Common Methodology for Information Technology Security Evaluation

Evaluation methodology

July 2005

Version 3.0
Revision 2

CCMB-2005-07-004

Foreword

This version of the Common Methodology for Information Technology Security Evaluation (CEM v3.0) is the first major revision since being published as CEM v1.0 in 1999 and CEM v2.2 in 2004.

CEM v3.0 is released for public comment and aims to eliminate redundant evaluation activities; reduce/eliminate those activities that contributed little to the final assurance of a product; clarify CC terminology to reduce misunderstandings; and restructure and refocus the evaluation activities to those areas where security assurance would truly be gained.

This revision 2 of CEM v3.0 includes all editorial updates as of the release date.

Trademarks:

- Microsoft is a registered trademark of Microsoft Corporation
- POSIX is a registered trademark of the IEEE
- UNIX is a registered trademark of The Open Group in the United States and other countries
- Windows is a registered trademark of Microsoft Corporation in the United States and other countries

Legal Notice:

The governmental organisations listed below contributed to the development of this version of the Common Methodology for Information Technology Security Evaluation. As the joint holders of the copyright in the Common Methodology for Information Technology Security Evaluation, version 3.0 *(called "CEM* 3.0*"), they hereby grant non-exclusive license to ISO/IEC to use CEM* 3.0 *in the continued development/maintenance of the ISO/IEC 18045 international standard. However, these governmental organisations retain the right to use, copy, distribute, translate or modify CEM* 3.0 *as they see fit.*

Australia/New Zealand:	*The Defence Signals Directorate and the Government Communications Security Bureau respectively;*
Canada:	*Communications Security Establishment;*
France:	*Direction Centrale de la Sécurité des Systèmes d'Information;*
Germany:	*Bundesamt für Sicherheit in der Informationstechnik;*
Japan:	*Information Technology Promotion Agency*
Netherlands:	*Netherlands National Communications Security Agency;*
Spain:	*Ministerio de Administraciones Públicas and Centro Criptológico Nacional;*
United Kingdom:	*Communications-Electronics Security Group;*
United States:	*The National Security Agency and the National Institute of Standards and Technology.*

Table of Contents

Table of contents

Table of contents

List of figures

List of tables

1 Introduction

1 The target audience for the Common Methodology for Information Technology Security Evaluation (CEM) is primarily evaluators applying the CC and certifiers confirming evaluator actions; evaluation sponsors, developers, PP/ST authors and other parties interested in IT security may be a secondary audience.

2 The CEM recognises that not all questions concerning IT security evaluation will be answered herein and that further interpretations will be needed. Individual schemes will determine how to handle such interpretations, although these may be subject to mutual recognition agreements. A list of methodology-related activities that may be handled by individual schemes can be found in Annex A.

2 Scope

3 The Common Methodology for Information Technology Security Evaluation (CEM) is a companion document to the Common Criteria for Information Technology Security Evaluation (CC). The CEM describes the minimum actions to be performed by an evaluator in order to conduct a CC evaluation, using the criteria and evaluation evidence defined in the CC.

3 Normative references

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

CC Common Criteria for Information Technology Security Evaluation, Version 3.0, revision 2, June 2005.

4 Terms and definitions

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

6 Terms which are presented in bold-faced type are themselves defined in this section.

7 **action —** evaluator action element of the CC Part 3. These actions are either explicitly stated as evaluator actions or implicitly derived from developer actions (implied evaluator actions) within the CC Part 3 assurance components.

8 **activity —** the application of an assurance class of the CC Part 3.

9 **check —** to generate a **verdict** by a simple comparison. Evaluator expertise is not required. The statement that uses this verb describes what is mapped.

10 **evaluation deliverable —** any resource required from the sponsor or developer by the evaluator or overseer to perform one or more evaluation or evaluation oversight activities.

11 **evaluation evidence —** a tangible **evaluation deliverable**.

12 **evaluation technical report —** a report that documents the **overall verdict** and its justification, produced by the evaluator and submitted to an overseer.

13 **examine —** to generate a **verdict** by analysis using evaluator expertise. The statement that uses this verb identifies what is analysed and the properties for which it is analysed.

14 **interpretation —** a clarification or amplification of a CC, CEM or **scheme** requirement.

15 **methodology —** the system of principles, procedures and processes applied to IT security evaluations.

16 **observation report —** a report written by the evaluator requesting a clarification or identifying a problem during the evaluation.

17 **overall verdict —** a *pass or fail* statement issued by an evaluator with respect to the result of an evaluation.

18 **oversight verdict —** a statement issued by an overseer confirming or rejecting an *overall verdict* based on the results of evaluation oversight activities.

19 **record —** to retain a written description of procedures, events, observations, insights and results in sufficient detail to enable the work performed during the evaluation to be reconstructed at a later time.

20 **report —** to include evaluation results and supporting material in the **Evaluation Technical Report** or an **Observation Report**.

21 **scheme —** set of rules, established by an evaluation authority, defining the evaluation environment, including criteria and **methodology** required to conduct IT security evaluations.

22 **sub-activity —** the application of an assurance component of the CC Part 3. Assurance families are not explicitly addressed in the CEM because evaluations are conducted on a single assurance component from an assurance family.

23 **tracing —** a simple directional relation between two sets of entities, which shows which entities in the first set correspond to which entities in the second.

24 **verdict —** a *pass, fail or inconclusive* statement issued by an evaluator with respect to a CC evaluator action element, assurance component, or class. Also see **overall verdict**.

25 **work unit —** the most granular level of evaluation work. Each CEM action comprises one or more work units, which are grouped within the CEM action by CC content and presentation of evidence or developer action element. The work units are presented in the CEM in the same order as the CC elements from which they are derived. Work units are identified in the left margin by a symbol such as *4:ALC_TAT.1-2*. In this symbol, the first digit (*4*) indicates the EAL; the string *ALC_TAT.1* indicates the CC component (i.e. the CEM sub-activity), and the final digit (*2*) indicates that this is the second work unit in the ALC_TAT.1 sub-activity.

5 Symbols and abbreviated terms

CEM	Common Methodology for Information Technology Security Evaluation
ETR	Evaluation Technical Report
OR	Observation Report

6 Overview

6.1 Organisation of the CEM

26 Chapter 7 defines the conventions used in the CEM.

27 Chapter 8 describes general evaluation tasks with no verdicts associated with them as they do not map to CC evaluator action elements.

28 Chapter 9 addresses the work necessary for reaching an evaluation result on a PP.

29 Chapters 10 to 16 define the evaluation activities, organised by Assurance Classes.

30 Annex A covers the basic evaluation techniques used to provide technical evidence of evaluation results.

31 Annex B provides an explanation of the Vulnerability Analysis criteria and examples of their application

7 Document Conventions

7.1 Terminology

32 Unlike the CC, where each element maintains the last digit of its identifying symbol for all components within the family, the CEM may introduce new work units when a CC evaluator action element changes from sub-activity to sub-activity; as a result, the last digit of the work unit's identifying symbol may change although the work unit remains unchanged.

33 Any methodology-specific evaluation work required that is not derived directly from CC requirements is termed *task* or *sub-task*.

7.2 Verb usage

34 All work unit and sub-task verbs are preceded by the auxiliary verb *shall* and by presenting both the verb and the *shall* in ***bold italic*** type face. The auxiliary verb *shall* is used only when the provided text is mandatory and therefore only within the work units and sub-tasks. The work units and sub-tasks contain mandatory activities that the evaluator must perform in order to assign verdicts.

35 Guidance text accompanying work units and sub-tasks gives further explanation on how to apply the CC words in an evaluation. The described method is normative, meaning that the verb usage is in accordance with ISO definitions for these verbs; that is: the auxiliary verb *should* is used when the described method is strongly preferred and the auxiliary verb *may* is used where the described method(s) is allowed but no preference is indicated. (The auxiliary verb *shall* is used only for the text of work units.)

36 The verbs *check*, *examine*, *report* and *record* are used with a precise meaning within this part of the CEM and the chapter 4 should be referenced for their definitions.

7.3 General evaluation guidance

37 Material that has applicability to more than one sub-activity is collected in one place. Guidance whose applicability is widespread (across activities and EALs) has been collected into Annex A. Guidance that pertains to multiple sub-activities within a single activity has been provided in the introduction to that activity. If guidance pertains to only a single sub-activity, it is presented within that sub-activity.

7.4 Relationship between CC and CEM structures

38 There are direct relationships between the CC structure (i.e. class, family, component and element) and the structure of the CEM. Figure 1 illustrates the correspondence between the CC constructs of class, family and evaluator action elements and CEM activities, sub-activities and actions. However,

several CEM work units may result from the requirements noted in CC developer action and content and presentation elements.

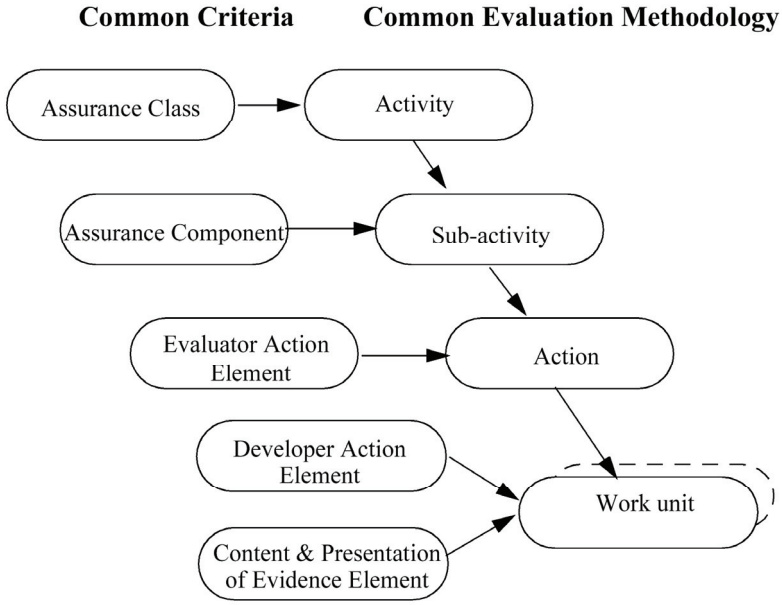

Figure 1 - Mapping of the CC and CEM structures

8 Evaluation process and related tasks

8.1 Introduction

39 This chapter provides an overview of the evaluation process and defines the tasks an evaluator is intended to perform when conducting an evaluation.

40 Each evaluation, whether of a PP or TOE (including ST), follows the same process, and has four evaluator tasks in common: the input task, the output task, the evaluation sub-activities, and the demonstration of the technical competence to the evaluation authority task.

41 The input task and the output tasks, which are related to management of evaluation evidence and to report generation, are entirely described in this chapter. Each task has associated sub-tasks that apply to, and are normative for all CC evaluations (evaluation of a PP or a TOE).

42 The evaluation sub-activities are only introduced in this chapter, and fully described in the following chapters.

43 In contrast to the evaluation sub-activities, input and output tasks have no verdicts associated with them as they do not map to CC evaluator action elements; they are performed in order to ensure conformance with the universal principles and to comply with the CEM.

44 The demonstration of the technical competence to the evaluation authority task may be fulfilled by the evaluation authority analysis of the output tasks results, or may include the demonstration by the evaluators of their understanding of the inputs for the evaluation sub-activities. This task has no associated evaluator verdict, but has an evaluator authority verdict. The detailed criteria to pass this task are left to the discretion of the evaluation authority, as noted in Annex A.6.

8.2 Evaluation process overview

8.2.1 Objectives

45 This section presents the general model of the methodology and identifies:

a) roles and responsibilities of the parties involved in the evaluation process;

b) the general evaluation model.

8.2.2 Responsibilities of the roles

46 The general model defines the following roles: sponsor, developer, evaluator and evaluation authority.

47 The sponsor is responsible for requesting and supporting an evaluation. This means that the sponsor establishes the different agreements for the evaluation (e.g. commissioning the evaluation). Moreover, the sponsor is responsible for ensuring that the evaluator is provided with the evaluation evidence.

48 The developer produces the TOE and is responsible for providing the evidence required for the evaluation (e.g. training, design information), on behalf of the sponsor.

49 The evaluator performs the evaluation tasks required in the context of an evaluation: the evaluator receives the evaluation evidence from the developer on behalf of the sponsor or directly from the sponsor, performs the evaluation sub-activities and provides the results of the evaluation assessment to the evaluation authority.

50 The evaluation authority establishes and maintains the scheme, monitors the evaluation conducted by the evaluator, and issues certification/validation reports as well as certificates based on the evaluation results provided by the evaluator.

8.2.3 Relationship of roles

51 To prevent undue influence from improperly affecting an evaluation, some separation of roles is required. This implies that the roles described above are fulfilled by different entities, except that the roles of developer and sponsor may be satisfied by a single entity.

52 Moreover, some evaluations (e.g. EAL1 evaluation) may not require the developer to be involved in the project. In this case, it is the sponsor who provides the TOE to the evaluator and who generates the evaluation evidence.

8.2.4 General evaluation model

53 The evaluation process consists of the evaluator performing the evaluation input task, the evaluation output task and the evaluation sub-activities. Figure 2 provides an overview of the relationship between these tasks and sub-activities.

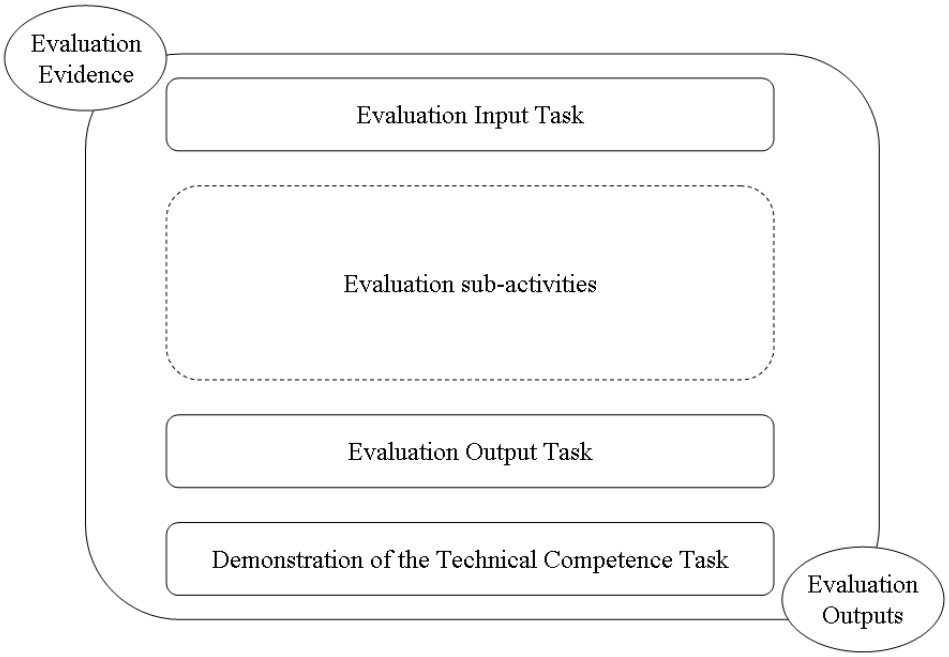

Figure 2 - Generic evaluation model

54 The evaluation process may be preceded by a preparation phase where initial contact is made between the sponsor and the evaluator. The work that is performed and the involvement of the different roles during this phase may vary. It is typically during this step that the evaluator performs a feasibility analysis to assess the likelihood of a successful evaluation.

8.2.5 Evaluator verdicts

55 The evaluator assigns verdicts to the requirements of the CC and not to those of the CEM. The most granular CC structure to which a verdict is assigned is the evaluator action element (explicit or implied). A verdict is assigned to an applicable CC evaluator action element as a result of performing the corresponding CEM action and its constituent work units. Finally, an evaluation result is assigned, as described in CC Part 1, Chapter 9, Evaluation results

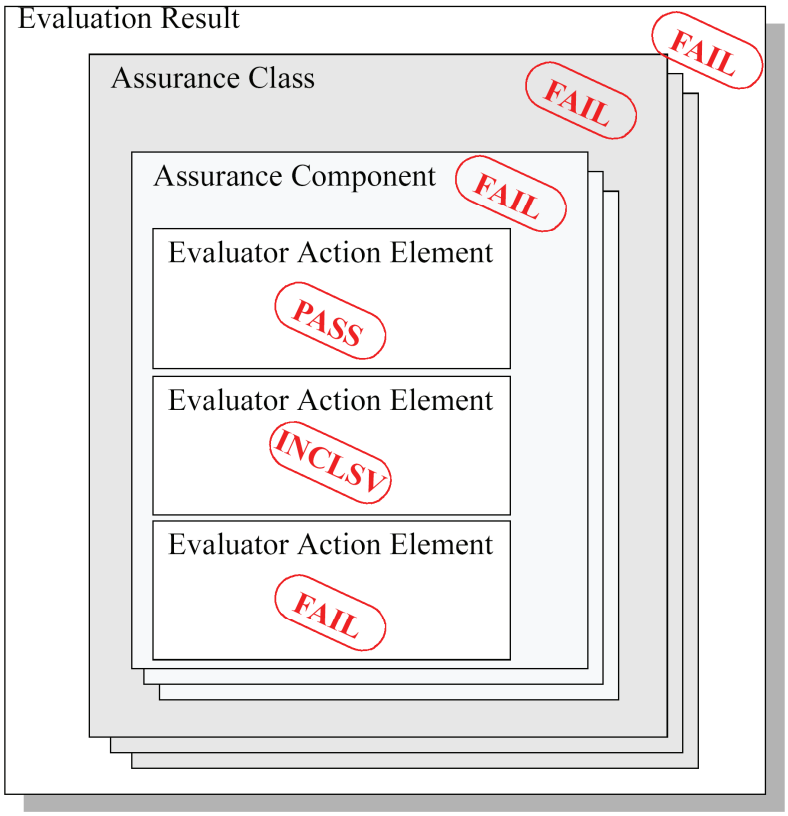

Figure 3 - Example of the verdict assignment rule

56 The CEM recognises three mutually exclusive verdict states:

a) Conditions for a *pass* verdict are defined as an evaluator completion of the CC evaluator action element and determination that the requirements for the PP, ST or TOE under evaluation are met. The conditions for passing the element are defined as:

1) the constituent work units of the related CEM action, and;

2) all evaluation evidence required for performing these work units is coherent, that is it can be fully and completely understood by the evaluator, and

3) all evaluation evidence required for performing these work units does not have any obvious internal inconsistencies or inconsistencies with other evaluation evidence. Note that obvious means here that the evaluator discovers this inconsistency while performing the work units: the evaluator should not undertake a full consistency analysis across the entire evaluation evidence every time a work unit is performed.

b) Conditions for a *fail* verdict are defined as an evaluator completion of the CC evaluator action element and determination that the requirements for the PP, ST, or TOE under evaluation are not met, or

that the evidence is incoherent, or an obvious inconsistency in the evaluation evidence has been found;

c) All verdicts are initially *inconclusive* and remain so until either a *pass* of *fail* verdict is assigned.

57 The overall verdict is *pass* if and only if all the constituent verdicts are also *pass*. In the example illustrated in Figure 3, if the verdict for one evaluator action element is *fail* then the verdicts for the corresponding assurance component, assurance class, and overall verdict are also *fail*.

8.3 Evaluation input task

8.3.1 Objectives

58 The objective of this task is to ensure that the evaluator has available the correct version of the evaluation evidence necessary for the evaluation and that it is adequately protected. Otherwise, the technical accuracy of the evaluation cannot be assured, nor can it be assured that the evaluation is being conducted in a way to provide repeatable and reproducible results.

8.3.2 Application notes

59 The responsibility to provide all the required evaluation evidence lies with the sponsor. However, most of the evaluation evidence is likely to be produced and supplied by the developer, on behalf of the sponsor.

60 Since the assurance requirements apply to the entire TOE, all evaluation evidence pertaining to all parts of the TOE is to be made available to the evaluator. The scope and required content of such evaluation evidence is independent of the level of control that the developer has over each of the parts of the TOE. For example, if design is required, then the TOE design (ADV_TDS) requirements will apply to all subsystems that are part of the TSF. In addition, assurance requirements that call for procedures to be in place (for example, CM capabilities (ALC_CMC) and Delivery (ALC_DEL)) will also apply to the entire TOE (including any part produced by another developer).

61 It is recommended that the evaluator, in conjunction with the sponsor, produce an index to required evaluation evidence. This index may be a set of references to the documentation. This index should contain enough information (e.g. a brief summary of each document, or at least an explicit title, indication of the sections of interest) to help the evaluator to find easily the required evidence.

62 It is the information contained in the evaluation evidence that is required, not any particular document structure. Evaluation evidence for a sub-activity may be provided by separate documents, or a single document may satisfy several of the input requirements of a sub-activity.

63 The evaluator requires stable and formally-issued versions of evaluation evidence. However, draft evaluation evidence may be provided during an evaluation, for example, to help an evaluator make an early, informal assessment, but is not used as the basis for verdicts. It may be helpful for the evaluator to see draft versions of particular appropriate evaluation evidence, such as:

a) test documentation, to allow the evaluator to make an early assessment of tests and test procedures;

b) design documents, to provide the evaluator with background for understanding the TOE design;

c) source code or hardware drawings, to allow the evaluator to assess the application of the developer's standards.

64 Draft evaluation evidence is more likely to be encountered where the evaluation of a TOE is performed concurrently with its development. However, it may also be encountered during the evaluation of an already-developed TOE where the developer has had to perform additional work to address a problem identified by the evaluator (e.g. to correct an error in design or implementation) or to provide evaluation evidence of security that is not provided in the existing documentation (e.g. in the case of a TOE not originally developed to meet the requirements of the CC).

8.3.3 Management of evaluation evidence sub-task

8.3.3.1 Configuration control

65 The evaluator ***shall perform*** configuration control of the evaluation evidence.

66 The CC implies that the evaluator is able to identify and locate each item of evaluation evidence after it has been received and is able to determine whether a specific version of a document is in the evaluator's possession.

67 The evaluator ***shall protect*** the evaluation evidence from alteration or loss while it is in the evaluator's possession.

8.3.3.2 Disposal

68 Schemes may wish to control the disposal of evaluation evidence at the conclusion of an evaluation. The disposal of the evaluation evidence should be achieved by one or more of:

a) returning the evaluation evidence;

b) archiving the evaluation evidence;

c) destroying the evaluation evidence.

8.3.3.3 Confidentiality

69 An evaluator may have access to sponsor and developer commercially-sensitive information (e.g. TOE design information, specialist tools), and may have access to nationally-sensitive information during the course of an evaluation. Schemes may wish to impose requirements for the evaluator to maintain the confidentiality of the evaluation evidence. The sponsor and evaluator may mutually agree to additional requirements as long as these are consistent with the scheme.

70 Confidentiality requirements affect many aspects of evaluation work, including the receipt, handling, storage and disposal of evaluation evidence.

8.4 Evaluation sub-activities

71 The evaluation sub-activities vary depending whether it is a PP or a TOE evaluation. Moreover, in the case of a TOE evaluation, the sub-activities depend upon the selected assurance requirements.

8.5 Evaluation output task

8.5.1 Objectives

72 The objective of this section is to describe the Observation Report (OR) and the Evaluation Technical Report (ETR). Schemes may require additional evaluator reports such as reports on individual units of work, or may require additional information to be contained in the OR and the ETR. The CEM does not preclude the addition of information into these reports as the CEM specifies only the minimum information content.

73 Consistent reporting of evaluation results facilitates the achievement of the universal principle of repeatability and reproducibility of results. The consistency covers the type and the amount of information reported in the ETR and OR. ETR and OR consistency among different evaluations is the responsibility of the overseer.

74 The evaluator performs the two following sub-tasks in order to achieve the CEM requirements for the information content of reports:

a) write OR sub-task (if needed in the context of the evaluation);

b) write ETR sub-task.

8.5.2 Management of evaluation outputs

75 The evaluator delivers the ETR to the evaluation authority, as well as any ORs as they become available. Requirements for controls on handling the ETR and ORs are established by the scheme which may include delivery to the sponsor or developer. The ETR and ORs may include sensitive or proprietary information and may need to be sanitised before they are given to the sponsor.

8.5.3 Application notes

76 In this version of the CEM, the requirements for the provision of evaluator evidence to support re-evaluation and re-use have not been explicitly stated. Where information for re-evaluation or re-use is required by the sponsor, the scheme under which the evaluation is being performed should be consulted.

8.5.4 Write OR sub-task

77 ORs provide the evaluator with a mechanism to request a clarification (e.g. from the overseer on the application of a requirement) or to identify a problem with an aspect of the evaluation.

78 In the case of a fail verdict, the evaluator ***shall provide*** an OR to reflect the evaluation result. Otherwise, the evaluator may use ORs as one way of expressing clarification needs.

79 For each OR, the evaluator ***shall report*** the following:

a) the identifier of the PP or TOE evaluated;

b) the evaluation task/sub-activity during which the observation was generated;

c) the observation;

d) the assessment of its severity (e.g. implies a fail verdict, holds up progress on the evaluation, requires a resolution prior to evaluation being completed);

e) the identification of the organisation responsible for resolving the issue;

f) the recommended timetable for resolution;

g) the assessment of the impact on the evaluation of failure to resolve the observation.

80 The intended audience of an OR and procedures for handling the report depend on the nature of the report's content and on the scheme. Schemes may distinguish different types of ORs or define additional types, with associated differences in required information and distribution (e.g. evaluation ORs to overseers and sponsors).

8.5.5 Write ETR sub-task

8.5.5.1 Objectives

81 The evaluator ***shall provide*** an ETR to present technical justification of the verdicts.

82 The CEM defines the ETR's minimum content requirement; however, schemes may specify additional content and specific presentational and structural requirements. For instance, schemes may require that certain introductory material (e.g. disclaimers and copyright chapters) be reported in the ETR.

83 The reader of the ETR is assumed to be familiar with general concepts of information security, the CC, the CEM, evaluation approaches and IT.

84 The ETR supports the evaluation authority to confirm that the evaluation was done to the required standard, but it is anticipated that the documented results may not provide all of the necessary information, so additional information specifically requested by the scheme may be necessary. This aspect is outside the scope of the CEM.

8.5.5.2 ETR for a PP Evaluation

85 This section describes the minimum content of the ETR for a PP evaluation. The contents of the ETR are portrayed in Figure 4; this figure may be used as a guide when constructing the structural outline of the ETR document.

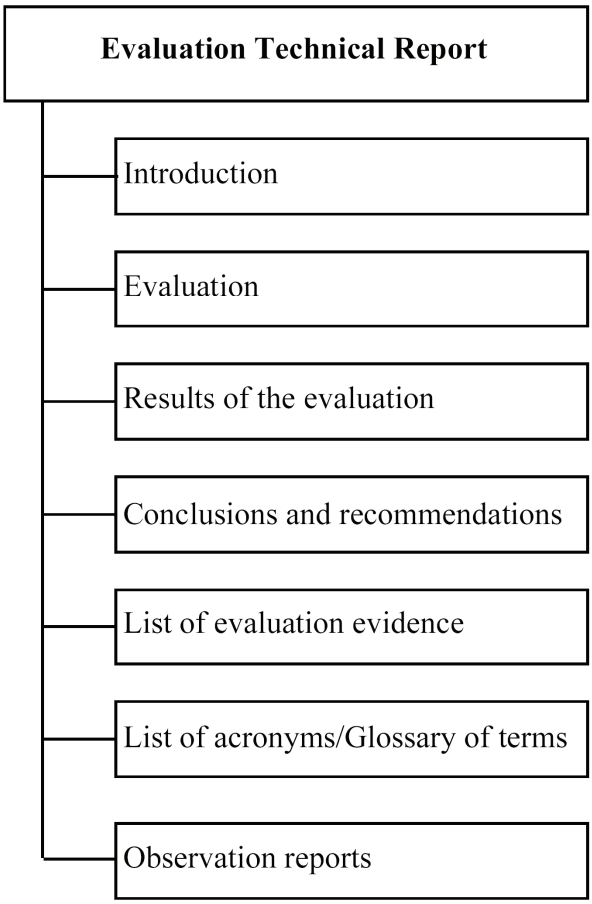

Figure 4 - ETR information content for a PP evaluation

8.5.5.2.1 Introduction

86 The evaluator ***shall report*** evaluation scheme identifiers.

87 Evaluation scheme identifiers (e.g. logos) are the information required to unambiguously identify the scheme responsible for the evaluation oversight.

88 The evaluator ***shall report*** ETR configuration control identifiers.

89 The ETR configuration control identifiers contain information that identifies the ETR (e.g. name, date and version number).

90 The evaluator ***shall report*** PP configuration control identifiers.

91 PP configuration control identifiers (e.g. name, date and version number) are required to identify what is being evaluated in order for the overseer to verify that the verdicts have been assigned correctly by the evaluator.

92 The evaluator ***shall report*** the identity of the developer.

93 The identity of the PP developer is required to identify the party responsible for producing the PP.

94 The evaluator ***shall report*** the identity of the sponsor.

95 The identity of the sponsor is required to identify the party responsible for providing evaluation evidence to the evaluator.

96 The evaluator ***shall report*** the identity of the evaluator.

97 The identity of the evaluator is required to identify the party performing the evaluation and responsible for the evaluation verdicts.

8.5.5.2.2 Evaluation

98 The evaluator ***shall report*** the evaluation methods, techniques, tools and standards used.

99 The evaluator references the evaluation criteria, methodology and interpretations used to evaluate the PP.

100 The evaluator ***shall report*** any constraints on the evaluation, constraints on the handling of evaluation results and assumptions made during the evaluation that have an impact on the evaluation results.

101 The evaluator may include information in relation to legal or statutory aspects, organisation, confidentiality, etc.

8.5.5.2.3 Results of the evaluation

102 The evaluator ***shall report*** a verdict and a supporting rationale for each assurance component that constitutes an APE activity, as a result of performing the corresponding CEM action and its constituent work units.

103 The rationale justifies the verdict using the CC, the CEM, any interpretations and the evaluation evidence examined and shows how the evaluation evidence does or does not meet each aspect of the criteria. It contains a description of the work performed, the method used, and any derivation of results. The rationale may provide detail to the level of a CEM work unit.

8.5.5.2.4 Conclusions and recommendations

104 The evaluator ***shall report*** the conclusions of the evaluation, in particular the overall verdict as defined in CC Part 1 Chapter 9, Evaluation results, and determined by application of the verdict assignment described in 8.2.5.

105 The evaluator provides recommendations that may be useful for the overseer. These recommendations may include shortcomings of the PP discovered during the evaluation or mention of features which are particularly useful.

8.5.5.2.5 List of evaluation evidence

106 The evaluator ***shall report*** for each item of evaluation evidence the following information:

- the issuing body (e.g. the developer, the sponsor);
- the title;
- the unique reference (e.g. issue date and version number).

8.5.5.2.6 List of acronyms/Glossary of terms

107 The evaluator ***shall report*** any acronyms or abbreviations used in the ETR.

108 Glossary definitions already defined by the CC or CEM need not be repeated in the ETR.

8.5.5.2.7 Observation reports

109 The evaluator ***shall report*** a complete list that uniquely identifies the ORs raised during the evaluation and their status.

110 For each OR, the list should contain its identifier as well as its title or a brief summary of its content.

8.5.5.3 ETR for a TOE Evaluation

111 This section describes the minimum content of the ETR for a TOE evaluation. The contents of the ETR are portrayed in Figure 5; this figure may be used as a guide when constructing the structural outline of the ETR document.

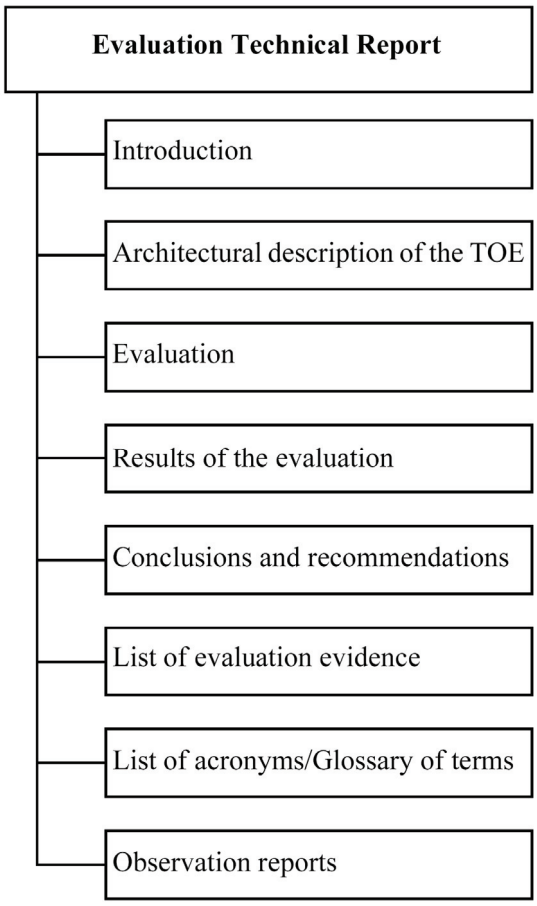

Figure 5 - ETR information content for a TOE evaluation

8.5.5.3.1 Introduction

112 The evaluator ***shall report*** evaluation scheme identifiers.

113 Evaluation scheme identifiers (e.g. logos) are the information required to unambiguously identify the scheme responsible for the evaluation oversight.

114 The evaluator ***shall report*** ETR configuration control identifiers.

115 The ETR configuration control identifiers contain information that identifies the ETR (e.g. name, date and version number).

116 The evaluator ***shall report*** ST and TOE configuration control identifiers.

117 ST and TOE configuration control identifiers identify what is being evaluated in order for the overseer to verify that the verdicts have been assigned correctly by the evaluator.

118 If the ST claims that the TOE conforms to the requirements of one or more PPs, the ETR shall report the reference of the corresponding PPs.

119 The PPs reference contains information that uniquely identifies the PPs (e.g. title, date, and version number).

120 The evaluator ***shall report*** the identity of the developer.

121 The identity of the TOE developer is required to identify the party responsible for producing the TOE.

122 The evaluator ***shall report*** the identity of the sponsor.

123 The identity of the sponsor is required to identify the party responsible for providing evaluation evidence to the evaluator.

124 The evaluator ***shall report*** the identity of the evaluator.

125 The identity of the evaluator is required to identify the party performing the evaluation and responsible for the evaluation verdicts.

8.5.5.3.2 Architectural description of the TOE

126 The evaluator ***shall report*** a high level description of the TOE and its major components based on the evaluation evidence described in the CC assurance family entitled TOE design (ADV_TDS), where applicable.

127 The intent of this section is to characterise the degree of architectural separation of the major components. If there is no TOE design (ADV_TDS) requirement in the ST, this is not applicable and is considered to be satisfied.

8.5.5.3.3 Evaluation

128 The evaluator ***shall report*** the evaluation methods, techniques, tools and standards used.

129 The evaluator may reference the evaluation criteria, methodology and interpretations used to evaluate the TOE or the devices used to perform the tests.

130 The evaluator ***shall report*** any constraints on the evaluation, constraints on the distribution of evaluation results and assumptions made during the evaluation that have an impact on the evaluation results.

