are not SFR-related need not be specified or described, since they are not a factor in testing.

SFR-related interfaces are described in terms of how they are invoked, and any values that are returned. This description would include a list of SFR-related parameters, and descriptions of these parameters. Note that global data would also be considered parameters if used by the module (either as inputs or outputs) when invoked. If a parameter were expected to take on a set of values (e.g., a “flag” parameter), the complete set of values the parameter could take on that would have an effect on module processing would be specified. Likewise, parameters representing data structures are described such that each field of the data structure is identified and described.

Note that different programming languages may have additional “interfaces” that would be non-obvious; an example would be operator/function overloading in C++. This “implicit interface” in the class description would also be described as part of the low-level TOE design.

Note that although a module could present only one interface, it is more common that a module presents a small set of related interfaces.

Invocation conventions are a programming-reference-type description that one could use to correctly invoke a module's interface if one were writing a program to make use of the module's functionality through that interface. This includes necessary inputs and outputs, including any set-up that may need to be performed with respect to global variables.

Values returned through the interface refer to values that are either passed through parameters or messages; values that the function call itself returns in the style of a “C” program function call; or values passed through global means (such as certain error routines in *ix-style operating systems).

Related Evaluator Actions: The evaluator will check the TOE design to determine that the description of each SFR-enforcing module provides sufficient detail.

Requirement: EAL3

m) The design shall summarise the non-SFR-enforcing behaviour of the SFR-enforcing subsystems.

Role in the evaluation process: The purpose of this requirement is

- to provide the evaluator greater understanding of the way each subsystem works,
- to assure that all SFR-enforcing behaviour exhibited by a subsystem has been described.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that it provides a complete and accurate high-level description of the non-SFR-enforcing behaviour of the SFR-enforcing subsystems.

Requirement: EAL4

n) The design shall describe each SFR-supporting module in terms of its purpose and interaction with other modules.

Related Evaluator Actions: The evaluator will check the TOE design to determine that the non-SFR-supporting modules are correctly categorised, and that the description of each non-SFR-supporting module contains its purpose and the interaction with other modules.

Requirement: EAL5

o) The design shall describe each SFR-supporting module in terms of its purpose, SFR-related interfaces, return values from those interfaces, and called interfaces to other modules. For details refer to requirement l).

Related Evaluator Actions: The evaluator will check the TOE design to determine that the description of each SFR-supporting module provides sufficient detail.

Requirement: EAL4 EAL5

p) The design shall describe each SFR-non-interfering module in terms of its purpose and interaction with other modules.

Related Evaluator Actions: The evaluator will check the TOE design to determine that the SFR-non-interfering modules are correctly categorised, and that the description of each SFR-non-interfering module contains its purpose and the interaction with other modules.

Requirement: EAL2

q) The design shall provide a description of the interactions among SFR-enforcing subsystems of the TSF, and between SFR-enforcing subsystems of the TSF and other subsystems of the TSF. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a subsystem to a routine in a different subsystem; global variables; hardware signals (e.g., interrupts) from a hardware subsystem to an interrupt-handling subsystem), but the data elements identified for a particular subsystem that are going to be used by another subsystem should be covered in this discussion. Any control relationships between subsystems (e.g., a subsystem responsible for configuring a rule base for a firewall system and the subsystem that actually implements these rules) should also be described.

Role in the evaluation process: The goal of describing the interactions involving the SFR-enforcing subsystems is to help provide the reader a better understanding of how the TSF performs it functions.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that interactions involving the SFR-enforcing subsystems of the TSF are described.

Requirement: EAL3 EAL4 EAL5

r) The design shall provide a description of the interactions among all subsystems of the TSF. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a subsystem to a routine in a different subsystem; global variables; hardware signals (e.g., interrupts) from a hardware subsystem to an interrupt-handling subsystem), but the data elements identified for a particular subsystem that are going to be used by another subsystem should be covered in this discussion. Any control relationships between subsystems (e.g., a subsystem responsible for configuring a rule base for a firewall system and the subsystem that actually implements these rules) should also be described.

Role in the evaluation process: The goal of describing the interactions between the subsystems is to help provide the reader a better understanding of how the TSF performs it functions.

Related Evaluator Actions: The evaluator will analyse the TOE design to determine that interactions between the subsystems of the TSF are described.

Requirement: EAL2 EAL3

s) The developer shall provide a mapping from the TSFI of the functional specification to the lowest level of decomposition available in the TOE design. The mapping shall demonstrate that all behaviour described in the TOE design is mapped to the TSFIs that invoke it.

Background: The subsystems described in the TOE design provide a description of how the TSF works at a detailed level for SFR-enforcing portions of the TSF, and at a higher level for other portions of the TSF. The TSFI provide a description of how the implementation is exercised. The evidence from the developer identifies the subsystem that is initially involved when an operation is requested at the TSFI, and identify the various subsystems that are primarily responsible for implementing the functionality.

Note: A complete “call tree” for each TSFI is not required.

Related Evaluator Actions: The evaluator will check the TOE design to determine that it contains a complete and accurate mapping from the TSFI described in the functional specification to the subsystems of the TSF described in the TOE design.

Requirement: EAL4 EAL5

t) The developer shall provide a mapping from the TSFI of the functional specification to the lowest level of decomposition available in the TOE design. The mapping shall demonstrate that all behaviour described in the TOE design is mapped to the TSFIs that invoke it.

Background: The modules described in the TOE design provide a description of the implementation of the TSF. The TSFI provide a description of how the implementation is exercised. The evidence from the developer identifies the module that is initially invoked when an operation is requested at the TSFI, and identify the chain of modules invoked up to the module that is primarily responsible for implementing the functionality.

Note: A complete “call tree” for each TSFI is not required.

Related Evaluator Actions: The evaluator will check the TOE design to determine that it contains a complete and accurate mapping from the TSFI described in the functional specification to the subsystems of the TSF described in the TOE design.

8 Class AGD: Guidance Documents

Assurance class AGD defines requirements directed at the understandability, coverage and completeness of the preparative and operational documentation provided by the developer for the user. A **user** in this context is a person, which is authorised to perform TOE related operations in accordance with the SFRs. This documentation, which provides information for all user roles, is an important factor in the secure preparation and operation of the TOE.

8.1 Operational user guidance (AGD_OPE)

Requirements for operational user guidance help ensure that all types of users are able to operate the TOE in a secure manner. It should be excluded that the TOE can be used in a manner that is insecure but that the user of the TOE would reasonably believe to be secure. Operational user guidance is the primary vehicle available to the developer for providing the TOE users with the necessary background and specific information on how to correctly use the TOE's protection functions.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

a) The operational user guidance shall contain an overview about the security functionality that is visible at the user interfaces. The user guidance shall identify and describe for each user role the visible security interfaces, their purpose, and the functions available through these interfaces. For each user-accessible interface, the user guidance should identify the method(s) by which the interface is invoked (e.g. command-line, programming-language system call, menu selection, command button).

Related Evaluator Actions: The evaluator will check that the operational user guidance provides such overview information.

The evaluator will check that the operational user guidance in **all** its sections is clear (i.e. it cannot reasonably be misconstrued by an administrator or user and can due to this be used in a way detrimental to the TOE, or to the security provided by the TOE.) and reasonable (i.e. it does not make demands on the TOE's usage or operational environment that are inconsistent with the ST or unduly onerous to maintain security.).
The evaluator will also check that the user guidance is consistent with the other evaluation deliverables (specifically Security Target and functional specification).

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

b) The configuration of the TOE may allow different user roles to have dissimilar privileges in making use of the different functions of the TOE. This means that some users are authorised to perform certain functions, while other users may not be so authorised. These functions and privileges should be described, for each user role, by the user guidance. The user guidance shall describe, for each user role, the user-accessible functions and privileges that must be controlled in a secure processing environment, the types of commands required for them, and the reasons for such commands. The user guidance should contain warnings regarding the use of these functions and privileges. Warnings should address expected effects, possible side effects, and possible interactions with other functions and privileges. The user guidance should provide advise regarding the effective use of the security functions of the TOE.

Background: Various users will typically have different TOE related roles, i.e. tasks and responsibilities. There will typically be end-users who are aiming at using the end-user oriented functions and services of the TOE. There will typically also be privileged administrators who have to configure, administrator the TOE and monitor its operation. There can also be revisors who can review actions of and settings for other user groups. Depending on the type of TOE there can also be programmers making use of services provided by a software library or a hardware device.
Depending on their role users can access the security functions provided by the TOE in different ways and with different privileges. Administrators can typically specify the password building rules mandatory for all users. End-users on the other side can typically change only their own passwords. In specific products log files can only be deleted by a privileged revisor. The behaviour of other security functions (like storage reuse) will not be influenced by any user.

Related Evaluator Actions: The evaluator will check that the operational user guidance provides such information in sufficient detail.

Examples:

- The TOE might distinguish various administrative roles including administrators with the ability to fully control all configuration parameters and auditors who are only authorised to inspect the configuration parameters.
- The TOE might provide the capability to temporarily or permanently turn off security functions (like disabling logging or weakening authentication for specific methods of access). This might be acceptable in specific environment or in special situations. The privilege to perform such functions obviously needs to be restricted and clearly documented in the user guidance.
- The TOE might provide the capability to perform session traces which contain all information entered by users including passwords (like telnet sessions). This might be acceptable in special situations (like troubleshooting). The privilege to activate such functions obviously needs to be restricted and clearly documented in the user guidance. The user guidance has also to clearly describe how such information is removed.
- A firewall product might be intended to be used via a dedicated management station. The use of the firewall console for administration activities will then be discouraged. In special situations – like all users of the graphical user interface are locked out, or the graphical user interface is not operational, or hardware problems have to be debugged and resolved – console access will be necessary. This has to be performed under very restrictive boundary conditions. The administrator and the organisation's security officer have to be aware that some security functionality (like logging) might be limited in such situations.

- The advice to be provided regarding the effective use of the security functions might address aspects like reviewing password composition practises, suggested frequency of user file backups, or discussion on the effects of changing user access privileges.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

c) The operational user guidance shall describe for each user role and each user-accessible interface:
   - the parameters to be set by the user, their particular purposes, valid and default values, and secure and insecure use settings of such parameters, both individually or in combination.
   - the immediate TOE response, message, or code returned.

Related Evaluator Actions: The evaluator will check that the operational user guidance provides such information in sufficient detail.

Examples:
- If the TOE provides command-line access it is adequate – at least in the Unix/Linux system environment - to provide detailed usage information in the form of manpages.
- If the TOE provides a graphical user interface it is adequate to visualise the various windows/forms and to explain all fields, admissible values, default values, and secure and insecure use values.
- If the TOE provides an application program interface it is adequate to specify the interface in formal notation and supplementing this specification by an information description of the provided methods/functions, signatures, admissible values, default values, and secure and insecure use values.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

d) 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. All types of security-relevant events are detailed for each user role, such that each user knows what events may occur and what action (if any) he may have to take in order to maintain security. Security-relevant events that may occur during operation of the TOE are adequately defined to allow user intervention to maintain secure operation.

Related Evaluator Actions: The evaluator will check that the operational user guidance provides such information in sufficient detail.

Examples:
- Example events to be considered are audit trail overflow, system crash, specific audit trail entries indicating penetration attempts, and specific audit trail entries indicating system misfunctions. The administrator has be informed about the nature of the event and about the actions to be taken. The range of actions to be taken includes clearing specific files, collection information to report the problem to the developer, disconnecting the TOE from all or specific networks, reinstall the TOE, power off the TOE.
- Example events to be considered are also when a user account is to be removed when the user leaves the organisation or is assigned different responsibilities.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

e) 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.

Related Evaluator Actions: The evaluator will check that the operational user guidance provides such information in sufficient detail. He will make intensive use of the other evaluation documents to assess the completeness of the description.

Examples:

- Unix/Linux type operating systems distinguish between single-user-mode and multi-user-mode.
- Some operating systems support beside normal operating mode a maintenance operating mode in which specific diagnostic utilities are enabled and in which security functions might be disabled.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

f) The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfil the security objectives for the operational environment as described in the ST. The security measures described in the user guidance should include all relevant external procedural, physical, personnel and connectivity measures.

Related Evaluator Actions: The evaluator analyses the security objectives for the operational environment in the ST and determines that for each user role, the relevant security measures are described appropriately in the user guidance.

8.2 Preparative procedures (AGD_PRE)

Preparation requires that the delivered copy of the TOE is accepted, configured and activated by the user to exhibit the protection properties as needed during operation of the TOE. The preparative procedures provide confidence that the user will be aware of the TOE configuration parameters and how they can affect the TSF.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

a) The developer shall provide the TOE including its preparative procedures. The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with developer's delivery procedures. The acceptance procedures shall reflect the steps the user has to perform in order to accept the delivered TOE that are implied by developer's delivery procedures. The acceptance procedures shall include as a minimum, that the user has to check that all parts of the TOE as indicated in the ST have been delivered in the correct version. The acceptance procedures should also describe whether a user can verify the integrity and/or authenticity of the delivered TOE or detect a non-authorised delivery and should describe any corresponding procedures. When designing such procedures it should be considered that the preparative procedures provided to the user could itself be subject to unauthorised modification or unauthorised delivery. If the description of the delivery procedures states that no user acceptance procedure takes place, such statement shall be re-stated in the preparative procedures together with an adeaquate rationale.

Related Evaluator Actions: The evaluator will check that the developer has provided suitable acceptance procedures as part of the preparative procedures and that these acceptance procedures are consistent with the developer's delivery procedures. The evaluator will repeat all user procedures necessary to accept the TOE to determine that the TOE can be accepted securely using only the supplied preparative user guidance.

Examples:

- The user shall be provided with the information how he can identify the identity of the TOE or TOE parts. A comprehensive example is provided in chapter 2 of appendix B.
- If the developer's delivery procedures state that a user can verify the integrity and/or authenticity of the delivered TOE, the preparative procedures shall inform the user about corresponding activities. It has to be considered that the preparative procedures provided to the user could itself be subject to unauthorised modification.
- If the developer's delivery procedures state that a user can detect a non-authorised delivery, the preparative procedures shall inform the user about corresponding activities. It has to be considered that the preparative procedures provided to the could itself be part of an unauthorised delivery.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

b) The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment. The installation procedures should provide detailed information about the following, if applicable: minimum system requirements for secure installation; requirements for the operational environment in accordance with the security objectives provided by the ST; changing the installation specific security characteristics of entities under the control of the TOE (for example parameters, settings, passwords); handling exceptions and problems.

Related Evaluator Actions: The evaluator will check that the developer has provided suitable installation procedures and that these procedures are consistent with the security objectives for the operational environment as described in the ST. The evaluator will repeat all user procedures necessary to install the TOE to determine that the TOE and its operational environment can be installed securely using only the supplied preparative user guidance.

Examples:

- If passwords have to changed accounts have to be deleted/inactivated during the installation process, this has to be documented in the installation procedures..
- If the ST mandates a specific network configuration (like installation of a dedicated network for management access), this has to be documented in the installation procedures.
- If the ST forbids the activation of specific networking or system capabilities (like activating a telnet access port for administration), this has to be documented in the installation procedures.

9 Class ALC: Life-Cycle Support

Assurance class ALC: Life-cycle support defines requirements for assurance through the adoption of a well defined life-cycle model for all the steps of the TOE development, including flaw remediation procedures and policies, correct use of tools and techniques and the security measures used to protect the development environment.

9.1 CM capabilities (ALC_CMC)

Configuration management capabilities define the characteristics of the configuration management system.

Whereas this chapter deals with the requirements on the **capabilities** of the (CM) system, chapter 9.2 specifies requirements for the **scope** of the CM system.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

a) The developer has to provide the TOE which is labelled with its reference which is stated in the ST. The labelling must be performed in a way that the TOE can be uniquely identified by a consumer at the point of purchase or use. If multiple labels are assigned then the labels have to be consistent.

Background: If the TOE is labelled more than once then the labels have to be consistent. For example, it should be possible to relate any labelled guidance documentation supplied as part of the TOE to the evaluated operational TOE. This ensures that consumers can be confident that they have purchased the evaluated version of the TOE, that they have installed this version, and that they have the correct version of the guidance to operate the TOE in accordance with its ST.

Related Evaluator Actions: The evaluator will check that the TOE is labelled with its reference and that all labels are consistent if multiple labels are assigned. Although the evaluator will have to confirm this for the final version of the TOE, he will seek for related evidence in earlier phases of the evaluation.

There is no explicit requirement for the developer to provide documentation for this evaluation activity which supplements the delivered TOE. To support the evaluation process, however, the developer is encouraged to consider the following aspects:
The evaluator will analyse the evaluation documentation to determine how the labelling is performed. Supplemental notes from the developer provided with the TOE might facility this activity.

The evaluator will analyse whether labels are used consistently if multiple labels are used. To facilitate this analysis even pre-versions of the TOE should be delivered and labelled as it is intended for the final version of the TOE. This also applies to labels of CDs and packaging.

Examples:

Labelling a TOE with its reference

- An application library can be labelled by providing a method/function which returns a string indicating the reference.
- A command line program can be labelled by providing an option (like –v) which displays the reference.
- A window based application can be labelled by providing a window which displays the reference.
- A software product can be labelled by providing a log entry which displays the reference.

- A software product can be labelled by providing a start-up message which contains the reference.
- A hardware or firmware product can be labelled by a part number physically stamped on the TOE.
- Documents (like guidance delivered with the TOE) can be labelled by a version number and a date printed on the cover.
- Products can also be identified through labelled packaging or media.

Labelling a multi-part TOE

- A multi-part TOE can be labelled by providing the same label for each part of the TOE.
- A multi-part TOE can also be labelled by providing a specific label for each part of the TOE. This principle can for instance be applied for a firewall engine and the corresponding administration GUI.

Requirement: EAL2 EAL3 EAL4 EAL5

b) The developer must use a CM system.

Background: A CM system is the overall term for the set of procedures and tools (including their documentation) used by a developer to develop and maintain configurations of his products during their life-cycles. CM systems may have varying degrees of rigour and function. At higher levels, CM systems have to be automated, with flaw remediation, change controls, and other tracking mechanisms.

All documentation which needs to be delivered by the developer for this evaluation aspect is addressed in the next section. The documentation which has to be provided reflects the various requirements regarding capabilities and evidence increasing with the evaluation assurance level.

This requirement for developer action provides a mean to provide assurance that the developer has implemented adequate procedures to control the processes of refinement and modification of the TOE and the related information. CM systems are put in place to ensure the integrity of the portions of the TOE controlled by them.

Related Evaluator Actions: The evaluator will analyse the CM documentation provided (see next section) to determine whether the capabilities of the CM system are sufficient for the target evaluation level and whether the evidence given provides sufficient information that the CM system is used as described.

Starting with EAL3 the evaluator needs to get access to the developers CM system and its supporting tools.

Requirement: EAL2 EAL3 EAL4 EAL5

c) The developer shall provide the CM documentation.

Background: CM documentation is the overall term for the CM usage documentation and the CM output documentation. CM usage documentation is the part of the CM system, which describes how the CM system is defined and applied by using e.g. handbooks, regulations and/or documentation of tools and procedures. CM output comprises the results produced or enforced by the CM system. These CM related results could occur as documents (e.g. filled paper forms, CM system records, logging data, hard copies and electronic output data) as well as actions (e.g. manual measures to fulfil CM instructions). Examples of such CM outputs are configuration lists, CM plans and/or behaviours during the product life-cycle.

The fulfilment of this requirement also helps to convince the evaluator that the developer uses a CM system as described. The CM usage documentation might reflect an intention of the developer to implement a CM system and might support the implementation of the described CM system. The CM output documentation on the other side will also provide evidence that a CM system has been implemented and is used as described.

Related Evaluator Actions: The evaluator will analyse the CM documentation provided to determine whether the CM system is sufficiently documented and whether the capabilities of the CM system are sufficient for the target evaluation level whether the evidence given provides sufficient information that the CM system is used as described.
The specific requirements are discussed in detail below.

Requirement: EAL2 EAL3 EAL4 EAL5

d) The developer shall provide an overview about the CM system and describe how it is used.

Background: This part of the CM documentation will typically base on or incorporate documentation and other information from the development environment. Its main purpose is to allow the **evaluator** to understand how the CM system is built up and how it is used.

Related Evaluator Actions: The evaluator will analyse the documentation provided to determine whether it is sufficiently detailed to allow to understand the technical and procedural components and elements of the CM system.

Requirement: EAL2 EAL3 EAL4 EAL5

e) The CM system must have the capability to distinguish different versions of configuration items. The developer shall describe the method used to uniquely identify the configuration items and justify that the CM system in use provides the claimed capability. The description should encompass the approach to version control and unique referencing of the TOE. The documentation of related procedures should show how the status of each configuration item can be tracked throughout the life-cycle of the TOE. The procedures may be detailed in the CM plan or throughout the CM documentation. The information included should describe:

- the method how each configuration item is uniquely identified, such that it is possible to track versions of the same configuration item;
- the method how configuration items are assigned unique identifiers and how they are entered into the CM system;
- the method to be used to identify superseded versions of a configuration item.

Related Evaluator Actions: The evaluator will analyse the CM documentation to determine whether it provides sufficient information to explain how the various configuration items are identified and how uniqueness of versions is achieved.

Examples for Identification of a configuration item: A configuration name is uniquely identified by a unique name (name of a file, title of a document) and a version identifier. Whenever a configuration item is changed, its identification must change, too.
Version identifier typically contain multi-part version numbers, such as “2.3” or “1.2.4”. The rules regarding incrementing version numbers have to be documented in the CM plan.

Care has to be taken when a date is used as part of the version identifier for several reasons:

- A configuration item might be changed several times a day. The description of the identification method used has to take this into account. The integration of the corresponding time might help to solve related problems.
- Depending on the operating system several dates may be assigned to a file (like creation date, date of last modification, date of last access). The description of the identification method should clearly state which date is used for identification and should describe **how** it is assured that the identifier changes if the configuration item is changed.

The methods used may vary for different types of configuration items.

Requirement: EAL3 EAL4 EAL5

f) The CM plan (as part of the CM documentation) shall describe how the CM system is used for the development of the TOE. The main objective of issuing a CM plan is that each staff can see clearly what he has to do. The CM plan also defines and describes the output of the CM usage (CM system records). As such the CM plan contains a usage guide for the users (developers, testers, quality assurance) of the CM system. The CM plan shall identify all activities performed in the TOE development that are subject to configuration management procedures (e.g. creation, modification or deletion of a configuration item, data-backup, archiving). The CM plan shall also describe how CM instances (e.g. change control boards, interface control working groups) are introduced and staffed. The CM plan should also address the avoidance of concurrency problems as a result of simultaneous changes to configuration items. The information expected comprises technical information (handbooks, guidance, reference documents) as well as the specification of roles, responsibilities, forms and procedures.

Related Evaluator Actions: The evaluator will analyse the CM plan whether it contains a CM usage guide for the users of the CM system and whether it describes what the expected output of the CM usage.

Requirement: EAL3 EAL4 EAL5

g) The CM system shall provide measures such that only authorised changes are made to the configuration items. The CM documentation shall describe the access control measures.

Related Evaluator Actions: The evaluator will analyse the CM documentation describing also the CM access control measures to determine whether they are effective in preventing unauthorised access to the configuration items and are thus suitable to maintain the integrity of the TOE configuration items.

Requirement: EAL3 EAL4 EAL5

h) The developer shall provide evidence which demonstrates that the CM system is operating in accordance with the CM plan and that all configuration items have been and are being maintained under the CM system.

Role in the evaluation process: The purpose of this requirement is to support the evaluation process. It requires evidence to be provided.

Related Evaluator Actions: The evaluator will analyse the evidence provided to determine whether the CM system is used as described in the CM plan and that the actual CM output

includes the CM system records as described in the CM plan. He will also analyse the evidence provided to determine whether the CM systems maintains the configuration items identified in the configuration list.

Requirement: EAL4 EAL5

i) The CM system shall provide **automated** measures such that only authorised changes are made to the configuration items. The CM documentation shall describe the automated access control measures.

Role in the evaluation process: The purpose of this requirement is to support the development process. It shall assure that the CM system is less susceptible to human error or negligence by requesting support by automated means (tools).

Related Evaluator Actions: The evaluator will analyse the CM documentation describing also the automated CM access control measures to determine whether they are effective in preventing unauthorised access to the configuration items and are thus suitable to maintain the integrity of the TOE configuration items.

Requirement: EAL4 EAL5

j) The CM system shall support the production of the TOE by automated means.

Role in the evaluation process: The purpose of this requirement is to support the development process. It shall assure that the CM is system less susceptible to human error or negligence by requesting that the production of the TOE within CM system is supported by tools (tools).

Related Evaluator Actions: The evaluator will analyse the CM plan to determine whether it contains a description of automated procedures and the related tools to support the generation of the TOE.

Requirement: EAL4 EAL5

k) The CM plan shall describe the acceptance procedures which are used for the TOE and its configuration items. There are several types of acceptance situations to consider, some of which may overlap:

- accepting an item into the CM system for the first time, in particular inclusion of software, firmware and hardware components from other manufacturers into the TOE (“integration”);
- modifying configuration items;
- moving configuration items to the next life-cycle phase at each stage of the construction of the TOE (e.g. module, subsystem, system);
- subsequent to transports between different development sites;
- subsequent to the delivery of the TOE to the consumer.

The information provided must describe the roles or individuals involved and their responsibilities. The evaluator must be convinced that only configuration items that have passed an adequate and appropriate review and are of adeaquate quality are incorporated in the TOE. Note, that for EAL4 and EAL5 it is **not** required that the person accepting a configuration item is different from the person who has developed it.

Related Evaluator Actions: The evaluator will analyse the CM plan to determine whether it includes the procedures used to accept modified or newly created configuration items as parts of the TOE.

9.2 CM scope (ALC_CMS)

This chapter deals with the **scope** of the CM system which indicates the TOE items that need to be controlled by the CM system. This is reflected by the required contents of the configuration list to be provided.

A **configuration item** is an object managed by the CM system during the TOE development. These may be either parts of the TOE or objects related to the development of the TOE like evaluation documents or development tools. CM items may be stored in the CM system directly (e.g. for files) or by reference (e.g. for hardware parts) together with their version.

The **configuration list** is a CM output document listing all configuration items for a specific product together with the exact version of each CM item relevant for a specific version of the complete product. - 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 CM 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 CM tool. 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.).

Requirement: EAL1

a) The configuration list includes the following as configuration items: the TOE itself; and the other evaluation evidence required for EAL1 (security target, functional specification, guidance documentation). The configuration list shall uniquely identify these configuration items.

Related Evaluator Actions: The evaluator will check that the configuration list includes the above listed configuration items and that these configuration items are uniquely referenced.

Requirement: EAL2

b) The configuration list includes the following as configuration items: the TOE itself; the parts that comprise the TOE (software modules, hardware components), and the other evaluation evidence required for EAL2 (security target, architectural design, functional specification, guidance documentation, TOE design, CM documentation, description of delivery procedures, test documentation). The configuration list shall uniquely identify these configuration items.

Related Evaluator Actions: The evaluator will check that the configuration list includes the above listed configuration items and that these configuration items are uniquely referenced.

Requirement: EAL2 EAL3 EAL4 EAL5

c) For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

Related Evaluator Actions: The evaluator will analyse the configuration list provided to determine whether such information is supplied for the configuration items for which the identification of a developer is applicable.

Requirement: EAL3

d) The configuration list includes the following as configuration items: the TOE itself; the parts that comprise the TOE (software modules, hardware components), the TOE implementation representation, and the other evaluation evidence required for EAL3 (security target, architectural design, functional specification, guidance documentation, TOE design, CM documentation, description of delivery procedures, documentation of development security, test documentation). The configuration list shall uniquely identify these configuration items.