131 The evaluator may include information in relation to legal or statutory aspects, organisation, confidentiality, etc.

8.5.5.3.4 Results of the evaluation

132 For each activity on which the TOE is evaluated, the evaluator ***shall report***:

- the title of the activity considered;
- a verdict and a supporting rationale for each assurance component that constitutes this activity, as a result of performing the corresponding CEM action and its constituent work units.

133 The rationale justifies the verdict using the CC, the CEM, any interpretations and the evaluation evidence examined and shows how the evaluation evidence does or does not meet each aspect of the criteria. It contains a description of the work performed, the method used, and any derivation of results. The rationale may provide detail to the level of a CEM work unit.

134 The evaluator ***shall report*** all information specifically required by a work unit.

135 For the AVA and ATE activities, work units that identify information to be reported in the ETR have been defined.

8.5.5.3.5 Conclusions and recommendations

136 The evaluator ***shall report*** the conclusions of the evaluation, which will relate to whether the TOE has satisfied its associated ST, in particular the overall verdict as defined in CC Part 1 Chapter 9, Evaluation results, and determined by application of the verdict assignment described in 8.2.5.

137 The evaluator provides recommendations that may be useful for the overseer. These recommendations may include shortcomings of the IT product discovered during the evaluation or mention of features which are particularly useful.

8.5.5.3.6 List of evaluation evidence

138 The evaluator ***shall report*** for each item of evaluation evidence the following information:

– the issuing body (e.g. the developer, the sponsor);

– the title;

– the unique reference (e.g. issue date and version number).

8.5.5.3.7 List of acronyms/Glossary of terms

139 The evaluator ***shall report*** any acronyms or abbreviations used in the ETR.

140 Glossary definitions already defined by the CC or CEM need not be repeated in the ETR.

8.5.5.3.8 Observation reports

141 The evaluator ***shall report*** a complete list that uniquely identifies the ORs raised during the evaluation and their status.

142 For each OR, the list should contain its identifier as well as its title or a brief summary of its content.

9 Class APE: Protection Profile evaluation

9.1 Introduction

143 This Chapter describes the evaluation of a PP. The requirements and methodology for PP evaluation are identical for each PP evaluation, regardless of the EAL (or other set of assurance requirements) that is claimed in the PP. The evaluation methodology in this chapter is based on the requirements on the PP as specified in CC Part 3 class APE.

144 This Chapter should be used in conjunction with Annexes A, B and C in CC Part 1, as these Annexes clarify the concepts here and provide many examples.

9.2 Application notes

9.2.1 Re-using the evaluation results of certified PPs

145 While evaluating a PP that is based on one or more certified PPs, it may be possible to re-use the fact that these PPs were certified. The potential for re-use of the result of a certified PP is greater if the PP under evaluation does not add threats, OSPs, assumptions, security objectives and/or security requirements to those of the PP that compliance is being claimed to.

146 The evaluator is allowed to re-use the PP evaluation results by doing certain analyses only partially or not at all if these analyses or parts thereof were already done as part of the PP evaluation. While doing this, the evaluator should assume that the analyses in the PP were performed correctly.

147 An example would be where the PP that compliance is being claimed to contains a set of security requirements, and these were determined to be internally consistent during its evaluation. If the PP under evaluation uses the exact same requirements, the consistency analysis does not have to be repeated during the ST evaluation. If the PP under evaluation adds one or more requirements, or performs operations on these requirements, the analysis will have to be repeated. However, it may be possible to save work in this consistency analysis by using the fact that the original requirements are internally consistent. If the original requirements are internally consistent, the evaluator only has to determine that:

a) the set of all new and/or changed requirements is internally consistent, and

b) the set of all new and/or changed requirements is consistent with the original requirements.

148 The evaluator notes in the ETR each case where analyses are not done or only partially done for this reason.

9.3 PP introduction (APE_INT)

9.3.1 Evaluation of sub-activity (APE_INT.1)

9.3.1.1 Objectives

149 The objective of this sub-activity is to determine whether the PP is correctly identified, and whether the PP reference and TOE overview are consistent with each other.

9.3.1.2 Input

150 The evaluation evidence for this sub-activity is:

a) the PP.

9.3.1.3 Action APE_INT.1.1E

APE_INT.1.1C ***The PP introduction shall contain a PP reference and a TOE overview.***

APE_INT.1-1 The evaluator ***shall check*** that the PP introduction contains a PP reference and a TOE overview.

APE_INT.1.2C ***The PP reference shall uniquely identify the PP.***

APE_INT.1-2 The evaluator ***shall examine*** the PP reference to determine that it uniquely identifies the PP.

151 The evaluator determines that the PP reference identifies the PP itself, so that it can be easily distinguished from other PPs, and that it also uniquely identifies each version of the PP, e.g. by including a version number and/or a date of publication.

152 The PP should have some referencing system that is capable of supporting unique references (e.g. use of numbers, letters or dates).

APE_INT.1.3C ***The TOE overview shall summarise the usage and major security features of the TOE.***

APE_INT.1-3 The evaluator ***shall examine*** the TOE overview to determine that it describes the usage and major security features of the TOE.

153 The TOE overview should briefly (i.e. several paragraphs) describe the usage and major security features expected of the TOE. The TOE overview should enable consumers and potential TOE developers to quickly determine whether the PP is of interest to them.

154 The evaluator determines that the overview is clear enough for TOE developers and consumers, and sufficient to give them a general understanding of the intended usage and major security features of the TOE.

APE_INT.1.4C ***The TOE overview shall identify the TOE type.***

APE_INT.1-4 The evaluator ***shall check*** that the TOE overview identifies the TOE type.

APE_INT.1.5C ***The TOE overview shall identify any non-TOE hardware/software/firmware available to the TOE.***

APE_INT.1-5 The evaluator ***shall examine*** the TOE overview to determine that it identifies any non-TOE hardware/software/firmware available to the TOE.

155 While some TOEs can run stand-alone, other TOEs (notably software TOEs) need additional hardware, software or firmware to operate. In this section of the PP, the PP author lists all hardware, software, and/or firmware that will be available for the TOE to run on.

156 This identification should be detailed enough for potential consumers and TOE developers to determine whether their TOE can operate with the listed hardware, software and firmware.

9.4 Conformance claims (APE_CCL)

9.4.1 Evaluation of sub-activity (APE_CCL.1)

9.4.1.1 Objectives

157 The objective of this sub-activity is to determine the validity of various conformance claims. These describe how the PP conforms to the CC, other PPs and packages.

9.4.1.2 Input

158 The evaluation evidence for this sub-activity is:

a) the PP;

b) the PP(s) that the PP claims conformance to;

c) the package(s) that the PP claims conformance to.

9.4.1.3 Action APE_CCL.1.1E

APE_CCL.1.1C ***The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the PP claims conformance.***

APE_CCL.1-1 The evaluator ***shall check*** that the conformance claim contains a CC conformance claim that identifies the version of the CC to which the PP claims conformance.

159 The evaluator determines that the CC conformance claim identifies the version of the CC that was used to develop this PP. This should include the version number of the CC and, unless the International English version of the CC was used, the language of the version of the CC that was used.

APE_CCL.1.2C ***The CC conformance claim shall describe the conformance of the PP to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.***

APE_CCL.1-2 The evaluator ***shall check*** that the CC conformance claim states a claim of either CC Part 2 conformant or CC Part 2 extended for the PP.

APE_CCL.1.3C ***The CC conformance claim shall describe the conformance of the PP to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.***

APE_CCL.1-3 The evaluator ***shall check*** that the CC conformance claim states a claim of either CC Part 3 conformant or CC Part 3 extended for the PP.

APE_CCL.1.4C ***The CC conformance claim shall be consistent with the extended components definition.***

APE_CCL.1-4 The evaluator ***shall examine*** the CC conformance claim for CC Part 2 to determine that it is consistent with the extended components definition.

160 If the CC conformance claim contains CC Part 2 conformant, the evaluator determines that the extended components definition does not define functional components.

161 If the CC conformance claim contains CC Part 2 extended, the evaluator determines that the extended components definition defines at least one extended functional component.

APE_CCL.1-5 The evaluator ***shall examine*** the CC conformance claim for CC Part 3 to determine that it is consistent with the extended components definition.

162 If the CC conformance claim contains CC Part 3 conformant, the evaluator determines that the extended components definition does not define assurance components.

163 If the CC conformance claim contains CC Part 3 extended, the evaluator determines that the extended components definition defines at least one extended assurance component.

APE_CCL.1.5C ***The conformance claim shall identify all PPs and security requirement packages to which the PP claims conformance.***

APE_CCL.1-6 The evaluator ***shall check*** that the conformance claim contains a PP claim that identifies all PPs for which the PP claims conformance.

164 If the PP does not claim conformance to another PP, this work unit is not applicable and therefore considered to be satisfied.

165 The evaluator determines that any referenced PPs are unambiguously identified (e.g. by title and version number, or by the identification included in the introduction of that PP).

166 The evaluator is reminded that claims of partial conformance to a PP are not permitted.

APE_CCL.1-7 The evaluator ***shall check*** that the conformance claim contains a package claim that identifies all packages to which the PP claims conformance.

167 If the PP does not claim conformance to a package, this work unit is not applicable and therefore considered to be satisfied.

168 The evaluator determines that any referenced packages are unambiguously identified (e.g. by title and version number, or by the identification included in the introduction of that package).

169 The evaluator is reminded that claims of partial conformance to a package are not permitted.

APE_CCL.1.6C ***The conformance claim shall describe any conformance of the PP to a package as either package-conformant or package-augmented.***

APE_CCL.1-8 The evaluator ***shall check*** that, for each identified package, the conformance claim states a claim of either package-name conformant or package-name augmented.

170 If the PP does not claim conformance to a package, this work unit is not applicable and therefore considered to be satisfied.

171 If the package conformance claim contains package-name conformant, the evaluator determines that:

a) If the package is an assurance package, then the PP contains all SARs included in the package, but no additional SARs.

b) If the package is a functional package, then the PP contains all SFRs included in the package, but no additional SFRs.

172 If the package conformance claim contains package-name augmented, the evaluator determines that:

a) If the package is an assurance package, then the PP contains all SARs included in the package, and at least one additional SAR or at least one SAR that is hierarchical to a SAR in the package.

b) If the package is a functional package, then the PP contains all SFRs included in the package, and at least one additional SFR or at least one SFR that is hierarchical to a SFR in the package.

APE_CCL.1.7C ***The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.***

APE_CCL.1-9 The evaluator ***shall examine*** the conformance claim rationale to determine that the TOE type of the TOE is consistent with all TOE types of the PPs.

173 If the PP does not claim conformance to another PP, this work unit is not applicable and therefore considered to be satisfied.

174 The relation between the types could be simple: a firewall PP claiming conformance to another firewall PP, or more complex: a smartcard PP claiming conformance to a number of other PPs at the same time: a PP for the integrated circuit, a PP for the smartcard OS, and two PPs for two applications on the smartcard.

APE_CCL.1.8C ***The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.***

APE_CCL.1-10 The evaluator ***shall examine*** the conformance claim rationale to determine that it demonstrates that the statement of security problem definition is consistent, as defined by the conformance statement of the PP, with the

statements of security problem definition stated in the PPs for which compliance is being claimed.

175 If the PP under evaluation does not claim conformance with another PP, this work unit is not applicable and therefore considered to be satisfied.

176 If the PP to which conformance is being claimed does not have a statement of security problem definition, this work unit is not applicable and therefore considered to be satisfied.

177 If exact or strict conformance is required by the PP to which compliance is being claimed, no conformance claim rationale is required. Instead, the evaluator determines whether:

a) the statements of security problem definition in the PPs are identical (for exact conformance);

b) the statement of security problem definition in the PP under evaluation is a superset of or identical to the statement of security problem definition in the PP to which conformance is being claimed (for strict conformance).

178 If demonstrable compliance is required by the PP to which conformance is being claimed, the evaluator examines the conformance claim rationale to determine that it demonstrates that the statement of security problem definition of the PP under evaluation is at least equivalent to the statement of security problem definition in the PP to which conformance is being claimed.

APE_CCL.1.9C ***The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.***

APE_CCL.1-11 The evaluator ***shall examine*** the conformance claim rationale to determine that the statement of security objectives is consistent, as defined by the conformance statement of the PPs, with the statement of security objectives in the PPs.

179 If the PP does not claim conformance to another PP, this work unit is not applicable and therefore considered to be satisfied.

180 If exact or strict conformance is required by the PP to which compliance is being claimed, no conformance claim rationale is required. Instead, the evaluator determines whether:

a) the statements of security objectives in the PPs are identical (for exact conformance);

b) the statement of security objectives in the PP under evaluation is a superset of or identical to the statement of security objectives in the PP to which conformance is being claimed (for strict conformance).

181 If demonstrable compliance is required by the PP to which conformance is being claimed, the evaluator examines the conformance claim rationale to determine that it demonstrates that the statement of security objectives of the PP under evaluation is at least equivalent to the statement of security objectives in the PP that conformance is being claimed to.

APE_CCL.1.10C ***The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.***

APE_CCL.1-12 The evaluator ***shall examine*** the PP to determine that it is consistent, as defined by the conformance statement of the PP, with all security requirements in the PPs for which conformance is being claimed.

182 If the PP does not claim conformance to another PP, this work unit is not applicable and therefore considered to be satisfied.

183 If exact or strict conformance is required by the PP to which compliance is being claimed, no conformance claim rationale is required. Instead, the evaluator determines whether:

a) the statements of security requirements in the PPs are identical (for exact conformance);

b) the statement of security requirements in the PP under evaluation is a superset of or identical to the statement of security requirements in the PP to which conformance is being claimed (for strict conformance).

184 If demonstrable compliance is required by the PP to which conformance is being claimed, the evaluator examines the conformance claim rationale to determine that it demonstrates that the statement of security requirements of the PP under evaluation is at least equivalent to the statement of security requirements in the PP to which conformance is being claimed.

185 In all cases, SFRs based on FMI_CHO.1 play a special role, in that if the PP that conformance is being claimed to contains FMI_CHO.1 (which provides a choice between two sets of SFRs), and the PP under evaluation contains either of those two sets of SFRs, this set is considered to be identical to the FMI_CHO.1 requirement for the purpose of this work unit.

APE_CCL.1.11C ***The conformance statement shall describe the conformance required of any PPs/STs to the PP as exact-PP, strict-PP or demonstrable-PP - conformance.***

APE_CCL.1-13 The evaluator ***shall check*** that the PP conformance statement states a claim of exact-PP, strict-PP, demonstrable-PP conformance.

9.5 Security problem definition (APE_SPD)

9.5.1 Evaluation of sub-activity (APE_SPD.1)

9.5.1.1 Objectives

186 The objective of this sub-activity is to determine that the security problem intended to be addressed by the TOE, its operational environment, and its development environment, is clearly defined.

9.5.1.2 Input

187 The evaluation evidence for this sub-activity is:

a) the PP.

9.5.1.3 Action APE_SPD.1.1E

APE_SPD.1.1C ***The security problem definition shall describe the threats.***

APE_SPD.1-1 The evaluator ***shall check*** that the security problem definition describes the threats.

188 If all security objectives are derived from assumptions and/or OSPs only, the statement of threats need not be present in the PP. In this case, this work unit is not applicable and therefore considered to be satisfied.

189 The evaluator determines that the security problem definition describes the threats that must be countered by the TOE, its development environment, its operational environment or combinations of these three.

APE_SPD.1.2C ***All threats shall be described in terms of a threat agent, an asset, and an adverse action.***

APE_SPD.1-2 The evaluator ***shall examine*** the security problem definition to determine that all threats are described in terms of a threat agent, an asset, and an adverse action.

190 If all security objectives are derived from assumptions and OSPs only, the statement of threats need not be present in the PP. In this case, this work unit is not applicable and therefore considered to be satisfied.

191 Threat agents may be further described by aspects such as expertise, resource, opportunity, and motivation.

APE_SPD.1.3C ***The security problem definition shall describe the OSPs.***

APE_SPD.1-3 The evaluator ***shall check*** that the security problem definition describes the OSPs.

192 If all security objectives are derived from assumptions and/or threats only, OSPs need not be present in the PP. In this case, this work unit is not applicable and therefore considered to be satisfied.

193 The evaluator determines that OSP statements are made in terms of rules, practises or guidelines that must be followed by the TOE, its development environment, its operational environment or combinations of these three.

194 The evaluator determines that each OSP is explained and/or interpreted in sufficient detail to make it clearly understandable; a clear presentation of policy statements is necessary to permit tracing security objectives to them.

APE_SPD.1.4C ***The security problem definition shall describe the assumptions about the operational environment of the TOE.***

APE_SPD.1-4 The evaluator ***shall examine*** the security problem definition to determine that it describes the assumptions about the operational environment of the TOE.

195 If the threats and/or OSPs already sufficiently address the physical, personnel, and connectivity aspects of the TOE, this work unit is not applicable and is therefore considered to be satisfied.

196 The evaluator determines that each assumption about the operational environment of the TOE is explained in sufficient detail to enable consumers to determine that their operational environment matches the assumption. If the assumptions are not clearly understood, the end result may be that the TOE is used in an operational environment in which it will not function in a secure manner.

9.6 Security objectives (APE_OBJ)

9.6.1 Evaluation of sub-activity (APE_OBJ.1)

9.6.1.1 Objectives

197 The objective of this sub-activity is to determine whether the security objectives for the operational environment are clearly defined and internally consistent.

9.6.1.2 Input

198 The evaluation evidence for this sub-activity is:

a) the PP.

9.6.1.3 Action APE_OBJ.1.1E

APE_OBJ.1.1C ***The statement of security objectives shall describe the security objectives for the operational environment.***

APE_OBJ.1-1 The evaluator ***shall check*** that the statement of security objectives defines the security objectives for the operational environment.

199 The evaluator checks that the security objectives for the operational environment are identified.

9.6.2 Evaluation of sub-activity (APE_OBJ.2)

9.6.2.1 Objectives

200 The objective of this sub-activity is to determine whether the security objectives adequately and completely address the security problem definition, that the division of this problem between the TOE, its development environment, and its operational environment is clearly defined, and whether the security objectives are internally consistent.

9.6.2.2 Input

201 The evaluation evidence for this sub-activity is:

a) the PP.

9.6.2.3 Action APE_OBJ.2.1E

APE_OBJ.2.1C ***The statement of security objectives shall describe the security objectives for the TOE, the security objectives for the development environment and the security objectives for the operational environment.***

APE_OBJ.2-1 The evaluator ***shall check*** that the statement of security objectives defines the security objectives for the TOE, the security objectives for the development environment and the security requirement.

202 The evaluator checks that each category of security objectives is clearly identified and separated from the other categories.

APE_OBJ.2.2C ***The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.***

APE_OBJ.2-2 The evaluator ***shall check*** that the security objectives rationale traces all security objectives for the TOE back to threats countered by the objectives and/or OSPS enforced by the objectives.

203 Each security objective for the TOE may trace back to threats or OSPs, or a combination of threats and OSPs, but it must trace back to at least one threat or OSP.

204 Failure to trace implies that either the security objectives rationale is incomplete, the security problem definition is incomplete, or the security objective for the TOE has no useful purpose.

APE_OBJ.2.3C ***The security objectives rationale shall trace each security objective for the development environment back to threats countered by that security objective and OSPs enforced by that security objective.***

APE_OBJ.2-3 The evaluator ***shall check*** that the security objectives rationale traces the security objectives for the development environment back to threats countered by that security objective and OSPs enforced by that security objective.

205 Each security objective for the development environment may trace back threats or OSPs, or a combination of threats and OSPs, but it must trace back to at least one threat or OSP.

206 Failure to trace implies that either the security objectives rationale is incomplete, the security problem definition is incomplete, or the security objective for the development environment has no useful purpose.

APE_OBJ.2.4C ***The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.***

APE_OBJ.2-4 The evaluator ***shall check*** that the security objectives rationale traces the security objectives for the operational environment back to threats countered by that security objective, to OSPs enforced by that security objective, and to assumptions upheld by that security objective.

207 Each security objective for the operational environment may trace back to threats, OSPs, assumptions, or a combination of threats, OSPs and/or assumptions, but it must trace back to at least one threat, OSP or assumption.

208 Failure to trace implies that either the security objectives rationale is incomplete, the security problem definition is incomplete, or the security objective for the operational environment has no useful purpose.

APE_OBJ.2.5C ***The security objectives rationale shall demonstrate that the security objectives counter all threats.***

APE_OBJ.2-5 The evaluator ***shall examine*** the security objectives rationale to determine that it justifies for each threat that the security objectives are suitable to counter that threat.

209 If no security objectives trace back to the threat, this work unit fails.

210 The evaluator determines that the justification for a threat shows whether the threat is removed, diminished or mitigated.

211 The evaluator determines that the justification for a threat demonstrates that the security objectives are sufficient: if all security objectives that trace back to the threat are achieved, the threat is removed, sufficiently diminished, or the effects of the threat are sufficiently mitigated.

212 Note that the tracings from security objectives to threats provided in the security objectives rationale may be part of a justification, but do not constitute a justification by themselves. Even in the case that a security objective is merely a statement reflecting the intent to prevent a particular threat from being realised, a justification is required, but this justification could be as minimal as “Security Objective X directly counters Threat Y”.

213 The evaluator also determines that each security objective that traces back to a threat is necessary: when the security objective is achieved it actually contributes to the removal, diminishing or mitigation of that threat.

APE_OBJ.2.6C ***The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.***

APE_OBJ.2-6 The evaluator ***shall examine*** the security objectives rationale to determine that for each OSP it justifies that the security objectives are suitable to enforce that OSP.

214 If no security objectives trace back to the OSP, this work unit fails.

215 The evaluator determines that the justification for an OSP demonstrates that the security objectives are sufficient: if all security objectives that trace back to that OSP are achieved, the OSP is enforced.

216 The evaluator also determines that each security objective that traces back to an OSP is necessary: when the security objective is achieved it actually contributes to the enforcement of the OSP.

217 Note that the tracings from security objectives to OSPs provided in the security objectives rationale may be part of a justification, but do not constitute a justification by themselves. In the case that a security objective is

merely a statement reflecting the intent to enforce a particular OSP, a justification is required, but this justification could be as minimal as "Security Objective X directly enforces OSP Y".

APE_OBJ.2.7C ***The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.***

APE_OBJ.2-7 The evaluator ***shall examine*** the security objectives rationale to determine that for each assumption for the operational environment it contains an appropriate justification that the security objectives for the operational environment are suitable to uphold that assumption.

218 If no security objectives for the operational environment trace back to the assumption, this work unit fails.

219 The evaluator determines that the justification for an assumption about the operational environment of the TOE demonstrates that the security objectives are sufficient: if all security objectives for the operational environment that trace back to that assumption are achieved, the operational environment upholds the assumption.

220 The evaluator also determines that each security objective for the operational environment that traces back to an assumption about the operational environment of the TOE is necessary: when the security objective is achieved it actually contributes to the operational environment upholding the assumption.

221 Note that the tracings from security objectives for the operational environment to assumptions provided in the security objectives rationale may be a part of a justification, but do not constitute a justification by themselves. Even in the case that a security objective of the operational environment is merely a restatement of an assumption, a justification is required, but this justification could be as minimal as "Security Objective X directly upholds Assumption Y".

9.7 Extended components definition (APE_ECD)

9.7.1 Evaluation of sub-activity (APE_ECD.1)

9.7.1.1 Objectives

222 The objective of this sub-activity is to determine whether extended components have been clearly and unambiguously defined, and whether they are necessary, i.e. they could not have been clearly expressed using existing CC Part 2 or CC Part 3 components.

9.7.1.2 Input

223 The evaluation evidence for this sub-activity is:

a) the PP.

9.7.1.3 Action APE_ECD.1.1E

APE_ECD.1.1C ***The statement of security requirements shall identify all extended security requirements.***

APE_ECD.1-1 The evaluator ***shall check*** that all security requirements in the statement of security requirements that are not identified as extended requirements are present in CC Part 2 or in CC Part 3.

APE_ECD.1.2C ***The extended components definition shall define an extended component for each extended security requirement.***

APE_ECD.1-2 The evaluator ***shall check*** that the extended components definition defines an extended component for each extended security requirement.

224 If the PP does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

225 A single extended component may be used to define multiple iterations of an extended security requirement, it is not necessary to repeat this definition for each iteration.

APE_ECD.1.3C ***The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.***

APE_ECD.1-3 The evaluator ***shall examine*** the extended components definition to determine that it describes how each extended component fits into the existing CC components, families, and classes.

226 If the PP does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

227 The evaluator determines that each extended component is either:

a) a member of an existing CC Part 2 or CC Part 3 family, or

b) a member of a new family defined in the PP.

228 If the extended component is a member of an existing CC Part 2 or CC Part 3 family, the evaluator determines that the extended components definition adequately describes why the extended component should be a member of that family and how it relates to other components of that family.

229 If the extended component is a member of a new family defined in the PP, the evaluator confirms that the extended component is not appropriate for an existing family.

230 If the PP defines new families, the evaluator determines that each new family is either:

a) a member of an existing CC Part 2 or CC Part 3 class, or

b) a member of a new class defined in the PP.

231 If the family is a member of an existing CC Part 2 or CC Part 3 class, the evaluator determines that the extended components definition adequately describes why the family should be a member of that class and how it relates to other families in that class.

232 If the family is a member of a new class defined in the PP, the evaluator confirms that the family is not appropriate for an existing class.

APE_ECD.1-4 The evaluator ***shall examine*** the extended components definition to determine that each definition of an extended component identifies all applicable dependencies of that component.

233 If the PP does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

234 The evaluator confirms that no applicable dependencies have been overlooked by the PP author.

APE_ECD.1.4C ***The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.***

APE_ECD.1-5 The evaluator ***shall examine*** the extended components definition to determine that each extended functional component uses the existing CC Part 2 components as a model for presentation.

235 If the PP does not contain extended SFRs, this work unit is not applicable and therefore considered to be satisfied.

236 The evaluator determines that the extended functional component is consistent with CC Part 2 Section 7.1.3, Functional component structure.

237 If the extended functional component uses operations, the evaluator determines that the extended functional component is consistent with CC Part 1 Annex C.4, Operations.

238 If the extended functional component is hierarchical to an existing functional component, the evaluator determines that the extended functional component is consistent with CC Part 2 Section 7.1.4.1, Component changes highlighting.

APE_ECD.1-6 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new functional family uses the existing CC functional families as a model for presentation.

239 If the PP does not define new functional families, this work unit is not applicable and therefore considered to be satisfied.

240 The evaluator determines that all new functional families are defined consistent with CC Part 2 Section 7.1.2, Functional family structure.

APE_ECD.1-7 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new functional class uses the existing CC functional classes as a model for presentation.

241 If the PP does not define new functional classes, this work unit is not applicable and therefore considered to be satisfied.

242 The evaluator determines that all new functional classes are defined consistent with CC Part 2 Section 7.1.1, Functional class structure

APE_ECD.1-8 The evaluator ***shall examine*** the extended components definition to determine that each definition of an extended assurance component uses the existing CC Part 3 components as a model for presentation.

243 If the PP does not contain extended SARs, this work unit is not applicable and therefore considered to be satisfied.

244 The evaluator determines that the extended assurance component definition is consistent with CC Part 3 Section 8.1.3, Assurance component structure.

245 If the extended assurance component uses operations, the evaluator determines that the extended assurance component is consistent with CC Part 1 Annex C.4, Operations.

246 If the extended assurance component is hierarchical to an existing assurance component, the evaluator determines that the extended assurance component is consistent with CC Part 3 Section 8.1.3, Assurance component structure.

APE_ECD.1-9 The evaluator ***shall examine*** the extended components definition to determine that, for each defined extended assurance component, applicable methodology has been provided.

247 If the PP does not contain extended SARs, this work unit is not applicable and therefore considered to be satisfied.

248 The evaluator determines that, for each evaluator action element of each extended SAR, one or more work units is provided and that successfully

performing all work units for a given evaluator action element will demonstrate that the element has been achieved.

APE_ECD.1-10 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new assurance family uses the existing CC assurance families as a model for presentation.

249 If the PP does not define new assurance families, this work unit is not applicable and therefore considered to be satisfied.

250 The evaluator determines that all new assurance families are defined consistent with CC Part 3 Section 8.1.2, Assurance family structure.

APE_ECD.1-11 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new assurance class uses the existing CC assurance classes as a model for presentation.

251 If the PP does not define new assurance classes, this work unit is not applicable and therefore considered to be satisfied.

252 The evaluator determines that all new assurance classes are defined consistent with CC Part 3 Section 8.1.1, Assurance class structure.

APE_ECD.1.5C ***The extended components shall consist of measurable and objective elements such that compliance or noncompliance to these elements can be demonstrated.***

APE_ECD.1-12 The evaluator ***shall examine*** the extended components definition to determine that each element in each extended component is measurable and states objective evaluation requirements, such that compliance or noncompliance can be demonstrated.

253 If the PP does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

254 The evaluator determines that elements of extended functional components are stated in such a way that they are testable, and traceable through the appropriate TSF representations.

255 The evaluator also determines that elements of extended assurance components avoid the need for subjective evaluator judgement.

256 The evaluator is reminded that whilst being measurable and objective is appropriate for all evaluation criteria, it is acknowledged that no formal method exists to prove such properties. Therefore the existing CC functional and assurance components are to be used as a model for determining what constitutes compliance with this requirement.

9.7.1.4 Action APE_ECD.1.2E

APE_ECD.1-13 The evaluator ***shall examine*** the extended components definition to determine that each extended component can not be clearly expressed using existing components.

257 If the PP does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

258 The evaluator should take components from CC Part 2 and CC Part 3, other extended components that have been defined in the PP, combinations of these components, and possible operations on these components into account when making this determination.

259 The evaluator is reminded that the role of this work unit is to preclude unnecessary duplication of components, that is, components that can be clearly expressed by using other components. The evaluator should not undertake an exhaustive search of all possible combinations of components including operations in an attempt to find a way to express the extended component by using existing components.

9.8 Security requirements (APE_REQ)

9.8.1 Evaluation of sub-activity (APE_REQ.1)

9.8.1.1 Objectives

260 The objective of this sub-activity is to determine whether the SFRs and SARs are clear, unambiguous and well-defined and whether they are internally consistent.

9.8.1.2 Input

261 The evaluation evidence for this sub-activity is:

a) the PP.

9.8.1.3 Action APE_REQ.1.1E

APE_REQ.1.1C ***The statement of security requirements shall describe the SFRs and the SARs.***

APE_REQ.1-1 The evaluator ***shall check*** that the statement of security requirements describes the SFRs.

262 The evaluator determines that each SFR is identified by one of the following means:

a) by reference to an individual component in CC Part 2;

b) by reference to an extended component in the extended components definition of the PP;

c) by reference to a PP that the PP claims to be compliant with;

d) by reference to a security requirements package that the PP claims to be compliant with;

e) by reproduction in the PP.

263 It is not required to use the same means of identification for all SFRs.

APE_REQ.1-2 The evaluator ***shall check*** that the statement of security requirements describes the SARs.

264 The evaluator determines that each SAR is identified by one of the following means:

a) by reference to an individual component in CC Part 3;

b) by reference to an extended component in the extended components definition of the PP;

c) by reference to a PP that the PP claims to be compliant with;

d) by reference to a security requirements package that the PP claims to be compliant with;

e) by reproduction in the PP.

265 It is not required to use the same means of identification for all SARs.

APE_REQ.1-3 The evaluator ***shall examine*** the PP to determine that the PP lists and explains all terms that are used in the requirements that are not defined in the CC.

266 The evaluator determines that the PP lists and explains all:

– (types of) subjects and objects that are used in the SFRs

– (types of) security attributes of these subjects and objects (with possible values that these attributes may take)

– (types of) operations that are used in the SFRs

– (types of) users in the SFRs

– other terms that are introduced in the SFRs and/or SARs by completing operations, if these terms are not immediately clear to a wide audience, or are used outside their dictionary definition.

267 The goal of this work unit is to ensure that the SFRs and SARs are well-defined and that no misunderstanding can occur due to the introduction of vague terms. This work unit should not be taken into extremes, by forcing the PP writer to define every single word. The general audience of a set of security requirements can be assumed to have a reasonable knowledge of IT, security and Common Criteria.

268 All of the above may be presented in groups, classes, roles, types or other groupings or characterisations that allow easy understanding.

269 The evaluator is reminded that these lists and definitions do not have to be part of the statement of security requirements, but could be placed (in part or in whole) in different sections. This may be especially applicable if the same terms are used in the rest of the PP.

APE_REQ.1.2C ***The statement of security requirements shall identify all operations on the security requirements.***

APE_REQ.1-4 The evaluator ***shall check*** that the statement of security requirements identifies all operations on the security requirements.

270 The evaluator determines that all operations are identified in each SFR or SAR where such an operation is used. This includes both completed operations and uncompleted operations. Identification can be achieved by typographical distinctions, or by explicit identification in the surrounding text, or by any other distinctive means.

APE_REQ.1.3C ***All operations shall be performed correctly.***

APE_REQ.1-5 The evaluator ***shall examine*** the statement of security requirements to determine that all assignment operations are performed correctly.

271 An assignment operation is only allowed where specifically permitted in a component.

272 The evaluator compares each assignment with the component from which it is derived, to determine that the assignment has been left uncompleted, has been completed, or has been transformed into a selection.

273 If the assignment has been left uncompleted, the evaluator determines that the uncompleted assignment has been correctly copied from the component.

274 If the assignment has been completed with a value, the evaluator determines that the type of the value is consistent with the type required by the assignment.

275 If the assignment has been transformed into a selection, the evaluator determines that each value in the selection is an allowable value of the assignment.

APE_REQ.1-6 The evaluator ***shall examine*** the statement of security requirements to determine that all iteration operations are performed correctly.

276 The evaluator determines that each iteration of a requirement is different from each other iteration of that requirement (at least one element is different), or that the requirement applies to a different part of the TOE.

APE_REQ.1-7 The evaluator ***shall examine*** the statement of security requirements to determine that all selection operations are performed correctly.

277 A selection operation is only allowed where specifically permitted in a component.

278 The evaluator compares each selection with the component from which it is derived, to determine that the selection has been left uncompleted, has been completed, or has been restricted.

279 If the selection has been left uncompleted, the evaluator determines that the uncompleted selection has been correctly copied from the component.

280 If the selection has been completed, the evaluator determines that selected item or items are one or more of the items indicated within the selection portion of the component.

281 If the selection has been restricted, the evaluator determines that the remaining selectable items are a subset of the selectable items in the component.

APE_REQ.1-8 The evaluator ***shall examine*** the statement of security requirements to determine that all refinement operations are performed correctly.

282 The evaluator determines for each refinement that the component is refined in such manner that a TOE meeting the refined requirement also meets the unrefined requirement. If the refined requirement exceeds this boundary it is considered to be an extended requirement.

283 A special case of refinement is an editorial refinement, where a small change is made in a requirement, i.e. rephrasing a sentence due to adherence to proper English grammar. This change is not allowed to modify the meaning of the requirement in any way. The evaluator is reminded that editorial refinements have to be clearly identified.

284 Another special case of refinement is where multiple iterations of the same requirement are used, each with different refinements, where some of the refined iterations do not meet the full scope of the original requirement. This is acceptable, provided that all iterations of the refined requirement, taken collectively, meet the entire scope of the original requirement.

285 In addition, a refinement should be related to the original requirement. Refining an audit requirement with an extra element on prevention of electromagnetic radiation is normally not allowed. This refinement should be added to another requirement or, if no applicable requirement to refine can be found, be formulated as an extended requirement.