Related Evaluator Actions: The evaluator will check that the configuration list includes the above listed configuration items and that these configuration items are uniquely referenced.

Requirement: EAL4

e) The configuration list includes the following as configuration items: the TOE itself; the parts that comprise the TOE (software modules, hardware components), the TOE implementation representation, the other evaluation evidence required for EAL4 (security target, architectural design, functional specification, guidance documentation, TOE design, CM documentation, description of delivery procedures, documentation of development security, test documentation, life-cycle description, description of tools and techniques), and security flaw reports and resolution status. The configuration list shall uniquely identify these configuration items.

Related Evaluator Actions: The evaluator will check that the configuration list includes the above listed configuration items and that these configuration items are uniquely referenced.

Requirement: EAL5

f) The configuration list includes the following as configuration items: the TOE itself; the parts that comprise the TOE (software modules, hardware components), the TOE implementation representation, the other evaluation evidence required for EAL5 (security target, architectural design, functional specification, guidance documentation, TOE design, TSF internals documentation, CM documentation, description of delivery procedures, documentation of development security, test documentation, life-cycle description, description of tools and techniques), security flaw reports and resolution status, and all tools involved in the development and production of the TOE including the names, versions, configurations and roles of each development tool, and related documentation. The configuration list shall uniquely identify these configuration items.

Related Evaluator Actions: The evaluator will check that the configuration list includes the above listed configuration items and that these configuration items are uniquely referenced.

9.3 Delivery (ALC_DEL)

Delivery is the product life-cycle phase which is concerned with the transmission of the finished TOE from the production environment into the hands of the consumer. This may include packaging and storage at the development site, but does not include transports of the unfinished TOE or parts of the TOE between different developers or different development sites. The concern of this family is the secure transfer of the finished TOE from the development environment into the responsibility of the user.

Requirement: EAL2 EAL3 EAL4 EAL5

a) The developer shall document procedures for delivery of the TOE or parts of it to the consumer. The delivery documentation shall describe all procedures that are necessary to maintain security when distributing instances of the TOE or parts to the consumer. It shall describe proper procedures to maintain security of the TOE or its parts and to determine the identification of the TOE. The delivery documentation should cover the entire TOE, but may contain different procedures for different parts of the TOE. The delivery procedures should be applicable across all phases of delivery from the production environment to the installation environment (e.g. packaging, storage and distribution).

Related Evaluator Actions: The evaluator will analyse the delivery documentation to determine whether all procedures that are necessary to maintain security when distributing versions of the TOE to the consumer are described. In assessing the necessity of procedures he will also consider the overall security level stated for the TOE by the chosen level of the Vulnerability Assessment. If the TOE is required to be resistant against attackers of a certain potential in its intended environment, this should also apply to the delivery of the TOE. The evaluator should determine that a balanced approach has been taken, such that delivery does not present a weak point in an otherwise secure development process.

Note: The CC does not mandate specific delivery practises. Standard commercial practise for packaging and delivery may be acceptable. This includes shrink wrapped packaging, a security tape or a sealed envelope. For the distribution, physical (e.g. public mail or a private distribution service) or electronic (e.g. electronic mail or downloading off the Internet) procedures may be used. Cryptographic checksums or a software signature may be used by the developer to ensure that tampering or masquerading can be detected. Tamper proof seals additionally indicate if the confidentiality has been broken. For software TOEs, confidentiality might be assured by using encryption. If availability is of concern, a security transport might be required.

Requirement: EAL2 EAL3 EAL4 EAL5

b) The developer shall use the delivery procedures.

Related Evaluator Actions: The evaluator will search for evidence that the described procedures are used as described.

9.4 Development Security (ALC_DVS)

Development security covers the physical, procedural, personnel, and other security measures used in the development environment. It includes physical security of the development location(s) and controls on the selection and hiring of development staff.

Requirement: EAL3 EAL4 EAL5

a) The developer shall produce development security documentation. This documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment. Appendix C indicates aspects which should be addressed in the developer documentation. The developer documentation shall also specify the underlying confidentiality and integrity policies.

Related Evaluator Actions: The evaluator will analyse the development security documentation to determine whether it details all security measures used in the development environment that are necessary to protect the confidentiality and integrity of the TOE design and implementation. The evaluator will also analyse the development security documentation whether the security measures employed are sufficient to enforce underlying confidentiality and integrity policies and whether they are consistent with any objectives for the development environment stated in the ST.

Requirement : EAL3 EAL4 EAL5

b) The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Related Evaluator Actions: The evaluator will use the provided evidence as one means to determine whether the described procedures are used as described. He will also conduct a site visit to analyse whether the described measures and procedures are in place. This site visit will typically be conducted together with the analysis of the CM system.

9.5 Flaw remediation (ALC_FLR)

Flaw remediation ensures that flaws discovered by the TOE consumers will be tracked and corrected while the TOE is supported by the developer. While future compliance with the flaw remediation requirements cannot be determined when a TOE is evaluated, it is possible to evaluate the procedures and policies that a developer has in place to track and repair flaws, and to distribute the repairs to consumers.

Requirement (ALC_FLR.1 and above):

a) The developer shall document flaw remediation procedures addressed to TOE developers. These flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE. 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. The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws. The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether such documented procedures exist and that the flaw remediation procedures have the described properties.

Requirement (ALC_FLR.2 and above):

b) The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws. 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. The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users. The procedures for processing reported security flaws shall provide safeguards (such as analysis, testing, or a combination of the two) that any corrections to these security flaws do not introduce any new flaws.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether such documented procedures exist and that the flaw remediation procedures have the described properties.

Requirement (ALC_FLR.2 and above):

c) The developer shall document flaw remediation guidance addressed to TOE users. This guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether the flaw remediation procedures describe a means by which TOE users can report any suspected flaws in the TOE to the developer and that such means are adequately communicated to users.

Requirement (ALC_FLR.3):

d) The flaw remediation procedures shall include a procedure requiring timely responses for the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw. The issue of *timeliness* applies

to the issuance of both security flaw reports and the associated corrections. However, these need not be issued at the same time. It is recognised that flaw reports should be generated and issued as soon as an interim solution is found, even if that solution is as drastic as Turn off the TOE . Likewise, when a more permanent (and less drastic) solution is found, it should be issued without undue delay. *Automatic distribution* does not mean that human interaction with the distribution method is not permitted. In fact, the distribution method could consist entirely of manual procedures, perhaps through a closely monitored procedure with prescribed escalation upon the lack of issue of reports or corrections.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether such documented procedures exist.

Requirement (ALC_FLR.3):

e) 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. *Enabling the TOE users to register with the developer* simply means having a way for each TOE user to provide the developer with a point of contact; this point of contact is to be used to provide the TOE user with information related to security flaws that might affect that TOE user, along with any corrections to the security flaw. Registering the TOE user may be accomplished as part of the standard procedures that TOE users undergo to identify themselves to the developer, for the purposes of registering a software licence, or for obtaining update and other useful information. It should be noted that TOE users need not register; they must only be provided with a means of doing so. However, users who choose to register must be directly sent the information (or a notification of its availability).

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether such means are described.

Requirement (ALC_FLR.3):

f) The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether the specific points of contact are clearly identified.

9.6 Life-cycle definition (ALC_LCD)

Life-cycle definition establishes that the engineering practises used by a developer to produce the TOE include the considerations and activities identified in the development process and operational support requirements. Confidence in the correspondence between the requirements and the TOE is greater when security analysis and the production of evidence are done on a regular basis as an integral part of the development process and operational support activities. It is not the intent of this component to dictate any specific development process.

Requirement: EAL3 EAL4 EAL5

a) The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE. The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE. The developer shall provide life-cycle definition documentation.

The description of the life-cycle model should include:

- information on the life-cycle phases of the TOE and the boundaries between the subsequent phases;
- information on the procedures, tools and techniques used by the developer (e.g. for design, coding, testing, bug-fixing);
- overall management structure governing the application of the procedures (e.g. an identification and description of the individual responsibilities for each of the procedures required by the development and maintenance process covered by the life-cycle model);
- information which parts of the TOE are delivered by subcontractors, if subcontractors are involved.

It is not required that the model used conforms to any standard life-cycle model.

Related Evaluator Actions: The evaluator will analyse the life-cycle model definition documentation to determine whether the used life-cycle model addresses the development of the TOE as well as its maintenance. He will analyse the life-cycle model to determine whether use of the procedures, tools and techniques described by the life-cycle model will make the necessary positive contribution to the development and maintenance of the TOE. The information provided in the life-cycle model should convince the evaluator that the development and maintenance procedures adopted would minimise the likelihood of security flaws.

9.7 Tools and techniques (ALC_TAT)

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.

Requirement: EAL4 EAL5

a) The developer shall identify the development tools (programming languages and compiler, CAD systems) being used for the TOE. All development tools used for implementation shall be well-defined. The documentation of the development tools shall unambiguously define the meaning of all statements as well as all conventions and directives used in the implementation. For example, a well-defined language, compiler or CAD system may be considered to be one that conforms to a recognised standard, such as the ISO standards. A well-defined language is one that has a clear and complete description of its syntax, and a detailed description of the semantics of each construct.

Related Evaluator Actions: The evaluator will analyse the development tool documentation to determine whether all tools are addressed and whether the tools are well-defined and whether the meaning of all statements as well as all conventions and directives used in the implementation are unambiguously defined. This work will typically be performed in parallel with the evaluator's examination of the implementation representation. The key test the evaluator will apply is whether or not the documentation is sufficiently clear for the evaluator to be able to understand the implementation representation.

Requirement: EAL4 EAL5

b) The developer shall document the selected implementation-dependent options of the development tools. The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options. The documentation of **software** development tools should include definitions of implementation-dependent options that may affect the meaning of the executable code, and those that are different from the standard language as documented. Where source code is provided to the evaluator, information should also be provided on compilation and linking options used. The documentation for **hardware** design and development tools should describe the use of all options that affect the output from the tools (e.g. detailed hardware specifications, or actual hardware).

Related Evaluator Actions: The evaluator will analyse the development tool documentation to determine whether the meaning of all implementation-dependent options is unambiguously defined.

Requirement: EAL5

c) The developer shall describe the implementation standards that are being applied.

Related Evaluator Actions: The evaluator compares the provided implementation representation with the description of the applied implementation standards and verifies their use. He might supplement documentary evidence by visiting the development environment. A visit to the development environment will allow the evaluator to:

- observe the application of defined standards;
- examine documentary evidence of application of procedures describing the use of defined standards;
- interview development staff to check awareness of the application of defined standards and procedures.

10 Class ATE: Tests

The goal of this activity is to determine whether the TOE behaves as described in the ST and as specified in the evaluation evidence described in the ADV class. This determination is achieved through some combination of the developer's own functional testing of the TSF and independent testing of the TSF by the evaluator.

Note: The following subchapters are ordered in the sequence ATE_FUN, ATE_COV, ATE_DPT, and ATE_IND.

10.1 Functional tests (ATE_FUN)

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.

Requirement: EAL2 EAL3 EAL4 EAL5

a) The developer shall test the TSF and provide test documentation. The test documentation shall include a test plan which identifies the TOE to be tested and the tests to be performed. These descriptions shall include the description of any test pre-requisites which are necessary to establish the required initial conditions. The test description shall also contain the test procedure description which specifies how the test is executed, which inputs have to be performed, how the expected results should be verified, and which cleanup processing has to be performed. The test description should be clear and sufficiently detailed to ensure that the test is reproducible. The test plan shall provide information about the test configuration being used: both on the configuration of the TOE and on any test equipment being used. This information should be detailed enough to ensure that the test configuration is reproducible.

Background: Test pre-requisites are necessary to establish the required initial conditions for the test. They may be expressed in terms of parameters that must be set or in terms of test ordering in cases where the completion of one test establishes the necessary pre-requisites for another test.

Related Evaluator Actions: For the tests performed by the developer, the evaluator determines from the documentation provided, whether the tests are repeatable and whether the pre-requisites are complete and appropriate in that they will not bias the observed test results towards the expected test results. Regarding consistence of the test configuration the evaluator will also consider the security objectives for the operational environment described in the ST that may apply to the test environment. There may be some objectives for the operational environment that do not apply to the test environment. For example, an objective about user clearances may not apply; however, an objective about a single point of connection to a network would apply.

Examples:

Establishing initial conditions

- User accounts need to exist before they can be deleted.
- An ordering dependency would be where one test generates a file of data to be used as input for another test.

Requirement: EAL3 EAL4 EAL5

b) The test documentation shall include descriptions of the behaviour of TSF subsystems and of their interactions.

Background: This requirement derives from a requirement of an evaluator action of family ATE_DPT.

Related Evaluator Actions: The evaluator will analyse the test documentation to determine whether it describes the behaviour of TSF subsystems and of their interactions.

Requirement: EAL4

c) The test documentation shall include descriptions of the behaviour of the interfaces of TSF-enforcing modules.

Background: This requirement derives from a requirement of an evaluator action of family ATE_DPT.

Related Evaluator Actions: The evaluator will analyse the test documentation to determine whether it describes the behaviour of the interfaces of TSF-enforcing modules.

Requirement: EAL5

d) The test documentation shall include descriptions of the behaviour of the interfaces of all modules.

Background: This requirement derives from a requirement of an evaluator action of family ATE_DPT.

Related Evaluator Actions: The evaluator will analyse the test documentation to determine whether it describes the behaviour of the interfaces of all modules.

Requirement: EAL2 EAL3 EAL4 EAL5

e) The test documentation shall include the expected test results which show the anticipated outputs from a successful execution of the tests.

Background: The expected test results are needed to determine whether or not a test has been successfully performed. The description of the expected test results are sufficient if they are unambiguous and consistent with the expected behaviour given the testing approach.

Related Evaluator Actions: The evaluator will analyse the test documentation to determine whether it contains sufficient information about the anticipated outputs from a successful execution of the tests. The evaluator will determine whether the test steps and expected results are consistent with the descriptions of the TSFI in the functional specification.

Requirement: EAL2 EAL3 EAL4 EAL5

f) The test documentation shall include the information about actual test results which demonstrates that the actual test results are consistent with the expected test results. If a direct comparison of actual results cannot be made until some data reduction or synthesis has been first performed, the developer's test documentation should describe the process to reduce or synthesise the actual data. Since the TOE might be subject to change in the

course of the development process, the test documentation should also reference the actually tested version of the TOE.

Background: A comparison of the actual and expected test results will reveal any inconsistencies between the results.

Related Evaluator Actions: The evaluator will analyse the test documentation to determine whether the actual test results are consistent with the expected test results.

10.2 Coverage (ATE_COV)

This family assures that the TSF has been tested against its functional specification. This is achieved through an examination of developer evidence of correspondence.

Requirement: EAL2

a) The developer shall provide evidence of the test coverage. It shall show the correspondence between the tests in the test documentation and the TSFIs and their characteristics in the functional specification. Correspondence may take the form of a table or matrix. The coverage evidence required for this component will reveal the extent of coverage, rather than to show complete coverage.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it shows the correspondence between the tests in the test documentation and the TSFIs in the functional specification.

Requirement: EAL3 EAL4 EAL5

b) The developer shall provide an analysis of the test coverage. It shall show the correspondence between the tests in the test documentation and the TSFIs and the characteristics of the TSFI behaviour described in the functional specification. Correspondence may take the form of a table or matrix. The analysis of the test coverage shall also demonstrate that all TSFIs that are described in the functional specification are present in the test coverage analysis and mapped to tests in order for completeness to be claimed, although exhaustive specification testing of interfaces is not required.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it demonstrates the correspondence between the tests in the test documentation and the TSFIs in the functional specification and whether it demonstrates that all TSFIs in the functional specification have been tested.

10.3 Depth (ATE_DPT)

The components in this family deal with the level of detail to which the TSF is tested by the developer.

Requirement: EAL3

a) The developer shall provide the analysis of the depth of testing. The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the TSF subsystems and the characteristics of the behaviour at their interfaces described in the TOE design. A table or matrix may be sufficient to show test correspondence. The identification of the tests and the behaviour/interaction presented in the depth-of coverage analysis has to be unambiguous.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it demonstrates the requested correspondence.

Requirement: EAL4 EAL5

b) The developer shall provide the analysis of the depth of testing. The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation

and the TSF subsystems and modules and the characteristics of the behaviour at their interfaces described in the TOE design. A table or matrix may be sufficient to show test correspondence. The identification of the tests and the behaviour/interaction presented in the depth-of coverage analysis has to be unambiguous.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it demonstrates the requested correspondence.

Requirement: EAL3 EAL4 EAL5

c) The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have been tested. All descriptions of TSF subsystem behaviour and of interactions among TSF subsystems that are provided in the TOE design have to be tested. This means that each characteristic of the behaviour of the TSF subsystems at its interfaces explicitly described in the TOE design should have tests and expected results to verify that behaviour. The behaviour of TSF subsystems may be tested directly from those interfaces. Otherwise, the behaviour of those subsystems is tested from the TSFI interfaces. Or a combination of the two may be employed.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it demonstrates that all subsystem interfaces have been tested with sufficient rigour.

Requirement: EAL4

d) The analysis of the depth of testing shall demonstrate that the SFR-enforcing modules, i.e. all interfaces of SFR-enforcing modules in the TOE design, have been tested. This means that each characteristic of the behaviour of the SFR-enforcing modules at its interfaces explicitly described in the TOE design should have tests and expected results to verify that behaviour. Testing of an interface may be performed directly at that interface, or at the external interfaces, or a combination of both. A simple table or matrix may be sufficient to show test correspondence. The identification of the tests and the behaviour/interaction presented in the depth-of coverage analysis has to be unambiguous.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it demonstrates that all interfaces of SFR-enforcing modules have been tested with sufficient rigour.

Requirement: EAL5

e) The analysis of the depth of testing shall demonstrate that all modules, i.e. all interfaces of TSF modules that are provided in the TOE design, have to be tested. This means that each characteristic of the behaviour of the all modules at its interfaces explicitly described in the TOE design should have tests and expected results to verify that behaviour. Testing of an interface may be performed directly at that interface, or at the external interfaces, or a combination of both. A simple table or matrix may be sufficient to show test correspondence. The identification of the tests and the behaviour/interaction presented in the depth-of coverage analysis has to be unambiguous.

Related Evaluator Actions: The evaluator will analyse the provided documentation to determine whether it demonstrates that all module interfaces have been tested with sufficient rigour.

10.4 Independent testing (ATE_IND)

This family deals with the degree to which there is independent functional testing of the TSF.

Requirement: EAL1 EAL2 EAL3 EAL4 EAL5

a) The developer shall provide the TOE for testing. The TOE shall be suitable for testing.

Background: To be able to test the TOE the evaluator will need access to tools and other means or needs to make use of publicly available tools or tools from other sources.

Related Evaluator Actions: The evaluator will use the TOE and any test resources provided for its own testing.

Requirement: EAL2 EAL3 EAL4 EAL5

b) The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF. The resource set may include laboratory access and special test equipment, among others. Resources that are not identical to those used by the developer need to be equivalent in terms of any impact they may have on test results.

Background: The evaluator is put into a position to repeat any developer tests.

Related Evaluator Actions: The evaluator will use the TOE and the test resources provided for its own testing.

11 Class AVA: Vulnerability assessment

This class addresses the possibility of exploitable vulnerabilities introduced in the development or the operation of the TOE. It 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) could allow attackers to violate the security functional requirements.

This class contains only family Vulnerability analysis (AVA_VAN). It deals with the threat 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 authorised capabilities of other users.

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.

In contrast to previous versions of the CC, CC Version 3.1 does **not** mandate that the developer performs and documents a vulnerability analysis. The requirements of this class do all address analysis, documentation and testing activities which have to be performed by the evaluator.

However, the description of architectural soundness which has to be provided for ADV_ARC 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.

Appendix A Sample Document Structure for Class ADV

This appendix proposes structure elements for the documentation to be provided for assurance class “Development (ADV)”. Headings in bold font (like **Identification of TOE subsystems**) are directly derived from the CC requirements and are thus strongly recommended. Headings in italic bold font (like ***Mapping between Implementation representation and TOE Design description***) are suggestions provided by the author.

1 Family Security Architecture (ADV_ARC)

The following structure is proposed for the Security architecture:

Security Architecture EAL2 EAL3 EAL4 EAL5

[The mapping which has to be provided between the TOE Design description and the implementation representation can be achieved by a simple table.]

Self-protection
[This section has to demonstrate that the TSF protects itself from interference and tampering.

Examples:
- Self-protection can be achieved by processor-based separation mechanisms such as privilege levels or rings.
- Self-protection can also be achieved by software-based means. The security architecture description might contain information pertaining to coding conventions for parameter checking that would prevent TSF compromises (e.g. buffer overflows), and information on stack management for call and return operations.
- Self-protection can also be achieved by physical and logical restrictions on access to the TOE.
- The description might include protection placed upon the executable images of the TSF, and protection placed on objects (e.g. files used by the TSF).]

Contributions of TSF
Contributions of IT-environment
Contributions of non-IT-environment

Domain isolation
[This section has to describe the security domains maintained by the TSF.

Examples:
- A secure operating system provides a security domain by supplying a set of resources (address space, per-process environment variables) for use by processes with limited access rights and security properties.
- A packet-filter firewall is an example of a TOE which does not provide a security domain. Users on the LAN or WAN do not interact with the TOE, so there is no need for security domains; there are only data structures maintained by the TSF to keep the users' packets separated.]

Contributions of TSF
Contributions of IT-environment
Contributions of non-IT-environment

Non-bypassibility
[This section has to demonstrate that the TSF prevents bypass of the SFR-enforcing functionality. This means that the TSF protection mechanisms are always invoked to prevent that a TSFI can be used to access protected data or resources in an unauthorised way.

Examples:
Suppose the TSF provides file protection. Further suppose that although the "traditional" system call TSFIs for open, read, and write invoke the file protection mechanism described in the TOE design, there exists a TSFI that allows access to a batch job facility (creating batch jobs, deleting jobs, modifying unprocessed jobs). The evaluator should be able to determine from the vendor-provided description that this TSFI invokes the same protection mechanisms as do the "traditional" interfaces. This could be done, for example, by referencing the appropriate sections of the TOE design that discuss *how* the batch job facility TSFI achieves its security objectives.]

Contributions of TSF
Contributions of IT-environment

Contributions of non-IT-environment

Secure initialisation process

[This section has to demonstrate that the TSF initialisation process preserves security. This portion of the security architecture description should list the system initialisation components and describe the processing that occurs in transitioning from the “down” state to the initial secure stage (i.e. when all parts of the TSF are operational) when power-on or a reset is applied.

Examples:

- The TSF can preserve security by ensuring that the initialisation components are not accessible to untrusted entities after the secure stage is reached.
- The TSF can preserve security by ensuring that the interfaces provided by the initialisation components to untrusted users can not be used to tamper with the TSF.]

Contributions of TSF

Contributions of IT-environment

Contributions of non-IT-environment

2 Family Functional specification (ADV_FSP)

The following structure is a worked out example for a Functional Specification:

The Example firewall is used between an internal network and an external network. It verifies the source address of data received (to ensure that external data is not attempting to masquerade as originating from the internal data); if it detects any such attempts, it saves the offending attempt to the audit log. The administrator connects to the firewall by establishing a telnet connection to the firewall from the internal network. Administrator actions consist of authenticating, changing passwords, reviewing the audit log, and setting or changing the addresses of the internal and external networks.

The firewall presents the following interfaces to the internal network:

- IP datagrams
- Administrator Commands

and the following interfaces to the external network:

- IP datagrams

Interfaces Descriptions: IP Datagrams

The datagrams are in the format specified by RFC 791:

- Purpose - to transmit blocks of data (“datagrams”) from source hosts to destination hosts identified by fixed length addresses; also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through small-packet networks.
- Method of Use - they arrive from the lower-level (e.g. data link) protocol.
- Parameters - the following fields of the IP datagram header: source address, destination address, don't-fragment flag.
- Parameter description - [As defined by RFC 791, section 3.1 (“Internet Header Format”)]
- Actions - Transmits datagrams that are not masquerading; fragments large datagrams if necessary; reassembles fragments into datagrams.
- Error messages - (none). No reliability guaranteed (reliability to be provided by upper-level protocols) Undeliverable datagrams (e.g. must be fragmented for transmission, but don't-fragment flag is set) dropped.

Interfaces Descriptions: Administrator Commands

The administrator commands provide a means for the administrator to interact with the firewall. These commands and responses ride atop a telnet (RFC 854) connection established from any host on the internal network. Available commands are:

- Passwd
  - Purpose - sets administrator password
  - Method of Use - Passwd <*password*>
  - Parameters - password
  - Parameter description - value of new password

- Actions - changes password to new value supplied. There are no restrictions.
- Error messages - none.

- Readaudit
  - Purpose - presents the audit log to the administrator
  - Method of Use - Readaudit
  - Parameters - none
  - Parameter description - none
  - Actions - provides the text of the audit log
  - Error messages - none.

- Setintaddr
  - Purpose - sets the address of the internal address.
  - Method of Use - Setintaddr <*address*>
  - Parameters - address
  - Parameter description - first three fields of an IP address (as defined in RFC 791). For example: 123.123.123.
  - Actions - changes the internal value of the variable defining the internal network, the value of which is used to judge attempted masquerades.
  - Error messages - “address in use”: indicates the identified internal network is the same as the external network.

- Setextaddr
  - Purpose - sets the address of the internal address
  - Method of Use - Setextaddr <*address*>
  - Parameters - address
  - Parameter description - first three fields of an IP address (as defined in RFC 791). For example: 123.123.123.
  - Actions - changes the internal value of the variable defining the external network.
  - Error messages - “address in use”: indicates the identified external network is the same as the internal network.

3 Family Implementation Representation (ADV_IMP)

The following structure is proposed for the implementation representation:

Mapping between Implementation representation and TOE Design description EAL4 EAL5

[The mapping which has to be provided between the TOE Design description and the implementation representation can be achieved by a simple table.]

4 Family TSF Internals (ADV_INT)

The following structure is proposed for the TSF internals documentation:

TSF internals justification EAL5

[The section has to explain the characteristics used to judge the meaning of “well-structured”.

For example, procedural software that executes linearly is traditionally viewed as well-structured if it adheres to software engineering programming practises, such as those defined in IEEE Std 610.12-1990. For example, it would identify the criteria for the procedural software portions of the TSF:

- the process used for modular decomposition
- coding standards used in the development of the implementation
- a description of the maximum acceptable level of intermodule coupling exhibited by the TSF
- a description of the minimum acceptable level of cohesion exhibited the modules of the TSF

For other types of technologies used in the TOE - such as non-procedural software (e.g. object-oriented programming), widespread commodity hardware (e.g. PC microprocessors), and special-purpose hardware (e.g. smart-card processors) - the evaluation authority should be consulted for determining the adequacy of criteria for being “well-structured”.]

TSF internals description EAL5

[The section has to demonstrate that the entire TSF is well-structured.