APE_REQ.1.4C ***Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.***

APE_REQ.1-9 The evaluator ***shall examine*** the statement of security requirements to determine that each dependency of the security requirements is either satisfied, or that the security requirements rationale justifies the dependency not being satisfied.

286 A dependency is satisfied by the inclusion of the relevant component (or one that is hierarchical to it) within the statement of security requirements. The component used to satisfy the dependency should, if necessary, be modified by operations to ensure that it actually satisfies that dependency.

287 A justification that a dependency is not met can address either:

a) why the dependency is not necessary or useful, in which case no further information is required; or

b) that the dependency has been addressed by the operational environment of the TOE, in which case the justification should describe how the security objectives for the operational environment address this dependency.

9.8.1.4 Action APE_REQ.1.2E

APE_REQ.1-10 The evaluator ***shall examine*** the statement of security requirements to determine that it is internally consistent.

288 The evaluator determines that the combined set of all SFRs and SARs is internally consistent.

289 The evaluator determines that on all occasions where different security requirements apply to the same types of developer evidence, events, operations, data, tests to be performed etc. or to "all objects", "all subjects" etc., that these requirements do not conflict.

290 Some possible conflicts are:

a) an extended SAR specifying that the design of a certain cryptographic algorithm is to be kept secret, and another extended SAR specifying an open source review;

b) FAU_GEN.1 Audit data generation without time specifying that subject identity is to be logged, FDP_ACC.1 Access control specifying who has access to these logs, and FDP_UNO.1 Limited unobservability specifying that some actions of subjects should be unobservable to other subjects. If the subject that should not be able to see an activity can access logs of this activity, these SFRs conflict;

c) FPT_RIP.1 Removal before use specifying deletion of information no longer needed, and FDP_ROL.1 Rollback specifying that a TOE can return to a previous state. If the information that is needed for the rollback to the previous state has been deleted, these requirements conflict;

d) Multiple iterations of FDP_ACC.1 Access control and/or FDP_ACC.2 Access control with automatic modification of security attributes especially where some iterations cover the same subjects, objects, or operations. If one access control policy allows a subject to perform an operation on an object, while another policy does not allow this, these requirements conflict.

9.8.2 Evaluation of sub-activity (APE_REQ.2)

9.8.2.1 Objectives

291 The objective of this sub-activity is to determine whether the SFRs and SARs are clear, unambiguous and well-defined, whether they are internally consistent, and whether they meet the security objectives of the TOE and the security objectives for the development environment.

9.8.2.2 Input

292 The evaluation evidence for this sub-activity is:

a) the PP.

9.8.2.3 Action APE_REQ.2.1E

APE_REQ.2.1C ***The statement of security requirements shall describe the SFRs and the SARs.***

APE_REQ.2-1 The evaluator ***shall check*** that the statement of security requirements describes the SFRs.

293 The evaluator determines that each SFR is identified by one of the following means:

a) by reference to an individual component in CC Part 2;

b) by reference to an extended component in the extended components definition of the PP;

c) by reference to an individual component in a PP that the PP claims to be compliant with;

d) by reference to an individual component in a security requirements package that the PP claims to be compliant with;

e) by reproduction in the PP.

294 It is not required to use the same means of identification for all SFRs.

APE_REQ.2-2 The evaluator ***shall check*** that the statement of security requirements describes the SARs.

295 The evaluator determines that each SAR is identified by one of the following means:

a) by reference to an individual component in CC Part 3;

b) by reference to an extended component in the extended components definition of the PP;

c) by reference to an individual component in a PP that the PP claims to be compliant with;

d) by reference to an individual component in a security requirements package that the PP claims to be compliant with;

e) by reproduction in the PP.

296 It is not required to use the same means of identification for all SARs.

APE_REQ.2-3 The evaluator ***shall examine*** the PP to determine that the PP lists and explains all terms that are used in the requirements that are not defined in the CC.

297 The evaluator determines that the PP lists and explains all:

- (types of) subjects and objects that are used in the SFRs
- (types of) security attributes of these subjects and objects (with possible values that these attributes may take)
- (types of) operations that are used in the SFRs
- (types of) users in the SFRs
- other terms that are introduced in the SFRs and/or SARs by completing operations, if these terms are not immediately clear to a wide audience, or are used outside their dictionary definition.

298 The goal of this work unit is to ensure that the SFRs and SARs are well-defined and that no misunderstanding can occur due to the introduction of vague terms. This work unit should not be taken into extremes, by forcing the PP writer to define every single word. The general audience of a set of security requirements can be assumed to have a reasonable knowledge of IT, security and Common Criteria.

299 All of the above may be presented in groups, classes, roles, types or other groupings or characterisations that allow easy understanding.

300 The evaluator is reminded that these lists and definitions do not have to be part of the statement of security requirements, but could be placed (in part or in whole) in different sections. This may be especially applicable if the same terms are used in the rest of the PP (e.g. security problem definition or security objectives).

APE_REQ.2.2C ***The statement of security requirements shall identify all operations on the security requirements.***

APE_REQ.2-4 The evaluator ***shall check*** that the statement of security requirements identifies all operations on the security requirements.

301 The evaluator determines that all operations are identified in each SFR or SAR where such an operation is used. This includes both completed operations and uncompleted operations. Identification can be achieved by typographical distinctions, or by explicit identification in the surrounding text, or by any other distinctive means.

APE_REQ.2.3C ***All operations shall be performed correctly.***

APE_REQ.2-5 The evaluator ***shall examine*** the statement of security requirements to determine that all assignment operations are performed correctly.

302 An assignment operation is only allowed where specifically permitted in a component.

303 The evaluator compares each assignment with the component from which it is derived, to determine that the assignment has been left uncompleted, has been completed, or has been transformed into a selection.

304 If the assignment has been left uncompleted, the evaluator determines that the uncompleted assignment has been correctly copied from the component.

305 If the assignment has been completed with a value, the evaluator determines that the type of the value is consistent with the type required by the assignment.

306 If the assignment has been transformed into a selection, the evaluator determines that each value in the selection is an allowable value of the assignment.

APE_REQ.2-6 The evaluator ***shall examine*** the statement of security requirements to determine that all iteration operations are performed correctly.

307 The evaluator determines that each iteration of a requirement is different from each other iteration of that requirement (at least one element is different), or that the requirement applies to a different part of the TOE.

APE_REQ.2-7 The evaluator ***shall examine*** the statement of security requirements to determine that all selection operations are performed correctly.

308 A selection operation is only allowed where specifically permitted in a component.

309 The evaluator compares each selection with the component from which it is derived, to determine that the selection has been left uncompleted, has been completed, or has been restricted.

310 If the selection has been left uncompleted, the evaluator determines that the uncompleted selection has been correctly copied from the component.

311 If the selection has been completed, the evaluator determines that selected item or items are one or more of the items indicated within the selection portion of the component.

312 If the selection has been restricted, the evaluator determines that the remaining selectable items are a subset of the selectable items in the component.

APE_REQ.2-8 The evaluator ***shall examine*** the statement of security requirements to determine that all refinement operations are performed correctly.

313 The evaluator determines for each refinement that the component is refined in such manner that a TOE meeting the refined requirement also meets the unrefined requirement. If the refined requirement exceeds this boundary it is considered to be an extended requirement.

314 A special case of refinement is an editorial refinement, where a small change is made in a requirement, i.e. rephrasing a sentence due to adherence to proper English grammar. This change is not allowed to modify the meaning of the requirement in any way. The evaluator is reminded that editorial refinements have to be clearly identified.

315 Another special case of refinement is where multiple iterations of the same requirement are used, each with different refinements, where some of the refined iterations do not meet the full scope of the original requirement. This is acceptable, provided that all iterations of the refined requirement, taken collectively, meet the entire scope of the original requirement.

316 In addition, a refinement should be related to the original requirement. Refining an audit requirement with an extra element on prevention of electromagnetic radiation is normally not allowed. This refinement should be added to another requirement or, if no applicable requirement to refine can be found, be formulated as an extended requirement.

APE_REQ.2.4C ***Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.***

APE_REQ.2-9 The evaluator ***shall examine*** the statement of security requirements to determine that each dependency of the security requirements is either satisfied, or that the security requirements rationale justifies the dependency not being satisfied.

317 A dependency is satisfied by the inclusion of the relevant component (or one that is hierarchical to it) within the statement of security requirements. The component used to satisfy the dependency should, if necessary, be modified by operations to ensure that it actually satisfies that dependency.

318 A justification that a dependency is not met can address either:

a) why the dependency is not necessary or useful, in which case no further information is required; or

b) that the dependency has been addressed by the operational environment of the TOE, in which case the justification should describe how the security objectives for the operational environment address this dependency.

APE_REQ.2.5C ***The security requirements rationale shall trace each SFR back to the security objectives for the TOE.***

APE_REQ.2-10 The evaluator ***shall check*** that the security requirements rationale traces each SFR back to the security objectives for the TOE.

319 The evaluator determines that each SFR is traced back to at least one security objective for the TOE.

320 Failure to trace implies that either the security requirements rationale is incomplete, the security objectives for the TOE are incomplete, or the SFR has no useful purpose.

APE_REQ.2.6C ***The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.***

APE_REQ.2-11 The evaluator ***shall examine*** the security requirements rationale to determine that for each security objective for the TOE it justifies that the SFRs are suitable to meet that security objective for the TOE.

321 If no SFRs trace back to the security objective for the TOE, this work unit fails.

322 The evaluator determines that the justification for a security objective for the TOE demonstrates that the SFRs are sufficient: if all SFRs that trace back to the objective are satisfied, the security objective for the TOE is achieved.

323 If the SFRs that trace back to a security objective for the TOE have any uncompleted assignments, or uncompleted or restricted selections, the evaluator determines that for every conceivable completion or combination of completions of these operations, the security objective is still met.

324 The evaluator also determines that each SFR that traces back to a security objective for the TOE is necessary: when the SFR is satisfied, it actually contributes to achieving the security objective.

325 Note that the tracings from SFRs to security objectives for the TOE provided in the security requirements rationale may be a part of the justification, but do not constitute a justification by themselves.

326 While examining the justification, the evaluator takes into account that, unless the security objectives of the operational environment preclude this, TOEs may be physically attacked, or repeatedly attacked without this being detected, which may prevent the SFRs to achieve the security objectives.

APE_REQ.2.7C ***The security requirements rationale shall trace each SAR back to the security objectives for the development environment.***

APE_REQ.2-12 The evaluator ***shall check*** that the security requirements rationale traces each SAR back to the security objectives for the development environment.

327 The evaluator determines that each SAR is traced back to at least one security objective for the development environment.

328 Failure to trace implies that either the security requirements rationale is incomplete, the security objectives for the development environment are incomplete, or the SAR has no useful purpose.

APE_REQ.2.8C ***The security requirements rationale shall demonstrate that the SARs meet all security objectives for the development environment.***

APE_REQ.2-13 The evaluator ***shall examine*** the security requirements rationale to determine that for each security objective for the development environment it justifies that the SARs are suitable to meet that security objective for the development environment.

329 If no SARs trace back to the security objective for the development environment, this work unit fails.

330 The evaluator determines that the justification for a security objective for the development environment demonstrates that the SARs are sufficient: if all SARs that trace back to the objective are satisfied, the security objective for the development environment is achieved.

331 If the SARs that trace back to a security objective for the development environment have any uncompleted assignments, or uncompleted or restricted selections, the evaluator determines that for every conceivable completion or combination of completions of these operations, the security objective is still met.

332 The evaluator also determines that each SAR that traces back to a security objective for the development environment is necessary, when the SAR is satisfied, it actually contributes to achieving the security objective.

333 Note that the tracings from SARs to security objectives for the development environment provided in the security requirements rationale may be a part of the justification, but do not constitute a justification by themselves.

9.8.2.4 Action APE_REQ.2.2E

APE_REQ.2-14 The evaluator ***shall examine*** the statement of security requirements to determine that it is internally consistent.

334 The evaluator determines that the combined set of all SFRs and SARs is internally consistent.

335 The evaluator determines that on all occasions where different security requirements apply to the same types of developer evidence, events, operations, data, tests to be performed etc. or to “all objects”, “all subjects” etc., that these requirements do not conflict.

336 Some possible conflicts are:

a) an extended SAR specifying that the design of a certain cryptographic algorithm is to be kept secret, and another extended SAR specifying an open source review;

b) FAU_GEN.1 Audit data generation without time specifying that subject identity is to be logged, FDP_ACC.1 Access control specifying who has access to these logs, and FDP_UNO.1 Limited unobservability specifying that some actions of subjects should be unobservable to other subjects. If the subject that should not be able to see an activity can access logs of this activity, these SFRs conflict;

c) FPT_RIP.1 Removal before use specifying deletion of information no longer needed, and FDP_ROL.1 Rollback specifying that a TOE can return to a previous state. If the information that is needed for the rollback to the previous state has been deleted, these requirements conflict;

d) Multiple iterations of FDP_ACC.1 Access control and/or FDP_ACC.2 Access control with automatic modification of security attributes especially where some iterations cover the same subjects, objects, or operations. If one access control policy allows a subject to perform an operation on an object, while another policy does not allow this, these requirements conflict.

10 Class ASE: Security Target evaluation

10.1 Introduction

337 This Chapter describes the evaluation of an ST. The ST evaluation should be started prior to any TOE evaluation sub-activities since the ST provides the basis and context to perform these sub-activities. The evaluation methodology in this section is based on the requirements on the ST as specified in CC Part 3 class ASE.

338 This Chapter should be used in conjunction with Annexes A, B and C in CC Part 1, as these Annexes clarify the concepts here and provide many examples.

10.2 Application notes

10.2.1 Re-using the evaluation results of certified PPs

339 While evaluating an ST that is based on one or more certified PPs, it may be possible to re-use the fact that these PPs were certified. The potential for re-use of the result of a certified PP is greater if the ST does not add threats, OSPs, assumptions, security objectives and/or security requirements to those of the PP.

340 The evaluator is allowed to re-use the PP evaluation results by doing certain analyses only partially or not at all if these analyses or parts thereof were already done as part of the PP evaluation. While doing this, the evaluator should assume that the analyses in the PP were performed correctly.

341 An example would be where the PP contains a set of security requirements, and these were determined to be internally consistent during the PP evaluation. If the ST uses the exact same requirements, the consistency analysis does not have to be repeated during the ST evaluation. If the ST adds one or more requirements, or performs operations on these requirements, the analysis will have to be repeated. However, it may be possible to save work in this consistency analysis by using the fact that the original requirements are internally consistent. If the original requirements are internally consistent, the evaluator only has to determine that:

a) the set of all new and/or changed requirements is internally consistent, and

b) the set of all new and/or changed requirements is consistent with the original requirements.

342 The evaluator notes in the ETR each case where analyses are not done or only partially done for this reason.

10.3 ST introduction (ASE_INT)

10.3.1 Evaluation of sub-activity (ASE_INT.1)

10.3.1.1 Objectives

343 The objective of this sub-activity is to determine whether the ST and the TOE are correctly identified, whether the TOE is correctly described in a narrative way at three levels of abstraction (TOE reference, TOE overview and TOE description), and whether these three descriptions are consistent with each other.

10.3.1.2 Input

344 The evaluation evidence for this sub-activity is:

a) the ST.

10.3.1.3 Action ASE_INT.1.1E

ASE_INT.1.1C ***The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE description.***

ASE_INT.1-1 The evaluator ***shall check*** that the ST introduction contains an ST reference, a TOE reference, a TOE overview and a TOE description.

ASE_INT.1.2C ***The ST reference shall uniquely identify the ST.***

ASE_INT.1-2 The evaluator ***shall examine*** the ST reference to determine that it uniquely identifies the ST.

345 The evaluator determines that the ST reference identifies the ST itself, so that it can be easily distinguished from other STs, and that it also uniquely identifies each version of the ST, e.g. by including a version number and/or a date of publication.

346 In evaluations where a CM system is provided, the evaluator could validate the uniqueness of the reference by checking the configuration list. In the other cases, the ST should have some referencing system that is capable of supporting unique references (e.g. use of numbers, letters or dates).

ASE_INT.1.3C ***The TOE reference shall identify the TOE.***

ASE_INT.1-3 The evaluator ***shall examine*** the TOE reference to determine that it identifies the TOE.

347 The evaluator determines that the TOE reference identifies the TOE, so that it is clear to which TOE the ST refers, and that it also identifies the version of the TOE, e.g. by including a version/release/build number, or a date of release.

ASE_INT.1-4 The evaluator ***shall examine*** the TOE reference to determine that it is not misleading.

348 If the TOE is related to one or more well-known products, it is allowed to reflect this in the TOE reference. However, this should not be used to mislead consumers: situations where only a small part of a product is evaluated, yet the TOE reference does not reflect this, are not allowed.

ASE_INT.1.4C ***The TOE overview shall summarise the usage and major security features of the TOE.***

ASE_INT.1-5 The evaluator ***shall examine*** the TOE overview to determine that it describes the usage and major security features of the TOE.

349 The TOE overview should briefly (i.e. several paragraphs) describe the usage and major security features of the TOE. The TOE overview should enable potential consumers to quickly determine whether the TOE may be suitable for their security needs.

350 The TOE overview in an ST for a composed TOE should describe the usage and major security feature of the composed TOE, rather than those of the individual component TOEs.

351 The evaluator determines that the overview is clear enough for consumers, and sufficient to give them a general understanding of the intended usage and major security features of the TOE.

ASE_INT.1.5C ***The TOE overview shall identify the TOE type.***

ASE_INT.1-6 The evaluator ***shall check*** that the TOE overview identifies the TOE type.

ASE_INT.1-7 The evaluator ***shall examine*** the TOE overview to determine that the TOE type is not misleading.

352 There are situations where the general consumer would expect certain functionality of the TOE because of its TOE type. If this functionality is absent in the TOE, the evaluator determines that the TOE overview adequately discusses this absence.

353 There are also TOEs where the general consumer would expect that the TOE should be able to operate in a certain operational environment because of its TOE type. If the TOE can not operate in such an operational environment, the evaluator determines that the TOE overview adequately discusses this.

ASE_INT.1.6C ***The TOE overview shall identify any non-TOE hardware/software/firmware required by the TOE.***

ASE_INT.1-8 The evaluator ***shall examine*** the TOE overview to determine that it identifies any non-TOE hardware/software/firmware required by the TOE.

354 While some TOEs can run stand-alone, other TOEs (notably software TOEs) need additional hardware, software or firmware to operate. If the TOE does

not require any hardware, software or firmware, this work unit is not applicable and therefore considered to be satisfied.

355 The evaluator determines that the TOE overview identifies any additional hardware, software and firmware needed by the TOE to operate. This identification does not have to be exhaustive, but detailed enough for potential consumers of the TOE to determine whether their current hardware, software and firmware support use of the TOE, and, if this is not the case, which additional hardware, software and/or firmware is needed.

ASE_INT.1.7C ***The TOE description shall describe the physical scope and boundaries of the TOE.***

ASE_INT.1-9 The evaluator ***shall examine*** the TOE description to determine that it describes the physical scope and boundaries of the TOE.

356 The evaluator determines that the TOE description discusses the hardware, firmware and software components and/or modules that constitute the TOE at a level of detail that is sufficient to give the reader a general understanding of those components and/or modules.

357 The evaluator also determines that the TOE description lists all guidance that is part of the TOE.

358 The evaluator also determines that the TOE description describes exactly where the physical boundary lies between the TOE hardware/software/firmware and any non-TOE hardware/software/firmware.

359 The evaluator also determines that the TOE description describes exactly where the boundary lies between the TOE guidance and any non-TOE guidance.

ASE_INT.1.8C ***The TOE description shall describe the logical scope and boundaries of the TOE.***

ASE_INT.1-10 The evaluator ***shall examine*** the TOE description to determine that it describes the logical scope and boundaries of the TOE.

360 The evaluator determines that the TOE description discusses the logical security features offered by the TOE at a level of detail that is sufficient to give the reader a general understanding of those features.

361 The evaluator also determines that the TOE description describes exactly where the logical boundary lies between functionality provided by the TOE, and functionality provided by any non-TOE hardware/software/firmware.

362 An ST for a composed TOE can refer out to the description of the logical scope and boundaries of the component TOEs, provided in the component TOE STs to provide the majority of this description for the composed TOE. However, the evaluator determines that the composed TOE ST clearly discusses which boundaries of the individual components are now within the composed TOE, and therefore not a boundary of the composed TOE. E.g. for

a software dependent component TOE a boundary may lie between the TOE and the underlying operating system. When the application is integrated with an operating system in the composed TOE, the boundary between the application and operating system becomes internal and does not form part of the composed TOE logical boundary.

10.3.1.4 Action ASE_INT.1.2E

ASE_INT.1-11 The evaluator ***shall examine*** the TOE reference, TOE overview and TOE description to determine that they are consistent with each other.

10.4 Conformance claims (ASE_CCL)

10.4.1 Evaluation of sub-activity (ASE_CCL.1)

10.4.1.1 Objectives

363 The objective of this sub-activity is to determine the validity of various conformance claims. These describe how the ST and the TOE conform to the CC and how the ST conforms to PPs and packages.

10.4.1.2 Input

364 The evaluation evidence for this sub-activity is:

a) the ST;

b) the PP(s) that the ST claims conformance to;

c) the package(s) that the ST claims conformance to.

10.4.1.3 Action ASE_CCL.1.1E

ASE_CCL.1.1C ***The conformance claim shall contain a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.***

ASE_CCL.1-1 The evaluator ***shall check*** that the conformance claim contains a CC conformance claim that identifies the version of the CC to which the ST and the TOE claim conformance.

365 The evaluator determines that the CC conformance claim identifies the version of the CC that was used to develop this ST. This should include the version number of the CC and, unless the International English version of the CC was used, the language of the version of the CC that was used.

366 For a composed TOE, the evaluator will consider any differences between the version of the CC claimed for a component and the version of the CC claimed for the composed TOE. If the versions differ the evaluator will assess whether the differences between the versions will lead to conflicting claims.

367 For instances where the CC conformance claims for the base TOE and dependent TOE are for different major releases of the CC (e.g. one component TOE conformance claim is CC v2.x and the other component TOE conformance claim is CC v3.x), the conformance claim for the composed TOE will be the earlier release of the CC, as the CC is developed with an aim to provide backwards compatibility (although this may not be achieved in the strictest sense, it is understood to be achieved in principle).

ASE_CCL.1.2C ***The CC conformance claim shall describe the conformance of the ST to CC Part 2 as either CC Part 2 conformant or CC Part 2 extended.***

ASE_CCL.1-2 The evaluator ***shall check*** that the CC conformance claim states a claim of either CC Part 2 conformant or CC Part 2 extended for the ST.

368 For a composed TOE, the evaluator will consider whether this claim is consistent not only with the CC Part 2, but also with the claims of conformance to CC Part 2 by each of the component TOEs. i.e. if one or more component TOEs claims to be CC Part 2 extended, then the composed TOE should also claim to be CC Part 2 extended.

369 The CC conformance claim for the composed TOE may be CC Part 2 extended, even though the component TOEs are Part 2 conformant, in the event that additional SFRs are claimed for the base TOE (see composed TOE guidance for ASE_CCL.1.6C)

ASE_CCL.1.3C ***The CC conformance claim shall describe the conformance of the ST to CC Part 3 as either CC Part 3 conformant or CC Part 3 extended.***

ASE_CCL.1-3 The evaluator ***shall check*** that the CC conformance claim states a claim of either CC Part 3 conformant or CC Part 3 extended for the ST.

ASE_CCL.1.4C ***The CC conformance claim shall be consistent with the extended components definition.***

ASE_CCL.1-4 The evaluator ***shall examine*** the CC conformance claim for CC Part 2 to determine that it is consistent with the extended components definition.

370 If the CC conformance claim contains CC Part 2 conformant, the evaluator determines that the extended components definition does not define functional components.

371 If the CC conformance claim contains CC Part 2 extended, the evaluator determines that the extended components definition defines at least one extended functional component.

ASE_CCL.1-5 The evaluator ***shall examine*** the CC conformance claim for CC Part 3 to determine that it is consistent with the extended components definition.

372 If the CC conformance claim contains CC Part 3 conformant, the evaluator determines that the extended components definition does not define assurance components.

373 If the CC conformance claim contains CC Part 3 extended, the evaluator determines that the extended components definition defines at least one extended assurance component.

ASE_CCL.1.5C ***The conformance claim shall identify all PPs and security requirement packages to which the ST claims conformance.***

ASE_CCL.1-6 The evaluator ***shall check*** that the conformance claim contains a PP claim that identifies all PPs for which the ST claims conformance.

374 If the ST does not claim conformance to a PP, this work unit is not applicable and therefore considered to be satisfied.

375 The evaluator determines that any referenced PPs are unambiguously identified (e.g. by title and version number, or by the identification included in the introduction of that PP).

376 The evaluator is reminded that claims of partial conformance to a PP are not permitted. Therefore, conformance to a PP requiring a composite solution can be claimed in an ST for a composed TOE. Conformance to such a PP would not have been possible during the evaluation of the component TOEs, as these components would not have satisfied the composed solution. This is only possible in the instances where the "composite" PP permits use of the composition evaluation approach (use of ACO components).

377 The ST for a composed TOE will identify the STs of the component TOEs from which the composed ST is comprised. The composed TOE is essentially claiming conformance to the STs of the component TOEs.

ASE_CCL.1-7 The evaluator ***shall check*** that the conformance claim contains a package claim that identifies all packages to which the ST claims conformance.

378 If the ST does not claim conformance to a package, this work unit is not applicable and therefore considered to be satisfied.

379 The evaluator determines that any referenced packages are unambiguously identified (e.g. by title and version number, or by the identification included in the introduction of that package).

380 The evaluator determines that the component TOE STs from which the composed TOE is derived are also unambiguously identified.

381 The evaluator is reminded that claims of partial conformance to a package are not permitted.

ASE_CCL.1.6C ***The conformance claim shall describe any conformance of the ST to a package as either package-conformant or package-augmented.***

ASE_CCL.1-8 The evaluator ***shall check*** that, for each identified package, the conformance claim states a claim of either package-name conformant or package-name augmented.

382 If the ST does not claim conformance to a package, this work unit is not applicable and therefore considered to be satisfied.

383 If the package conformance claim contains package-name conformant, the evaluator determines that:

a) If the package is an assurance package, then the ST contains all SARs included in the package, but no additional SARs.

b) If the package is a functional package, then the ST contains all SFRs included in the package, but no additional SFRs.

384 If the package conformance claim contains package-name augmented, the evaluator determines that:

a) If the package is an assurance package then the ST contains all SARs included in the package, and at least one additional SAR or at least one SAR that is hierarchical to a SAR in the package.

b) If the package is a functional package, then the ST contains all SFRs included in the package, and at least one additional SFR or at least one SFR that is hierarchical to a SFR in the package.

385 For a composed TOE, the evaluator will consider whether any additional SFRs have been added to the base TOE. This may be necessary as a result of dependencies from the dependent TOE on the base TOE (as considered to be the IT environment during the dependent TOE evaluation). In this case the conformance claim will be ST-augmented. If the conformance claim is ST-augmented, it should be made clear in the composed TOE ST which SFRs are only included as a result of composition, and were not examined as SFRs in the dependent TOE (e.g. EALx) evaluation.

ASE_CCL.1.7C ***The conformance claim rationale shall demonstrate that the TOE type is consistent with the TOE type in the PPs for which conformance is being claimed.***

ASE_CCL.1-9 The evaluator ***shall examine*** the conformance claim rationale to determine that the TOE type of the TOE is consistent with all TOE types of the PPs.

386 If the ST does not claim conformance to a PP, this work unit is not applicable and therefore considered to be satisfied.

387 The relation between the types could be simple: a firewall ST claiming conformance to a firewall PP, or more complex: a smartcard ST claiming conformance to a number of PPs at the same time (a PP for the integrated circuit, a PP for the smartcard OS, and two PPs for two applications on the smartcard).

388 For a composed TOE, the evaluator will determine whether the conformance claim rationale demonstrates that the TOE types of the component TOEs are consistent with the composed TOE type. This does not mean that both the component and the composed TOE types have to be the same, but rather that the component TOEs are suitable for integration to provide the composed TOE.

ASE_CCL.1.8C ***The conformance claim rationale shall demonstrate that the statement of the security problem definition is consistent with the statement of the security problem definition in the PPs for which conformance is being claimed.***

ASE_CCL.1-10 The evaluator ***shall examine*** the conformance claim rationale to determine that it demonstrates that the statement of security problem definition is consistent, as defined by the conformance statement of the ST, with the statements of security problem definition stated in the PPs to which compliance is being claimed.

389 If the ST does not claim conformance with a PP, this work unit is not applicable and therefore considered to be satisfied.

390 If the PP does not have a statement of security problem definition, this work unit is not applicable and therefore considered to be satisfied.

391 If exact or strict conformance is required by the PP to which compliance is being claimed no conformance claim rationale is required. Instead, the evaluator determines whether:

a) the statements of security problem definition in the PP and ST are identical (for exact conformance);

b) the statement of security problem definition in the ST is a superset of or identical to the statement of security problem definition in the PP (for strict conformance).

392 If demonstrable compliance is required by the PP, the evaluator examines the conformance claim rationale to determine that it demonstrates that the statement of security problem definition of the ST is at least equivalent to the statement of security problem definition in the PP.

393 For a composed TOE, the evaluator will consider whether the security problem definition of the composed TOE is consistent with that specified in the STs for the component TOEs. This is determined in terms of demonstrable compliance. In particular, the evaluator examines the conformance claim rationale to determine that:

a) Threat statements and OSPs in the composed TOE ST do not contradict those from the component STs.

b) Any assumptions made in the component STs are upheld in the composed TOE ST. That is, either the assumption should also be present in the composed ST, or the assumption should be positively addressed in the composed ST. The assumption could be positively addressed through specification of requirements in the composed TOE to provide functionality fulfilling the concern captured in the assumption.

ASE_CCL.1.9C ***The conformance claim rationale shall demonstrate that the statement of security objectives is consistent with the statement of security objectives in the PPs for which conformance is being claimed.***

ASE_CCL.1-11 The evaluator ***shall examine*** the conformance claim rationale to determine that the statement of security objectives is consistent, as defined by the

conformance statement of the PP, with the statement of security objectives in the PPs.

394 If the ST does not claim conformance to a PP, this work unit is not applicable and therefore considered to be satisfied.

395 If exact or strict conformance is required by the PP, no conformance claim rationale is required. Instead, the evaluator determines whether:

a) the statements of security objectives in the PP and ST are identical (for exact conformance);

b) the statement of security objectives in the ST is a superset of or identical to the statement of security objectives in the PP (for strict conformance).

396 For a composed TOE, the evaluator will consider whether the security objectives of the composed TOE are consistent with that specified in the STs for the component TOEs. This is determined in terms of demonstrable compliance. In particular, the evaluator examines the conformance claim rationale to determine that:

a) The statement of security objectives in the dependent TOE ST relevant to any IT in the operational environment are consistent with the statement of security objectives for the TOE in the base TOE ST. It is not expected that the statement of security objectives for the environment within in the dependent TOE ST will cover all aspects of the statement of security objectives for the TOE in the base TOE ST.

b) The statement of security objectives in the composed ST is consistent with the statements of security objectives in the STs for the component TOEs.

397 If demonstrable compliance is required by the PP, the evaluator examines the conformance claim rationale to determine that it demonstrates that the statement of security objectives of the ST is at least equivalent to the statement of security objectives in the PP, or component TOE ST in the case of a composed TOE ST.

ASE_CCL.1.10C ***The conformance claim rationale shall demonstrate that the statement of security requirements is consistent with the statement of security requirements in the PPs for which conformance is being claimed.***

ASE_CCL.1-12 The evaluator ***shall examine*** the ST to determine that it is consistent, as defined by the conformance statement of the PP, with all security requirements in the PPs for which conformance is being claimed.

398 If the ST does not claim conformance to a PP, this work unit is not applicable and therefore considered to be satisfied.

399 If exact or strict conformance is required by the PP to which compliance is being claimed, no conformance claim rationale is required. Instead, the evaluator determines whether:

a) the statements of security requirements in the PP and ST are identical (for exact conformance);

b) the statement of security requirements in the ST is a superset of or identical to the statement of security requirements in the PP (for strict conformance).

400 For a composed TOE, the evaluator will consider whether the security requirements of the composed TOE are consistent with that specified in the STs for the component TOEs. This is determined in terms of demonstrable compliance. In particular, the evaluator examines the conformance rationale to determine that:

a) The statement of security requirements in the dependent TOE ST relevant to any IT in the operational environment is consistent with the statement of security requirements for the TOE in the base TOE ST. It is not expected that the statement of security requirements for the environment within in the dependent TOE ST will cover all aspects of the statement of security requirements for the TOE in the base TOE ST.

b) The statement of security objectives in the dependent TOE ST relevant to any IT in the operational environment is consistent with the statement of security requirements for the TOE in the base TOE ST. It is not expected that the statement of security objectives for the environment within in the dependent TOE ST will cover all aspects of the statement of security requirements for the TOE in the base TOE ST.

c) The statement of security requirements in the composed is consistent with the statements of security requirements in the STs for the component TOEs.

401 If demonstrable compliance is required by the PP to which conformance is being claimed, the evaluator examines the conformance claim rationale to determine that it demonstrates that the statement of security requirements of the ST is at least equivalent to the statement of security requirements in the PP, or component TOE ST in the case of a composed TOE ST.

402 In all cases, SFRs based on FMI_CHO.1 play a special role, in that if the PP contains FMI_CHO.1 (which provides a choice between two sets of SFRs), and the ST contains either of those two sets of SFRs, this set is considered to be identical to the FMI_CHO.1 requirement for the purpose of this work unit.

10.5 Security problem definition (ASE_SPD)

10.5.1 Evaluation of sub-activity (ASE_SPD.1)

10.5.1.1 Objectives

403 The objective of this sub-activity is to determine that the security problem intended to be addressed by the TOE, its operational environment, and its development environment, is clearly defined.

10.5.1.2 Input

404 The evaluation evidence for this sub-activity is:

a) the ST.

10.5.1.3 Action ASE_SPD.1.1E

ASE_SPD.1.1C ***The security problem definition shall describe the threats.***

ASE_SPD.1-1 The evaluator ***shall check*** that the security problem definition describes the threats.

405 If all security objectives are derived from assumptions and/or OSPs only, the statement of threats need not be present in the ST. In this case, this work unit is not applicable and therefore considered to be satisfied.

406 The evaluator determines that the security problem definition describes the threats that must be countered by the TOE, its development environment, its operational environment or combinations of these three.

ASE_SPD.1.2C ***All threats shall be described in terms of a threat agent, an asset, and an adverse action.***

ASE_SPD.1-2 The evaluator ***shall examine*** the security problem definition to determine that all threats are described in terms of a threat agent, an asset, and an adverse action.

407 If all security objectives are derived from assumptions and/or OSPs only, the statement of threats need not be present in the ST. In this case, this work unit is not applicable and therefore considered to be satisfied.

408 Threat agents may be further described by aspects such as expertise, resource, opportunity, and motivation.

ASE_SPD.1.3C ***The security problem definition shall describe the OSPs.***

ASE_SPD.1-3 The evaluator ***shall check*** that the security problem definition describes the OSPs.

409 If all security objectives are derived from assumptions and threats only, OSPs need not be present in the ST. In this case, this work unit is not applicable and therefore considered to be satisfied.

410 The evaluator determines that OSP statements are made in terms of rules, practises or guidelines that must be followed by the TOE, its development environment, its operational environment or combinations of these three.

411 The evaluator determines that each OSP is explained and/or interpreted in sufficient detail to make it clearly understandable; a clear presentation of policy statements is necessary to permit tracing security objectives to them.

ASE_SPD.1.4C ***The security problem definition shall describe the assumptions about the operational environment of the TOE.***

ASE_SPD.1-4 The evaluator ***shall examine*** the security problem definition to determine that it describes the assumptions about the operational environment of the TOE.

412 If the threats and/or OSPs already sufficiently address the physical, personnel, and connectivity aspects of the TOE, this work unit is not applicable and is therefore considered to be satisfied.

413 The evaluator determines that each assumption about the operational environment of the TOE is explained in sufficient detail to enable consumers to determine that their operational environment matches the assumption. If the assumptions are not clearly understood, the end result may be that the TOE is used in an operational environment in which it will not function in a secure manner.

10.6 Security objectives (ASE_OBJ)

10.6.1 Evaluation of sub-activity (ASE_OBJ.1)

10.6.1.1 Objectives

414 The objective of this sub-activity is to determine whether the security objectives for the operational environment are clearly defined and internally consistent.

10.6.1.2 Input

415 The evaluation evidence for this sub-activity is:

a) the ST.

10.6.1.3 Action ASE_OBJ.1.1E

ASE_OBJ.1.1C ***The statement of security objectives shall describe the security objectives for the operational environment.***

ASE_OBJ.1-1 The evaluator ***shall check*** that the statement of security objectives defines the security objectives for the operational environment.

416 The evaluator checks that the security objectives for the operational environment are identified.

10.6.2 Evaluation of sub-activity (ASE_OBJ.2)

10.6.2.1 Objectives

417 The objective of this sub-activity is to determine whether the security objectives adequately and completely address the security problem definition, that the division of this problem between the TOE, its development environment, and its operational environment is clearly defined, and whether the security objectives are internally consistent.

10.6.2.2 Input

418 The evaluation evidence for this sub-activity is:

a) the ST.

10.6.2.3 Action ASE_OBJ.2.1E

ASE_OBJ.2.1C ***The statement of security objectives shall describe the security objectives for the TOE, the security objectives for the development environment and the security objectives for the operational environment.***

ASE_OBJ.2-1 The evaluator ***shall check*** that the statement of security objectives defines the security objectives for the TOE, the security objectives for the

development environment and the security objectives for the operational environment.

419 The evaluator checks that each category of security objectives is clearly identified and separated from the other categories.

ASE_OBJ.2.2C ***The security objectives rationale shall trace each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective.***

ASE_OBJ.2-2 The evaluator ***shall check*** that the security objectives rationale traces all security objectives for the TOE back to threats countered by the objectives and/or OSPs enforced by the objectives.

420 Each security objective for the TOE may trace back to threats or OSPs, or a combination of threats and OSPs, but it must trace back to at least one threat or OSP.

421 Failure to trace implies that either the security objectives rationale is incomplete, the security problem definition is incomplete, or the security objective for the TOE has no useful purpose.

ASE_OBJ.2.3C ***The security objectives rationale shall trace each security objective for the development environment back to threats countered by that security objective and OSPs enforced by that security objective.***

ASE_OBJ.2-3 The evaluator ***shall check*** that the security objectives rationale traces the security objectives for the development environment back to threats countered by that security objective and OSPs enforced by that security objective.

422 Each security objective for the development environment may trace back to threats or OSPs, or a combination of threats and OSPs, but it must trace back to at least one threat or OSP.

423 Failure to trace implies that either the security objectives rationale is incomplete, the security problem definition is incomplete, or the security objective for the development environment has no useful purpose.

ASE_OBJ.2.4C ***The security objectives rationale shall trace each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective.***

ASE_OBJ.2-4 The evaluator ***shall check*** that the security objectives rationale traces the security objectives for the operational environment back to threats countered by that security objective, to OSPs enforced by that security objective, and to assumptions upheld by that security objective.

424 Each security objective for the operational environment may trace back to threats, OSPs, assumptions, or a combination of threats, OSPs and/or assumptions, but it must trace back to at least one threat, OSP or assumption.

425 Failure to trace implies that either the security objectives rationale is incomplete, the security problem definition is incomplete, or the security objective for the operational environment has no useful purpose.

ASE_OBJ.2.5C ***The security objectives rationale shall demonstrate that the security objectives counter all threats.***

ASE_OBJ.2-5 The evaluator ***shall examine*** the security objectives rationale to determine that it justifies for each threat that the security objectives are suitable to counter that threat.

426 If no security objectives trace back to the threat, this work unit fails.

427 The evaluator determines that the justification for a threat shows whether the threat is removed, diminished or mitigated.

428 The evaluator determines that the justification for a threat demonstrates that the security objectives are sufficient: if all security objectives that trace back to the threat are achieved, the threat is removed, sufficiently diminished, or the effects of the threat are sufficiently mitigated.

429 Note that the tracings from security objectives to threats provided in the security objectives rationale may be part of a justification, but do not constitute a justification by themselves. Even in the case that a security objective is merely a statement reflecting the intent to prevent a particular threat from being realised, a justification is required, but this justification could be as minimal as "Security Objective X directly counters Threat Y".

430 The evaluator also determines that each security objective that traces back to a threat is necessary: when the security objective is achieved it actually contributes to the removal, diminishing or mitigation of that threat.

ASE_OBJ.2.6C ***The security objectives rationale shall demonstrate that the security objectives enforce all OSPs.***

ASE_OBJ.2-6 The evaluator ***shall examine*** the security objectives rationale to determine that for each OSP it justifies that the security objectives are suitable to enforce that OSP.

431 If no security objectives trace back to the OSP, this work unit fails.

432 The evaluator determines that the justification for an OSP demonstrates that the security objectives are sufficient: if all security objectives that trace back to that OSP are achieved, the OSP is enforced.

433 The evaluator also determines that each security objective that traces back to an OSP is necessary: when the security objective is achieved it actually contributes to the enforcement of the OSP.

434 Note that the tracings from security objectives to OSPs provided in the security objectives rationale may be part of a justification, but do not constitute a justification by themselves. In the case that a security objective is

merely a statement reflecting the intent to enforce a particular OSP, a justification is required, but this justification could be as minimal as "Security Objective X directly enforces OSP Y".

ASE_OBJ.2.7C ***The security objectives rationale shall demonstrate that the security objectives for the operational environment uphold all assumptions.***

ASE_OBJ.2-7 The evaluator ***shall examine*** the security objectives rationale to determine that for each assumption for the operational environment it contains an appropriate justification that the security objectives for the operational environment are suitable to uphold that assumption.

435 If no security objectives for the operational environment trace back to the assumption, this work unit fails.

436 The evaluator determines that the justification for an assumption about the operational environment of the TOE demonstrates that the security objectives are sufficient: if all security objectives for the operational environment that trace back to that assumption are achieved, the operational environment upholds the assumption.

437 The evaluator also determines that each security objective for the operational environment that traces back to an assumption about the operational environment of the TOE is necessary: when the security objective is achieved it actually contributes to the operational environment upholding the assumption.

438 Note that the tracings from security objectives for the operational environment to assumptions provided in the security objectives rationale may be a part of a justification, but do not constitute a justification by themselves. Even in the case that a security objective of the operational environment is merely a restatement of an assumption, a justification is required, but this justification could be as minimal as "Security Objective X directly upholds Assumption Y".

10.7 Extended components definition (ASE_ECD)

10.7.1 Evaluation of sub-activity (ASE_ECD.1)

10.7.1.1 Objectives

439 The objective of this sub-activity is to determine whether extended components have been clearly and unambiguously defined, and whether they are necessary, i.e. they could not have been clearly expressed using existing CC Part 2 or CC Part 3 components.

10.7.1.2 Input

440 The evaluation evidence for this sub-activity is:

a) the ST.

10.7.1.3 Action ASE_ECD.1.1E

ASE_ECD.1.1C ***The statement of security requirements shall identify all extended security requirements.***

ASE_ECD.1-1 The evaluator ***shall check*** that all security requirements in the statement of security requirements that are not identified as extended requirements are present in CC Part 2 or in CC Part 3.

ASE_ECD.1.2C ***The extended components definition shall define an extended component for each extended security requirement.***

ASE_ECD.1-2 The evaluator ***shall check*** that the extended components definition defines an extended component for each extended security requirement.

441 If the ST does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

442 A single extended component may be used to define multiple iterations of an extended security requirement, it is not necessary to repeat this definition for each iteration.

ASE_ECD.1.3C ***The extended components definition shall describe how each extended component is related to the existing CC components, families, and classes.***

ASE_ECD.1-3 The evaluator ***shall examine*** the extended components definition to determine that it describes how each extended component fits into the existing CC components, families, and classes.

443 If the ST does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

444 The evaluator determines that each extended component is either:

a) a member of an existing CC Part 2 or CC Part 3 family, or

b) a member of a new family defined in the ST.

445 If the extended component is a member of an existing CC Part 2 or CC Part 3 family, the evaluator determines that the extended components definition adequately describes why the extended component should be a member of that family and how it relates to other components of that family.

446 If the extended component is a member of a new family defined in the ST, the evaluator confirms that the extended component is not appropriate for an existing family.

447 If the ST defines new families, the evaluator determines that each new family is either:

a) a member of an existing CC Part 2 or CC Part 3 class, or

b) a member of a new class defined in the ST.

448 If the family is a member of an existing CC Part 2 or CC Part 3 class, the evaluator determines that the extended components definition adequately describes why the family should be a member of that class and how it relates to other families in that class.

449 If the family is a member of a new class defined in the ST, the evaluator confirms that the family is not appropriate for an existing class.

ASE_ECD.1-4 The evaluator ***shall examine*** the extended components definition to determine that each definition of an extended component identifies all applicable dependencies of that component.

450 If the ST does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

451 The evaluator confirms that no applicable dependencies have been overlooked by the ST author.

ASE_ECD.1.4C ***The extended components definition shall use the existing CC components, families, classes, and methodology as a model for presentation.***

ASE_ECD.1-5 The evaluator ***shall examine*** the extended components definition to determine that each extended functional component uses the existing CC Part 2 components as a model for presentation.

452 If the ST does not contain extended SFRs, this work unit is not applicable and therefore considered to be satisfied.

453 The evaluator determines that the extended functional component is consistent with CC Part 2 Section 7.1.3, Functional component structure.

454 If the extended functional component uses operations, the evaluator determines that the extended functional component is consistent with CC Part 1 Annex C.4, Operations.

455 If the extended functional component is hierarchical to an existing functional component, the evaluator determines that the extended functional component is consistent with CC Part 2 Section 7.1.4.1, Component changes highlighting.

ASE_ECD.1-6 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new functional family uses the existing CC functional families as a model for presentation.

456 If the ST does not define new functional families, this work unit is not applicable and therefore considered to be satisfied.

457 The evaluator determines that all new functional families are defined consistent with CC Part 2 Section 7.1.2, Functional family structure.

ASE_ECD.1-7 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new functional class uses the existing CC functional classes as a model for presentation.

458 If the ST does not define new functional classes, this work unit is not applicable and therefore considered to be satisfied.

459 The evaluator determines that all new functional classes are defined consistent with CC Part 2 Section 7.1.1, Functional class structure.

ASE_ECD.1-8 The evaluator ***shall examine*** the extended components definition to determine that each definition of an extended assurance component uses the existing CC Part 3 components as a model for presentation.

460 If the ST does not contain extended SARs, this work unit is not applicable and therefore considered to be satisfied.

461 The evaluator determines that the extended assurance component definition is consistent with CC Part 3 Section 8.1.3, Assurance component structure.

462 If the extended assurance component uses operations, the evaluator determines that the extended assurance component is consistent with CC Part 1 Annex C.4, Operations.

463 If the extended assurance component is hierarchical to an existing assurance component, the evaluator determines that the extended assurance component is consistent with CC Part 3 Section 8.1.3, Assurance component structure.

ASE_ECD.1-9 The evaluator ***shall examine*** the extended components definition to determine that, for each defined extended assurance component, applicable methodology has been provided.

464 If the ST does not contain extended SARs, this work unit is not applicable and therefore considered to be satisfied.

465 The evaluator determines that, for each evaluator action element of each extended SAR, one or more work units is provided and that successfully

performing all work units for a given evaluator action element will demonstrate that the element has been achieved.

ASE_ECD.1-10 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new assurance family uses the existing CC assurance families as a model for presentation.

466 If the ST does not define new assurance families, this work unit is not applicable and therefore considered to be satisfied.

467 The evaluator determines that all new assurance families are defined consistent with CC Part 3 Section 8.1.2, Assurance family structure.

ASE_ECD.1-11 The evaluator ***shall examine*** the extended components definition to determine that each definition of a new assurance class uses the existing CC assurance classes as a model for presentation.

468 If the ST does not define new assurance classes, this work unit is not applicable and therefore considered to be satisfied.

469 The evaluator determines that all new assurance classes are defined consistent with CC Part 3 Section 8.1.1, Assurance class structure.

ASE_ECD.1.5C ***The extended components shall consist of measurable and objective elements such that compliance or noncompliance to these elements can be demonstrated.***

ASE_ECD.1-12 The evaluator ***shall examine*** the extended components definition to determine that each element in each extended component is measurable and states objective evaluation requirements, such that compliance or noncompliance can be demonstrated.

470 If the ST does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

471 The evaluator determines that elements of extended functional components are stated in such a way that they are testable, and traceable through the appropriate TSF representations.

472 The evaluator also determines that elements of extended assurance components avoid the need for subjective evaluator judgement.

473 The evaluator is reminded that whilst being measurable and objective is appropriate for all evaluation criteria, it is acknowledged that no formal method exists to prove such properties. Therefore the existing CC functional and assurance components are to be used as a model for determining what constitutes compliance with this requirement.

10.7.1.4 Action ASE_ECD.1.2E

ASE_ECD.1-13 The evaluator ***shall examine*** the extended components definition to determine that each extended component can not be clearly expressed using existing components.

474 If the ST does not contain extended security requirements, this work unit is not applicable and therefore considered to be satisfied.

475 The evaluator should take components from CC Part 2 and CC Part 3, other extended components that have been defined in the ST, combinations of these components, and possible operations on these components into account when making this determination.

476 The evaluator is reminded that the role of this work unit is to preclude unnecessary duplication of components, that is, components that can be clearly expressed by using other components. The evaluator should not undertake an exhaustive search of all possible combinations of components including operations in an attempt to find a way to express the extended component by using existing components.

10.8 Security requirements (ASE_REQ)

10.8.1 Evaluation of sub-activity (ASE_REQ.1)

10.8.1.1 Objectives

477 The objective of this sub-activity is to determine whether the SFRs and SARs are clear, unambiguous and well-defined and whether they are internally consistent.

10.8.1.2 Input

478 The evaluation evidence for this sub-activity is:

a) the ST.

10.8.1.3 Action ASE_REQ.1.1E

ASE_REQ.1.1C ***The statement of security requirements shall describe the SFRs and the SARs.***

ASE_REQ.1-1 The evaluator ***shall check*** that the statement of security requirements describes the SFRs.

479 The evaluator determines that each SFR is identified by one of the following means:

a) by reference to an individual component in CC Part 2;

b) by reference to an extended component in the extended components definition of the ST;

c) by reference to a PP that the ST claims to be compliant with;

d) by reference to a security requirements package that the ST claims to be compliant with;

e) by reproduction in the ST.

480 It is not required to use the same means of identification for all SFRs.

ASE_REQ.1-2 The evaluator ***shall check*** that the statement of security requirements describes the SARs.

481 The evaluator determines that each SAR is identified by one of the following means:

a) by reference to an individual component in CC Part 3;

b) by reference to an extended component in the extended components definition of the ST;

c) by reference to a PP that the ST claims to be compliant with;

d) by reference to a security requirements package that the ST claims to be compliant with;

e) by reproduction in the ST.

482 It is not required to use the same means of identification for all SARs.

ASE_REQ.1-3 The evaluator ***shall examine*** the ST to determine that the ST lists and explains all terms that are used in the requirements that are not defined in the CC.

483 The evaluator determines that the ST lists and explains all:

– (types of) subjects and objects that are used in the SFRs

– (types of) security attributes of these subjects and objects (with possible values that these attributes may take)

– (types of) operations that are used in the SFRs

– (types of) users in the SFRs

– other terms that are introduced in the SFRs and/or SARs by completing operations, if these terms are not immediately clear to a wide audience, or are used outside their dictionary definition.

484 The goal of this work unit is to ensure that the SFRs and SARs are well-defined and that no misunderstanding can occur due to the introduction of vague terms. This work unit should not be taken into extremes, by forcing the PP writer to define every single word. The general audience of a set of security requirements can be assumed to have a reasonable knowledge of IT, security and Common Criteria.

485 All of the above may be presented in groups, classes, roles, types or other groupings or characterisations that allow easy understanding.

486 The evaluator is reminded that these lists and definitions do not have to be part of the statement of security requirements, but could be placed (in part or in whole) in different sections. This may be especially applicable if the same terms are used in the rest of the ST.

ASE_REQ.1.2C ***The statement of security requirements shall identify all operations on the security requirements.***

ASE_REQ.1-4 The evaluator ***shall check*** that the statement of security requirements identifies all operations on the security requirements.

487 The evaluator determines that all operations are identified in each SFR or SAR where such an operation is used. Identification can be achieved by typographical distinctions, or by explicit identification in the surrounding text, or by any other distinctive means.

ASE_REQ.1.3C ***All assignment and selection operations shall be completed.***

ASE_REQ.1-5 The evaluator ***shall examine*** the statement of security requirements to determine that each assignment and each selection operation is completed.

488 The evaluator determines that there are no choices left in the assignments and selections of all SFRs and all SARs.

ASE_REQ.1.4C ***All operations shall be performed correctly.***

ASE_REQ.1-6 The evaluator ***shall examine*** the statement of security requirements to determine that all assignment operations are performed correctly.

489 An assignment operation is only allowed where specifically permitted in a component.

490 The evaluator compares each assignment with the component from which it is derived, to determine that the values of the parameters or variables chosen comply with the indicated type required by the assignment.

ASE_REQ.1-7 The evaluator ***shall examine*** the statement of security requirements to determine that all iteration operations are performed correctly.

491 The evaluator determines that each iteration of a requirement is different from each other iteration of that requirement (at least one element is different), or that the requirement applies to a different part of the TOE.

ASE_REQ.1-8 The evaluator ***shall examine*** the statement of security requirements to determine that all selection operations are performed correctly.

492 A selection operation is only allowed where specifically permitted in a component.

493 The evaluator compares each selection with the component from which it is derived, to determine that the selected item or items are one or more of the items indicated within the selection portion of the component.

ASE_REQ.1-9 The evaluator ***shall examine*** the statement of security requirements to determine that all refinement operations are performed correctly.

494 The evaluator determines for each refinement that the component is refined in such manner that a TOE meeting the refined requirement also meets the unrefined requirement. If the refined requirement exceeds this boundary it is considered to be an extended requirement.

495 A special case of refinement is an editorial refinement, where a small change is made in a requirement, i.e. rephrasing a sentence due to adherence to proper English grammar. This change is not allowed to modify the meaning of the requirement in any way. The evaluator is reminded that editorial refinements have to be clearly identified.

496 Another special case of refinement is where multiple iterations of the same requirement are used, each with different refinements, where some of the refined iterations do not meet the full scope of the original requirement. This is acceptable, provided that all iterations of the refined requirement, taken collectively, meet the entire scope of the original requirement.

497 In addition, a refinement should be related to the original requirement. Refining an audit requirement with an extra element on prevention of electromagnetic radiation is normally not allowed. This refinement should be added to another requirement or, if no applicable requirement to refine can be found, be formulated as an extended requirement.

ASE_REQ.1.5C ***Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.***

ASE_REQ.1-10 The evaluator ***shall examine*** the statement of security requirements to determine that each dependency of the security requirements is either satisfied, or that the security requirements rationale justifies the dependency not being satisfied.

498 A dependency is satisfied by the inclusion of the relevant component (or one that is hierarchical to it) within the statement of security requirements. The component used to satisfy the dependency should, if necessary, be modified by operations to ensure that it actually satisfies that dependency.

499 A justification that a dependency is not met can address either:

a) why the dependency is not necessary or useful, in which case no further information is required; or

b) that the dependency has been addressed by the operational environment of the TOE, in which case the justification should describe how the security objectives for the operational environment address this dependency.

10.8.1.4 Action ASE_REQ.1.2E

ASE_REQ.1-11 The evaluator ***shall examine*** the statement of security requirements to determine that it is internally consistent.

500 The evaluator determines that the combined set of all SFRs and SARs is internally consistent.

501 The evaluator determines that on all occasions where different security requirements apply to the same types of developer evidence, events, operations, data, tests to be performed etc. or to “all objects”, “all subjects” etc., that these requirements do not conflict.

502 Some possible conflicts are:

a) an extended SAR specifying that the design of a certain cryptographic algorithm is to be kept secret, and another extended SAR specifying an open source review;

b) FAU_GEN.1 Audit data generation without time specifying that subject identity is to be logged, FDP_ACC.1 Access control specifying who has access to these logs, and FDP_UNO.1 Limited unobservability specifying that some actions of subjects should be unobservable to other subjects. If the subject that should not be able to see an activity can access logs of this activity, these SFRs conflict;

c) FPT_RIP.1 Removal before use specifying deletion of information no longer needed, and FDP_ROL.1 Rollback specifying that a TOE can return to a previous state. If the information that is needed for the rollback to the previous state has been deleted, these requirements conflict;

d) Multiple iterations of FDP_ACC.1 Access control and/or FDP_ACC.2 Access control with automatic modification of security attributes especially where some iterations cover the same subjects, objects, or operations. If one access control policy allows a subject to perform an operation on an object, while another policy does not allow this, these requirements conflict.

10.8.2 Evaluation of sub-activity (ASE_REQ.2)

10.8.2.1 Objectives

503 The objective of this sub-activity is to determine whether the SFRs and SARs are clear, unambiguous and well-defined, whether they are internally consistent, and whether they meet the security objectives of the TOE and the security objectives for the development environment.

10.8.2.2 Input

504 The evaluation evidence for this sub-activity is:

a) the ST.

10.8.2.3 Action ASE_REQ.2.1E

ASE_REQ.2.1C ***The statement of security requirements shall describe the SFRs and the SARs.***

ASE_REQ.2-1 The evaluator ***shall check*** that the statement of security requirements describes the SFRs.

505 The evaluator determines that each SFRs is identified by one of the following means:

a) by reference to an individual component in CC Part 2;

b) by reference to an extended component in the extended components definition of the ST;

c) by reference to an individual component in a PP that the ST claims to be compliant with;

d) by reference to an individual component in a security requirements package that the ST claims to be compliant with;

e) by reproduction in the ST.

506 It is not required to use the same means of identification for all SFRs.

ASE_REQ.2-2 The evaluator ***shall check*** that the statement of security requirements describes the SARs.

507 The evaluator determines that all SARs are identified by one of the following means:

a) by reference to an individual component in CC Part 3;

b) by reference to an extended component in the extended components definition of the ST;

c) by reference to an individual component in a PP that the ST claims to be compliant with;

d) by reference to an individual component in a security requirements package that the ST claims to be compliant with;

e) by reproduction in the ST.

508 It is not required to use the same means of identification for all SARs.

ASE_REQ.2-3 The evaluator ***shall examine*** the ST to determine that the ST lists and explains all terms that are used in the requirements that are not defined in the CC.

509 The evaluator determines that the ST lists and explains all:

- (types of) subjects and objects that are used in the SFRs
- (types of) security attributes of these subjects and objects (with possible values that these attributes may take)
- (types of) operations that are used in the SFRs
- (types of) users in the SFRs
- other terms that are introduced in the SFRs and/or SARs by completing operations, if these terms are not immediately clear to a wide audience, or are used outside their dictionary definition.

510 The goal of this work unit is to ensure that the SFRs and SARs are well-defined and that no misunderstanding can occur due to the introduction of vague terms. This work unit should not be taken into extremes, by forcing the PP writer to define every single word. The general audience of a set of security requirements can be assumed to have a reasonable knowledge of IT, security and Common Criteria.

511 All of the above may be presented in groups, classes, roles, types or other groupings or characterisations that allow easy understanding.

512 The evaluator is reminded that these lists and definitions do not have to be part of the statement of security requirements, but could be placed (in part or in whole) in different sections. This may be especially applicable if the same terms are used in the rest of the ST (e.g. security problem definition or security objectives).

ASE_REQ.2.2C ***The statement of security requirements shall identify all operations on the security requirements.***

ASE_REQ.2-4 The evaluator ***shall check*** that the statement of security requirements identifies all operations on the security requirements.

513 The evaluator determines that all operations are identified in each SFR or SAR where such an operation is used. Identification can be achieved by typographical distinctions, or by explicit identification in the surrounding text, or by any other distinctive means.

ASE_REQ.2.3C ***All assignment and selection operations shall be completed.***

ASE_REQ.2-5 The evaluator ***shall examine*** the statement of security requirements to determine that each assignment and each selection operation is completed.

514 The evaluator determines that there are no choices left in the assignments and selections of all SFRs and all SARs.

ASE_REQ.2.4C ***All operations shall be performed correctly.***

ASE_REQ.2-6 The evaluator ***shall examine*** the statement of security requirements to determine that all assignment operations are performed correctly.

515 An assignment operation is only allowed where specifically permitted in a component.

516 The evaluator compares each assignment with the component from which it is derived, to determine that the values of the parameters or variables chosen comply with the indicated type required by the assignment.

ASE_REQ.2-7 The evaluator ***shall examine*** the statement of security requirements to determine that all iteration operations are performed correctly.

517 The evaluator determines that each iteration of a requirement is different from each other iteration of that requirement (at least one element is different), or that the requirement applies to a different part of the TOE.

ASE_REQ.2-8 The evaluator ***shall examine*** the statement of security requirements to determine that all selection operations are performed correctly.

518 A selection operation is only allowed where specifically permitted in a component.

519 The evaluator compares each selection with the component from which it is derived, to determine that the selected item or items are one or more of the items indicated within the selection portion of the component.

ASE_REQ.2-9 The evaluator ***shall examine*** the statement of security requirements to determine that all refinement operations are performed correctly.

520 The evaluator determines for each refinement that the component is refined in such manner that a TOE meeting the refined requirement also meets the unrefined requirement. If the refined requirement exceeds this boundary it is considered to be an extended requirement.

521 A special case of refinement is an editorial refinement, where a small change is made in a requirement, i.e. rephrasing a sentence due to adherence to proper English grammar. This change is not allowed to modify the meaning of the requirement in any way. The evaluator is reminded that editorial refinements have to be clearly identified.

522 Another special case of refinement is where multiple iterations of the same requirement are used, each with different refinements, where some of the refined iterations do not meet the full scope of the original requirement. This is acceptable, provided that all iterations of the refined requirement, taken collectively, meet the entire scope of the original requirement.

523 In addition, a refinement should be related to the original requirement. Refining an audit requirement with an extra element on prevention of electromagnetic radiation is normally not allowed. This refinement should be added to another requirement or, if no applicable requirement to refine can be found, be formulated as an extended requirement.

ASE_REQ.2.5C ***Each dependency of the security requirements shall either be satisfied, or the security requirements rationale shall justify the dependency not being satisfied.***

ASE_REQ.2-10 The evaluator ***shall examine*** the statement of security requirements to determine that each dependency of the security requirements is either satisfied, or that the security requirements rationale justifies the dependency not being satisfied.

524 A dependency is satisfied by the inclusion of the relevant component (or one that is hierarchical to it) within the statement of security requirements. The

component used to satisfy the dependency should, if necessary, be modified by operations to ensure that it actually satisfies that dependency.

525 A justification that a dependency is not met can address either:

a) why the dependency is not necessary or useful, in which case no further information is required; or

b) that the dependency has been addressed by the operational environment of the TOE, in which case the justification should describe how the security objectives for the operational environment address this dependency.

ASE_REQ.2.6C ***The security requirements rationale shall trace each SFR back to the security objectives for the TOE.***

ASE_REQ.2-11 The evaluator ***shall check*** that the security requirements rationale traces each SFR back to the security objectives for the TOE.

526 The evaluator determines that each SFR is traced back to at least one security objective for the TOE.

527 Failure to trace implies that either the security requirements rationale is incomplete, the security objectives for the TOE are incomplete, or the SFR has no useful purpose.

ASE_REQ.2.7C ***The security requirements rationale shall demonstrate that the SFRs meet all security objectives for the TOE.***

ASE_REQ.2-12 The evaluator ***shall examine*** the security requirements rationale to determine that for each security objective for the TOE it demonstrates that the SFRs are suitable to meet that security objective for the TOE.

528 If no SFRs trace back to the security objective for the TOE, this work unit fails.

529 The evaluator determines that the justification for a security objective for the TOE demonstrates that the SFRs are sufficient: if all SFRs that trace back to the objective are satisfied, the security objective for the TOE is achieved.

530 The evaluator also determines that each SFR that traces back to a security objective for the TOE is necessary: when the SFR is satisfied, it actually contributes to achieving the security objective.

531 Note that the tracings from SFRs to security objectives for the TOE provided in the security requirements rationale may be a part of the justification, but do not constitute a justification by themselves.

532 While examining the justification, the evaluator takes into account that, unless the security objectives of the operational environment preclude this, TOEs may be physically attacked, or repeatedly attacked without this being detected, which may prevent the SFRs to achieve the security objectives.

533 and these cases are not or not sufficiently addressed by the security objectives for the operational environment.

ASE_REQ.2.8C ***The security requirements rationale shall trace each SAR back to the security objectives for the development environment.***

ASE_REQ.2-13 The evaluator ***shall check*** that the security requirements rationale traces each SAR back to the security objectives for the development environment.

534 The evaluator determines that each SAR is traced back to at least one security objective for the development environment.

535 Failure to trace implies that either the security requirements rationale is incomplete, the security objectives for the development environment are incomplete, or the SAR has no useful purpose.

ASE_REQ.2.9C ***The security requirements rationale shall demonstrate that the SARs meet all security objectives for the development environment.***

ASE_REQ.2-14 The evaluator ***shall examine*** the security requirements rationale to determine that for each security objective for the development environment it justifies that the SARs are suitable to meet that security objective for the development environment.

536 If no SARs trace back to the security objective for the development environment, this work unit fails.

537 The evaluator determines that the justification for a security objective for the development environment demonstrates that the SARs are sufficient: if all SARs that trace back to the objective are satisfied, the security objective for the development environment is achieved.

538 The evaluator also determines that each SAR that traces back to a security objective for the development environment is necessary, when the SAR is satisfied, it actually contributes to achieving the security objective.

539 Note that the tracings from SARs to security objectives for the development environment provided in the security requirements rationale may be a part of the justification, but do not constitute a justification by themselves.

10.8.2.4 Action ASE_REQ.2.2E

ASE_REQ.2-15 The evaluator ***shall examine*** the statement of security requirements to determine that it is internally consistent.

540 The evaluator determines that the combined set of all SFRs and SARs is internally consistent.

541 The evaluator determines that on all occasions where different security requirements apply to the same types of developer evidence, events, operations, data, tests to be performed etc. or to “all objects”, “all subjects” etc., that these requirements do not conflict.

542 Some possible conflicts are:

a) an extended SAR specifying that the design of a certain cryptographic algorithm is to be kept secret, and another extended assurance requirement specifying an open source review;

b) FAU_GEN.1 Audit data generation without time specifying that subject identity is to be logged, FDP_ACC.1 Access control specifying who has access to these logs, and FDP_UNO.1 Limited unobservability specifying that some actions of subjects should be unobservable to other subjects. If the subject that should not be able to see an activity can access logs of this activity, these SFRs conflict;

c) FPT_RIP.1 Removal before use specifying deletion of information no longer needed, and FDP_ROL.1 Rollback specifying that a TOE can return to a previous state. If the information that is needed for the rollback to the previous state has been deleted, these requirements conflict;

d) Multiple iterations of FDP_ACC.1 Access control and/or FDP_ACC.2 Access control with automatic modification of security attributes especially where some iterations cover the same subjects, objects, or operations. If one access control policy allows a subject to perform an operation on an object, while another policy does not allow this, these requirements conflict.

10.9 TOE summary specification (ASE_TSS)

10.9.1 Evaluation of sub-activity (ASE_TSS.1)

10.9.1.1 Objectives

543 The objective of this sub-activity is to determine whether the TOE summary specification addresses all SFRs, and whether the TOE summary specification is consistent with other narrative descriptions of the TOE.

10.9.1.2 Input

544 The evaluation evidence for this sub-activity is:

a) the ST.

10.9.1.3 Action ASE_TSS.1.1E

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

ASE_TSS.1-1 The evaluator ***shall examine*** the TOE summary specification to determine that it describes how the TOE meets each SFR.

545 The evaluator determines that the TOE summary specification provides, for each SFR from the statement of security requirements, a description on how that SFR is met.

546 The evaluator is reminded that the objective of each description is to provide potential consumers of the TOE with a high-level view of how the developer intends to satisfy each SFR and that the descriptions therefore should not be overly detailed.

547 For a composed TOE, the evaluator also determines that it is clear which component provides each SFR or how the components combine to meet each SFR.

10.9.1.4 Action ASE_TSS.1.2E

ASE_TSS.1-2 The evaluator ***shall examine*** the TOE summary specification to determine that it is consistent with the TOE overview and the TOE description.

548 The TOE overview, TOE description, and TOE summary specification describe the TOE in a narrative form at increasing levels of detail. These descriptions therefore need to be consistent.

11 Class ADV: Development

11.1 Introduction

549 The purpose of the development activity is to assess the design documentation in terms of its adequacy to understand how the TSF meets the SFRs and how the implementation of these SFRs cannot be tampered with or bypassed. This understanding is achieved through examination of increasingly refined descriptions of the TSF design documentation. Design documentation consists of a functional specification (which describes the interfaces of the TSF), a TOE design description (which describes the architecture of the TSF in terms of how it works in order to perform the functions related to the SFRs being claimed), and an implementation description (a source code level description). In addition, there is an architectural design document (which describes the architectural properties of the TSF to explain how its security enforcement cannot be compromised or bypassed), an internals description (which describes how the TSF was constructed in a manner that encourages understandability), and a security policy model (which formally describes the security policies enforced by the TSF).

11.2 Application notes

550 The CC requirements for design documentation are levelled by the amount, and detail of information provided, and the degree of formality of the presentation of the information. At lower levels, the most security-critical portions of the TSF are described with the most detail, while less security-critical portions of the TSF are merely summarised; added assurance is gained by increasing the amount of information about the most security-critical portions of the TSF, and increasing the details about the less security-critical portions. The most assurance is achieved when thorough details and information of all portions are provided.

551 The CC considers a document's degree of formality (that is, whether it is informal or semiformal) to be hierarchical. An informal document is one that is expressed in a natural language. The methodology does not dictate the specific language that must be used; that issue is left for the scheme. The following paragraphs differentiate the contents of the different informal documents.