For example, it would explain how the procedural software portions of the TSF meet the following:

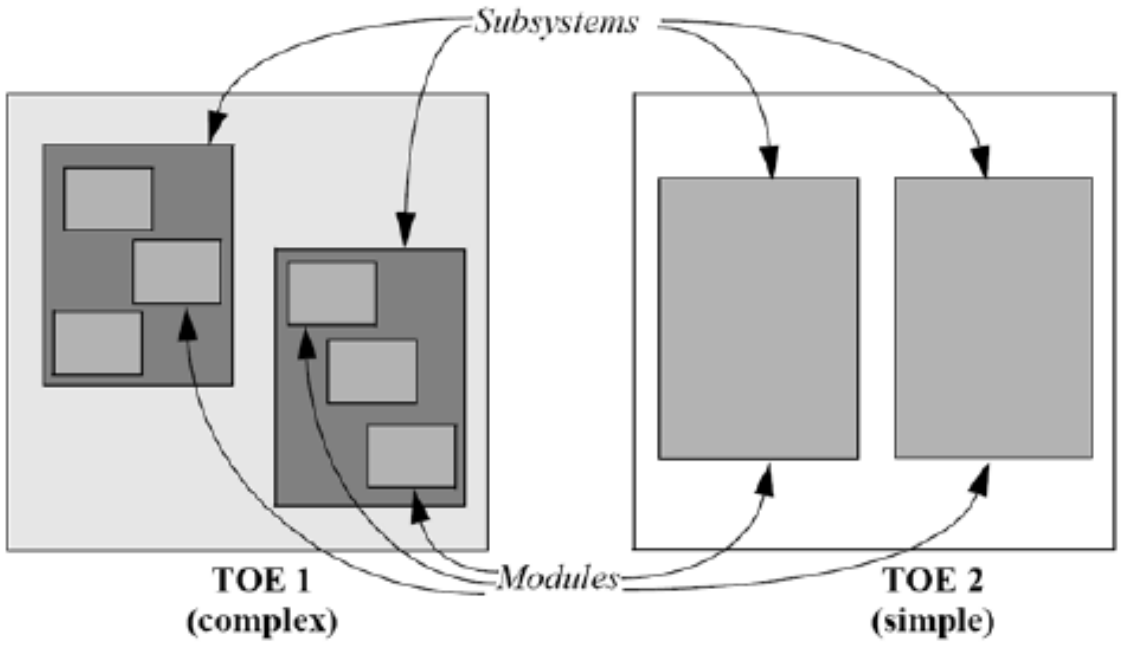
- that there is a one-to-one correspondence between the modules identified in the TSF and the modules described in the TOE design
- how the TSF design is a reflection of the modular decomposition process
- a justification for all instances where the coding standards were not used or met
- a justification for any coupling or cohesion outside the acceptable bounds]

5 Family Security policy modelling (ADV_SPM)

There are no requirements of this family for EAL1 to EAL5.

6 Family TOE design (ADV_TDS)

Depending on the complexity of the TSF, the design may be described in terms of subsystems *and* modules (where subsystems are at a higher level of abstraction than modules); or it may just be described in terms of one level of abstraction (e.g., *subsystems* at lower assurance levels, *modules* at higher levels). In cases where a lower level of abstraction (modules) is presented, requirements levied on higher-level abstractions (subsystems) are essentially met by default. The following figure illustrates these principles:



The following structure is proposed for the TOE design description of a complex TOE:

TOE and TSF Subsystems EAL2 EAL3 EAL4 EAL5

Identification of TOE subsystems
[The subsystems are identified with a simple list of what they are.]

TSF subsystems

Identification of TSF subsystems
[This section has to identify those TOE subsystems which make up the TSF. This can be performed within an own section or by indicating the TSF subsystems in the previous section.]

Behaviour of TSF Subsystems
[A subsystem's behaviour is what it does. A behaviour summary of a subsystem is an overview of the actions it performs.]

EAL2 [This section has to describe the behaviour of each SFR-supporting or SFR-non-interfering TSF subsystem in detail sufficient to determine that it is not SFR-enforcing. It has to summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.]

EAL3 [This section has to describe the behaviour of each SFR-non-interfering TSF subsystem in detail sufficient to determine that it is SFR-non-interfering. It has to describe the SFR-enforcing behaviour of the SFR-enforcing subsystems. SFR-enforcing behaviour refers to how a subsystem provides the functionality that implements an SFR. While not at the level of an algorithmic description, a detailed description of behaviour typically discusses how the functionality is provided in terms of what key data and data structures are, what control relationships exist within a subsystem, and how these elements work together to provide the SFR-enforcing behaviour. It has to summarise the non-SFR-enforcing behaviour of the SFR-enforcing subsystems.]

EAL4 EAL5 [This section has to describe each subsystem of the TSF. The subsystem-level description has to contain a description of how the security functional requirements are achieved in the design, but at a level of abstraction above the modular description.]

Interactions among TSF Subsystems

EAL2 [This section has to describe the interactions among SFR-enforcing subsystems of the TSF, and between SFR-enforcing subsystems of the TSF and other subsystems of the TSF. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a subsystem to a routine in a different subsystem; global variables; hardware signals (e.g., interrupts) from a hardware subsystem to an interrupt-handling subsystem), but the data elements identified for a particular subsystem that are going to be used by another subsystem should be covered in this discussion. Any control relationships between subsystems (e.g., a subsystem responsible for configuring a rule base for a firewall system and the subsystem that actually implements these rules) should also be described.]

EAL3 EAL4 EAL5 [This section has to describe the interactions among all subsystems of the TSF. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a subsystem to a routine in a different subsystem; global variables; hardware signals (e.g., interrupts) from a hardware subsystem to an interrupt-handling subsystem), but the data elements identified for a particular subsystem that are going to be used by another subsystem should be covered in this discussion. Any control relationships between subsystems (e.g., a subsystem responsible for configuring a rule base for a firewall system and the subsystem that actually implements these rules) should also be described.]

TSF Modules

Description of TSF Modules

EAL4 EAL5 [This section has to describe the structure of the TOE in terms of modules. It has to describe each SFR-enforcing module in terms of its purpose, SFR-related interfaces, return values from those interfaces, and called interfaces to other modules. The description of the interfaces presented by each SFR-enforcing module has to contain an accurate and complete description of the SFR-related parameters, the invocation conventions for each interface, and any values returned directly by the interface. It has to describe each SFR-non-interfering module in terms of its purpose and interaction with other modules.]

EAL4 [This section has to describe each SFR-supporting module in terms of its purpose and interaction with other modules.]

EAL5 [This section has to designate each TSF module as either SFR-enforcing, SFR-supporting, or SFR-non-interfering. It has to describe each SFR-supporting module in terms of its purpose, SFR-related interfaces, return values from those interfaces, and called interfaces to other modules.]

Mapping from TSF Subsystems to TSF Modules EAL4 EAL5

[This section has to provide a mapping from the subsystems of the TSF to the modules of the TSF. This can be achieved by a simple table.]

Mapping from TSFI to Subsystems EAL2 EAL3

[This section has to provide a mapping from the TSFI of the functional specification to the subsystems of the TSF. The mapping shall demonstrate that all behaviour described in the TOE design is mapped to the TSFIs that invoke it. This can be achieved by a simple table.]

Mapping from TSFI to Modules EAL4 EAL5

[This section has to provide a mapping from the TSFI of the functional specification to the modules of the TSF. The mapping shall demonstrate that all behaviour described in the TOE design is mapped to the TSFIs that invoke it. This can be achieved by a simple table.]

Appendix B Sample Document Structure for Class AGD

This appendix proposes structure elements for the documentation to be provided for assurance class “Guidance Documents (AGD)”. Headings in bold font (like **Acceptance Procedures**) are directly derived from the CC requirements and are thus strongly recommended. Headings in italic bold font (like ***Identification of user roles***) are suggestions provided by the author.

All user guidance can be contained in a single document. However, in many cases it may be appropriate that guidance is provided in separate documents for preparation and operation of the TOE, or even separate for different user roles as end-users, administrators, application programmers using software or hardware interfaces, etc. To be as general as possible the proposed structure follows the separation into “operational user guidance” (addressing issues which are relevant for the ongoing use of the TOE) and “preparative procedures” (addressing issues which are relevant for the acceptance of the TOE after delivery and for the installation of the TOE).

1 Operational user guidance (AGD_OPE)

The following structure is proposed for the portion of the user guidance addressing operational use of the TOE*:* EAL1 EAL2 EAL3 EAL4 EAL5

Generally many parties within an organisation are involved in the elaboration of the guidance documentation and its layout. Nevertheless this guidance strongly recommends a specific structure of the guidance to facilitate the proper coverage of all CC requirements. This includes the requirement that the user guidance shall be clear and reasonable.

The proposed structure distinguishes two parts. The first part (“General Description”) provides an overview about the information which is not role specific. This includes an initial identification and characterisation of the various roles supported by the TOE. The second part (“User Roles Specific Description”) specifies in detail the information required by the CC. Not all aspects might be applicable to all TOEs. It is strongly recommended to provide a commented “not applicable” statement in such situations.

General Description
[see note above]

Identification and characterisation of user roles
[Since the TOE will support several types of users (end-users, application programmers, administrators, auditors), this initial section shall identify the types and about the functions and interfaces accessible to this user and the privileges granted. Various users will typically have different TOE related roles, i.e. tasks and responsibilities. There will typically be end-users who are aiming at using the end-user oriented functions and services of the TOE. There will typically also be privileged administrators who have to configure, administrator the TOE and monitor its operation. There can also be revisors who can review actions of and settings for other user groups. Depending on the type of TOE there can also be programmers making use of services provided by a software library or a hardware device. This section should also refer to the portions of the guidance documentation which are relevant for the various users roles.]

Modes of operations
[This section shall identify all possible modes of operation of the TOE (including operation following failure or operational error). Unix/Linux type operating systems for example distinguish between single-user-mode and multi-user-mode. Some operating systems support beside normal operating mode a maintenance operating mode in which specific diagnostic utilities are enabled and in which security functions might be confined or even completely disabled.
For each operating mode such consequences and the resulting implications for maintaining secure operation should be described.]

Setting up a secure operational environment
[This section shall describe the security measures which have to be implemented for a secure operational environment. The security measures described should include all relevant external procedural, physical, personnel and connectivity measures. The information in this section has to be consistent with the security objectives for the operational environment as specified in the Security Target.]

User Role Specific Description
[see note above – The following section shall be specific to **one** user role.]

Description of security functions
[This section shall describe the available security functions of the TOE which are visible at user interfaces, and identify the interfaces by which the security functions can be invoked. It shall describe the purpose, behaviour, and interrelationships of the security functions The description has to be in accordance with the TOE summary specification of the Security Target.]

Description of privileges
[This section shall describe the privileges associated with this security function. Depending on their role users can access the security functions provided by the TOE in different ways and with different privileges. The TOE might distinguish various administrative roles including administrators with the ability to fully control all configuration parameters (such as password building rules mandatory for all users, while end-users on the other side can typically change only their own passwords) and auditors who are only authorised to inspect the configuration parameters. The TOE might provide the capability to temporarily or permanently turn off security functions (like disabling logging or weakening authentication for specific methods of access). This might be acceptable in specific environment or in special situations. The privilege to perform such functions obviously needs to be restricted and clearly documented in the user guidance. The TOE might also provide the capability to perform session traces which contain all information entered by users including passwords (like telnet sessions). This might be acceptable in special situations (like troubleshooting). The privilege to activate such functions obviously needs to be restricted and clearly documented in the operational guidance. A firewall product might be intended to be administered and operated via a dedicated management station. The use of the firewall console for administration activities will then be discouraged. In special situations – like all users of the graphical user interface are locked out, or the graphical user interface is not operational, or hardware problems have to be debugged and resolved – console access will be necessary. This has to be performed under very restrictive boundary conditions. The administrator and the organisation's security officer have to be aware that some security functionality (like logging) might be limited in such situations. The operational guidance has also to clearly describe how such information is removed. In specific products log files can only be deleted by a privileged

revisor. The behaviour of other security functions (like storage reuse) will not be modifiable by any user.]

Warnings

[This section shall identify those user-accessible functions and privileges that must be controlled in a secure processing environment and the types of commands required for them. The reasons for such commands should be explained. This section shall contain warnings regarding the use of these functions and privileges. Warnings should address expected effects, possible side effects, and possible interactions with other functions and privileges. Example 1: The TOE might provide the capability to perform session traces which contain all information entered by users including passwords (like telnet sessions). This might be acceptable in special situations (like troubleshooting). The privilege to activate such functions obviously needs to be restricted and clearly documented in the user guidance. The user guidance has also to clearly describe how such information is removed. Example 2: A firewall product might be intended to be used via a dedicated management station. The use of the firewall console for administration activities will then be discouraged. In special situations – like all users of the graphical user interface are locked out, or the graphical user interface is not operational, or hardware problems have to be debugged and resolved – console access will be necessary. This has to be performed under very restrictive boundary conditions. The administrator and the organisation's security officer have to be aware that some security functionality (like logging) might be limited in such situations.]

Description of interfaces

[This section shall describe the corresponding interfaces by describing the purpose, behaviour, and interrelationships of the interfaces]

Method of invocation

[This section shall describe the method(s) by which the interface is invoked (e.g. command-line, programming-language system call, menu selection, command button).]

Specification of interfaces

[This section shall describe the parameters to be set by the user, their particular purposes, valid and default values, and secure and insecure use settings of such parameters, both individually or in combination. This section should also describe the immediate TOE response, message, or code returned. If the TOE provides command-line access it is adequate – at least in the Unix/Linux system environment - to provide detailed usage information in the form of manpages. If the TOE provides a graphical user interface it is adequate to visualise the various windows/forms and to explain all fields, admissible values, default values, and secure and insecure use values. If the TOE provides an application program interface it is adequate to specify the interface in formal notation and supplementing this specification by an information description of the provided methods/functions, signatures, admissible values, default values, and secure and insecure use values.]

Events

[This section shall describe each type of security-relevant event relative to the user-accessible functions that need to be performed by users carrying this role. All types of security-relevant events have to be specified in such detail, such that each user knows what events may occur and what action (if any) he may have to take in order to maintain security. Security-relevant events that may occur during operation of the TOE have to be adequately defined to allow user intervention to maintain secure operation. Example events to be considered are audit trail overflow, system crash, specific audit trail entries indicating

penetration attempts, and specific audit trail entries indicating system misfunctions. The user with a corresponding role has be informed about the nature of the event and about the actions to be taken. The range of actions to be taken includes clearing specific files, collection information to report the problem to the developer, disconnecting the TOE from all or specific networks, reinstall the TOE, power off the TOE. Example events to be considered are also when a user account is to be removed when the user leaves the organisation or is assigned different responsibilities.]