552 A functional specification provides a description of the purpose and method-of-use of interfaces to the TSF. For 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 each of these and how they are realised through interactions with the externally-visible interfaces to the TSF. If there is also audit functionality that detects and record the occurrences of such events, descriptions 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.

553 A design description is expressed in terms of logical divisions (subsystems or components) that each provide a comprehensible service or function. For 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, in terms of what actions each subsystem takes when an incoming packet arrives at the firewall.

554 A security policy model describes the policies enforced by the functionality and services that make up the TSF. At the most informal level, this would be a broad characterisation at the level of detail included in the set of security objectives in the TOE's ST/PP. A semiformal model would have a constant structure presentation and would best be exemplified by the set of SFRs that the TSF claims. A formal model would be a mathematically precise description of the TSF's security behaviour.

11.3 Architectural design (ADV_ARC)

11.3.1 Evaluation of sub-activity (ADV_ARC.1)

11.3.1.1 Objectives

555 The objective of this sub-activity is to determine whether the TSF is structured such that it cannot be interfered with or bypassed, and whether TSFs that provide security domains isolate those domains from each other.

11.3.1.2 Input

556 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the architectural design;

e) the operational user guidance;

11.3.1.3 Application notes

557 The notions of self-protection, domain isolation, and non-bypassability are distinct from security functionality expressed in Part 2 SFRs because self-protection and non-bypassability largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are achieved through the design of the TOE, and enforced by the correct implementation of that design. Also, the evaluation of these properties is less straight-forward than the evaluation of mechanisms; it is more difficult to check for the absence of functionality than for its presence. However, the determination that these properties are being satisfied is just as critical as the determination that the mechanisms are properly implemented.

558 The overall approach used is that the developer provides a TSF that meets the above-mentioned properties, and provides evidence (in the form of documentation) that can be analysed to show that the properties are indeed met. The evaluator has the responsibility for looking at the evidence and, coupled with other evidence delivered for the TOE, determining that the properties are achieved. The work units can be characterised as those detailing with what information has to be provided, and those dealing with the actual analysis the evaluator performs.

559 The analyses the evaluator performs must be done in the context of all of the development evidence provided for the TOE, at the level of detail the evidence is provided. At lower assurance levels there should not be the expectation that, for example, TSF self-protection is completely analysed, because only high-level design representations will be available. The evaluator also needs to be sure to use information gleaned from other

portions of their analysis (e.g., analysis of the TOE design) in making their assessments for the properties being examined in the following work units.

11.3.1.4 Action ADV_ARC.1.1E

ADV_ARC.1.1C ***The descriptive information contained in the architectural design shall be at a level of detail commensurate with the description of the SFR-enforcing abstractions described in the TOE design document.***

ADV_ARC.1-1 The evaluator ***shall examine*** the architectural design to determine that the information provided in the evidence is presented at a level of detail commensurate with the descriptions of the SFR-enforcing abstractions contained in the TOE design document.

560 With respect to the functional specification, the evaluator should ensure that the self-protection functionality described cover those effects that are evident at the TSFI. Such a 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 evaluator ensures that the functionality that might be invoked through the TSFI is described.

561 If Evaluation of sub-activity (ADV_TDS.1) or Evaluation of sub-activity (ADV_TDS.2) are included, in addition to the information above the evaluator also ensures the description contains information on how architectural components that contribute to TSF domain separation work.

562 If Evaluation of sub-activity (ADV_TDS.3) or higher is available, in addition to the information previously covered the architectural description should also contain information that is fairly close to the actual implementation of the functionality, and not so abstract that it is implementation independent. For example, such a description might contain information pertaining to coding conventions that would prevent TSF compromises (e.g. buffer overflows), and information on stack management for call and return operations. The evaluator checks the descriptions of the mechanisms to ensure that the level of detail is such that there is little ambiguity between the description in the architectural design and the implementation representation.

ADV_ARC.1.2C ***The architectural design shall describe the security domains maintained by the TSF consistently with the SFRs.***

ADV_ARC.1-2 The evaluator ***shall examine*** the architectural design to determine that it describes the security domains maintained by the TSF.

563 Security domains refer to environments supplied by the TSF for use by potentially-harmful entities; for example, a typical secure operating system supplies a set of resources (address space, per-process environment variables) for use by processes with limited access rights and security properties. The evaluator determines that the developer's description of the security domains takes into account all of the SFRs claimed by the TOE.

564 For some TOEs such domains do not exist because all of the interactions available to users are severely constrained by the TSF. A packet-filter firewall is an example of such a TOE. Users on the LAN or WAN do not interact with the TOE, so there need be no security domains; there are only data structures maintained by the TSF to keep the users' packets separated. The evaluator ensures that any claim that there are no domains is supported by the evidence and that no such domains are, in fact, available.

ADV_ARC.1.3C ***The architectural design shall describe how the TSF initialisation process is secure.***

ADV_ARC.1-3 The evaluator ***shall examine*** the architectural design to determine that the initialisation process preserves security.

565 The information provided in the architectural design relating to TSF initialisation is directed at the TOE components that are involved in bringing the TSF into an initial secure state (i.e. when all parts of the TSF are operational) when power-on or a reset is applied. This discussion in the architectural design should list the system initialisation components and the processing that occurs in transitioning from the “down” state to the initial secure state.

566 It is often the case that the components that perform the initialisation function are not accessible after the secure state is achieved; if this is the case then the architectural design identifies the components and explains how that they are not reachable by untrusted entities after the TSF has been established. In this respect, the property that needs to be preserved is that these components either 1) cannot be accessed by untrusted entities after the secure state is achieved, or 2) if they provide interfaces to untrusted entities, these TSFI cannot be used to violate the TSP.

567 The TOE components related to TSF initialisation, then, are treated themselves as part of the TSF, and analysed from that perspective. It should be noted that even though these are treated as part of the TSF, it is likely that a justification (as allowed by TSF internals (ADV_INT)) can be made that they do not have to meet the internal structuring requirements of ADV_INT.

ADV_ARC.1.4C ***The architectural design shall demonstrate that the TSF protects itself from interference and tampering.***

ADV_ARC.1-4 The evaluator ***shall examine*** the architectural design to determine that it contains information sufficient to support a determination that the TSF is able to protect itself from interference and tampering by untrusted active entities.

568 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 does not, strictly speaking, protect itself because it depends on the other IT entities to provide some of the protection.

For the purposes of the architectural design, 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.

569 Self-protection is typically achieved by a variety of means, ranging from physical and logical restrictions on access to the TOE; to hardware-based means (e.g. "execution rings" and memory management functionality); to software-based means (e.g. boundary checking of inputs on a trusted server). The evaluator determines that all such mechanisms are described.

570 The evaluator determines that the design description covers how user input is handled by the TSF in such a way that the TSF does not subject itself to being corrupted by that user input. For example, the TSF might implement the notion of privilege and protect itself by using privileged-mode routines to handle user data. The TSF might make use of processor-based separation mechanisms such as privilege levels or rings. The TSF might implement software protection constructs or coding conventions that contribute to implementing isolation of software domains, perhaps by delineating user address space from system address space. And the TSF might have reliance its environment to provide some support to the protection of the TSF.

571 All of the mechanisms contributing to the domain separation functions are described. The evaluator should use knowledge gained from other evidence (functional specification, TOE design, TSF internals description, other parts of the architectural description, or implementation representation, as included in the assurance package for the TOE) in determining if any functionality contributing to self-protection was described that is not present in the architectural design.

572 Accuracy of the description of the self-protection mechanisms is the property that the description faithfully describes what is implemented. The evaluator should use other evidence (functional specification, TOE design, TSF Internals documentation, other parts of the architectural description, implementation representation, as included in the ST for the TOE) in determining whether there are discrepancies in any descriptions of the self-protection mechanisms. If Implementation representation (ADV_IMP) is included in the assurance package for the TOE, the evaluator will choose a sample of the implementation representation; the evaluator should also ensure that the descriptions are accurate for the sample chosen. If an evaluator cannot understand how a certain self-protection mechanism works or could work in the system architecture, it may be the case that the description is not accurate.

ADV_ARC.1.5C ***The architectural design shall demonstrate that the TSF prevents bypass of the SFR-enforcing functionality.***

ADV_ARC.1-5 The evaluator ***shall examine*** the architectural design to determine that it presents an analysis that adequately describes how the SFR-enforcing mechanisms cannot be bypassed.

573 Non-bypassability is a property that the security functions of the TSF (as specified by the SFRs) are always invoked. For example, if access control to files is specified as a capability of the TSF via an SFR, there must be no interfaces through which files can be accessed without invoking the TSF's access control mechanism (such as an interface through which a raw disk access takes place).

574 Describing how the TSF mechanisms cannot be bypassed generally requires a systematic argument based on the TSP and the TSFI. The description of how the TSF works (contained in the design decomposition evidence, such as the functional specification, TOE design documentation) - along with the information in the TSS - provides the background necessary for the evaluator to understand what resources are being protected and what security functions are being provided. The functional specification provides descriptions of the TSFI through which the resources/functions are accessed.

575 The evaluator assesses the description provided (and other information provided by the developer, such as the functional specification) to ensure that no available interface can bypass the TSF. This means that every available interface must be either unrelated to the SFRs that are claimed in the ST (and does not interact with anything that is used to satisfy SFRs) or else uses the security functionality that is described in other development evidence in the manner described. For example, a game would likely be unrelated to the SFRs, so there must be an explanation of how it cannot affect security. Access to user data, however, is likely to be related to access control SFRs, so the explanation would describe how the security functionality works when invoked through the data-access interfaces. Such a description in needed for every available interface.

576 An example of a description follows. Suppose the TSF provides file protection. Further suppose that although the “traditional” system call TSFI 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.

577 Using this same example, suppose there is a TSFI whose sole purpose is to display the time of day. The evaluator should determine that the description adequately argues that this TSFI is not capable of manipulating any protected resources and should not invoke any security function.

578 For a final example using a security function rather than a protected resource, consider an ST that contains FCO_NRE.2 Non-repudiation of origin of exported data, which requires that the TSF provides evidence of origination for information types specified in the ST. Suppose that the “information types” included all information that is sent by the TOE via e-mail. In this case the evaluator should examine the description to ensure that all TSFI that

can be invoked to send e-mail perform the "evidence of origination generation" function are detailed. The description might point to user guidance to show all places where e-mail can originate (e.g., e-mail program, notification from scripts/batch jobs) and then how each of these places invokes the evidence generation function.

579 The evaluator should also ensure that the description is comprehensive, in that each interface is analysed with respect to the entire TSP. This may require the evaluator to examine supporting information (functional specification, TOE design, other parts of the architectural description, operational user guidance, and perhaps even the implementation representation, as provided for the TOE) to determine that the description has correctly capture all aspects of an interface. The evaluator should consider what portions of the TSP each TSFI might affect (from the description of the TSFI and its implementation in the supporting documentation), and then examine the description to determine whether it covers those aspects.

11.4 Functional specification (ADV_FSP)

11.4.1 Evaluation of sub-activity (ADV_FSP.1)

11.4.1.1 Objectives

580 The objective of this sub-activity is to determine whether the developer has provided a high-level description of at least the SFR-enforcing and SFR-supporting TSFI, in terms of descriptions of their parameters. There is no other required evidence that can be expected to be available to measure the accuracy of these descriptions; the evaluator merely ensures the descriptions seem plausible.

11.4.1.2 Input

581 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the operational user guidance;

11.4.1.3 Action ADV_FSP.1.1E

ADV_FSP.1.1C ***The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.***

ADV_FSP.1-1 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each SFR-supporting and SFR-enforcing TSFI.

582 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 operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for. The evaluator should not only determine that the purpose exists, but also that it accurately reflects the TSFI by taking into account other information about the interface, such as the description of the parameters; this can be done in association with other work units for this component.

583 If an operation available through an interface plays a role in enforcing any security policy on the TOE (that is, if one of the operations 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 operations and results, some of which may be SFR-enforcing and some of which may not.

584 Interfaces to (or operations available through an interface relating to) operations 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 operations on which SFR-enforcing functionality has no dependence are termed *SFR non-interfering*.

585 It should be noted that in order for an interface to be SFR supporting or SFR non-interfering it must have *no* SFR-enforcing operations or results. In contrast, an SFR-enforcing interface may have SFR-supporting operations (for example, the ability to set the system clock may be an SFR-enforcing operation of an interface, but if that same interface is used to display the system date that operation may only be SFR supporting). An example of a purely SFR-supporting interface is a system call interface that is used both by untrusted users and by a portion of the TSF that is running in user mode.

586 At this level, it is unlikely that a developer will have expended effort to label interfaces as SFR-enforcing and SFR-supporting. In the case that this has been done, the evaluator should verify to the extent that supporting documentation (e.g., operational user guidance) allows that this identification is correct. Note that this identification activity is necessary for several work units for this component.

587 In the more likely case that the developer has not labelled the interfaces, the evaluator must perform their own identification of the interfaces first, and then determine whether the required information (for this work unit, the purpose) is present. Again, because of the lack of supporting evidence this identification will be difficult and have low assurance that all appropriate interfaces have been correctly identified, but nonetheless the evaluator examines other evidence available for the TOE to ensure as complete coverage as is possible.

ADV_FSP.1.2C ***The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.***

ADV_FSP.1-2 The evaluator ***shall examine*** the functional specification to determine that the method of use for each SFR-supporting and SFR-enforcing TSFI is given.

588 See work unit ADV_FSP.1-1 for a discussion on the identification of SFR-supporting and SFR-enforcing TSFI.

589 The method of use for a TSFI summarises how the interface is manipulated in order to invoke the operations and obtain the results associated with the TSFI. The evaluator should be able to determine, from reading this material in the functional specification, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each TSFI, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing

the functional specification, as well as by the evaluator evaluating the functional specification.

590 For administrative interfaces whose functionality is documented as being inaccessible to untrusted users, the evaluator ensures that the method of making the functions inaccessible is described in the functional specification. It should be noted that this inaccessibility should be tested by the developer in their test suite.

ADV_FSP.1-3 The evaluator ***shall examine*** the presentation of the TSFI to determine that it identifies all parameters associated with each SFR-enforcing and SFR-supporting TSFI.

591 See work unit ADV_FSP.1-1 for a discussion on the identification of SFR-supporting and SFR-enforcing TSFI.

592 The evaluator examines the functional specification to ensure that all of the parameters are described for identified TSFI. Parameters are explicit inputs or outputs to an interface that control the behaviour of that interface. For examples, parameters are the arguments supplied to an API; the various fields in packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

593 While difficult to obtain much assurance that all parameters for the applicable TSFI have been identified, the evaluator should also check other evidence provided for the evaluation (e.g., operational user guidance) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.1.3C ***The functional specification shall provide rationale for the implicit categorisation of interfaces as SFR-non-interfering.***

ADV_FSP.1-4 The evaluator ***shall examine*** the rationale provided by the developer for the implicit categorisation of interfaces as SFR-non-interfering to determine that it is accurate.

594 In the case where the developer has provided adequate documentation to perform the analysis called for by the rest of the work units for this component without explicitly identifying SFR-enforcing and SFR-supporting interfaces, this work unit should be considered satisfied.

595 This work unit is intended to apply to cases where the developer has not described a portion of the TSFI, claiming that it is SFR-non-interfering and therefore not subject to other requirements of this component. In such a case, the developer provides a rationale for this characterisation in sufficient detail such that the evaluator understands the rationale, the characteristics of the interfaces affected (e.g., their high-level function with respect to the TOE, such as “colour palette manipulation"), and that the claim that these are SFR-non-interfering is supported. Given the level of assurance the evaluator should not expect more detail than is provided for the non-SFR-non-interfering interfaces, and in fact the detail should be much less. In most

cases, individual interfaces should not need to be addressed in the developer-provided rationale section.

11.4.1.4 Action ADV_FSP.1.2E

ADV_FSP.1-5 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the TOE security functional requirements.

596 To ensure that all ST security functional requirements are covered by the functional specification, as well as the test coverage analysis, the evaluator may construct a map between the TOE security functional requirements and the TSFI. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

597 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were covered by three different TSFI, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to TSFI A, B, and C and claim they had completed the work unit. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to TSFI A; FDP_ACC.1 Access control (rule 2) to TSFI B; etc. It might also be the case that the interface is a wrapper interface (e.g., IOCTL), in which case the mapping would need to be specific to certain set of parameters for a given interface.

598 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that they completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (TOE design (ADV_TDS)) when included in the ST. It is also important to note that since the parameters, operations, and error messages associated with TSFIs must be fully specified, the evaluator should be able to determine if all aspects of an SFR appear to be implemented at the interface level.

ADV_FSP.1-6 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the TOE security functional requirements.

599 For each functional requirement in the ST that results in effects visible at the TSF boundary, the information in the associated TSFI for that requirement specifies the required functionality described by the requirement. For example, if the ST contains a requirement for access control lists, and the only TSFI that map to that requirement specify functionality for Unix-style protection bits, then the functionality specification is not accurate with respect to the requirements.

600 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that the evaluator completely map those

requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (ADV_TDS) when included in the ST.

11.4.2 Evaluation of sub-activity (ADV_FSP.2)

11.4.2.1 Objectives

601 The objective of this sub-activity is to determine whether the developer has provided a description of the TSFIs in terms of their purpose, method of use, and parameters. In addition, the SFR-enforcing operations, results and error messages of each TSFI that is SFR-enforcing are also described.

11.4.2.2 Input

602 The evaluation evidence for this sub-activity that is required by the work-units is:

a) the ST;

b) the functional specification;

c) the TOE design.

603 The evaluation evidence for this sub-activity that is used if included in the ST for the TOE is:

a) the architectural design document;

b) the operational user guidance;

11.4.2.3 Action ADV_FSP.2.1E

ADV_FSP.2.1C ***The functional specification shall completely represent the TSF.***

ADV_FSP.2-1 The evaluator ***shall examine*** the functional specification to determine that the TSF is fully represented.

604 The identification of the TSFI is a necessary prerequisite to all other activities in this sub-activity. The TSF must be identified (done as part of the TOE design (ADV_TDS) work units) in order to identify the TSFI. This activity can be done at a high level to ensure that no large groups of interfaces have been missed (network protocols, hardware interfaces, configuration files), or at a low level as the evaluation of the functional specification proceeds.

605 In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of TOE design (ADV_TDS) analysis, but is implicitly performed (through identification and description of the TSFI) by the developer in cases where ADV_TDS is not included in the assurance package. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

a) The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST, or contributes to TSF run-time initialisation, self-protection, domain isolation, and non-bypassability. This is typically all software that 1) runs prior to the TSF being able to protect itself in interactions with untrusted entities (e.g., while booting up); 2) runs in a privileged state (of the underlying hardware or software environment) that allows access to code or data depended on for correct operation of the security functionality; and 3) runs with no privileges with respect to other portions of the system, but either implements or directly supports security functionality.

b) The software used by administrators in order to perform security management activities specified in the guidance documentation.

606 Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI includes interfaces to the security services/resources the TSF is providing. This is especially relevant for TSFs that have a reliance on the non-TSF hardware and/or software, because not only is the TSF providing security services (and thus exposing TSFI), but it is also using services outside of the TSF. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence (for integration and analysis purposes), and thus documentation requirements for these interfaces are specified in Reliance of dependent component (ACO_REL).

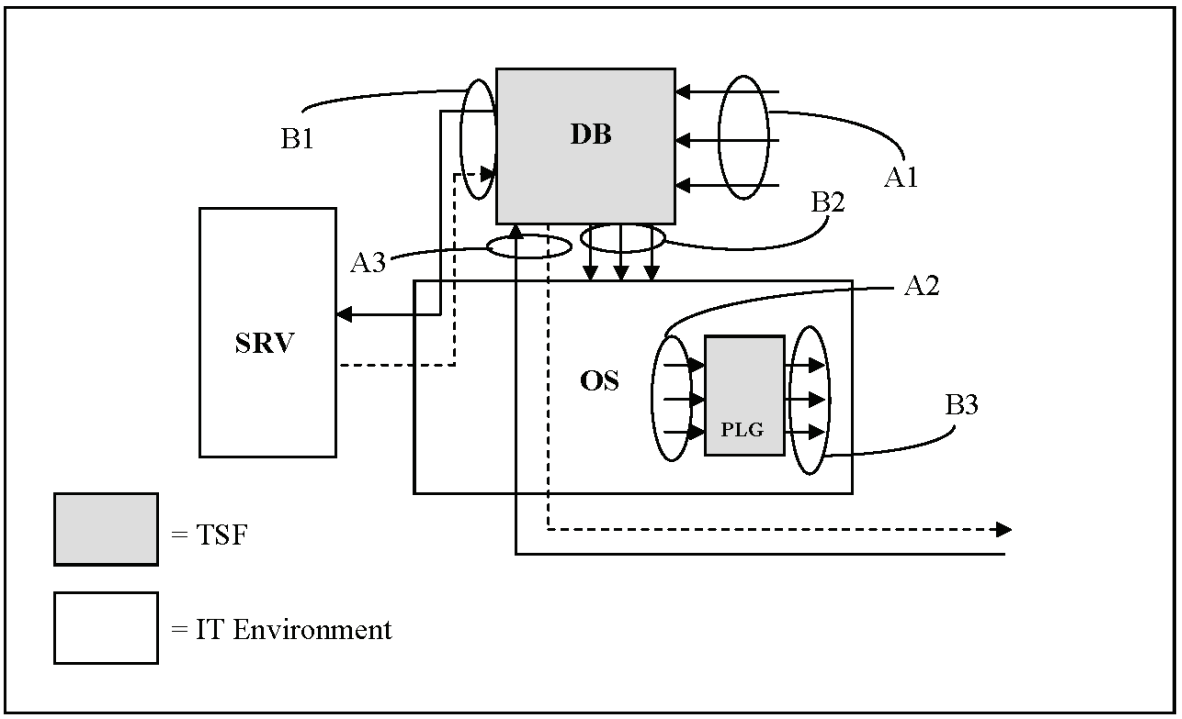

Figure 6 - Interfaces in a DBMS system

607 Figure 6 illustrates a TOE (a database management system) that relies on hardware and software that is not part of the TSF (referred to as the "IT

environment" in the rest of this discussion). The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labelled *OS*) itself, as well as an external server (labelled *SRV*). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *Ax* for TSFI, and *Bx* for interfaces to be documented in Reliance of dependent component (ACO_REL). Each of these groups of interfaces is now discussed.

608 Interface group A1 represent the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

609 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

610 Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

611 Note that the *B* interfaces are not TSFI; requirements related to these interfaces are covered in work units for the Reliance of dependent component (ACO_REL) family.

612 Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion puts the TSFI into the two categories mentioned previously: non-administrative interfaces to the TSF (such as those used by users, the TSF itself, and programs) and TSFI used by administrators.

613 TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the *.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

614 TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an additional requirement that the functions that an administrator uses to perform their duties - as documented in administrative guidance - also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

a) The administrative tool used is also accessible to untrusted users, and runs with some privilege itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.

b) The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (Operational user guidance (AGD_OPE)) are part of the TSFI. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view.

c) The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool.

615 It is also important to note that some TOEs will have interfaces that one might consider part of the TSFI, but environmental factors remove them from consideration. Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration, no firewall-provided services such as telnet).

616 Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the TOE, or they could use a GUI-based tool that essentially translated the GUI-based check boxes, text boxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

617 The term *administrator* above is used in the sense of an entity that has complete trust with respect to all policies implemented by the TSF. There may be entities that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an “administrator”, they need to be treated as

untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

618 Another example of interfaces that are not necessarily TSFI concerns the use of a protection mechanism to remove interfaces from consideration during the evaluation. This might be a set of configuration parameters stored in a database. If the database is inaccessible to untrusted users and administrators are directed not to change the parameters, these parameters need not be considered part of the TSFI.

619 Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a wrapper interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered interfaces. Switches that can change the behaviour of the hardware are also part of the interface.

620 In making an assessment for this work unit, the evaluator determines that all portions of the TSF are addressed in terms of the interfaces listed in the functional specification. All portions of the TSF should have a corresponding interface description, or if there are no corresponding interfaces for a portion of the TSF, the evaluator determines that that is acceptable.

ADV_FSP.2.2C ***The functional specification shall describe the purpose and method of use for all TSFI.***

ADV_FSP.2-2 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each TSFI.

621 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 operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for. The evaluator should not only determine that the purpose exists, but also that it accurately reflects the TSFI by taking into account other information about the interface, such as the description of operations and error messages.

ADV_FSP.2-3 The evaluator ***shall examine*** the functional specification to determine that the method of use for each TSFI is given.

622 The method of use for a TSFI summarises how the interface is manipulated in order to invoke the operations and obtain the results associated with the TSFI. The evaluator should be able to determine, from reading this material in the functional specification, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each

TSFI, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the functional specification, as well as by the evaluator evaluating the functional specification.

623 For administrative interfaces whose functionality is documented as being inaccessible to untrusted users, the evaluator ensures that the method of making the functions inaccessible is described in the functional specification. It should be noted that this inaccessibility should be tested by the developer in their test suite.

624 The evaluator should not only determine that the set of method of use descriptions exist, but also that they accurately cover each TSFI.

ADV_FSP.2.3C ***The functional specification shall identify and describe all parameters associated with each TSFI.***

ADV_FSP.2-4 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely identifies all parameters associated with every TSFI.

625 The evaluator examines the functional specification to ensure that all of the parameters are described for each TSFI. Parameters are explicit inputs or outputs to an interface that control the behaviour of that interface. For examples, parameters are the arguments supplied to an API; the various fields in packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

626 In order to determine that all of the parameters are present in the TSFI, the evaluator should examine the rest of the interface description (operations, error messages, etc.) to determine if the effects of the parameter are accounted for in the description. The evaluator should also check other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.2-5 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all parameters associated with every TSFI.

627 Once all of the parameters have been identified, the evaluator needs to ensure that they are accurately described, and that the description of the parameters is complete. A parameter description tells what the parameter is in some meaningful way. For instance, the interface *foo(i)* could be described as having “parameter i which is an integer"; this is not an acceptable parameter description. A description such as “parameter i is an integer that indicates the number of users currently logged in to the system” is much more acceptable.

628 In order to determine that the description of the parameters is complete, the evaluator should examine the rest of the interface description (purpose, method of use, operations, error messages, etc.) to determine if the descriptions of the parameter(s) are accounted for in the description. The evaluator should also check other evidence provided (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.2.4C ***For SFR-enforcing TSFIs, the functional specification shall describe the SFR-enforcing operations associated with the TSFI.***

ADV_FSP.2-6 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes the SFR-enforcing operations associated with the SFR-enforcing TSFIs.

629 If an operation available through an interface plays a role in enforcing any security policy on the TOE (that is, if one of the operations 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 operations and results, some of which may be SFR-enforcing and some of which may not.

630 The developer is not required to "label" interfaces as SFR-enforcing, and likewise is not required to identify operations available through an interface as SFR-enforcing. It is the evaluator's responsibility to examine the evidence provided by the developer and determine that the required information is present. In the case where the developer has identified the SFR-enforcing TSFI and SFR-enforcing operations available through those TSFI, the evaluator must judge completeness and accuracy based on other information supplied for the evaluation (e.g., TOE design, architectural design, operational user guidance), and on the other information presented for the interfaces (parameters and parameter descriptions, error messages, etc.).

631 In this case (where the developer has provided only the SFR-enforcing information for SFR-enforcing TSFI) the evaluator also ensures that no interfaces have been mis-categorised. This is done by examining other information supplied for the evaluation (e.g., TOE design, architectural design, operational user guidance), and the other information presented for the interfaces (parameters and parameter descriptions, for example) not labelled as SFR-enforcing.

632 In the case where the developer has provided the same level of information on all interfaces, the evaluator performs the same type of analysis mentioned in the previous paragraphs. The evaluator should determine which interfaces are SFR-enforcing and which are not, and subsequently ensure that the SFR-enforcing aspects of the SFR-enforcing operations are appropriately described.

633 The SFR-enforcing operations 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 operations would be SFR-enforcing and therefore must be described, even if the result of that operation 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).

634 The level of description that is required is that sufficient for the reader to understand what role the TSFI operations play with respect to the SFR. The evaluator should keep in mind that the description should be detailed enough to support the generation (and assessment) of test cases against that interface. If the description is unclear or lacking detail such that meaningful testing cannot be conducted against the TSFI, it is likely that the description is inadequate.

ADV_FSP.2.5C ***For SFR-enforcing TSFIs, the functional specification shall describe error messages resulting from processing associated with the SFR-enforcing operations.***

ADV_FSP.2-7 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes error messages that may result from SFR-enforcing operations associated with each SFR-enforcing TSFI.

635 This work unit should be performed in conjunction with, or after, work unit ADV_FSP.2-6 in order to ensure the set of SFR-enforcing TSFI and SFR-enforcing operations is correctly identified. The developer may provide more information than is required (for example, all error messages associated with each interface), in which the case the evaluator should restrict their assessment of completeness and accuracy to only those that they determine to be associated with SFR-enforcing operations of SFR-enforcing TSFI.

636 Errors can take many forms, depending on the interface being described. 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.

637 Errors (and the associated error messages) come about through the invocation of an interface. The processing that occurs in response to the interface invocation may encounter error conditions, which trigger (through an implementation-specific mechanism) an error message to be generated. In some instances this may be a return value from the interface itself; in other instances a global value may be set and checked after the invocation of an interface. It is likely that a TOE will have a number of low-level error messages that may result from fundamental resource conditions, such as “disk full” or “resource locked”. While these error messages may map to a large number of TSFI, they could be used to detect instances where detail from an interface description has been omitted. For instance, a TSFI that produces a “disk full” message, but has no obvious description of why that

TSFI should cause an access to the disk in its description of operations, might cause the evaluator to examine other evidence (Architectural design (ADV_ARC), TOE design (ADV_TDS)) related that TSFI to determine if the description is accurate.

638 In order to determine that the description of the error messages of a TSFI is accurate and complete, the evaluator measures the interface description against the other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance), as well as other evidence available for that TSFI (parameters, analysis from work unit ADV_FSP.2-6).

11.4.2.4 Action ADV_FSP.2.2E

ADV_FSP.2-8 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the TOE security functional requirements.

639 To ensure that all ST security functional requirements are covered by the functional specification, as well as the test coverage analysis, the evaluator may construct a map between the TOE security functional requirements and the TSFI. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

640 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were covered by three different TSFI, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to TSFI A, B, and C and claim they had completed the work unit. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to TSFI A; FDP_ACC.1 Access control (rule 2) to TSFI B; etc. It might also be the case that the interface is a wrapper interface (e.g., IOCTL), in which case the mapping would need to be specific to certain set of parameters for a given interface.

641 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that they completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (TOE design (ADV_TDS)) when included in the ST. It is also important to note that since the parameters, operations, and error messages associated with TSFIs must be fully specified, the evaluator should be able to determine if all aspects of an SFR appear to be implemented at the interface level.

ADV_FSP.2-9 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the TOE security functional requirements.

642 For each functional requirement in the ST that results in effects visible at the TSF boundary, the information in the associated TSFI for that requirement specifies the required functionality described by the requirement. For

example, if the ST contains a requirement for access control lists, and the only TSFI that map to that requirement specify functionality for Unix-style protection bits, then the functionality specification is not accurate with respect to the requirements.

643 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that the evaluator completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (ADV_TDS) when included in the ST.

11.4.3 Evaluation of sub-activity (ADV_FSP.3)

11.4.3.1 Objectives

644 The objective of this sub-activity is to determine whether the developer has provided a description of the TSFIs in terms of their purpose, method of use, and parameters. In addition, the operations, results and error messages of each TSFI are also described sufficiently that it can be determined whether they are SFR-enforcing, with the SFR-enforcing TSFI being described in more detail than other TSFIs.

11.4.3.2 Input

645 The evaluation evidence for this sub-activity that is required by the work-units is:

a) the ST;

b) the functional specification;

c) the TOE design.

646 The evaluation evidence for this sub-activity that is used if included in the ST for the TOE is:

a) the architectural design document;

b) the implementation representation;

c) the TSF internals description;

d) the operational user guidance;

11.4.3.3 Action ADV_FSP.3.1E

ADV_FSP.3.1C ***The functional specification shall completely represent the TSF.***

ADV_FSP.3-1 The evaluator ***shall examine*** the functional specification to determine that the TSF is fully represented.

647 The identification of the TSFI is a necessary prerequisite to all other activities in this sub-activity. The TSF must be identified (done as part of the TOE design (ADV_TDS) work units) in order to identify the TSFI. This activity can be done at a high level to ensure that no large groups of interfaces have been missed (network protocols, hardware interfaces, configuration files), or at a low level as the evaluation of the functional specification proceeds.

648 In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of TOE design (ADV_TDS) analysis, but is implicitly performed (through identification and description of the TSFI) by the developer in cases where ADV_TDS is not included in the assurance package. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

a) The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST, or contributes to TSF run-time initialisation, self-protection, domain isolation, and non-bypassability. This is typically all software that 1) runs prior to the TSF being able to protect itself in interactions with untrusted entities (e.g., while booting up); 2) runs in a privileged state (of the underlying hardware or software environment) that allows access to code or data depended on for correct operation of the security functionality; and 3) runs with no privileges with respect to other portions of the system, but either implements or directly supports security functionality.

b) The software used by administrators in order to perform security management activities specified in the guidance documentation.

649 Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI includes interfaces to the security services/resources the TSF is providing. This is especially relevant for TSFs that have a reliance on the non-TSF hardware and/or software, because not only is the TSF providing security services (and thus exposing TSFI), but it is also using services outside of the TSF. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence (for integration and analysis purposes), and thus documentation requirements for these interfaces are specified in Reliance of dependent component (ACO_REL).

650 Figure 6 illustrates a TOE (a database management system) that relies on hardware and software that is not part of the TSF (referred to as the “IT environment” in the rest of this discussion). The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is

providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labelled *OS*) itself, as well as an external server (labelled *SRV*). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *Ax* for TSFI, and *Bx* for interfaces to be documented in Reliance of dependent component (ACO_REL). Each of these groups of interfaces is now discussed.

651 Interface group A1 represent the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

652 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

653 Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

654 Note that the *B* interfaces are not TSFI; requirements related to these interfaces are covered in work units for the Reliance of dependent component (ACO_REL) family.

655 Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion puts the TSFI into the two categories mentioned previously: non-administrative interfaces to the TSF (such as those used by users, the TSF itself, and programs) and TSFI used by administrators.

656 TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the *.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

657 TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an additional requirement that the functions that an administrator uses to perform their duties - as documented in administrative guidance - also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

a) The administrative tool used is also accessible to untrusted users, and runs with some privilege itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.

b) The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (Operational user guidance (AGD_OPE)) are part of the TSFI. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view.

c) The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool.

658 It is also important to note that some TOEs will have interfaces that one might consider part of the TSFI, but environmental factors remove them from consideration. Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration, no firewall-provided services such as telnet).

659 Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the TOE, or they could use a GUI-based tool that essentially translated the GUI-based check boxes, text boxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

660 The term *administrator* above is used in the sense of an entity that has complete trust with respect to all policies implemented by the TSF. There may be entities that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an “administrator”, they need to be treated as untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

661 Another example of interfaces that are not necessarily TSFI concerns the use of a protection mechanism to remove interfaces from consideration during the evaluation. This might be a set of configuration parameters stored in a database. If the database is inaccessible to untrusted users and administrators are directed not to change the parameters, these parameters need not be considered part of the TSFI.

662 Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a wrapper interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered interfaces. Switches that can change the behaviour of the hardware are also part of the interface.

663 In making an assessment for this work unit, the evaluator determines that all portions of the TSF are addressed in terms of the interfaces listed in the functional specification. All portions of the TSF should have a corresponding interface description, or if there are no corresponding interfaces for a portion of the TSF, the evaluator determines that that is acceptable.

ADV_FSP.3.2C ***The functional specification shall describe the purpose and method of use for all TSFI.***

ADV_FSP.3-2 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each TSFI.

664 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 operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for. The evaluator should not only determine that the purpose exists, but also that it accurately reflects the TSFI by taking into account other information about the interface, such as the description of operations and error messages.

ADV_FSP.3-3 The evaluator ***shall examine*** the functional specification to determine that the method of use for each TSFI is given.

665 The method of use for a TSFI summarises how the interface is manipulated in order to invoke the operations and obtain the results associated with the TSFI. The evaluator should be able to determine, from reading this material in the functional specification, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each TSFI, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the functional specification, as well as by the evaluator evaluating the functional specification.

666 For administrative interfaces whose functionality is documented as being inaccessible to untrusted users, the evaluator ensures that the method of making the functions inaccessible is described in the functional specification. It should be noted that this inaccessibility should be tested by the developer in their test suite.

667 The evaluator should not only determine that the set of method of use descriptions exist, but also that they accurately cover each TSFI.

ADV_FSP.3.3C ***The functional specification shall identify and describe all parameters associated with each TSFI.***

ADV_FSP.3-4 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely identifies all parameters associated with every TSFI.

668 The evaluator examines the functional specification to ensure that all of the parameters are described for each TSFI. Parameters are explicit inputs or outputs to an interface that control the behaviour of that interface. For examples, parameters are the arguments supplied to an API; the various fields in packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

669 In order to determine that all of the parameters are present in the TSFI, the evaluator should examine the rest of the interface description (operations, error messages, etc.) to determine if the effects of the parameter are accounted for in the description. The evaluator should also check other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.3-5 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all parameters associated with every TSFI.

670 Once all of the parameters have been identified, the evaluator needs to ensure that they are accurately described, and that the description of the parameters is complete. A parameter description tells what the parameter is in some meaningful way. For instance, the interface *foo(i)* could be described as having “parameter i which is an integer"; this is not an acceptable parameter description. A description such as “parameter i is an integer that indicates the number of users currently logged in to the system” is much more acceptable.

671 In order to determine that the description of the parameters is complete, the evaluator should examine the rest of the interface description (purpose, method of use, operations, error messages, etc.) to determine if the descriptions of the parameter(s) are accounted for in the description. The evaluator should also check other evidence provided (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.3.4C ***For SFR-enforcing TSFIs, the functional specification shall describe the SFR-enforcing operations associated with the TSFI.***

ADV_FSP.3-6 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes the SFR-enforcing operations associated with the SFR-enforcing TSFIs.

672 If an operation available through an interface plays a role in enforcing any security policy on the TOE (that is, if one of the operations 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 operations and results, some of which may be SFR-enforcing and some of which may not.

673 The developer is not required to "label" interfaces as SFR-enforcing, and likewise is not required to identify operations available through an interface as SFR-enforcing. It is the evaluator's responsibility to examine the evidence provided by the developer and determine that the required information is present. In the case where the developer has identified the SFR-enforcing TSFI and SFR-enforcing operations available through those TSFI, the evaluator must judge completeness and accuracy based on other information supplied for the evaluation (e.g., TOE design, architectural design, operational user guidance), and on the other information presented for the interfaces (parameters and parameter descriptions, error messages, etc.).

674 In this case (developer has provided only the SFR-enforcing information for SFR-enforcing TSFI) the evaluator also ensures that no interfaces have been mis-categorised. This is done by examining other information supplied for the evaluation (e.g., TOE design, architectural design, operational user guidance), and the other information presented for the interfaces (parameters and parameter descriptions, for example) not labelled as SFR-enforcing. The analysis done for work units ADV_FSP.3-7 and ADV_FSP.3-8 are also used in making this determination.

675 In the case where the developer has provided the same level of information on all interfaces, the evaluator performs the same type of analysis mentioned in the previous paragraphs. The evaluator should determine which interfaces are SFR-enforcing and which are not, and subsequently ensure that the SFR-enforcing aspects of the SFR-enforcing operations are appropriately described. Note that in this case, the evaluator should be able to perform the bulk of the work associated with work unit ADV_FSP.3-8 in the course of performing this SFR-enforcing analysis.

676 The SFR-enforcing operations 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 operations would be SFR-enforcing and therefore must be described, even if the result of that operation 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).

677 The level of description that is required is that sufficient for the reader to understand what role the TSFI operations play with respect to the SFR. The evaluator should keep in mind that the description should be detailed enough to support the generation (and assessment) of test cases against that interface. If the description is unclear or lacking detail such that meaningful testing cannot be conducted against the TSFI, it is likely that the description is inadequate.

ADV_FSP.3.5C ***For SFR-enforcing TSFIs, the functional specification shall describe error messages resulting from invocation of the TSFI.***

ADV_FSP.3-7 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes error messages that may result from an invocation of each SFR-enforcing TSFI.

678 This work unit should be performed in conjunction with, or after, work unit ADV_FSP.3-6 in order to ensure the set of SFR-enforcing TSFI is correctly identified. The evaluator should note that the requirement and associated work unit is that all error messages associated with an SFR-enforcing TSFI must be described, not just those associated with SFR-enforcing operations. This is because at this level of assurance, the “extra” information provided by the error message descriptions should be used in determining whether all of the SFR-enforcing aspects of an interface have been appropriately described. For instance, if an error message associated with a TSFI (e.g., “access denied") indicated that an SFR-enforcing decision or operation had taken place, but in the description of the SFR-enforcing operations there was no mention of that particular SFR-enforcing mechanism, then the description may not be complete.

679 Errors can take many forms, depending on the interface being described. 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.

680 Errors (and the associated error messages) come about through the invocation of an interface. The processing that occurs in response to the interface invocation may encounter error conditions, which trigger (through an implementation-specific mechanism) an error message to be generated. In some instances this may be a return value from the interface itself; in other instances a global value may be set and checked after the invocation of an interface. It is likely that a TOE will have a number of low-level error messages that may result from fundamental resource conditions, such as “disk full” or “resource locked". While these error messages may map to a large number of TSFI, they could be used to detect instances where detail from an interface description has been omitted. For instance, a TSFI that produces a “disk full" message, but has no obvious description of why that

TSFI should cause an access to the disk in its description of operations, might cause the evaluator to examine other evidence (Architectural design (ADV_ARC), TOE design (ADV_TDS)) related that TSFI to determine if the description is accurate.

681 In order to determine that the description of the error messages of a TSFI is accurate and complete, the evaluator measures the interface description against the other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance), as well as for other evidence supplied for that TSFI (description of SFR-enforcing operations, summary of non-SFR-enforcing operations and results).

ADV_FSP.3.6C ***The functional specification shall summarise the non-SFR-enforcing operations associated with each TSFI.***

ADV_FSP.3-8 The evaluator ***shall examine*** the presentation of the TSFI to determine that it summarises the non-SFR-enforcing operations associated with each TSFI.

682 The purpose of this work unit is to supplement the details about the SFR-enforcing operations (provided in work unit ADV_FSP.3-6) with a summary of the remaining operations (i.e., those that are not SFR-enforcing). This covers *all* non-SFR-enforcing operations, whether invokable through SFR-enforcing TSFI or through non-SFR-enforcing TSFI. Such a summary about all non-SFR-enforcing operations helps to provide a more complete picture of the functions provided by the TSF, and is to be used by the evaluator in determining whether an operation or TSFI may have been mis-categorized.

683 The information to be provided is more abstract than that required for SFR-enforcing operations. While it should still be detailed enough so that the reader can understand what the operation does, the description does not have to be detailed enough to support writing tests against it, for instance. For the evaluator, the key is that the information must be sufficient to make a positive determination that the operation is non-SFR-enforcing. If that level of information is missing, the summary is insufficient and more information must be obtained.

11.4.3.4 Action ADV_FSP.3.2E

ADV_FSP.3-9 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the TOE security functional requirements.

684 To ensure that all ST security functional requirements are covered by the functional specification, as well as the test coverage analysis, the evaluator may construct a map between the TOE security functional requirements and the TSFI. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

685 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the

FDP_ACC.1 Access control assignment, and these ten rules were covered by three different TSFI, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to TSFI A, B, and C and claim they had completed the work unit. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to TSFI A; FDP_ACC.1 Access control (rule 2) to TSFI B; etc. It might also be the case that the interface is a wrapper interface (e.g., IOCTL), in which case the mapping would need to be specific to certain set of parameters for a given interface.

686 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that they completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (TOE design (ADV_TDS)) when included in the ST. It is also important to note that since the parameters, operations, and error messages associated with TSFIs must be fully specified, the evaluator should be able to determine if all aspects of an SFR appear to be implemented at the interface level.

ADV_FSP.3-10 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the TOE security functional requirements.

687 For each functional requirement in the ST that results in effects visible at the TSF boundary, the information in the associated TSFI for that requirement specifies the required functionality described by the requirement. For example, if the ST contains a requirement for access control lists, and the only TSFI that map to that requirement specify functionality for Unix-style protection bits, then the functionality specification is not accurate with respect to the requirements.

688 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that the evaluator completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (ADV_TDS) when included in the ST.

11.4.4 Evaluation of sub-activity (ADV_FSP.4)

11.4.4.1 Objectives

689 The objective of this sub-activity is to determine whether the developer has completely described all of the TSFI in a manner such that the evaluator is able to determine whether the TSFI are completely and accurately described, and appears to implement the security functional requirements of the ST.

11.4.4.2 Input

690 The evaluation evidence for this sub-activity that is required by the work-units is:

a) the ST;

b) the functional specification;

c) the TOE design.

691 The evaluation evidence for this sub-activity that is used if included in the ST for the TOE is:

a) the architectural design document;

b) the implementation representation;

c) the TSF internals description;

d) the operational user guidance;

11.4.4.3 Application notes

692 The functional specification describes the interfaces to the TSF (the TSFI) in a structured manner. Because of the dependency on Evaluation of sub-activity (ADV_TDS.1), the evaluator is expected to have identified the TSF prior to beginning work on this sub-activity. Without firm knowledge of what comprises the TSF, it is not possible to assess the completeness of the TSFI.

693 In performing the various work units included in this family, the evaluator is asked to make assessments of accuracy and completeness of several factors (the TSFI itself, as well as the individual components (parameters, operations, error messages, etc.) of the TSFI). In doing this analysis, the evaluator is expected to use the documentation provided for the evaluation. This includes the ST, the TOE design, and may include other documentation such as the user and administrative guidance, architectural design, implementation representation, and composition information. The documentation should be examined in an iterative fashion. The evaluator may read, for example, in the TOE design how a certain function is implemented, but see no way to invoke that function from the interface. This might cause the evaluator to question the completeness of a particular TSFI description, or whether an interface has been left out of the functional specification altogether. Describing analysis activities of this sort in the ETR is a key method in providing rationale that the work units have been performed appropriately.

694 It should be recognised that there exist functional requirements whose functionality is manifested wholly or in part architecturally, rather than through a specific mechanism. An example of this is the implementation of mechanisms implementing the Residual information protection (FPT_RIP) requirements. Such mechanisms typically are implemented to ensure a behaviour isn't present, which is difficult to test and typically has been verified through analysis. In the cases where such functional requirements are included in the ST, it is expected that the evaluator recognise that there

may be SFRs of this type that have no interfaces, and that this should not be considered a deficiency in the functional specification.

11.4.4.4 Action ADV_FSP.4.1E

ADV_FSP.4.1C ***The functional specification shall completely represent the TSF.***

ADV_FSP.4-1 The evaluator ***shall examine*** the functional specification to determine that the TSF is fully represented.

695 The identification of the TSFI is a necessary prerequisite to all other activities in this sub-activity. The TSF must be identified (done as part of the TOE design (ADV_TDS) work units) in order to identify the TSFI. This activity can be done at a high level to ensure that no large groups of interfaces have been missed (network protocols, hardware interfaces, configuration files), or at a low level as the evaluation of the functional specification proceeds.

696 In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of ADV_TDS analysis, but is implicitly performed (through identification and description of the TSFI) by the developer in cases where ADV_TDS is not included in the assurance package. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

a) The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST, or contributes to TSF run-time initialisation, self-protection, domain isolation, and non-bypassability. This is typically all software that 1) runs prior to the TSF being able to protect itself in interactions with untrusted entities (e.g., while booting up); 2) runs in a privileged state (of the underlying hardware or software environment) that allows access to code or data depended on for correct operation of the security functionality; and 3) runs with no privileges with respect to other portions of the system, but either implements or directly supports security functionality.

b) The software used by administrators in order to perform security management activities specified in the guidance documentation.

697 Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI includes interfaces to the security services/resources the TSF is providing. This is especially relevant for TSFs that have a reliance on the non-TSF hardware and/or software, because not only is the TSF providing security services (and thus exposing TSFI), but it is also using services outside of the TSF. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence (for integration and analysis purposes), and thus documentation requirements for these interfaces are specified in Reliance of dependent component (ACO_REL).

698 Figure 6 illustrates a TOE (a database management system) that relies on hardware and software that is not part of the TSF (referred to as the "IT environment" in the rest of this discussion). The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labelled *OS*) itself, as well as an external server (labelled *SRV*). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *Ax* for TSFI, and *Bx* for interfaces to be documented in Reliance of dependent component (ACO_REL). Each of these groups of interfaces is now discussed.

699 Interface group A1 represent the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

700 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

701 Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

702 Note that the *B* interfaces are not TSFI; requirements related to these interfaces are covered in work units for the Reliance of dependent component (ACO_REL) family.

703 Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion puts the TSFI into the two categories mentioned previously: non-administrative interfaces to the TSF (such as those used by users, the TSF itself, and programs) and TSFI used by administrators.

704 TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the

*.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

705 TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an additional requirement that the functions that an administrator uses to perform their duties - as documented in administrative guidance - also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

a) The administrative tool used is also accessible to untrusted users, and runs with some privilege itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.

b) The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (Operational user guidance (AGD_OPE)) are part of the TSFI. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view.

c) The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool.

706 It is also important to note that some TOEs will have interfaces that one might consider part of the TSFI, but environmental factors remove them from consideration. Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration, no firewall-provided services such as telnet).

707 Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the TOE, or they could use a GUI-based tool that essentially translated the GUI-based check boxes, text boxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

708 The term *administrator* above is used in the sense of an entity that has complete trust with respect to all policies implemented by the TSF. There

may be entities that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an "administrator", they need to be treated as untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

709 Another example of interfaces that are not necessarily TSFI concerns the use of a protection mechanism to remove interfaces from consideration during the evaluation. This might be a set of configuration parameters stored in a database. If the database is inaccessible to untrusted users and administrators are directed not to change the parameters, these parameters need not be considered part of the TSFI.

710 Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a wrapper interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered interfaces. Switches that can change the behaviour of the hardware are also part of the interface.

711 In making an assessment for this work unit, the evaluator determines that all portions of the TSF are addressed in terms of the interfaces listed in the functional specification. All portions of the TSF should have a corresponding interface description, or if there are no corresponding interfaces for a portion of the TSF, the evaluator determines that that is acceptable.

ADV_FSP.4.2C ***The functional specification shall describe the purpose and method of use for all TSFI.***

ADV_FSP.4-2 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each TSFI.

712 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 operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for. The evaluator should not only determine that the purpose exists, but also that it accurately reflects the TSFI by taking into account other information about the interface, such as the description of operations and error messages.

ADV_FSP.4-3 The evaluator ***shall examine*** the functional specification to determine that the method of use for each TSFI is given.

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

TSFI. The evaluator should be able to determine, from reading this material in the functional specification, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each TSFI, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the functional specification, as well as by the evaluator evaluating the functional specification.

714 For administrative interfaces whose functionality is documented as being inaccessible to untrusted users, the evaluator ensures that the method of making the functions inaccessible is described in the functional specification. It should be noted that this inaccessibility should be tested by the developer in their test suite.

715 The evaluator should not only determine that the set of method of use descriptions exist, but also that they accurately cover each TSFI.

ADV_FSP.4.3C ***The functional specification shall identify and describe all parameters associated with each TSFI.***

ADV_FSP.4-4 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely identifies all parameters associated with every TSFI.

716 The evaluator examines the functional specification to ensure that all of the parameters are described for each TSFI. Parameters are explicit inputs or outputs to an interface that control the behaviour of that interface. For examples, parameters are the arguments supplied to an API; the various fields in packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

717 In order to determine that all of the parameters are present in the TSFI, the evaluator should examine the rest of the interface description (operations, error messages, etc.) to determine if the effects of the parameter are accounted for in the description. The evaluator should also check other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.4-5 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all parameters associated with every TSFI.

718 Once all of the parameters have been identified, the evaluator needs to ensure that they are accurately described, and that the description of the parameters is complete. A parameter description tells what the parameter is in some meaningful way. For instance, the interface *foo(i)* could be described as

having “parameter i which is an integer"; this is not an acceptable parameter description. A description such as “parameter i is an integer that indicates the number of users currently logged in to the system” is much more acceptable.

719 In order to determine that the description of the parameters is complete, the evaluator should examine the rest of the interface description (purpose, method of use, operations, error messages, etc.) to determine if the descriptions of the parameter(s) are accounted for in the description. The evaluator should also check other evidence provided (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.4.4C ***The functional specification shall describe all operations associated with each TSFI.***

ADV_FSP.4-6 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all operations associated with every TSFI.

720 The evaluator checks to ensure that all of the operations are described. Operations available through an interface describe what the interface does (as opposed to the TOE design, which describe how the operations are provided by the TSF).

721 Operations of an interface describe functionality that can be invoked through the interface, and can be categorised as *regular* operations, and *SFR-related* operations. Regular operations are descriptions of what the interface does. The amount of information provided for this description is dependant on the complexity of the interface. The SFR-related operations are those that are visible at any external interface (for instance, audit activity caused by the invocation of an interface (assuming audit requirements are included in the ST) should be described, even though the result of that operation is generally not visible through the invoked interface). Depending on the parameters of an interface, there may be many different operations able to be invoked through the interface (for instance, an API might have the first parameter be a “subcommand”, and the following parameters be specific to that subcommand. The IOCTL API in some Unix systems is an example of such an interface).

722 In order to determine that the description of the operations of a TSFI are complete, the evaluator should review the rest of the interface description (parameter descriptions, error messages, etc.) to determine if the operations described are accounted for. The evaluator should also analyse other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if there is evidence of operations that are described there but not in the functional specification.

ADV_FSP.4.5C ***The functional specification shall describe all error messages that may result from an invocation of each TSFI.***

ADV_FSP.4-7 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all errors messages resulting from an invocation of each TSFI.

723 Errors can take many forms, depending on the interface being described. 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.

724 Errors (and the associated error messages) come about through the invocation of an interface. The processing that occurs in response to the interface invocation may encounter error conditions, which trigger (through an implementation-specific mechanism) an error message to be generated. In some instances this may be a return value from the interface itself; in other instances a global value may be set and checked after the invocation of an interface. It is likely that a TOE will have a number of low-level error messages that may result from fundamental resource conditions, such as "disk full" or "resource locked". While these error messages may map to a large number of TSFI, they could be used to detect instances where detail from an interface description has been omitted. For instance, a TSFI that produces a "disk full" message, but has no obvious description of why that TSFI should cause an access to the disk in its description of operations, might cause the evaluator to examine other evidence (Architectural design (ADV_ARC), TOE design (ADV_TDS)) related that TSFI to determine if the description is complete and accurate.

725 The evaluator determines that, for each TSFI, the exact set of error messages that can be returned on invoking that interface can be determined. The evaluator reviews the evidence provided for the interface to determine if the set of errors seems complete. They cross-check this information with other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to ensure that there are no errors steaming from processing mentioned that are not included in the functional specification.

ADV_FSP.4-8 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes the meaning of all errors associated with each TSFI.

726 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)".

727 In order to determine that the description of the errors due to invoking a TSFI are complete, the evaluator examines the rest of the interface description (parameter descriptions, operations, etc.) to determine if potential error conditions that might be caused by using such an interface are accounted for. The evaluator also checks other evidence provided for the evaluation (e.g. TOE design, architectural design, operational user guidance, implementation representation) to see if error processing related to the TSFI is described there but is not described in the functional specification.

11.4.4.5 Action ADV_FSP.4.2E

ADV_FSP.4-9 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the TOE security functional requirements.

728 To ensure that all ST security functional requirements are covered by the functional specification, as well as the test coverage analysis, the evaluator may construct a map between the TOE security functional requirements and the TSFI. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

729 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were covered by three different TSFI, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to TSFI A, B, and C and claim they had completed the work unit. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to TSFI A; FDP_ACC.1 Access control (rule 2) to TSFI B; etc. It might also be the case that the interface is a wrapper interface (e.g., IOCTL), in which case the mapping would need to be specific to certain set of parameters for a given interface.

730 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that they completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (TOE design (ADV_TDS)) when included in the ST. It is also important to note that since the parameters, operations, and error messages associated with TSFIs must be fully specified, the evaluator should be able to determine if all aspects of an SFR appear to be implemented at the interface level.

ADV_FSP.4-10 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the TOE security functional requirements.

731 For each functional requirement in the ST that results in effects visible at the TSF boundary, the information in the associated TSFI for that requirement specifies the required functionality described by the requirement. For example, if the ST contains a requirement for access control lists, and the only TSFI that map to that requirement specify functionality for Unix-style

protection bits, then the functionality specification is not accurate with respect to the requirements.

732 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that the evaluator completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (ADV_TDS) when included in the ST.

11.4.5 Evaluation of sub-activity (ADV_FSP.5)

11.4.5.1 Objectives

733 The objective of this sub-activity is to determine whether the developer has completely described all of the TSFI in a manner such that the evaluator is able to determine whether the TSFI are completely and accurately described, and appears to implement the security functional requirements of the ST. The completeness of the interfaces is judged based upon the implementation representation.

11.4.5.2 Input

734 The evaluation evidence for this sub-activity that is required by the work-units is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the implementation representation.

735 The evaluation evidence for this sub-activity that is used if included in the ST for the TOE is:

a) the architectural design document;

b) the TSF internals description;

c) the formal security policy model;

d) the operational user guidance;

11.4.5.3 Action ADV_FSP.5.1E

ADV_FSP.5.1C ***The functional specification shall completely represent the TSF.***

ADV_FSP.5-1 The evaluator ***shall examine*** the functional specification to determine that the TSF is fully represented.

736 The identification of the TSFI is a necessary prerequisite to all other activities in this sub-activity. The TSF must be identified (done as part of the TOE design (ADV_TDS) work units) in order to identify the TSFI. This activity can be done at a high level to ensure that no large groups of interfaces have been missed (network protocols, hardware interfaces, configuration files), or at a low level as the evaluation of the functional specification proceeds.

737 In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must be identified. This identification is formally a part of ADV_TDS analysis, but is implicitly performed (through identification and description of the TSFI) by the developer in cases where ADV_TDS is not included in the assurance package. In this analysis, a portion of the TOE is considered to be in the TSF under two conditions:

a) The software contributes to the satisfaction of security functionality specified by a functional requirement in the ST, or contributes to TSF run-time initialisation, self-protection, domain isolation, and non-bypassability. This is typically all software that 1) runs prior to the TSF being able to protect itself in interactions with untrusted entities (e.g., while booting up); 2) runs in a privileged state (of the underlying hardware or software environment) that allows access to code or data depended on for correct operation of the security functionality; and 3) runs with no privileges with respect to other portions of the system, but either implements or directly supports security functionality.

b) The software used by administrators in order to perform security management activities specified in the guidance documentation.

738 Identification of the TSFI is a complex undertaking. The TSF is providing services and resources, and so the TSFI includes interfaces to the security services/resources the TSF is providing. This is especially relevant for TSFs that have a reliance on the non-TSF hardware and/or software, because not only is the TSF providing security services (and thus exposing TSFI), but it is also using services outside of the TSF. While these are (using the general term) interfaces between the TSF and the IT environment, they are not TSFI. Nonetheless, it is vital to document their existence (for integration and analysis purposes), and thus documentation requirements for these interfaces are specified in Reliance of dependent component (ACO_REL).

739 Figure 6 illustrates a TOE (a database management system) that relies on hardware and software that is not part of the TSF (referred to as the “IT environment” in the rest of this discussion). The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is

providing security services specified by functional requirements in the ST. The IT environment consists of the operating system (represented by the box labelled *OS*) itself, as well as an external server (labelled *SRV*). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *Ax* for TSFI, and *Bx* for interfaces to be documented in Reliance of dependent component (ACO_REL). Each of these groups of interfaces is now discussed.

740 Interface group A1 represent the prototypical set of TSFI. These are interfaces used to directly access the database and its security functionality and resources.

741 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

742 Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

743 Note that the *B* interfaces are not TSFI; requirements related to these interfaces are covered in work units for the Reliance of dependent component (ACO_REL) family.

744 Having discussed the interfaces in general, the types of TSFI are now discussed in more detail. This discussion puts the TSFI into the two categories mentioned previously: non-administrative interfaces to the TSF (such as those used by users, the TSF itself, and programs) and TSFI used by administrators.

745 TSFI in the first category are varied in their appearance in a TOE. Most commonly interfaces are thought of as those described in terms of Application Programming Interfaces (APIs), such as kernel calls in a Unix-like operating system. However, interfaces also may be described in terms of menu choices, check boxes, and edit boxes in a GUI; parameter files (the *.INI files and the registry for Microsoft Windows systems); and network communication protocols at all levels of the protocol stack.

746 TSFI in the second category are more complex. While there are three cases that need to be considered (discussed below), for all cases there is an additional requirement that the functions that an administrator uses to perform their duties - as documented in administrative guidance - also are part of the TSFI and must be documented and shown to work correctly. The individual cases are as follows:

a) The administrative tool used is also accessible to untrusted users, and runs with some "privilege" itself. In this case the TSFI to be described are similar to those in the first category because the tool itself is privileged.

b) The administrative tool uses the privileges of the invoker to perform its tasks. In this case, the interfaces supporting the activities that the administrator is directed to do by the administrative guidance (Operational user guidance (AGD_OPE)) are part of the TSFI. Other interfaces supported by the tool that the administrator is directed not to use (and thus play no role in supporting the TSP), but that are accessible to non-administrators, are not part of the TSFI because there are no privileges associated with their use. Note that this case differs from the previous one in that the tool does not run with privilege, and therefore is not in and of itself interesting from a security point of view.

c) The administrative tool is only accessible to administrative users. In this case the TSFI are identified in the same manner as the previous case. Unlike the previous case, however, the evaluator ascertains that an untrusted user is unable to invoke the tool.

747 It is also important to note that some TOEs will have interfaces that one might consider part of the TSFI, but environmental factors remove them from consideration. Most of these examples are for TOEs to which untrusted users have restricted access. For example, consider a firewall that untrusted users only have access to via the network interfaces, and further that the network interfaces available only support packet-passing (no remote administration, no firewall-provided services such as telnet).

748 Further suppose that the firewall had a command-line interface that logged-in administrators could use to administer the TOE, or they could use a GUI-based tool that essentially translated the GUI-based check boxes, text boxes, etc., into scripts that invoked the command-line utilities. Finally, suppose that the administrators were directed in the administrative guidance to use the GUI-based tool in administering the firewall. In this case, the command-line interface does not have to be documented because it is inaccessible to untrusted users, and because the administrators are instructed not use it.

749 There may be administrators that are trusted with respect to some policies (e.g., audit) and not to others (e.g., a flow control policy). In these cases, even though the entity may be referred to as an "administrator", they need to be treated as untrusted users with respect to policies to which they have no administrative access. So, in the previous firewall example, if there was an auditor role that was allowed direct log-on to the firewall machine, the command-line interfaces not related to audit are now part of the TSFI, because they are accessible to a user that is not trusted with respect to the policies the interfaces provide access to. The point is that such interfaces need to be addressed in the same manner as previously discussed.

750 Another example of interfaces that are not necessarily TSFI concerns the use of a protection mechanism to remove interfaces from consideration during the evaluation. This might be a set of configuration parameters stored in a database. If the database is inaccessible to untrusted users and administrators are directed not to change the parameters, these parameters need not be considered part of the TSFI.

751 Hardware interfaces exist as well. Functions provided by the BIOS of various devices may be visible through a wrapper interface such as the IOCTLs in a Unix operating system. If the TOE is or includes a hardware device (e.g., a network interface card), the bus interface signals, as well as the interface seen at the network port, must be considered "interfaces". Switches that can change the behaviour of the hardware are also part of the interface.

752 In making an assessment for this work unit, the evaluator determines that all portions of the TSF are addressed in terms of the interfaces listed in the functional specification. All portions of the TSF should have a corresponding interface description, or if there are no corresponding interfaces for a portion of the TSF, the evaluator determines that that is acceptable.

ADV_FSP.5.2C ***The functional specification shall describe the TSFI using a semi-formal style.***

ADV_FSP.5-2 The evaluator ***shall examine*** the functional specification to determine that it is presented using a semiformal style.

753 A semi-formal presentation is characterised by a standardised format with a well-defined syntax that reduces ambiguity that may occur in informal presentations. Since the intent of the semi-formal format is to enhance the reader's ability to understand the presentation, use of certain structured presentation methods (pseudo-code, flow charts, block diagrams) are appropriate, though not required.

754 For the purposes of this activity, the evaluator should ensure that the interface descriptions are formatted in a structured, consistent manner and use common terminology. A semiformal presentation of the interfaces also implies that the level of detail of the presentation for the interfaces is largely consistent across all TSFI. For the functional specification, it is acceptable to refer to external specifications for portions of the interface as long as those external specifications are themselves semiformal.

ADV_FSP.5.3C ***The functional specification shall describe the purpose and method of use for all TSFI.***

ADV_FSP.5-3 The evaluator ***shall examine*** the functional specification to determine that it states the purpose of each TSFI.

755 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 operations and results related to the interface, but rather a statement to help

the reader understand in general what the interface is intended to be used for. The evaluator should not only determine that the purpose exists, but also that it accurately reflects the TSFI by taking into account other information about the interface, such as the description of operations and error messages.

ADV_FSP.5-4 The evaluator ***shall examine*** the functional specification to determine that the method of use for each TSFI is given.

756 The method of use for a TSFI summarises how the interface is manipulated in order to invoke the operations and obtain the results associated with the TSFI. The evaluator should be able to determine, from reading this material in the functional specification, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each TSFI, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the functional specification, as well as by the evaluator evaluating the functional specification.

757 For administrative interfaces whose functionality is documented as being inaccessible to untrusted users, the evaluator ensures that the method of making the functions inaccessible is described in the functional specification. It should be noted that this inaccessibility should be tested by the developer in their test suite.

758 The evaluator should not only determine that the set of method of use descriptions exist, but also that they accurately cover each TSFI.

ADV_FSP.5.4C ***The functional specification shall identify and describe all parameters associated with each TSFI.***

ADV_FSP.5-5 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely identifies all parameters associated with every TSFI.

759 The evaluator examines the functional specification to ensure that all of the parameters are described for each TSFI. Parameters are explicit inputs or outputs to an interface that control the behaviour of that interface. For examples, parameters are the arguments supplied to an API; the various fields in packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; etc.

760 In order to determine that all of the parameters are present in the TSFI, the evaluator should examine the rest of the interface description (operations, error messages, etc.) to determine if the effects of the parameter are accounted for in the description. The evaluator should also check other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour

or additional parameters are described there but not in the functional specification.

ADV_FSP.5.5C ***The functional specification shall describe all operations associated with each TSFI.***

ADV_FSP.5-6 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all parameters associated with every TSFI.

761 Once all of the parameters have been identified, the evaluator needs to ensure that they are accurately described, and that the description of the parameters is complete. A parameter description tells what the parameter is in some meaningful way. For instance, the interface *foo(i)* could be described as having “parameter i which is an integer"; this is not an acceptable parameter description. A description such as “parameter i is an integer that indicates the number of users currently logged in to the system". is much more acceptable.

762 In order to determine that the description of the parameters is complete, the evaluator should examine the rest of the interface description (purpose, method of use, operations, error messages, etc.) to determine if the descriptions of the parameter(s) are accounted for in the description. The evaluator should also check other evidence provided (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if behaviour or additional parameters are described there but not in the functional specification.

ADV_FSP.5-7 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all operations associated with every TSFI.

763 The evaluator checks to ensure that all of the operations are described. Operations available through an interface describe what the interface does (as opposed to the TOE design, which describe how the operations are provided by the TSF).

764 Operations of an interface describe functionality that can be invoked through the interface, and can be categorised as *regular* operations, and *SFR-related* operations. Regular operations are descriptions of what the interface does. The amount of information provided for this description is dependant on the complexity of the interface. The SFR-related operations are those that are visible at any external interface (for instance, audit activity caused by the invocation of an interface (assuming audit requirements are included in the ST) should be described, even though the result of that operation is generally not visible through the invoked interface). Depending on the parameters of an interface, there may be many different operations able to be invoked through the interface (for instance, an API might have the first parameter be a “subcommand”, and the following parameters be specific to that subcommand. The IOCTL API in some Unix systems is an example of such an interface).

765 In order to determine that the description of the operations of a TSFI are complete, the evaluator should review the rest of the interface description (parameter descriptions, error messages, etc.) to determine if the operations described are accounted for. The evaluator should also analyse other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if there is evidence of operations that are described there but not in the functional specification.

ADV_FSP.5.6C ***The functional specification shall describe all error messages that may result from an invocation of each TSFI.***

ADV_FSP.5-8 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes all errors messages resulting from an invocation of each TSFI.

766 Errors can take many forms, depending on the interface being described. 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.

767 Errors (and the associated error messages) come about through the invocation of an interface. The processing that occurs in response to the interface invocation may encounter error conditions, which trigger (through an implementation-specific mechanism) an error message to be generated. In some instances this may be a return value from the interface itself; in other instances a global value may be set and checked after the invocation of an interface. It is likely that a TOE will have a number of low-level error messages that may result from fundamental resource conditions, such as “disk full” or “resource locked". While these error messages may map to a large number of TSFI, they could be used to detect instances where detail from an interface description has been omitted. For instance, a TSFI that produces a “disk full" message, but has no obvious description of why that TSFI should cause an access to the disk in its description of operations, might cause the evaluator to examine other evidence (ADV_ARC, ADV_TDS) related that TSFI to determine if the description is complete and accurate.

768 The evaluator determines that, for each TSFI, the exact set of error messages that can be returned on invoking that interface can be determined. The evaluator reviews the evidence provided for the interface to determine if the set of errors seems complete. They cross-check this information with other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to ensure that there are no errors steaming from processing mentioned that are not included in the functional specification.

ADV_FSP.5-9 The evaluator ***shall examine*** the presentation of the TSFI to determine that it completely and accurately describes the meaning of all errors associated with each TSFI.

769 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)".