Recommendation for secure usage of the TOE

[This section shall provide role specific advice regarding effective use of the security functions of the TOE (e.g. reviewing password composition practises, suggested frequency of user file backups, discussion on the effects of changing user access privileges).]

2 Preparative procedures (AGD_PRE)

The following structure is proposed for the portion of the user guidance addressing the preparative procedures: EAL1 EAL2 EAL3 EAL4 EAL5

Acceptance Procedures
[This section should describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the documented delivery procedures. The acceptance procedures should reflect the steps the user has to perform in order to accept the delivered TOE that are implied by the delivery procedures. The acceptance procedures should include as a minimum, that the user has to check that all parts of the TOE as indicated in the ST have been delivered in the correct version. Appended is an example which shows which information is adequate to check the delivery. The acceptance procedures should also describe whether a user can verify the integrity and/or authenticity of the delivered TOE or detect a non-authorised delivery and should describe any corresponding activities. When designing such procedures it should be considered that the preparative procedures provided to the user could itself be subject to unauthorised modification or unauthorised delivery. If the delivery procedures state that no user acceptance procedure takes place, such statement shall be re-stated in the preparative procedures together with an adeaquate rationale.]

Installation and Startup Procedures
[This section addresses the secure preparation of the operational environment of the TOE and the secure installation of the TOE.]

Preparation of the operational environment
[This section should describe all the steps necessary for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.]

Installation of the TOE
[This section should describe all the steps necessary for secure installation of the TOE. The installation procedures should provide detailed information about the following, if applicable: minimum system requirements for secure installation; changing the installation specific security characteristics of entities under the control of the TOE (for example initial parameter settings, changing of developer provided passwords); handling exceptions and problems.]

Startup of the TOE
[This section should describe all aspects relevant for the secure startup of the TOE. This might include interactions to enter specific operating modes.]

Example for Identification of the TOE (extract from Security Target)

The Target of Evaluation (TOE) is called ChipCompany Security Controller SC2006X1, Version 02. The following table outlines the TOE deliverables:

No	Type	Identifier	Release	Form of Delivery
1	HW	Security Controller SC2006X1	02	Wafer or package module
2	SW	IC Dedicated Test Software Test ROM software	1.5	Included in SC2006X1 Test ROM

No	Type	Identifier	Release	Form of Delivery
3	SW	RNG software	1.42	Software module (this is implemented in the Embedded Software by the User)
5	DOC	Hardware Manual	1.0	Hardcopy
6	DOC	User Guidance	3.40	Hardcopy

The TOE is identified by Security Controller SC2006X1, Version 02 (stored as a version number in the EEPROM) produced in Cologne (indicated by IC manufacturers ID number 4612 for Cologne).

Appendix C Sample Document Structure for Class ALC

This appendix proposes structure elements for the documentation to be provided for assurance class “Life-Cycle Support (ALC)”. Headings in bold font (like **CM usage documentation**) are directly derived from the CC requirements and are thus strongly recommended. Headings in italic bold font (like ***Description of roles and responsibilities***) are suggestions provided by the author.

1 Family CM capabilities (ALC_CMC)

The following table gives an overview about the documentation to be provided as CM documentation:

Documentation Requirements Overview	**EAL1**	**EAL2**	**EAL3**	**EAL4**	**EAL5**
CM Usage Documentation		**X**	**X**	**X**	**X**
Description of method of identifying configuration items		**X**	**X**	**X**	**X**
CM Plan			**X**	**X**	**X**
CM usage guide			X	X	X
Description of the output of the CM usage			X	X	X
Description of TOE access control measures			X	X	X
Description of automated TOE access control measures				X	X
Description of automated TOE generation procedures				X	X
Description of acceptance procedures				X	X
CM output documentation	**X**	**X**	**X**	**X**	**X**
Configuration list, *see chapter 9.2*	X	X	X	X	X
CM system records			X	X	X

Table 2: Documentation requirements overview (CM documentation)

The following structure is proposed for the CM documentation:

CM System Overview EAL2 EAL3 EAL4 EAL5

[CM system is the overall term for the set of procedures and tools including their documentation used by a developer to develop and maintain configurations of his products during their life-cycles. This section should give an overview about the CM system used or planned by the developer. It should summarise the technical and procedural means used by the developer to control and uniquely identify configuration items. If applicable it should also address the level of automation provided.]

CM Usage Documentation EAL2 EAL3 EAL4 EAL5

[This part of the CM documentation will typically base on or incorporate documentation and other information from the development environment. Its main purpose is to allow the evaluator to understand how the CM system is used.]

Technical information

Description of method for identifying configuration items

[An accurate and detailed description of the method(s) used for uniquely identifying the various types of configuration items including the TOE itself is requested. The evaluator will analyse the configuration list to determine whether the described method(s) is/are used by the developer.]

Identification of standard tools used

[The tools which are used to support the CM system have to be identified (name, version) and described (reference documents, URLs). The developer is reminded that URLs may be subject to change. He is thus encouraged to provide downloaded information in order to enforce the uniqueness of the reference documentation.]

Specification of extensions to and adaptations of standard tools

[Developers might customise tools or supplement them by own development activities. This section should be used to document such activities.]

Other supporting means

[The less a CM system makes use of manual or automated tools the more it has to make use of other means (typically text editors, or even “paper and pencil”). This section should specify such technical means and what they are used for.]

Procedural information

Description of roles and responsibilities

[The specification of roles and responsibilities provides the backbone of the description of the procedures. Number of roles and characteristics of the responsibilities will be heavily dominated by the size and structure of the development team and the complexity of the CM system.]

Description of procedures

CM Plan EAL3 EAL4 EAL5

[The CM plan is a major part of the CM documentation and describes how the CM system is used for the TOE. The main objective of issuing a CM plan is that each staff can see clearly what he has to do. The CM plan also defines and describes the output of the CM system.]

CM Usage Guide

[The “usage guide” comprises the documentation needed by the evaluator to understand and analyse how the CM system is used (denoted here as “CM Usage Documentation Plan”) as well as the documentation oriented towards the user of the CM system (denoted here as “Developer CM Guidance”). Since the “CM usage Documentation is already required for EAL2 it has been placed outside of the CM plan.]

Developer CM Guidance

[This section addresses the information provided to the individual CM system users (i.e. developers, testers, quality assurance). This section should also address the avoidance of concurrency problems as a result of simultaneous changes to configuration items. The information expected comprises technical information

(handbooks, guidance, reference documents) as well as the specification of roles, responsibilities, forms and procedures.]

Technical information

Procedural information

Specification of roles and responsibilities

Description of CM instances
[This section describes how CM instances (e.g. change control boards, interface control working groups) are introduced and staffed.]

Specification of procedures
[This section describes procedures related to change management and other CM related issued not addressed in other sections.]

Description of TOE access control measures
[This section addresses technical and procedural means to assure that only authorised changes to the configuration items maintained by the CM system are possible.]

Technical information

Tool related information
[This section is dedicated to the information regarding access control functions provided by the CM tools (standards + extensions/customisation) themselves.]

Platform related information
[This section is dedicated to the information regarding access control functions provided by the technical environment of the CM tools. Such functions are typically provided by the underlying operating system. The description should also include the discussion of networking aspects. To avoid redundancies references to information provided for ALC_DVS are encouraged.]

Description of automated TOE access control measures EAL4 EAL5
[This section contains the description of the automated access control measures if this description is not contained in above sections.]

Procedural information

Description of automated TOE generation procedures EAL4 EAL5
[This section addresses the increased requirements regarding the degree of CM automation. A description is needed how the used CM tools support the automated generation/building of the TOE. The make tool and its deviates for example provide such functionality.]

Description of acceptance procedures EAL4 EAL5
[This section addresses the acceptance procedures which are used for the TOE and its configuration items. The information provided must describe the roles or individuals involved and their responsibilities. The evaluator must be convinced that only configuration items that have passed an adequate and appropriate review and are of adeaquate quality are incorporated in the TOE. Note, that for EAL4 and EAL5 it is not required that the person accepting a configuration item is different from the person who has developed it.]

Description of the output of the CM usage EAL3 EAL4 EAL5

[This section shall describe the output of the usage of the CM system. The output consists at least of the configuration list and of CM system records. The evaluator uses the information in this section to verify whether the information provided as CM output covers the information identified in this section.]

Configuration list

CM Records

Description of CM output (logs) related to changes to configuration items

[This headline and the following headlines provide only examples of CM records which typically occur. The CM system actually used and its characteristics might suggest deviations from that structure.]

Description of CM output (logs) related to introduction of new configuration items

Description of CM output (logs) related to invocation of CM tools

Description of paper forms

CM output documentation EAL1 EAL2 EAL3 EAL4 EAL5

[This part of the documentation addresses the actual output of the usage of the CM system.]

[For EAL3 and above: The CM output documentation will be scattered over many documents/files and will also undergo changes in the evaluation process. It is therefore strongly recommended to treat this part of the documentation separate from the “CM System Overview” and the “CM Plan”.]

Configuration list *(see chapter 9.2)*

Samples of **CM system records** EAL3 EAL4 EAL5

[Samples of those CM output documents are requested which are produced during the operation of the CM system documenting important activities. Examples of CM system records are CM item change control forms or CM item access approval forms.]

Supplemental information EAL2

[Although the only CM output which is required for EAL2 is the configuration list, the developer is encouraged to provide supplemental CM output suitable to illustrate the realisation of the CM system and its use. Such supplemental information will typically be provided in separate documents (files).]

2 Family CM scope (ALC_CMS)

The configuration list is output from the CM system and will in most cases be maintained and generated by means of tools. There might be various tools in use. As an example the developer might use a different tool for maintaining control over the implementation representation than for maintaining control over the documentation or the reports about security flaws. As a consequence the configuration list to be provided might consist of several physically different portions with varying layout and contents.

The sample provided here thus does not propose a structure of a configuration list but provides a sample for the content of a configuration list.

Readme for Configuration list EAL1 EAL2 EAL3 EAL4 EAL5

[The configuration list(s) provided should be supplemented by an overview document which identifies the configuration list portions and provides all information which is necessary to understand their contents.]

Identification of TOE EAL1 EAL2 EAL3 EAL4 EAL5

S-Firewall	Version 1.1 consisting of
S-Firewallbox	Version 1.2
S-FirewallMgmt	Version 3.4.5

Evaluation Documents EAL1 EAL2 EAL3 EAL4 EAL5

Security Target for S-Firewall Version 1.1, Version 1.10, 01/12/2006
Administrator Handbook for S-Firewall Version 1.1, Version 1.2, October 2006
…

Parts of the TOE EAL2 EAL3 EAL4 EAL5

name	size	version	dev	date
engine.exe	345.678	3.4	jdoe	23/06/2005
crysup.dll	123.034	1.23	fmiller	01/09/2006
fw.hlp	798.072	3.15	gbeck	15/09/2006

…

Implementation representation of the TOE EAL3 EAL4 EAL5

name	size	version	dev	date
fwengine.c	86.324	3.4	jdoe	23/06/2005
fwengine.h	1.457	1.23	jdoe	01/09/2006
/xyz/engine/make	360	3.15	gbeck	15/09/2006

…

Security flaw reports and resolution status EAL4 EAL5

name	assigned	last status change
REP20060422	ffischer	23/06/2006
REP20060305	rgold	01/09/2006
REP20050713	gbeck	15/09/2006

…

Note: In many cases these configuration tools are handled by other tools than those used for version tracking of source files or documents. Not all relevant information can thus be expected to be contained in this list. It may therefore be adequate to supplement this part of the list by samples of the referenced reports.

Development Tools EAL5

gcc 4.1.0 Free Software Foundation

…

3 Family Delivery (ALC_DEL)

The documentation of the delivery procedures is strongly dependent on the TOE product type, and of the parties involved. The major objective of this section is thus to provide guidance to the developer regarding information to be incorporated in the documentation.

Delivery Procedures Overview EAL3 EAL4 EAL5

[The delivery documentation should cover the entire TOE, but may contain different procedures for different parts of the TOE. The delivery procedures should be applicable across all phases of delivery from the production environment to the installation environment (e.g. packaging, storage and distribution). This overview section should thus identify the parts of the TOE for which different delivery procedures are to be distinguished. The overview should also identify the delivery phases to be distinguished and the parties/organisations involved in each phase.]

Security Objectives EAL3 EAL4 EAL5

[This section should identify the security aspects (integrity, confidentiality, availability) relevant for the TOE or parts of the TOE during the various phases of the delivery process. A justification for the statement of the security objectives should also be provided.]

Delivery Procedures EAL3 EAL4 EAL5

[The contents of this sections is highly dependent on the nature of the TOE. It should encompass all technical and procedural means which are relevant for the delivery procedures and for maintaining the security of the TOE or parts of the TOE during delivery.]

Verification of Users EAL3 EAL4 EAL5

[If the delivery procedures use technical means to protect the integrity of the TOE or parts of the TOE, this section should describe how a user can verify the integrity of the TOE or parts of the TOE and how a user is informed about such verification procedures.]

Evidence EAL3 EAL4 EAL5

[This section contains evidence to convince the evaluator that the procedures are in use. In case of newly developed products the evaluator needs to be convinced that the developer is prepared and willing to apply the procedures. The information and material provided for this section is typically conveyed in physically separated form.]

4 Development Security (ALC_DVS)

The documentation of the security of the development security is strongly dependent on the developers organisation. The objective of this section is more to support a brainstorming activity of the developer than to provide a rigid framework for the documentation to be provided.

Organisation EAL3 EAL4 EAL5

[This section should describe the company or organisation which the development organisation is part of by characterising its activities and outlining its geographical and organisational structure.]