770 In order to determine that the description of the errors due to invoking a TSFI are complete, the evaluator examines the rest of the interface description (parameter descriptions, operations, etc.) to determine if potential error conditions that might be caused by using such an interface are accounted for. The evaluator also checks other evidence provided for the evaluation (e.g., TOE design, architectural design, operational user guidance, implementation representation) to see if error processing related to the TSFI is described there but is not described in the functional specification.

ADV_FSP.5.7C ***The functional specification shall describe all error messages that do not result from an invocation of a TSFI.***

ADV_FSP.5-10 The evaluator ***shall examine*** the functional specification to determine that it completely and accurately describes all errors messages that do not result from an invocation of any TSFI.

771 This work unit complements work unit ADV_FSP.5-8, which describes those error messages that result from an invocation of the TSFI. Taken together, these work units cover all error messages that might be generated by the TSF.

772 The evaluator assesses the completeness and accuracy of the functional specification by comparing its contents to instances of error message generation within the implementation representation. Most of these error messages will have already been covered by work unit ADV_FSP.5-8.

773 The error messages related to this work unit are typically those that are not expected to be generated, but are constructed as a matter of good programming practises. 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 practise 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.

ADV_FSP.5.8C ***The functional specification shall provide a rationale for each error message contained in the TSF implementation yet does not result from an invocation of a TSFI.***

ADV_FSP.5-11 The evaluator ***shall examine*** the functional specification to determine that it provides a rationale for each error message contained in the TSF implementation yet does not result from an invocation of a TSFI.

774 The evaluator ensures that every error message found under work unit ADV_FSP.5-10 contains a rationale describing why it cannot be invoked from the TSFI.

775 As was described in the previous work unit, this rationale might be as straightforward as the fact that the error message in question is provided for completeness of execution logic and that it is never expected to be generated. The evaluator ensures that the rationale for each such error message is logical.

11.4.5.4 Action ADV_FSP.5.2E

ADV_FSP.5-12 The evaluator ***shall examine*** the functional specification to determine that it is a complete instantiation of the TOE security functional requirements.

776 To ensure that all ST security functional requirements are covered by the functional specification, as well as the test coverage analysis, the evaluator may construct a map between the TOE security functional requirements and the TSFI. Note that this map may have to be at a level of detail “below” the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

777 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were covered by three different TSFI, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to TSFI A, B, and C and claim they had completed the work unit. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to TSFI A; FDP_ACC.1 Access control (rule 2) to TSFI B; etc. It might also be the case that the interface is a wrapper interface (e.g., IOCTL), in which case the mapping would need to be specific to certain set of parameters for a given interface.

778 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that they completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (TOE design (ADV_TDS)) when included in the ST. It is also important to note that since the parameters, operations, and error messages associated with TSFIs must be fully specified, the evaluator should be able to determine if all aspects of an SFR appear to be implemented at the interface level.

ADV_FSP.5-13 The evaluator ***shall examine*** the functional specification to determine that it is an accurate instantiation of the TOE security functional requirements.

779 For each functional requirement in the ST that results in effects visible at the TSF boundary, the information in the associated TSFI for that requirement specifies the required functionality described by the requirement. For example, if the ST contains a requirement for access control lists, and the only TSFI that map to that requirement specify functionality for Unix-style protection bits, then the functionality specification is not accurate with respect to the requirements.

780 The evaluator must recognise that for requirements that have little or no manifestation at the TSF boundary (e.g., Residual information protection (FPT_RIP)) it is not expected that the evaluator completely map those requirements to the TSFI. The analysis for those requirements will be performed in the analysis for the TOE design (ADV_TDS) when included in the ST.

11.4.6 Evaluation of sub-activity (ADV_FSP.6)

781 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

11.5 Implementation representation (ADV_IMP)

11.5.1 Evaluation of sub-activity (ADV_IMP.1)

11.5.1.1 Objectives

782 The objective of this sub-activity is to determine that the implementation representation made available by the developer is suitable for use in other analysis activities; *suitability* is judged by its conformance to the requirements for this component.

11.5.1.2 Input

783 The evaluation evidence for this sub-activity is:

a) the implementation representation;

b) the documentation of the development tools, as resulting from ALC_TAT ;

c) TOE design description.

11.5.1.3 Application notes

784 The entire implementation representation is made available to ensure that analysis activities are not curtailed due to lack of information. This does not, however, imply that all of the representation is examined when the analysis activities are being performed. This is likely impractical in almost all cases, in addition to the fact that it most likely will not result in a higher-assurance TOE vs. targeted sampling of the implementation representation. For this sub-activity, this is even truer. It would not be productive for the evaluator to spend large amounts of time verifying the requirements for one portion of the implementation representation, and then use a different portion of the implementation representation in performing analysis for other work units. Therefore, the evaluator is encouraged to select the sample of the implementation representation from the areas of the TOE that will be of most interest during the analysis performed during work units from other families (e.g. ATE_IND, AVA_VAN and ADV_INT).

11.5.1.4 Action ADV_IMP.1.1E

ADV_IMP.1.1C ***The implementation representation shall define the TSF to a level of detail such that the TSF can be generated without further design decisions.***

ADV_IMP.1-1 The evaluator ***shall check*** that the implementation representation defines the TSF to a level of detail such that the TSF can be generated without further design decisions.

785 Source code or hardware diagrams and/or chip specifications that are used to build the actual hardware are examples of parts of an implementation representation. 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. The evaluator is encouraged to perform a quick check when first looking at the implementation representation to assure themselves that the developer is on the right track. However, the evaluator is also encourage to perform the bulk of this check while working on other work units that call for examining the implementation; this will ensure the sample examined for this work unit is relevant.

ADV_IMP.1.2C ***The implementation representation shall be in the form used by the development personnel.***

ADV_IMP.1-2 The evaluator ***shall check*** that the implementation representation is in the form used by development personnel.

786 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.

787 The evaluator samples the implementation representation to gain confidence that it is the version that is usable by the developer. The sample is such that the evaluator has assurance that all areas of the implementation representation are in conformance with the requirement; however, a complete examination of the entire implementation representation is unnecessary.

788 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.

789 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 (in terms of structure) than the original, un-shrouded

representation, supplying such obfuscated code may cause significantly more time to be spent in analysis tasks involving the representation. When such forms of representation are created, the components require details on the shrouding tools/algorithms used so that the un-shrouded representation can be supplied, and the additional information can be used to gain confidence that the shrouding process does not compromise any security mechanisms.

790 The evaluator samples the implementation representation to gain confidence that all of the information needed to interpret the implementation representation has been supplied. Note that the tools are among those referenced by Tools and techniques (ALC_TAT) components. The evaluator is encouraged to perform a quick check when first looking at the implementation representation to assure themselves that the developer is on the right track. However, the evaluator is also encouraged to perform the bulk of this check while working on other work units that call for examining the implementation; this will ensure the sample examined for this work unit is relevant.

ADV_IMP.1.3C ***The mapping between the TOE design description and the sample of the implementation representation shall demonstrate their correspondence.***

ADV_IMP.1-3 The evaluator ***shall examine*** the mapping between the TOE design description and the sample of the implementation representation to determine that it is accurate.

791 The evaluator augments the determination of existence (specified in work unit ADV_IMP.1-1) by verifying 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 the implementation representation accurately reflects the description provided in the TOE design description.

792 For 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.

793 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.

11.5.2 Evaluation of sub-activity (ADV_IMP.2)

794 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

11.6 TSF internals (ADV_INT)

11.6.1 Evaluation of sub-activity (ADV_INT.1)

11.6.1.1 Objectives

795 The objective of this sub-activity is to determine whether the TSF is designed and structured in a modular fashion that ensures the TSF implementation is not overly complex and facilitates understanding by the developer and the evaluator.

11.6.1.2 Input

796 The evaluation evidence for this sub-activity is:

a) the ST;

b) the TOE design description;

c) the implementation representation;

d) the architectural description;

e) the documentation of the coding standards, as resulting from ALC_TAT.

11.6.1.3 Application notes

797 The role of the architectural description is to provide evidence of modularity of the TSF and to justify the complexity of the design and implementation. The focus of the architectural description is on the appropriateness of inclusion of functions in a module, and on the interaction between modules. The low-level TOE design, on the other hand, describes the design of the modules of the TSF and how those modules work to satisfy the SFRs.

798 The modules identified in the architectural description are the same as the modules identified in the TOE design (TOE design (ADV_TDS)). Some of the information from the TOE design may be applicable in meeting the required content for the architectural description, and can be included by reference.

799 For portions of the TSF implemented in software, a module consists of one or more source code files that cannot be decomposed into smaller compilable units.

800 This explicit component levies stricter modularity constraints on the SFR-enforcing modules than on the non-SFR-enforcing modules. The SFR modules are those modules that play a critical role in enforcing the SFR(s) that are explicitly identified in the assignment ADV_INT.1.4D. While the entire TSF is to be designed using good engineering principles and result in a modular TSF, the non-SFR-enforcing are not required to adhere as strictly to the coupling and cohesion metrics that are defined for SFR-enforcing

modules. The evaluator determines that the design of the TSF and resulting modules are not overly complex and the developer's application of coding standards result in a TSF that is understandable. The intent of this component is for the evaluator to focus on the SFR-enforcing modules and to ensure they exhibit the required characteristics. The evaluator performs limited analysis on the non-SFR-enforcing modules to ensure that these modules are non-SFR-enforcing and that their interaction with SFR-enforcing modules will not adversely affect the SFR-enforcing module's ability to enforce the explicitly assigned SFRs.

801 The primary goal of this component is to ensure the TSF's implementation representation is understandable to facilitate maintenance and analysis (of both the developer and evaluator).

802 Before performing this work unit, the evaluator should have already determined the TSFI that are responsible for enforcing the SFRs in the ST.

11.6.1.4 Action ADV_INT.1.1E

ADV_INT.1.1C ***The TSF internals description shall identify the Assigned SFR-enforcing and non-Assigned SFR-enforcing modules.***

ADV_INT.1-1 The evaluator ***shall check*** the TSF internals description to determine that it identifies the Assigned SFR-enforcing and non-Assigned SFR-enforcing modules.

803 The TSF internals description clearly identifies which modules are the SFR-enforcing that correspond to the SFRs in the ADV_INT.1.4D requirement, and which modules are the non-Assigned SFR-enforcing modules. The evaluator ensures that all modules in the TSF internals description are designated as either an Assigned SFR-enforcing or non-Assigned SFR-enforcing modules.

804 The non-Assigned SFR-enforcing portions of the TSF consist of the SFR-supporting modules, SFR non-interfering modules, and SFR-enforcing modules that do not play a role in the enforcement of the SFR(s) identified in ADV_INT.1.4D as depicted in the figure below, where in this example, non-SFR-enforcing is everything in the TSF other than the assigned SFR-enforcing functions.

805 The assigned SFR-enforcing modules are those modules that interact with the module or modules that provide a TSFI for the assigned SFRs with justified exceptions.

ADV_INT.1-2 The evaluator ***shall examine*** the coupling analysis to determine that the TSF internals description accurately identifies the modules as Assigned SFR-enforcing or non-SFR-enforcing.

806 The parts of the TSF that implement an SFR (e.g. access control, any other policy identified in ADV_INT.1.4D of the PP or ST) are those modules that

interact with the module or modules that provide the TSFIs for that SFR, with justified exceptions.

807 The evaluator uses the coupling analysis and global analysis to determine the interactions of modules. The coupling analysis may include a call tree, identify functions that are passed to other functions via pointers, or in a message passing system identifies the messages that are passed between modules. The global variable analysis identifies the global variables used in the TSF, and identifies the modules and their mode of reference (e.g., write, read) to each global variable.

808 The evaluator uses the coupling analysis provided with the TSF internals description and the module descriptions provided as part of TOE design (ADV_TDS) to ensure that all of the modules that interact with the module that exports the TSFI are correctly identified as SFR-enforcing and the remaining modules are identified as non-SFR-enforcing.

809 The evaluator uses the global variable analysis to determine if modules that are identified as non-SFR-enforcing modules set or write to SFR-enforcing global variables that are read by an SFR-enforcing module. Modules that write to SFR-enforcing global variables (variables that are related to an SFR for the SFRs identified in ADV_INT.1.4D) and are read by the module or modules that provide the TSFIs for that SFR, or read by other modules that are deemed SFR-enforcing because of their interactions with SFR-enforcing modules, are deemed SFR-enforcing. The use of global variables potentially plays as critical a role as making an explicit call to a module. The intent is that all of the modules that play a SFR related role (as opposed to modules that provide infrastructure support, such as scheduling, reading binary data from the disk) in enforcing an SFR are identified as SFR-enforcing.

810 For example, module A presents the *open()* TSFI, which plays a critical role in enforcing the FDP_ACC policy, and calls functions in modules B, C and D. Module B is responsible for making the access control decision (thereby being SFR-enforcing), which in turn calls modules X and Y. Module X is responsible for sorting the access control list, and module Y is called to allocate memory for module B. If the access control policy has a rule concerning the ordering of access control lists, then module X is considered SFR-enforcing. If there is no rule regarding ordering of access control lists (ordering of access control lists makes no difference), then module X can be considered non-SFR-enforcing. If the only role module Y plays is the allocation of memory for module B, then module Y could be considered non-SFR-enforcing (with respect to the FDP_ACC requirements, if a residual data protection policy was identified as an SFR in this component, then module Y may be SFR-enforcing).

ADV_INT.1.2C ***There shall be a one-to-one correspondence between the modules identified in TSF internals description and the modules identified in the modular design description.***

ADV_INT.1-3 The evaluator ***shall check*** the TSF internals description to determine that the identification of modules is identical to the modules that are described in the modular design description.

811 The evaluator determines that the modules identified and described in the modular description are the exact same modules identified in the TSF internals design. Any modules that are identified in either document that are not present in both signify a failure of this work unit.

ADV_INT.1.3C ***The TSF internals description shall provide a justification for the designation of non-Assigned SFR-enforcing modules that interact with the Assigned SFR-enforcing module(s).***

ADV_INT.1-4 The evaluator ***shall examine*** the TSF internals description to determine that the developer provided justification for the designation of non-Assigned SFR-enforcing modules that interact with the Assigned SFR-enforcing modules accurately reflects the module's role in enforcement.

812 The evaluator ensures that the justification for why a module is non-Assigned SFR-enforcing even though it interacts with an Assigned SFR-enforcing module, and therefore is relied upon by an Assigned SFR-enforcing module to perform some function, is reasonable. The justification should discuss that the non-Assigned SFR-enforcing module plays no direct role in satisfying an Assigned SFR or portions of an Assigned SFR. A module that interacts with an Assigned SFR-enforcing module is justifiably designated a non-Assigned SFR-enforcing module if it is only providing infrastructure support.

813 Modules that provide TSFIs that are not Assigned SFR-enforcing do not require justification for their designation, nor do the modules in the call tree for such modules. For example, Modules X, Y, and Z in the example below require no justification. From previous work units (e.g., Functional specification (ADV_FSP)) it has already been determined that the TSFI is non-SFR-enforcing and therefore the modules that implement or provide services to implement the TSFI are also non-SFR-enforcing. On the other hand, Modules D and E exist in the call tree for a module that exports a TSFI that is Assigned SFR-enforcing and therefore those modules potentially are Assigned SFR-enforcing and therefore require a justification (e.g., provides infrastructure support that allocates memory to a module) as to why they are non-Assigned SFR-enforcing. Module F writes to a global variable that is read by Assigned SFR-enforcing modules, therefore it also is potentially an Assigned SFR-enforcing module.

ADV_INT.1.4C ***The TSF internals description shall describe the process used for modular decomposition.***

ADV_INT.1-5 The evaluator ***shall examine*** the TSF internals description to determine that the developer's process for modular decomposition results in a modular TSF design.

814 Modules interact by providing services to one another, or by cooperating in achieving an overall goal.

815 The evaluator assesses the developer's design methodology to determine that the TSF design incorporates sound engineering principles to decompose the TSF into modules. Modules should be designed to include only related functions, to have a single, well-defined purpose, and to avoid unnecessary interactions with other modules. The methodology should describe what are acceptable forms of communication between modules and under what circumstances discouraged forms of coupling and cohesion are considered acceptable. The methodology should also address complexity and what steps are taken to ensure modules are not overly complex. The intent here is that the developer's process should cover/address the developer action elements for the TSF internals (ADV_INT) requirements and this process provides guidance to the developer on how to design and code their modules.

816 The methodology provided by the developer provides the evaluator with an understanding of the developer's approach to designing (or restructuring of) the TSF and aids the evaluator in performing other work units for this component.

ADV_INT.1.5C ***The TSF internals description shall describe how the TSF design is a reflection of the modular decomposition process.***

ADV_INT.1-6 The evaluator ***shall examine*** the TSF internals description to determine that the TSF's resulting design is an accurate reflection of the developer's decomposition process.

817 The evaluator compares the developer's documented process for decomposition presented in the TSF internals description with the implementation representation to determine if the developer followed their process. The evaluator will need to perform this work unit in conjunction with other work units. In fact, the evaluator should understand the process the developer has employed for decomposing their TSF into modules and while performing other work units ensure that process was followed.

ADV_INT.1.6C ***The TSF internals description shall provide a justification, on a per Assigned SFR-enforcing module basis, of any deviations from the coding standards.***

ADV_INT.1-7 The evaluator ***shall examine*** the implementation representation to determine that the developer adhered to the coding standards for the Assigned SFR-enforcing modules.

818 The evaluator ensures that the development of the implementation representation adhered to the coding standards that are included in the development tools (Tools and techniques (ALC_TAT)). A developer may have more than one coding standard for different components of the TOE, and this is acceptable, as long as each standard contains the necessary information. This is accomplished by determining that the format prescribed by the coding standards (e.g., comments, naming conventions, indentation) is

exhibited in the implementation representation. The evaluator ensures that the guidelines for programming logic (e.g., types of loops, control flow, error handling, the use of pointers) are exhibited in the implementation representation. The evaluator determines that other aspects of code development that are defined in the coding standards are followed by the developer.

ADV_INT.1-8 The evaluator ***shall examine*** the TSF internals description to determine if the justification for deviations from the coding standards for each Assigned SFR-enforcing module is reasonable.

819 There may be acceptable occurrences where the implementation representation does not follow the coding standards. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable (unless those modules are non-Assigned SFR-enforcing). Typically, deviations from the prescribed formatting aspects of the coding standards are not justifiable. Acceptable justification for deviation from the coding standards would generally be related to performance or efficiency reasons, and possibly for backward compatibility.

ADV_INT.1-9 The evaluator ***shall examine*** the implementation representation to determine if the deviations from the coding standards for each Assigned SFR-enforcing module do not make the offending modules overly complex.

820 The evaluator assesses the modules that have been identified in the TSF internals description as deviating from the coding standards to ensure the deviations do not make the Assigned SFR-enforcing modules too complex to understand the module's processing and interactions in a reasonable amount of time (e.g., no more than eight hours).

821 A complexity analysis performed by a tool, such as McCabe, may provide information that addresses the programmatic complexity of the module, but here, the evaluator assesses the modules not adhering to the coding standards to ensure the code is readable and readily understood.

ADV_INT.1.7C ***The TSF internals description shall include a coupling analysis that describes intermodule coupling for the Assigned SFR-enforcing modules.***

ADV_INT.1-10 The evaluator ***shall examine*** the coupling analysis to determine that it is a complete and accurate reflection of the implementation representation.

822 The evaluator ensures that the coupling analysis correctly identifies and describes the method of all of the interactions with other modules by examining the implementation representation. The coupling analysis includes any explicit calls made by functions in a module to other modules, the passing of pointers to function to another function, the use of global variables, shared memory, message passing, or any other means that functions may use to communicate with other modules.

823 The evaluator also uses the implementation representation to ensure modules do not reference global variables that are not contained in the coupling analysis and that the references modules make (e.g., read, write) are consistent with the references specified in the coupling analysis.

824 The evaluator makes an independent assessment of the type of coupling that exists between modules and ensures that they arrive at the same conclusion as the presented in the developer's analysis.

ADV_INT.1-11 The evaluator ***shall examine*** the coupling analysis to ensure that Assigned SFR-enforcing modules that exhibit coupling other than call or common coupling are justified.

825 Once the evaluator has completed work unit ADV_INT.1-10 and has determined that the coupling analysis is complete and accurate, the evaluator determines that the instances of coupling other than call or common coupling are justified by the TSF internals description. The evaluator determines if the justifications are reasonable, and therefore allowable in a separate work unit.

ADV_INT.1.8C ***The TSF internals description shall include a cohesion analysis that describes the types of cohesion for the Assigned SFR-enforcing modules.***

ADV_INT.1-12 The evaluator ***shall check*** the TSF internals description to determine it identifies and describes the type of cohesion for each Assigned SFR-enforcing module.

826 The evaluator determines that the cohesion analysis provided in the TSF internals description is complete, that is each module is associated with the type of cohesion that is exhibited, as well as a description of how the determination of the type of cohesion was made by the developer.

827 The evaluator will perform a cohesion analysis in work unit ADV_INT.1-14 and compare their findings to the cohesion analysis provided in the TSF internals description.

ADV_INT.1.9C ***The TSF internals description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by Assigned SFR-enforcing modules, other than those permitted.***

ADV_INT.1-13 The evaluator ***shall examine*** the TSF internals description to determine if the justification for coupling and cohesion for Assigned SFR-enforcing modules other than the ones permitted are reasonable.

828 The evaluator judges the developer's rationale for Assigned SFR-enforcing modules that exhibit coupling and cohesion other than those allowed and determines if the justifications are reasonable. The justification is provided for each module that is not conformant to the allowed forms of coupling and cohesion.

829 Reasons for justification may include efficiency or performance and the evaluator uses their judgement to determine that the justification is

acceptable and that the disallowed forms of cohesion or coupling are not pervasive. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable.

830 In order to assess the justification for either coupling or cohesion forms other than those permitted may require the evaluator to examine the implementation representation. The evaluator confirm the justification is reasonable, by understanding the offending software, and determining that requiring the code to strictly conform to the coupling or cohesion metrics would be unreasonable.

11.6.1.5 Action ADV_INT.1.2E

ADV_INT.1-14 The evaluator ***shall examine*** the implementation representation to determine that the type of cohesion exhibited by a subset of Assigned SFR-enforcing modules is consistent with that claimed in the TSF internals description.

831 Cohesion is a measure of the strength of relationship between the activities performed by a module. Cohesion ranges from the most desirable form, functional cohesion, to the least desirable form, coincidental cohesion. Allowable forms of cohesion include functional, sequential, and communicational cohesion; coincidental cohesion is unacceptable. Temporal cohesion is acceptable only for special-purpose use, such as initialisation, shutdown, and error recovery. Logical cohesion may be acceptable, based on the argument for its justification.

832 The evaluator uses the implementation representation and modular design description to determine the purpose of the modules and the relationship the functions have within a module to achieve that purpose. The processing performed by the functions in a module and the service they provide are an indication of the design in the grouping of functions and what type of cohesion is exhibited.

833 The evaluator performs an analysis to determine the type of cohesion exhibited by the modules and ensures that the evaluator's findings are consistent with the developer's claim in the TSF internals description. It is important to note that there may be some subjectivity involved in making a determination of the type of cohesion exhibited (e.g., sequential vs. temporal), but the type of cohesion that is deemed unacceptable (e.g., coincidental, logical) is more clear cut.

834 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.1.6 Action ADV_INT.1.3E

ADV_INT.1-15 The evaluator ***shall examine*** the implementation representation to determine that a subset of the Assigned SFR-enforcing modules are not overly complex.

835 A metric that simply relies on the numbers of lines of code, or the percentage of comments and statements is considered a weak complexity measure and does provide a good metric of software complexity. More meaningful metrics consider the number of unique operators and operands in the program, or which measures the amount of unstructured decision logic in software by examining cyclomatic complexity.

836 The evaluator determines the complexity of the implementation representation by measuring the length of time it takes the evaluator to understand the logic in a module and the module's relationship with other modules. When using the design documentation and TSF internals description, the evaluator should understand a module's implementation and interactions in a reasonable amount of time. The measurement of time is subjective and is based on the evaluator's experience and skill in reading and understanding source code. It is assumed that the evaluator has knowledge of computer programming, and is familiar with the programming language used by the developer. Given this minimum level of expertise, the amount of time it takes to understand a module's implementation should not exceed eight hours.

837 The evaluator needs to also consider how a module incorporates access to "global entities", such as functions, macros or data. One example of this is the use of header files in the C programming language. Often times header files are nested and a programmer does not understand all of the contents that are potentially accessible by their software. Another poor example of programming is the incorporation of header files without knowing whether the header file is actually needed. The problem arises when a programmer makes a mistake (e.g., mis-types a variable or macro name). The likelihood of the improper variable being referenced or macro being invoked is less if only the necessary header files are included. Having several or more unnecessary header files included increases the likelihood that the mis-typed variable or macro exists in one of those unnecessary header files which could lead to serious consequences.

838 This work unit (as does the previous work unit) specifies that the subset of code the evaluator examines contains all three types of modules (i.e., SFR-enforcing, SFR-supporting, and SFR non-interfering). One might argue that SFR non-interfering modules are unimportant and therefore the evaluator should not be spending any time analysing these types of modules. However, for this very reason it is not clear that developers spend much time analysing these types of modules and therefore these types of modules may be more susceptible to containing flaws. When determining the subset of the SFR non-interfering modules, the evaluator should ensure they include any SFR non-interfering modules that are accessible by untrusted entities.

839 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.2 Evaluation of sub-activity (ADV_INT.2)

11.6.2.1 Objectives

840 The objective of this sub-activity is to determine whether the TSF is designed and structured in a modular fashion that ensures the TSF implementation is not overly complex and facilitates understanding by the developer and the evaluator.

11.6.2.2 Input

841 The evaluation evidence for this sub-activity is:

a) the modular design description;

b) the implementation representation;

c) the TSF internals description;

d) the documentation of the coding standards, as resulting from ALC_TAT.

11.6.2.3 Application notes

842 The role of the TSF internals description is to provide evidence of modularity of the TSF and to justify the complexity of the design and implementation. The focus of the TSF internals description is on how the software is architected from a software engineering perspective, providing information regarding the appropriateness of inclusion of functions in a module, and on the interaction between modules. The modular design description (ADV_TDS), on the other hand, describes the functionality of the modules of the TSF and how those modules work to satisfy the SFRs.

843 The modules identified in the TSF internals description are the same as the modules identified in the modular design description. Some of the information from the modular design description may be applicable in meeting the required content for the TSF internals description, and can be included by reference.

844 For portions of the TSF implemented in software, a module consists of one or more source code files that cannot be decomposed into smaller compilable units.

845 The primary goal of this component is to ensure the TSF's implementation representation is understandable to facilitate maintenance and analysis by the developer (and to a lesser extent the evaluator).

846 In typical CEM fashion, work units are presented that correspond to determining if the developer's content elements are satisfied. There are two additional work units that ensure the two developer action elements are achieved. While one may be lead to believe that the analysis performed for this component can be segregated into separate individuals performing work

units autonomously, this is not the case. Ideally multiple individual evaluators will perform an architecture study activity to takes into account many of the work units described below. Many of the work units to be performed in this component are related and should be done simultaneously.

11.6.2.4 Action ADV_INT.2.1E

ADV_INT.2.1C ***There shall be a one-to-one correspondence between the modules identified in TSF internals description and the modules identified in the modular design description.***

ADV_INT.2-1 The evaluator ***shall check*** the TSF internals description to determine that the identification of modules is identical to the modules that are described in the modular design description.

847 The evaluator determines that the modules identified and described in the modular description are the exact same modules identified in the TSF internals design. Any modules that are identified in either document that are not present in both signify a failure of this work unit.

ADV_INT.2.2C ***The TSF internals description shall describe the process used for modular decomposition.***

ADV_INT.2-2 The evaluator ***shall examine*** the TSF internals description to determine that the developer's process for modular decomposition results in a modular TSF design.

848 Modules interact by providing services to one another, or by cooperating in achieving an overall goal.

849 The evaluator assesses the developer's design methodology to determine that the TSF design incorporates sound engineering principles to decompose the TSF into modules. Modules should be designed to include only related functions, to have a single, well-defined purpose, and to avoid unnecessary interactions with other modules. The methodology should describe what are acceptable forms of communication between modules and under what circumstances discouraged forms of coupling and cohesion are considered acceptable. The methodology should also address complexity and what steps are taken to ensure modules are not overly complex. The intent here is that the developer's process should cover/address the developer action elements for the TSF internals (ADV_INT) requirements and this process provides guidance to the developer on how to design and code their modules.

850 The methodology provided by the developer provides the evaluator with an understanding of the developer's approach to designing (or restructuring of) the TSF and aids the evaluator in performing other work units for this component.

ADV_INT.2.3C ***The TSF internals description shall describe how the TSF design is a reflection of the modular decomposition process.***

ADV_INT.2-3 The evaluator ***shall examine*** the TSF internals description to determine that the TSF's resulting design is an accurate reflection of the developer's decomposition process.

851 The evaluator compares the developer's documented process for decomposition presented in the TSF internals description with the implementation representation to determine if the developer followed their process. The evaluator will need to perform this work unit in conjunction with other work units. In fact, the evaluator should understand the process the developer has employed for decomposing their TSF into modules and while performing other work units ensure that process was followed.

ADV_INT.2.4C ***The TSF internals description shall provide a justification, on a per-module basis, of any deviations from the coding standards.***

ADV_INT.2-4 The evaluator ***shall examine*** the implementation representation to determine that the developer adhered to the coding standards for the TSF modules.

852 The evaluator ensures that the development of the implementation representation adhered to the coding standards that are included in the development tools (Tools and techniques (ALC_TAT)). A developer may have more than one coding standard for different components of the TOE, and this is acceptable, as long as each standard contains the necessary information. This is accomplished by determining that the format prescribed by the coding standards (e.g., comments, naming conventions, indentation) is exhibited in the implementation representation. The evaluator ensures that the guidelines for programming logic (e.g., types of loops, control flow, error handling, the use of pointers) are exhibited in the implementation representation. The evaluator determines that other aspects of code development that are defined in the coding standards are followed by the developer.

ADV_INT.2-5 The evaluator ***shall examine*** the TSF internals description to determine if the justification for deviations from the coding standards for each module is reasonable.

853 There may be acceptable occurrences where the implementation representation does not follow the coding standards. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable (unless those modules are non-Assigned SFR-enforcing). Typically, deviations from the prescribed formatting aspects of the coding standards are not justifiable. Acceptable justification for deviation from the coding standards would generally be related to performance or efficiency reasons, and possibly for backward compatibility.

ADV_INT.2-6 The evaluator ***shall examine*** the implementation representation to determine if the deviations from the coding standards for each module do not make the offending modules overly complex.

854 The evaluator assesses the modules that have been identified in the TSF internals description as deviating from the coding standards to ensure the deviations do not make the Assigned SFR-enforcing modules too complex to understand the module's processing and interactions in a reasonable amount of time (e.g., no more than eight hours).

855 A complexity analysis performed by a tool, such as McCabe, may provide information that addresses the programmatic complexity of the module, but here, the evaluator assesses the modules not adhering to the coding standards to ensure the code is readable and readily understood.

ADV_INT.2.5C ***The TSF internals description shall include a coupling analysis that describes intermodule coupling for all TSF modules.***

ADV_INT.2-7 The evaluator ***shall examine*** the coupling analysis to determine that it is a complete and accurate reflection of the implementation representation.

856 The evaluator ensures that the coupling analysis correctly identifies and describes the method of all of the interactions with other modules by examining the implementation representation. The coupling analysis includes any explicit calls made by functions in a module to other modules, the passing of pointers to function to another function, the use of global variables, shared memory, message passing, or any other means that functions may use to communicate with other modules.

857 The evaluator also uses the implementation representation to ensure modules do not reference global variables that are not contained in the coupling analysis and that the references modules make (e.g., read, write) are consistent with the references specified in the coupling analysis.

858 The evaluator makes an independent assessment of the type of coupling that exists between modules and ensures that they arrive at the same conclusion as the presented in the developer's analysis.

ADV_INT.2-8 The evaluator ***shall examine*** the coupling analysis to ensure that modules that exhibit coupling other than call or common coupling are justified.

859 Once the evaluator has completed work unit ADV_INT.2-6 and has determined that the coupling analysis is complete and accurate, the evaluator determines that the instances of coupling other than call or common coupling are justified by the TSF internals description. The evaluator determines if the justifications are reasonable, and therefore allowable in a separate work unit.

ADV_INT.2.6C ***The TSF internals description shall include a cohesion analysis that describes the types of cohesion for all TSF modules.***

ADV_INT.2-9 The evaluator ***shall check*** the TSF internals description to determine it identifies and describes the type of cohesion for each TSF module.

860 The evaluator determines that the cohesion analysis provided in the TSF internals description is complete, that is each module is associated with the

type of cohesion that is exhibited, as well as a description of how the determination of the type of cohesion was made by the developer.

861 The evaluator will perform a cohesion analysis in work unit ADV_INT.2-11 and compare their findings to the cohesion analysis provided in the TSF internals description.

ADV_INT.2.7C ***The TSF internals description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by modules of the TSF, other than those permitted.***

ADV_INT.2-10 The evaluator ***shall examine*** the TSF internals description to determine if the justification for coupling and cohesion other than the ones permitted are reasonable.

862 The evaluator judges the developer's rationale for modules that exhibit coupling and cohesion other than those allowed and determines if the justifications are reasonable. The justification is provided for each module that is not conformant to the allowed forms of coupling and cohesion.

863 Reasons for justification may include efficiency or performance and the evaluator uses their judgement to determine that the justification is acceptable and that the disallowed forms of cohesion or coupling are not pervasive. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable.

864 In order to assess the justification for either coupling or cohesion forms other than those permitted may require the evaluator to examine the implementation representation. The evaluator confirm the justification is reasonable, by understanding the offending software, and determining that requiring the code to strictly conform to the coupling or cohesion metrics would be unreasonable.

11.6.2.5 Action ADV_INT.2.2E

ADV_INT.2-11 The evaluator ***shall examine*** the implementation representation to determine that the type of cohesion exhibited by a subset of SFR-enforcing, SFR-supporting, and SFR non-interfering TSF modules is consistent with that claimed in the TSF internals description.

865 Cohesion is a measure of the strength of relationship between the activities performed by a module. Cohesion ranges from the most desirable form, functional cohesion, to the least desirable form, coincidental cohesion. Allowable forms of cohesion include functional, sequential, and communicational cohesion; coincidental cohesion is unacceptable. Temporal cohesion is acceptable only for special-purpose use, such as initialisation, shutdown, and error recovery. Logical cohesion may be acceptable, based on the argument for its justification.

866 The evaluator uses the implementation representation and modular design description to determine the purpose of the modules and the relationship the functions have within a module to achieve that purpose. The processing performed by the functions in a module and the service they provide are an indication of the design in the grouping of functions and what type of cohesion is exhibited.

867 The evaluator performs an analysis to determine the type of cohesion exhibited by the modules and ensures that the evaluator's findings are consistent with the developer's claim in the TSF internals description. It is important to note that there may be some subjectivity involved in making a determination of the type of cohesion exhibited (e.g., sequential vs. temporal), but the type of cohesion that is deemed unacceptable (e.g., coincidental, logical) is more clear cut.

868 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.2.6 Action ADV_INT.2.3E

ADV_INT.2-12 The evaluator ***shall examine*** the implementation representation to determine that a subset of the SFR-enforcing, SFR-supporting, and SFR non-interfering TSF modules are not overly complex.