Development Organisation EAL3 EAL4 EAL5

[This section should describe the development organisation by providing information about its organisational structure (commented organisation diagram), identifying all contributing locations and their contribution to the development of the TOE.]

Technical Development Environment EAL3 EAL4 EAL5

[This section should describe technical environment for TOE development. This includes the identification and characterisation of all relevant technical system, networks, and the relevant communication protocols.]

Security Policies EAL3 EAL4 EAL5

[This section should identify the security policies (regarding integrity, confidentiality, availability) relevant for the TOE or parts of the TOE during the development process. These include the policies governing:

1. what material must be protected from unauthorised modification in order to preserve the integrity of the TOE, and which members of the development staff are allowed to modify such material.
2. what information relating to the TOE development needs to be kept confidential, and which members of the development staff are allowed to access such material;
3. relevant availability aspects for information and systems

A justification for the statement of the security policies should also be provided.]

Personnel Security EAL3 EAL4 EAL5

[This section should describe which measures and procedures are in place to assure the trustworthiness of the development personnel. This includes the descriptions of screening procedures to be followed, obligations to be engaged and clearances required. This section should describe the relevant procedures for all phases of the employment (covering also the revocation of access rights when people leave the organisation).]

Access Control EAL3 EAL4 EAL5

[This section should describe which measures and procedures are in place to control physical and logical access to all locations, buildings, rooms, and systems relevant for the development of the TOE. This includes the access control measures itself as well as the related management procedures. The description should include a detailed discussion about network segregation and network access control measures. It should also describe the procedures relevant for visitor access to a development site. The information should cover the access to information in all form including paper (thus also discussing the disposal of sensitive information in paper form).]

Transfer of protected material EAL3 EAL4 EAL5

[The description should describe the measures which are in place in case protected material is transferred within and out of the development environment and between different development sites. This encompasses transfer of protected material in any form including email.]

Security Management EAL3 EAL4 EAL5

[The description should describe roles and responsibilities in ensuring the continued application of security measures, and the detection of security breaches.]

5 Flaw remediation (ALC_FLR)

The following structure is proposed for the security flaw remediation documentation:

Flaw Remediation Procedures addressed to Developers (**ALC_FLR.1 and above**)
[This part of the documentation mainly addresses documented internal procedures specifying actions and responsibilities. It should be accompanied by information which helps the evaluator to understand the procedures, related practises, and tools which are used to support the flaw remediation process. The proposed structure of the documentation addresses this supplemental information. Whenever the following text refers to requirements of the flaw remediation procedures the developer should provide references to documented procedures and arguments that convince the evaluator that the procedures fulfil the requirements.]

Tracking of reported security flaws
[The flaw remediation procedures must require that all potential security flaws are tracked starting with its detection, covering the analysis phase, and including the resolution phase. The procedures should handle suspected flaws from all possible sources (including developers and users). This section should address the developer actions and tracking mechanisms. There is no requirement – in the Common Criteria - to further track flaws in case the analysis reveals that the flaw is not security relevant. The procedures must however assure that the results of such an analysis are tracked. This section should thus describe how reported security flaws are tracked. This includes identification and descriptions of the tools used for that purpose.]

Analysis of security flaws
[The flaw remediation procedures must assure that an analysis of each reported flaw is performed and that a description of the nature and effect of the flaw is generated. This section should address such analysis and the description that is generated bases on this analysis. This section should also address how preliminary information provided by the reporter is used in this process. If the procedures foresee that a categorisation of flaws is performed, the identified categories and related classification criteria should also be described in this section.]

Correction of security flaws

Corrective actions
[The flaw remediation procedures must require that corrective actions are identified for each security flaw.]

Effectiveness of Flaw Remediation Procedures **(ALC_FLR.2 and above)**
[The documented procedures should be sufficiently detailed so that they describe how it is ensured that each reported security flaw is corrected. The procedures have to contain reasonable steps that show progress leading to the eventual, inevitable resolution. The procedures have to describe the process that is taken from the point at which the suspected security flaw is determined to be a security flaw to the point at which it is resolved. The procedures shall provide safeguards that any corrections to security flaws do not introduce any new flaw.]

Timely response **(ALC_FLR.3)**
[The documented flaw remediation procedures should contain a procedure which requires timely response.]

Status Provision of security flaws
[The flaw remediation procedures must assure that actual status information is assigned to each suspected security flaw. This should cover the early reporting phase as well as the analysis phase, the correction phase, and the phase in which the flaw remediation has been finalised. This section should identify and describe the various status identifiers and the related status transitions.]

Information of TOE Users
[It is not only necessary that flaw remediation procedures addressed to users exist but it is also important that users are aware of such information. This section should describe how users are informed about the relevant flaw remediation procedures involving users and about any changes in such procedures. It should specifically address providing flaw information, corrections and guidance on corrective actions to TOE users.]

Information supplied as part of the delivery process
[This section should specify which flaw remediation related information is communicated to users in the course of the delivery process. This might include information about procedures as well as supplemental information like account related information which a user might need to access actual information. Samples of related forms should also be provided.]

Information supplied in the guidance documentation
[This section should specify which flaw remediation related information is communicated to users in the guidance documentation. It should refer to the corresponding sections in the guidance documentation and summarise the information provided in these sections.]

Information supplied on the developer website
[This section should describe which flaw remediation related information is communicated to users on the developer website. The description should consider that information on websites is highly dynamic. The developer should thus understand the contents of this section as a commitment regarding the minimum of TOE related flaw remediation procedure information provided to users during the whole life time of the product.]

Information supplied by other means
[This section should describe other means by which flaw remediation related information is communicated to users. Such means might include but are not limited to emails, newsletters, and mailing lists.]

Involvement of TOE Users (ALC_FLR.2 and above)
[This section addresses the active involvement of the TOE user in not only passively receiving information from the developer but also actively requesting corrective actions and querying the status of such corrective actions. This section should describe the developer procedures which assure that these issues are adequately handled. The previous section (“Information of TOE Users”) should also address how information about such processes are communicated to TOE users.]

Extended Involvement of TOE Users (ALC_FLR.3)
[The flaw remediation procedures shall identify the specific points of contact for all reports and enquiries and shall include a means by which users may register with the developer to automatically receive flaw reports and associated corrections.]

Flaw Remediation Procedures addressed to TOE Users (**ALC_FLR.2 and above**)
[This part of the documentation will provide references to guidance information which is made available to users. Such guidance information shall include the description of the procedures to report suspected security flaws, request corrective actions, and query the status of such corrective actions.]

Extended Flaw Remediation Procedures addressed to TOE Users (**ALC_FLR.3**)
[This part of the documentation will also provide references to guidance information which is made available to users. Such guidance information shall also identify the specific points of contact for all reports and enquiries and shall describe the procedures by which users may register with the developer to automatically receive flaw reports and associated corrections.]

6 Family Life-cycle definition (ALC_LCD)

The following structure is proposed for the description of the life-cycle model:

Life-Cycle Model Overview EAL3 EAL4 EAL5
[This section should identify the live-cycle phases (covering at least product and version planning, development, release, and maintenance) of the TOE and the roles involved in support of the life-cycle. It should also summarise the possible transition between the life-cycle phases.]

Description Life-Cycle Model EAL3 EAL4 EAL5
[The description of the life-cycle and the underlying life-cycle model should be structured according to the life-cycle phases. For each phase the following information should be provided:]

Description of life-cycle phases
[This section should characterise a life-cycle. It should address the aspects listed below.]

Description of activities
[This section should related activities including verification, review, and other quality assurance sub-activities.]

Relationship to other phases
[This section should describe the boundaries of the phase to other phases by specifying the input and output of each phase. It should describe possible transitions to other phases and identify the corresponding conditions.]

Procedures, tools and techniques
[This section should describe information on the procedures, tools and techniques used by the developer (e.g. for design, coding, testing, bug-fixing).]

Roles and responsibilities
[This section should identify the entities acting in this phase and should describe the corresponding roles. This section should also describe the overall management structure governing the application of the procedures (e.g. an identification and description of the individual responsibilities for each of the procedures required by the development and maintenance process covered by the life-cycle model)]

Involvement of third parties
[This section should describe in how far third parties (subcontractors, consultants) are involved in the development and maintenance of the TOE. In such cases the description should cover the type and portions of the activities, and the interfaces to the developer.]

7 Family Tools and techniques (ALC_TAT)

The following structure is proposed for the development tool documentation:

Identification of development tools EAL4 EAL5

[This section should identify all development tools (programming languages and compilers, CAD systems). Identification should include version numbers and origin.]

Description of development tools EAL4 EAL5

[This section should describe each development tool by providing appropriate external documentation (programming language specifications and user manuals). The developer is advised to agree with the evaluator on documentation for which reference information only is sufficient.]

Description of implementation dependent options EAL4 EAL5

[This section should identify and describe the implementation dependent options (compiler options, linker objects, etc.) which might affect the generated code. Properly documented make-files together with the corresponding manuals will serve this purpose.]

Description of implementation standards EAL5

[This section should contain implementation standards and guidelines which the developers should follow. Such documentation can be internal standards or provided by other sources.]

Appendix D Sample Document Structure for Class ATE

This appendix proposes structure elements for the documentation to be provided for assurance class “Tests (ATE)”. Headings in bold font (like **Testplan**) are directly derived from the CC requirements and are thus strongly recommended. Headings in italic bold font (like ***Test identification***) are suggestions provided by the author.

Note: The following subchapters are ordered in the sequence ATE_FUN, ATE_COV, ATE_DPT, and ATE_IND.

1 Family Functional tests (ATE_FUN)

The following structure is proposed for the test documentation:

Testplan EAL2 EAL3 EAL4 EAL5

Identification of TOE
[This section should identify the version of the TOE to which this testplan applies.]

Test configuration
[This section should describe the test configuration. If multiple test configurations are used, it is recommended to identify an identifier to each test configuration and refer to this identifier in the description of the individual tests. This section should also provide instructions to connect and setup all required test equipment as required to conduct the test. This information should be detailed enough to ensure that the test configuration is reproducible.]

Description of TSF subsystems EAL3 EAL4 EAL5
[This section should describe the behaviour of TSF subsystems and their interactions. The information in this section shall be consistent with the TOE design and with all information contained in the test descriptions.]

Description of TSF-enforcing modules EAL4
[This section should describe the behaviour of TSF-enforcing modules and their interactions. The information in this section shall be consistent with the TOE design and with all information contained in the test descriptions.]

Description of all modules EAL5
[This section should describe the behaviour of all modules and their interactions. The information in this section shall be consistent with the TOE design and with all information contained in the test descriptions.]

Test descriptions
[This section should describe all tests (often also called test cases) of the developer’s test suite. The test descriptions should be clear and sufficiently detailed to ensure that the test is reproducible.]

Identification of test
[All tests should carry a unique identifier which allows an easy reference of this test in other test description and other documents like coverage analysis. It is recommended to choose a combination of a short identifier (which can be easily used in large tables) and a short name which gives a hint regarding the purpose of the test.]

Purpose of test
[This section should provide a test summary. It should identify the interface, summarise the interface behaviour and the related security functions to be tested.]

Description of test procedure

Instructions to establish initial conditions

[This section should describe any test pre-requisites which are necessary to establish the required initial conditions. They may be expressed in terms of parameters that must be set or in terms of test ordering in cases where the completion of one test establishes the necessary pre-requisites for another test. Examples to consider are: User accounts need to exist before they can be deleted, There is a need to perform actions in a test that will result in the generation of audit records, before performing a test to consider the searching and sorting of those audit records, generation of a file of data by one test which id to be used as input for another test.]

Instructions to stimulate the interface

[This section should describe the inputs which have to performed to stimulate the interface. It is recommended to structure this part of the test procedure into test steps.]

Description of expected test results

[This section should describe the expected test results which show the anticipated outputs from a successful execution of the tests. The description of the expected test results is needed to determine whether or not a test has been successfully performed. Expected test results are sufficient if they are unambiguous and consistent with the expected behaviour given the testing approach.]

Instructions to verify the results

[This section should provide instructions for observing the interface and should describe how the expected results should be verified. This might include performing an analysis on the observed behaviour for comparison against expected results.]

Instructions for test conclusion

[This section should provide instructions to conclude the test and establish the necessary post-test state for the TOE. This includes a description of the manual or automated cleanup processing which has to be performed.]

Report about actual test results EAL2 EAL3 EAL4 EAL5

[This section should include the information about actual test results which demonstrates that the actual test results are consistent with the expected test results.
It is strongly recommended to keep this part of the test documentation separate from the testplan, because it should be anticipated that different versions of the TOE are tested in course of the development and certification process resulting in multiple instances of this documentation part.]

Identification of tested TOE version

[Since the TOE might be subject to change in the course of the development process, the test documentation should also reference the actually tested version of the TOE.]

Results of individual tests

Identification of tester

[This section should contain the names of the tester and (if different from the tester) the authors of the test report]

Date and time

[If not specified in other portions of this report this section should specify the date and time this test was executed. The granularity of the information provided should allow to determine the sequence of the tests performed.]

Detailed results
[This section should contain detailed information about the actual test results. The level of detail required is governed by the requirement that it must be demonstrated that the actual test results are consistent with the expected test results.]

Procedures for data reduction or synthesis
[If a direct comparison of actual results can be made until some data reduction or synthesis has been first performed, this section should describe the process to reduce or synthesise the actual data.]

Verdict
[This section should just contain a clear statement regarding the success of a test. It is recommended to simply specify “pass”, “fail”, or “inconclusive” (in case a problem with the tests environment occurs).]

Example test descriptions

The following text provides two examples of test descriptions. The first example (Test IA-7) addresses an application providing user authentication by passwords. The second example (Test 4712) addresses a firewall dropping malformed packages.

Test IA-7 Excessive authentication attempts

Purpose of test
This test is performed via the Login Panel which is displayed after startup of *sample_application*. A valid userid *sample_userid* and a wrong password are entered three times. After these three attempts the same userid and the correct password are entered. It is checked that the login is rejected four times.

Description of test procedure

Instructions to establish initial conditions
Login as an administrator and define user *sample_userid* and assign a password. Login as user *sample_userid* and enter the correct password. Verify that user *sample_userid* can successfully login. Logoff user *sample_userid*.

Instructions to stimulate the interface
Step 1: Enter sample_userid and a correctly formed password but wrong password. If login is rejected, continue test. If not, terminate test.

Step 2: Enter sample_userid and same wrong password than before. If login is rejected, continue test. If not, terminate test.

Step 3: Enter sample_userid and same wrong password than before. If login is rejected, continue test. If not, terminate test.

Step 4: Enter sample_userid and correct password. Observe reaction of TOE.

Description of expected test results
The login should be rejected in all four cases. In the first three cases the login attempt should be rejected because the wrong password has been supplied. In the fourth case the login attempt should be rejected because user access should have been revoked after three unsuccesful attempts.

Instructions to verify the results
Beside observing the TOE behaviour at the Login Panel the application log has to be inspected. For steps 1 and 2 message 305 and 308 have to be issued. For step 3 messages 305, 311 and 308 have to be issued. For step 4 messages 305 and 307 have to be issued.

Instructions for test conclusion
Login as administrator and delete user sample_userid.

Test 4712 Dropping malformed packet.

Purpose of test
This test is performed via a network interface. A handcrafted malformed packet is sent to the TOE. It is check that the packet is dropped.

Description of test procedure

Instructions to establish initial conditions
Activate test configuration A.

Instructions to stimulate the interface
Use hpng2 to generate a malformed packet (SYN+FIN+RST flags set) and send it to the TOE.

Description of expected test results
The packet should be dropped and a log message should be generated.

Instructions to verify the results
Use a network sniffer (etherreal) to observe that the packet is not forwarded in the target network. Analyse system log to conform that a correct and clear message is logged.

Instructions for test conclusion
none

2 Family Coverage (ATE_COV)

Test coverage evidence EAL2 EAL3 EAL4 EAL5

[It is recommended to arrange the test coverage evidence in the form of a table in which the rows represent the tests and the columns represent the TSFIs and their characteristics.

T-1 … T-9 Identifiers for tests
TSFI-1 … TSFI-4 Identifiers for TSFI
Char_1 … Char_8 Characteristics of TSFI regarding security functionality. Since the table cell will not allow to provide an adequate description of the security functionality to be tested, it is encouraged to use identifiers, which refer to a description outside the table.]

The following sample table visualises such a table. It mainly serves for demonstration purposes and contains only a few rows and columns.

	TSFI-1			**TSFI-2**		**TSFI-3**	**TSFI-4**		
	Char_1	Char_2	Char_3	Char_4	Char_5		Char_6	Char_7	Char_8
T-1	X								
T-2		X							
T-3			X						
T-4				X			X		
T-5				X					
T-6							X		
T-7								X	
T-8									X
T-9									

Table 3: Sample coverage evidence

The following example addresses above table.

Assume the TOE to be tested can be referenced as “SecurePrd 2.1”.

Assume *TSFI-1* refers to the “Administration GUI”.

Characteristic *Char_1* refers to the following characteristic identified in the Functional Specification regarding administrator identification and authentication:

> SecurePrd 2.1 requires a user to provide a valid user name and the corresponding password before a user can gain access to the administration functions of SavePrd 2.1. In such a case message *100 Authentication successful* is issued. If no user name is provided message *200 Enter User Name* is issued. If user name or password is invalid, message *300 Authentication failed* is issued.

T-1 is a test consisting of three subtests. A summary of the test procedure is:

> All subtests of this test are performed by using the login screen of the administrative GUI entering data and observing returned messages. As a preparation activity for this test a user name *testuser* has to be defined and a valid password (*pAss123#*) assigned.

Test T-1.1: Hit *Enter* without entering a user name. Message *200 Enter User Name* must be issued.

Test T-1.2: Enter user name *testuser* and password *pAss124#*. Message *300 Authentication failed* must be issued.

Test T-1.3: Enter user name *testuser* and password *pAss123#*. Message *100 Authentication successful* must be issued and the *Main Administration Menu* must be displayed.

Test coverage analysis EAL3 EAL4 EAL5

TSFI overview

[This section should identify the TSFIs described in the functional specification and should list the characteristics of the TSFIs regarding security functionality as specified in the functional specification.]

Coverage analysis

[This section should analyse in how far the TSFIs and its characteristics as specified in the functional specification are covered by the tests. For this analysis it should refer to the contents of the table in the previous main section. Obviously this analysis can only work smoothly if the characteristics referred to in the table are of the granularity than those identified and discussed in this section.]

3 Family Depth (ATE_DPT)

Test subsystem correspondence EAL3

[It is recommended to arrange the test subsystem correspondence in the form of a table in which the rows represent the tests and the columns represent the subsystems and their characteristics.

T-1 … T-9 Identifiers for tests

Char_1 … Char_8 Characteristics of subsystem behaviour. Since the table cell will not allow to provide an adequate description of the subsystem behaviour to be tested, it is encouraged to use identifiers, which refer to a description outside the table. It is strongly recommended to use section "Description of TSF subsystems" in the testplan for that purpose.]

The following sample table visualises such a table. It mainly serves for demonstration purposes and contains only a few rows and columns.

	Subsystem A			**Subsystem B**			**Subsystem C**		
	Char_1	Char_2	Char_3	Char_4	Char_5	Char_6	Char_7	Char_8	Char_9
T-1	X					X			
T-2		X							
T-3			X						
T-4				X	X		X		
T-5				X					
T-6							X		
T-7								X	
T-8									X
T-9									

Table 4: Sample test subsystem correspondence EAL3

The following example addresses above table.

Assume the TOE to be tested can be referenced as "SecurePrd 2.1".

Assume *Subsystem A* refers to the "Subsystem Administrator GUI".

Characteristic *Char_1* refers to the following characteristic regarding administrator identification and authentication:

> SecurePrd 2.1 requires a user to provide a valid user name and the corresponding password before a user can gain access to the administration functions of SavePrd 2.1. In such a case message *100 Authentication successful* is issued. If no user name is provided message *200 Enter User Name* is issued. If user name or password is invalid, message *300 Authentication failed* is issued. Before testing the provided password against the password database, the Subsystem Administrator GUI validates user input by checking the length of the user name (must be 4-8 characters) and the length of the password (must be 6-10 characters).

T-1 is a test consisting of seven subtests. A summary of the test procedure is:

The first three subtests of this test are performed by using the login screen of the administrative GUI entering data and observing returned messages. As a preparation activity for this test a user name *testuser* has to be defined and a valid password (*pAss123#*) assigned.

Test T-1.1: Hit *Enter* without entering a user name. Message *200 Enter User Name* must be issued.

Test T-1.2: Enter user name *testuser* and password *pAss124#*. Message *300 Authentication failed* must be issued.

Test T-1.3: Enter user name *testuser* and password *pAss123#*. Message *100 Authentication successful* must be issued and the *Main Administration Menu* must be displayed.

The other subtests of this test are also performed by using the login screen of the administrative GUI entering data and observing returned messages. Additionally the event log of SecurePrd 2.1 has to be analysed.

Test T-1.4: Enter user name *tst* and password *pAss123#*. Message *300 Authentication failed* must be issued and log message *GU300 User Name too short* must be generated.

Test T-1.5: Enter user name *testuser1* and password *pAss123#*. Message *300 Authentication failed* must be issued and log message *GU301 User Name too long* must be generated.

Test T-1.6: Enter user name *testuser* and password *pAss1*. Message *302 Authentication failed* must be issued and log message *GU302 Password too short* must be generated.

Test T-2.5: Enter user name *testuser* and password *pAss123#1*. Message *302 Authentication failed* must be issued and log message *GU303 Password too short* must be generated.

Test subsystem & module correspondence EAL4 EAL5

[It is recommended to arrange the test subsystem & module correspondence in the form of a table in which the rows represent the tests and the columns represent the subsystems and their modules.

T-1 … T-9 Identifiers for tests

The following sample table visualises such a table. It mainly serves for demonstration purposes and contains only a few rows and columns.

	Subsystem A			**Subsystem B**			**Subsystem C**		
	Module a	Module b		Module c	Module d		Module e	Module f	
T-1		X	X						
T-2	X		X						
T-3			X						
T-4				X	X	X	X		
T-5				X		X			
T-6							X		

T-7								X	
T-8									X
T-9									

Table 5: Sample test subsystem & module correspondence, EAL4 and EAL5

The following example addresses above table.

Assume the TOE to be tested can be referenced as “SecurePrd 2.1”.

Assume *Subsystem A* refers to the “Subsystem Administrator GUI”, which contains the modules “GUI_UserInput” (*Module a*, which analyses the input from the users) and “GUI_PwdDataBase” (*Module b,* which checks the Password Database).

Characteristic *Char_1* refers to the following characteristic regarding administrator identification and authentication:

> SecurePrd 2.1 requires a user to provide a valid user name and the corresponding password before a user can gain access to the administration functions of SecurePrd 2.1. In such a case message *100 Authentication successful* is issued. If no user name is provided message *200 Enter User Name* is issued. If user name or password is invalid, message *300 Authentication failed* is issued. Testing the password is provided by module GUI_PwdDataBase of subsystem Administrator GUI. Module GUI_PwdDataBase provides expressive log messages prefixed by *GP*.

T-1 is a test consisting of four subtests. A summary of the test procedure is:

> The subtests of this test are performed by using the login screen of the administrative GUI entering data and observing returned messages. Additionally the event log of SavePrd 2.1 has to be analysed. As a preparation activity for this test a user name *testuser* has to be defined and a valid password (*pAss123#*) assigned.
>
> Test T-1.1: Enter user name *testuser* and password *pAss124#*. Message *300 Authentication failed* must be issued and log message *GP300 Wrong Password supplied* has to be generated.
>
> Test T-1.2: Enter user name *testuser* and password *pAss123#*. Message *100 Authentication successful* must be issued. log message *GP100 Authentication successful* has to be generated, and the *Main Administration Menu* must be displayed.
>
> Subtest T-1.3 requires additional preparation by renaming the Password Database.
>
> Test T-1.3: Enter user name *testuser* and password *pAss123#*. Message *302 Authentication failed* must be issued and log message *GP500 Password Database could not be opened* must be generated.
>
> Subtest T-1.4 requires additional preparation by replacing the Password Database by a text file.
>
> Test T-1.4: Enter user name *testuser* and password *pAss123#*. Message *302 Authentication failed* must be issued and log message *GP501 Password Database corrupted* must be generated.

Characteristic *Char_2* refers to the following characteristic regarding administrator identification and authentication:

> Before testing the provided password against the password database, module GUI_UserInput of Subsystem Administrator GUI validates user input by checking the length of the user name (must be 4-8 characters) and the length of the password (must be 6-10 characters). This functionality is provided by module GUI_UserInput of subsystem Administrator GUI. Module GUI_UserInput provides expressive log messages prefixed by *GU*.

T-2 is a test consisting of five subtests. A summary of the test procedure is:

> The subtests of this test are performed by using the login screen of the administrative GUI entering data and observing returned messages. Additionally the event log of SavePrd 2.1 has to be analysed.
>
> Test T-2.1: Hit *Enter* without entering a user name. Message *200 Enter User Name* must be issued and log message *GU200 No User Name supplied* has to be generated. No log message prefixed by *GP* must be issued.
>
> Test T-2.2: Enter user name *tst* and password *pAss123#*. Message *300 Authentication failed* must be issued and log message *GU300 User Name too short* must be generated. No log message prefixed by *GP* must be issued.
>
> Test T-2.3: Enter user name *testuser1* and password *pAss123#*. Message *300 Authentication failed* must be issued and log message *GU301 User Name too long* must be generated. No log message prefixed by *GP* must be issued.
>
> Test T-2.4: Enter user name *testuser* and password *pAss1*. Message *302 Authentication failed* must be issued and log message *GU302 Password too short* must be generated. No log message prefixed by *GP* must be issued.
>
> Test T-2.5: Enter user name *testuser* and password *pAss123#1*. Message *302 Authentication failed* must be issued and log message *GU303 Password too short* must be generated. No log message prefixed by *GP* must be issued.

Test depth analysis (subsystems) EAL3 EAL4 EAL5

[This section shall demonstrate that all TSF subsystems in the TOE design have been tested. All descriptions of TSF subsystem behaviour and of interactions among TSF subsystems that are provided in the TOE design have to be tested. The behaviour of TSF subsystems may be tested directly from those interfaces. Otherwise, the behaviour of those subsystems is tested from the TSFI interfaces. Or a combination of the two may be employed.]

Test depth analysis (security enforcing modules) EAL4

[This section shall demonstrate that the SFR-enforcing modules, i.e. all interfaces of SFR-enforcing modules in the TOE design, have been tested. Testing of an interface may be performed directly at that interface, or at the external interfaces, or a combination of both. A simple cross-table may be sufficient to show test correspondence. The identification of the tests and the behaviour/interaction presented in the depth-of coverage analysis has to be unambiguous.]

Test depth analysis (security enforcing modules) EAL5

[This section shall demonstrate that all modules, i.e. all interfaces to all modules in the TOE design, have been tested. Testing of an interface may be performed directly at that interface, or at the external interfaces, or a combination of both. A simple cross-table may be sufficient to

show test correspondence. The identification of the tests and the behaviour/interaction presented in the depth-of coverage analysis has to be unambiguous.]

4 Family Independent testing (ATE_IND)

There is no explicit requirement for the developer to provide documentation to support the testing of the evaluators. The developer should be reminded, however, that “an equivalent set of resources to those that were used in the developer's functional testing” also includes adequate usage documentation regarding the test tools and other means used for the test.

Appendix E Sample Document Structure for Class AVA

Since in this version of the CC there are no documentation requirements directed to the developer for this class, there is no need to provide a corresponding sample document structure.