869 A metric that simply relies on the numbers of lines of code, or the percentage of comments and statements is considered a weak complexity measure and does provide a good metric of software complexity. More meaningful metrics consider the number of unique operators and operands in the program, or which measures the amount of unstructured decision logic in software by examining cyclomatic complexity.

870 The evaluator determines the complexity of the implementation representation by measuring the length of time it takes the evaluator to understand the logic in a module and the module's relationship with other modules. When using the design documentation and TSF internals description, the evaluator should understand a module's implementation and interactions in a reasonable amount of time. The measurement of time is subjective and is based on the evaluator's experience and skill in reading and understanding source code. It is assumed that the evaluator has knowledge of computer programming, and is familiar with the programming language used by the developer. Given this minimum level of expertise, the amount of time it takes to understand a module's implementation should not exceed eight hours.

871 The evaluator needs to also consider how a module incorporates access to “global entities”, such as functions, macros or data. One example of this is the use of header files in the C programming language. Often times header files are nested and a programmer does not understand all of the contents that are potentially accessible by their software. Another poor example of programming is the incorporation of header files without knowing whether the header file is actually needed. The problem arises when a programmer

makes a mistake (e.g., mis-types a variable or macro name). The likelihood of the improper variable being referenced or macro being invoked is less if only the necessary header files are included. Having several or more unnecessary header files included increases the likelihood that the mis-typed variable or macro exists in one of those unnecessary header files which could lead to serious consequences.

872 This work unit (as does the previous work unit) specifies that the subset of code the evaluator examines contains all three types of modules (i.e., SFR-enforcing, SFR-supporting, and SFR non-interfering). One might argue that SFR non-interfering modules are unimportant and therefore the evaluator should not be spending any time analysing these types of modules. However, for this very reason it is not clear that developers spend much time analysing these types of modules and therefore these types of modules may be more susceptible to containing flaws. When determining the subset of the SFR non-interfering modules, the evaluator should ensure they include any SFR non-interfering modules that are accessible by untrusted entities.

873 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.3 Evaluation of sub-activity (ADV_INT.3)

11.6.3.1 Objectives

874 The objective of this component is to introduce the notion of a layered software architecture to facilitate the understanding of the TSF. Having a layered architecture allows the developer and evaluator to analyse and understand the TSF in more digestible pieces. A layered architecture with well defined interactions between layers allows for analysis of individual layers and the dependencies that exist on other layers of the system. This is desirable, since attempting to understand the interactions and dependencies of all the modules in the TSF may be more difficult than conceptualising the TSF from a layered architecture that provides a context for collections of modules.

875 This component eliminates the allowance of temporal cohesion with *limited* exceptions to facilitate a more desirable form of modularity. In addition, the developer must identify any redundant and unused code that may exist to add to a better understanding of the TSF architecture.

11.6.3.2 Input

876 The evaluation evidence for this sub-activity is:

a) the modular design description;

b) the implementation representation;

c) the TSF internals description;

d) the documentation of the coding standards, as resulting from ALC_TAT.

11.6.3.3 Application notes

877 While this component is hierarchical to ADV_INT.2 and introduces requirements on layers of the TSF, it is recommended that the evaluator understands the TSF internals from the layered representation before attempting to understand the modules and the internal structure presented at the module level. Once a layer of the TSF is well understood, including interactions of a layer, the evaluator should be able to examine modules within a layer with more context and will facilitate the understanding of the modules role within a layer. This will provide the evaluator with the ability to perform a more structured approach based upon TSF internal structuring in analysing modules.

878 This component also introduces a requirement that the TSF internals description identifies modules that contain unused or redundant code.

11.6.3.4 Action ADV_INT.3.1E

ADV_INT.3.1C ***There shall be a one-to-one correspondence between the modules identified in TSF internals description and the modules identified in the modular design description.***

ADV_INT.3-1 The evaluator ***shall check*** the TSF internals description to determine that the identification of modules is identical to the modules that are described in the modular design description.

879 The evaluator determines that the modules identified and described in the modular description are the exact same modules identified in the TSF internals design. Any modules that are identified in either document that are not present in both signify a failure of this work unit.

ADV_INT.3.2C ***The TSF internals description shall describe the process used for modular decomposition.***

ADV_INT.3-2 The evaluator ***shall examine*** the TSF internals description to determine that the developer's process for modular decomposition results in a modular TSF design.

880 Modules interact by providing services to one another, or by cooperating in achieving an overall goal.

881 The evaluator assesses the developer's design methodology to determine that the TSF design incorporates sound engineering principles to decompose the TSF into modules. Modules should be designed to include only related functions, to have a single, well-defined purpose, and to avoid unnecessary interactions with other modules. The methodology should describe what are acceptable forms of communication between modules and under what circumstances discouraged forms of coupling and cohesion are considered

acceptable. The methodology should also address complexity and what steps are taken to ensure modules are not overly complex. The intent here is that the developer's process should cover/address the developer action elements for the TSF internals (ADV_INT) requirements and this process provides guidance to the developer on how to design and code their modules.

882 The methodology provided by the developer provides the evaluator with an understanding of the developer's approach to designing (or restructuring of) the TSF and aids the evaluator in performing other work units for this component.

ADV_INT.3.3C ***The TSF internals description shall describe how the TSF design is a reflection of the modular decomposition process.***

ADV_INT.3-3 The evaluator ***shall examine*** the TSF internals description to determine that the TSF's resulting design is an accurate reflection of the developer's decomposition process.

883 The evaluator compares the developer's documented process for decomposition presented in the TSF internals description with the implementation representation to determine if the developer followed their process. The evaluator will need to perform this work unit in conjunction with other work units. In fact, the evaluator should understand the process the developer has employed for decomposing their TSF into modules and while performing other work units ensure that process was followed.

ADV_INT.3.4C ***The TSF internals description shall provide a justification, on a per-module basis, of any deviations from the coding standards.***

ADV_INT.3-4 The evaluator ***shall examine*** the implementation representation to determine that the developer adhered to the coding standards for the TSF modules.

884 The evaluator ensures that the development of the implementation representation adhered to the coding standards that are included in the development tools (Tools and techniques (ALC_TAT)). A developer may have more than one coding standard for different components of the TOE, and this is acceptable, as long as each standard contains the necessary information. This is accomplished by determining that the format prescribed by the coding standards (e.g., comments, naming conventions, indentation) is exhibited in the implementation representation. The evaluator ensures that the guidelines for programming logic (e.g., types of loops, control flow, error handling, the use of pointers) are exhibited in the implementation representation. The evaluator determines that other aspects of code development that are defined in the coding standards are followed by the developer.

ADV_INT.3-5 The evaluator ***shall examine*** the TSF internals description to determine if the justification for deviations from the coding standards for each module is reasonable.

885 There may be acceptable occurrences where the implementation representation does not follow the coding standards. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable (unless those modules are non-Assigned SFR-enforcing). Typically, deviations from the prescribed formatting aspects of the coding standards are not justifiable. Acceptable justification for deviation from the coding standards would generally be related to performance or efficiency reasons, and possibly for backward compatibility.

ADV_INT.3-6 The evaluator ***shall examine*** the implementation representation to determine if the deviations from the coding standards for each module do not make the offending modules overly complex.

886 The evaluator assesses the modules that have been identified in the TSF internals description as deviating from the coding standards to ensure the deviations do not make the Assigned SFR-enforcing modules too complex to understand the module's processing and interactions in a reasonable amount of time (e.g., no more than eight hours).

887 A complexity analysis performed by a tool, such as McCabe, may provide information that addresses the programmatic complexity of the module, but here, the evaluator assesses the modules not adhering to the coding standards to ensure the code is readable and readily understood.

ADV_INT.3.5C ***The TSF internals description shall include a coupling analysis that describes intermodule coupling for all TSF modules.***

ADV_INT.3-7 The evaluator ***shall examine*** the coupling analysis to determine that it is a complete and accurate reflection of the implementation representation.

888 The evaluator ensures that the coupling analysis correctly identifies and describes the method of all of the interactions with other modules by examining the implementation representation. The coupling analysis includes any explicit calls made by functions in a module to other modules, the passing of pointers to function to another function, the use of global variables, shared memory, message passing, or any other means that functions may use to communicate with other modules.

889 The evaluator also uses the implementation representation to ensure modules do not reference global variables that are not contained in the coupling analysis and that the references modules make (e.g., read, write) are consistent with the references specified in the coupling analysis.

890 The evaluator makes an independent assessment of the type of coupling that exists between modules and ensures that they arrive at the same conclusion as the presented in the developer's analysis.

ADV_INT.3-8 The evaluator ***shall examine*** the coupling analysis to ensure that modules that exhibit coupling other than call or common coupling are justified.

891 Once the evaluator has completed work unit ADV_INT.3-7 and has determined that the coupling analysis is complete and accurate, the evaluator determines that the instances of coupling other than call or common coupling are justified by the TSF internals description. The evaluator determines if the justifications are reasonable, and therefore allowable in a separate work unit.

ADV_INT.3.6C ***The TSF internals description shall include a cohesion analysis that describes the types of cohesion for all TSF modules.***

ADV_INT.3-9 The evaluator ***shall check*** the TSF internals description to determine it identifies and describes the type of cohesion for each TSF module.

892 The evaluator determines that the cohesion analysis provided in the TSF internals description is complete, that is each module is associated with the type of cohesion that is exhibited, as well as a description of how the determination of the type of cohesion was made by the developer.

893 The evaluator will perform a cohesion analysis in work unit ADV_INT.3-23 and compare their findings to the cohesion analysis provided in the TSF internals description.

ADV_INT.3.7C ***The TSF internals description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by modules of the TSF, other than those permitted.***

ADV_INT.3-10 The evaluator ***shall examine*** the TSF internals description to determine if the justification for coupling and cohesion other than the ones permitted are reasonable.

894 The evaluator judges the developer's rationale for modules that exhibit coupling and cohesion other than those allowed and determines if the justifications are reasonable. The justification is provided for each module that is not conformant to the allowed forms of coupling and cohesion.

895 Reasons for justification may include efficiency or performance and the evaluator uses their judgement to determine that the justification is acceptable and that the disallowed forms of cohesion or coupling are not pervasive. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable.

896 In order to assess the justification for either coupling or cohesion forms other than those permitted may require the evaluator to examine the implementation representation. The evaluator confirm the justification is reasonable, by understanding the offending software, and determining that requiring the code to strictly conform to the coupling or cohesion metrics would be unreasonable.

ADV_INT.3.8C ***The TSF internals description shall describe the layering architecture and shall describe the services that each layer provides.***

ADV_INT.3-11 The evaluator ***shall examine*** the TSF internals description to determine that the layering architecture is described.

897 The evaluator ensures the TSF internals description provides a presentation that illustrates the layers within the TSF and describes the purpose of each of the layers. Typically a figure or diagram will show the layers and their hierarchy, and at a high level, interactions between the layers. The purpose of each layer provides enough detail that makes clear each layer's role in the architecture, and allows the evaluator to determine how the modules associated with each layer work together to meet that purpose.

ADV_INT.3-12 The evaluator ***shall examine*** the TSF internals description to determine that the services provided with each layer is consistent with the description of the layering architecture.

898 The evaluator ensures that the description of the services provided by each layer is consistent with the descriptions of the layers. The description of each layer's purpose and role maps to the services that each layer is providing. An inconsistent description would exist if a service is being exported to another layer, but there is no indication in the purpose of the layer exporting the service that would suggest the layer is providing such a service.

ADV_INT.3.9C ***The TSF internals description shall describe the methodology used to determine the layering architecture.***

ADV_INT.3-13 The evaluator ***shall examine*** the TSF internals description to determine that it provides a description of the developer's methodology in determining the TSF's layered architecture.

899 The description of the layering architecture provides the evaluator an understanding of how the layered architecture was determined. It describes the process or methodology the developer used in determining the layered architecture, and provides confidence that the layered architecture resulted from sound software engineering principles, rather than an ad-hoc grouping of modules into layers. The methodology should describe how the requirements associated with layering in this component were considered, and any design decisions that were considered to satisfy the requirements or required exceptions to the principles of a sound layered architecture.

ADV_INT.3.10C ***The TSF internals description shall identify all modules associated with each layer of the TSF.***

ADV_INT.3-14 The evaluator ***shall check*** the TSF internals description to ensure it identifies all of the modules associated with each layer.

900 The evaluator ensures that all modules a in the TSF internals description are associated with a layer and that a module is not associated with more than one layer. The evaluator will want to perform the work unit determining that there is a one-to-one correspondence with the modules identified in the TSF internals description and the modular design description before performing this work unit to ensure all modules are correctly identified, as well as

associated with a layer. There should be no instances of a module not being associated with a layer or a module associated with multiple layers.

ADV_INT.3-15 The evaluator ***shall examine*** the TSF internals description and the modular design description to determine that the modules associated with each layer is consistent with each layer's role.

901 The evaluator ensures that the modules associated with each layer is appropriate and consistent with the activities the layer performs and the services it provides. By consulting the functional description of modules presented in the modular design, and reviewing the purpose/role of a layer, the evaluator determines that the modules associated with a layer are appropriate and work together to achieve the functionality presented by a layer.

902 If a module is associated with a layer and it is not clear from the modules description presented in the modular design description how the module supports or satisfies the purpose of a layer, there may be an inadequacy in the layer's description, there may be a concern regarding other requirements (e.g., interactions between layers), or the module may be simply misplaced and some form of corrective action is necessary.

ADV_INT.3.11C ***The TSF internals description shall identify all interactions between layers of the TSF.***

ADV_INT.3-16 The evaluator ***shall check*** the TSF internals description to determine the interactions between layers of the TSF are identified.

903 The evaluator ensures that the interactions between each layer have been identified in the TSF internals description. It is not necessary for the TSF internals description to repeat the detailed information of the interaction, a reference to where the details of the interaction (whether it be in the modular design description, TSF internals description as part of the developer's coupling analysis. or the functional specification) is sufficient.

ADV_INT.3-17 The evaluator ***shall examine*** the TSF internals description to determine that the interactions between layers is accurately and completely described.

904 The evaluator uses information contained in the coupling analysis of the TSF internals description to determine the interactions between layers is consistently (accurately and completely) described. The evaluator considers calls that are made from modules that reside in a layer to functions/routines contained in modules that reside in another layer. The evaluator also considers other means in which modules communicate (e.g., shared memory, global variables, passing of function pointers). All forms of module communication are presented in the developer's coupling analysis, and for this work unit the evaluator is only concerned with the module interactions that take place between layers (i.e., the evaluator need not be concerned with interactions between modules that reside in the same layer). The evaluator ensures the developer provided coupling analysis is complete and accurate before performing this work unit.

905 The evaluator ensures that the description of layer interactions make it clear which layer is initiating the call.

ADV_INT.3.12C ***The TSF internals description shall provide a justification of interactions that are initiated from a lower layer to a higher layer.***

ADV_INT.3-18 The evaluator ***shall check*** the TSF internals description to determine it contains a justification of interactions between layers that are initiated from a lower layer to a higher layer.

906 Since interactions that are initiated from a lower layer to a higher layer indicate an undesirable dependency, the evaluator ensures that for each interaction that is initiated from a layer to a layer higher in the hierarchy a justification exists. The evaluator uses the information garnered from the previous work unit to determine the set of initiated interactions of interest.

ADV_INT.3-19 The evaluator ***shall examine*** the TSF internals description to determine the justifications for lower layers initiating an interaction to a higher layer are acceptable.

907 The are instances where it may be acceptable for layering violations (for a lower layer to have a dependency on a higher layer). Typically, these exceptions are for efficiency or performance reasons. The evaluator assesses the developer's justification and exercises judgement to the acceptability of the argument. The evaluator considers how pervasive are the layering violations, does the justification for the laying violation make sense in the context of the developer's architecture, or does it appear that modest alterations could be made to the architecture resulting in the elimination of the violation without impacting performance too much. The evaluator considers the trade-off between understandable (and maintainable) code and the need for efficiency and performance.

ADV_INT.3.13C ***The TSF internals description shall identify all modules of the TSF that contain unused or redundant code.***

ADV_INT.3-20 The evaluator ***shall check*** the TSF internals description to determine it identifies modules that contain unused code.

908 The evaluator ensures that the TSF internals description identifies modules that contain unused code. The TSF internals description does not have to enumerate how much or where the code is located, just that module x, y, and z contain portions of code that are not used in our current implementation.

ADV_INT.3-21 The evaluator ***shall check*** the TSF internals description to determine it identifies the modules that contain redundant code.

909 The evaluator ensures that the TSF internals description identifies modules that contain redundant code. The TSF internals description does not have to enumerate how much or where the redundant code is located, just that module x, y, and z contain portions of redundant code.

ADV_INT.3-22 The evaluator ***shall examine*** the implementation representation to determine that modules that contain unused or redundant code have been accurately and completely identified in the TSF internals description.

910 The evaluator reviews the implementation representation to determine that the developer has accurately and completely identified code that is unused or redundant. While there is no requirement to remove this form of code in this component, the evaluator should ensure that code of this nature does not adversely impact the TSF's ability to enforce the SFRs, or the architectural requirements of domain isolation and non-bypassability.

911 The evaluator is only concerned with unused code that will be transformed (e.g., compiled) into the resulting implementation. Sections of code that will not be transformed (e.g., due to compiler directives) are not an issue for this work unit, however, this type of code could be an issue if improper directives are supplied to the compiler, or an *ifdef* statement is deleted or commented out.

912 In some cases, the inclusion of unused or redundant code will be obvious. For example, functions in a module that are never called, or macros that are never used. Cases of redundant code may be sections of code (within a function, a module, or in multiple modules) that are repeated rather than creating a function that performs the required task and is called when needed.

913 In other cases it may be more difficult to determine is a section of code should be considered unused or redundant. It may be the case where a line or multiple lines of code that are used as an aspect of defensive programming may not be reached under “normal operations”, but the programmer inserted the code as a defensive measure. This should not be considered unused code. There may be cases where two sections of code are very close and out of 60 statements, 58 are identical. The evaluator will have to use judgement, as to whether this should be considered redundant code. One cannot measure redundancy by a lines of code metric, without understanding the context of the code or the architecture of the TSF. Would it make sense to have these 60 statements recast into a function?

11.6.3.5 Action ADV_INT.3.2E

ADV_INT.3-23 The evaluator ***shall examine*** the implementation representation to determine that the type of cohesion exhibited by a subset of SFR-enforcing, SFR-supporting, and SFR non-interfering TSF modules is consistent with that claimed in the TSF internals description.

914 Cohesion is a measure of the strength of relationship between the activities performed by a module. Cohesion ranges from the most desirable form, functional cohesion, to the least desirable form, coincidental cohesion. Allowable forms of cohesion include functional, sequential, and communicational cohesion; coincidental cohesion is unacceptable. Temporal cohesion is acceptable only for special-purpose use, such as initialisation, shutdown, and error recovery. Logical cohesion may be acceptable, based on the argument for its justification.

915 The evaluator uses the implementation representation and modular design description to determine the purpose of the modules and the relationship the functions have within a module to achieve that purpose. The processing performed by the functions in a module and the service they provide are an indication of the design in the grouping of functions and what type of cohesion is exhibited.

916 The evaluator performs an analysis to determine the type of cohesion exhibited by the modules and ensures that the evaluator's findings are consistent with the developer's claim in the TSF internals description. It is important to note that there may be some subjectivity involved in making a determination of the type of cohesion exhibited (e.g., sequential vs. temporal), but the type of cohesion that is deemed unacceptable (e.g., coincidental, logical) is more clear cut.

917 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.3.6 Action ADV_INT.3.3E

ADV_INT.3-24 The evaluator ***shall examine*** the implementation representation to determine that a subset of the SFR-enforcing, SFR-supporting, and SFR non-interfering TSF modules are not overly complex.

918 A metric that simply relies on the numbers of lines of code, or the percentage of comments and statements is considered a weak complexity measure and does provide a good metric of software complexity. More meaningful metrics consider the number of unique operators and operands in the program, or which measures the amount of unstructured decision logic in software by examining cyclomatic complexity.

919 The evaluator determines the complexity of the implementation representation by measuring the length of time it takes the evaluator to understand the logic in a module and the module's relationship with other modules. When using the design documentation and TSF internals description, the evaluator should understand a module's implementation and interactions in a reasonable amount of time. The measurement of time is subjective and is based on the evaluator's experience and skill in reading and understanding source code. It is assumed that the evaluator has knowledge of computer programming, and is familiar with the programming language used by the developer. Given this minimum level of expertise, the amount of time it takes to understand a module's implementation should not exceed eight hours.

920 The evaluator needs to also consider how a module incorporates access to “global entities”, such as functions, macros or data. One example of this is the use of header files in the C programming language. Often times header files are nested and a programmer does not understand all of the contents that are potentially accessible by their software. Another poor example of programming is the incorporation of header files without knowing whether the header file is actually needed. The problem arises when a programmer

makes a mistake (e.g., mis-types a variable or macro name). The likelihood of the improper variable being referenced or macro being invoked is less if only the necessary header files are included. Having several or more unnecessary header files included increases the likelihood that the mis-typed variable or macro exists in one of those unnecessary header files which could lead to serious consequences.

921 This work unit (as does the previous work unit) specifies that the subset of code the evaluator examines contains all three types of modules (i.e., SFR-enforcing, SFR-supporting, and SFR non-interfering). One might argue that SFR non-interfering modules are unimportant and therefore the evaluator should not be spending any time analysing these types of modules. However, for this very reason it is not clear that developers spend much time analysing these types of modules and therefore these types of modules may be more susceptible to containing flaws. When determining the subset of the SFR non-interfering modules, the evaluator should ensure they include any SFR non-interfering modules that are accessible by untrusted entities.

922 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.4 Evaluation of sub-activity (ADV_INT.4)

11.6.4.1 Objectives

923 This component is focused on minimising the complexity of the TSF's implementation. While the primary focus of this family is to require the developer to design and implement a TSF that is well understood by the development staff, this component also requires the design and implementation to be easily understood by an independent third-party in an efficient manner. This translates into requirements such as the justification for any unused or redundant code, and the elimination of SFR-non-interfering modules.

11.6.4.2 Input

924 The evaluation evidence for this sub-activity is:

a) the modular design description;

b) the implementation representation;

c) the TSF internals description;

d) the documentation of the coding standards, as resulting from ALC_TAT.

11.6.4.3 Application notes

925 This component introduces the notion that redundant or unused code that exists in the TSF must be justified. In some cases, the inclusion of unused or redundant code will be obvious. For example, functions in a module that are

never called, or macros that are never used. Cases of redundant code may be sections of code (within a function, a module, or in multiple modules) that are repeated rather than creating a function that performs the required task and is called when needed. In other cases the evaluator will have to use judgement whether unused or redundant code is a source of concern. There may be a case where it is determined that a line of code, or section of code, is unreachable, but the developer put the code in as a form of defensive programming. This would be an example of where the unreachable code (at least in "normal" operation) is acceptable. Sections of code may appear to be redundant code, but there are very slight differences between the sections. Judgement should be exercised to determine if the code should be collected in one place (e.g., a function) and called, or if due to the tasks being performed, the pervasiveness of the code, and the differences between the apparent redundant code warrant restructuring. The evaluator should keep in mind, the purpose of these requirements is to facilitate an understanding of the software that ensures maintenance activities and future software enhancements will not adversely impact the TSF's ability to enforce the SFRs, or the architectural requirements of domain isolation and non-bypassability.

926 This component also requires the developer to eliminate any modules from the TSF that are not necessary for enforcing or supporting the SFRs (with the exception of modules necessary to provide domain isolation and non-bypassability). The evaluator ensures that only SFR-enforcing modules, SFR-supporting modules, and any modules that implement mechanisms necessary for domain isolation and non-bypassability are included in the TSF.

11.6.4.4 Action ADV_INT.4.1E

ADV_INT.4.1C ***There shall be a one-to-one correspondence between the modules identified in TSF internals description and the modules identified in the modular design description.***

ADV_INT.4-1 The evaluator ***shall check*** the TSF internals description to determine that the identification of modules is identical to the modules that are described in the modular design description.

927 The evaluator determines that the modules identified and described in the modular description are the exact same modules identified in the TSF internals design. Any modules that are identified in either document that are not present in both signify a failure of this work unit.

ADV_INT.4.2C ***The TSF internals description shall describe the process used for modular decomposition.***

ADV_INT.4-2 The evaluator ***shall examine*** the TSF internals description to determine that the developer's process for modular decomposition results in a modular TSF design.

928 Modules interact by providing services to one another, or by cooperating in achieving an overall goal.

929 The evaluator assesses the developer's design methodology to determine that the TSF design incorporates sound engineering principles to decompose the TSF into modules. Modules should be designed to include only related functions, to have a single, well-defined purpose, and to avoid unnecessary interactions with other modules. The methodology should describe what are acceptable forms of communication between modules and under what circumstances discouraged forms of coupling and cohesion are considered acceptable. The methodology should also address complexity and what steps are taken to ensure modules are not overly complex. The intent here is that the developer's process should cover/address the developer action elements for the TSF internals (ADV_INT) requirements and this process provides guidance to the developer on how to design and code their modules.

930 The methodology provided by the developer provides the evaluator with an understanding of the developer's approach to designing (or restructuring of) the TSF and aids the evaluator in performing other work units for this component.

ADV_INT.4.3C ***The TSF internals description shall describe how the TSF design is a reflection of the modular decomposition process.***

ADV_INT.4-3 The evaluator ***shall examine*** the TSF internals description to determine that the TSF's resulting design is an accurate reflection of the developer's decomposition process.

931 The evaluator compares the developer's documented process for decomposition presented in the TSF internals description with the implementation representation to determine if the developer followed their process. The evaluator will need to perform this work unit in conjunction with other work units. In fact, the evaluator should understand the process the developer has employed for decomposing their TSF into modules and while performing other work units ensure that process was followed.

ADV_INT.4.4C ***The TSF internals description shall provide a justification, on a per-module basis, of any deviations from the coding standards.***

ADV_INT.4-4 The evaluator ***shall examine*** the implementation representation to determine that the developer adhered to the coding standards for the TSF modules.

932 The evaluator ensures that the development of the implementation representation adhered to the coding standards that are included in the development tools (Tools and techniques (ALC_TAT)). A developer may have more than one coding standard for different components of the TOE, and this is acceptable, as long as each standard contains the necessary information. This is accomplished by determining that the format prescribed by the coding standards (e.g., comments, naming conventions, indentation) is exhibited in the implementation representation. The evaluator ensures that the guidelines for programming logic (e.g., types of loops, control flow, error

handling, the use of pointers) are exhibited in the implementation representation. The evaluator determines that other aspects of code development that are defined in the coding standards are followed by the developer.

ADV_INT.4-5 The evaluator ***shall examine*** the TSF internals description to determine if the justification for deviations from the coding standards for each module is reasonable.

933 There may be acceptable occurrences where the implementation representation does not follow the coding standards. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable (unless those modules are non-Assigned SFR-enforcing). Typically, deviations from the prescribed formatting aspects of the coding standards are not justifiable. Acceptable justification for deviation from the coding standards would generally be related to performance or efficiency reasons, and possibly for backward compatibility.

ADV_INT.4-6 The evaluator ***shall examine*** the implementation representation to determine if the deviations from the coding standards for each module do not make the offending modules overly complex.

934 The evaluator assesses the modules that have been identified in the TSF internals description as deviating from the coding standards to ensure the deviations do not make the Assigned SFR-enforcing modules too complex to understand the module's processing and interactions in a reasonable amount of time (e.g., no more than eight hours).

935 A complexity analysis performed by a tool, such as McCabe, may provide information that addresses the programmatic complexity of the module, but here, the evaluator assesses the modules not adhering to the coding standards to ensure the code is readable and readily understood.

ADV_INT.4.5C ***The TSF internals description shall include a coupling analysis that describes intermodule coupling for all TSF modules.***

ADV_INT.4-7 The evaluator ***shall examine*** the coupling analysis to determine that it is a complete and accurate reflection of the implementation representation.

936 The evaluator ensures that the coupling analysis correctly identifies and describes the method of all of the interactions with other modules by examining the implementation representation. The coupling analysis includes any explicit calls made by functions in a module to other modules, the passing of pointers to function to another function, the use of global variables, shared memory, message passing, or any other means that functions may use to communicate with other modules.

937 The evaluator also uses the implementation representation to ensure modules do not reference global variables that are not contained in the coupling

analysis and that the references modules make (e.g., read, write) are consistent with the references specified in the coupling analysis.

938 The evaluator makes an independent assessment of the type of coupling that exists between modules and ensures that they arrive at the same conclusion as the presented in the developer's analysis.

ADV_INT.4-8 The evaluator ***shall examine*** the coupling analysis to ensure that modules that exhibit coupling other than call or common coupling are justified.

939 Once the evaluator has completed work unit ADV_INT.4-7 and has determined that the coupling analysis is complete and accurate, the evaluator determines that the instances of coupling other than call or common coupling are justified by the TSF internals description. The evaluator determines if the justifications are reasonable, and therefore allowable in a separate work unit.

ADV_INT.4.6C ***The TSF internals description shall include a cohesion analysis that describes the types of cohesion for all TSF modules.***

ADV_INT.4-9 The evaluator ***shall check*** the TSF internals description to determine it identifies and describes the type of cohesion for each TSF module.

940 The evaluator determines that the cohesion analysis provided in the TSF internals description is complete, that is each module is associated with the type of cohesion that is exhibited, as well as a description of how the determination of the type of cohesion was made by the developer.

941 The evaluator will perform a cohesion analysis in work unit ADV_INT.4-27 and compare their findings to the cohesion analysis provided in the TSF internals description.

ADV_INT.4.7C ***The TSF internals description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by modules of the TSF, other than those permitted.***

ADV_INT.4-10 The evaluator ***shall examine*** the TSF internals description to determine if the justification for coupling and cohesion other than the ones permitted are reasonable.

942 The evaluator judges the developer's rationale for modules that exhibit coupling and cohesion other than those allowed and determines if the justifications are reasonable. The justification is provided for each module that is not conformant to the allowed forms of coupling and cohesion.

943 Reasons for justification may include efficiency or performance and the evaluator uses their judgement to determine that the justification is acceptable and that the disallowed forms of cohesion or coupling are not pervasive. A justification that is solely based on the use of third-party code or code that was inherited from previous life-cycles of the TOE is not acceptable.

944 In order to assess the justification for either coupling or cohesion forms other than those permitted may require the evaluator to examine the implementation representation. The evaluator confirm the justification is reasonable, by understanding the offending software, and determining that requiring the code to strictly conform to the coupling or cohesion metrics would be unreasonable.

ADV_INT.4.8C ***The TSF internals description shall describe the layering architecture and shall describe the services that each layer provides.***

ADV_INT.4-11 The evaluator ***shall examine*** the TSF internals description to determine that the layering architecture is described.

945 The evaluator ensures the TSF internals description provides a presentation that illustrates the layers within the TSF and describes the purpose of each of the layers. Typically a figure or diagram will show the layers and their hierarchy, and at a high level, interactions between the layers. The purpose of each layer provides enough detail that makes clear each layer's role in the architecture, and allows the evaluator to determine how the modules associated with each layer work together to meet that purpose.

ADV_INT.4-12 The evaluator ***shall examine*** the TSF internals description to determine that the services provided with each layer is consistent with the description of the layering architecture.

946 The evaluator ensures that the description of the services provided by each layer is consistent with the descriptions of the layers. The description of each layer's purpose and role maps to the services that each layer is providing. An inconsistent description would exist if a service is being exported to another layer, but there is no indication in the purpose of the layer exporting the service that would suggest the layer is providing such a service.

ADV_INT.4.9C ***The TSF internals description shall describe the methodology used to determine the layering architecture.***

ADV_INT.4-13 The evaluator ***shall examine*** the TSF internals description to determine that it provides a description of the developer's methodology in determining the TSF's layered architecture.

947 The description of the layering architecture provides the evaluator an understanding of how the layered architecture was determined. It describes the process or methodology the developer used in determining the layered architecture, and provides confidence that the layered architecture resulted from sound software engineering principles, rather than an ad-hoc grouping of modules into layers. The methodology should describe how the requirements associated with layering in this component were considered, and any design decisions that were considered to satisfy the requirements or required exceptions to the principles of a sound layered architecture.

ADV_INT.4.10C ***The TSF internals description shall identify all modules associated with each layer of the TSF.***

ADV_INT.4-14 The evaluator ***shall check*** the TSF internals description to ensure it identifies all of the modules associated with each layer.

948 The evaluator ensures that all modules a in the TSF internals description are associated with a layer and that a module is not associated with more than one layer. The evaluator will want to perform the work unit determining that there is a one-to-one correspondence with the modules identified in the TSF internals description and the modular design description before performing this work unit to ensure all modules are correctly identified, as well as associated with a layer. There should be no instances of a module not being associated with a layer or a module associated with multiple layers.

ADV_INT.4-15 The evaluator ***shall examine*** the TSF internals description and the modular design description to determine that the modules associated with each layer is consistent with each layer's role.

949 The evaluator ensures that the modules associated with each layer is appropriate and consistent with the activities the layer performs and the services it provides. By consulting the functional description of modules presented in the modular design, and reviewing the purpose/role of a layer, the evaluator determines that the modules associated with a layer are appropriate and work together to achieve the functionality presented by a layer.

950 If a module is associated with a layer and it is not clear from the modules description presented in the modular design description how the module supports or satisfies the purpose of a layer, there may be an inadequacy in the layer's description, there may be a concern regarding other requirements (e.g., interactions between layers), or the module may be simply misplaced and some form of corrective action is necessary.

ADV_INT.4.11C ***The TSF internals description shall identify all interactions between layers of the TSF.***

ADV_INT.4-16 The evaluator ***shall check*** the TSF internals description to determine the interactions between layers of the TSF are identified.

951 The evaluator ensures that the interactions between each layer have been identified in the TSF internals description. It is not necessary for the TSF internals description to repeat the detailed information of the interaction, a reference to where the details of the interaction (whether it be in the modular design description, TSF internals description as part of the developer's coupling analysis. or the functional specification) is sufficient.

ADV_INT.4-17 The evaluator ***shall examine*** the TSF internals description to determine that the interactions between layers is accurately and completely described.

952 The evaluator uses information contained in the coupling analysis of the TSF internals description to determine the interactions between layers is consistently (accurately and completely) described. The evaluator considers calls that are made from modules that reside in a layer to functions/routines

contained in modules that reside in another layer. The evaluator also considers other means in which modules communicate (e.g., shared memory, global variables, passing of function pointers). All forms of module communication are presented in the developer's coupling analysis, and for this work unit the evaluator is only concerned with the module interactions that take place between layers (i.e., the evaluator need not be concerned with interactions between modules that reside in the same layer). The evaluator ensures the developer provided coupling analysis is complete and accurate before performing this work unit.

953 The evaluator ensures that the description of layer interactions make it clear which layer is initiating the call.

ADV_INT.4.12C ***The TSF internals description shall provide a justification of interactions that are initiated from a lower layer to a higher layer.***

ADV_INT.4-18 The evaluator ***shall check*** the TSF internals description to determine it contains a justification of interactions between layers that are initiated from a lower layer to a higher layer.

954 Since interactions that are initiated from a lower layer to a higher layer indicate an undesirable dependency, the evaluator ensures that for each interaction that is initiated from a layer to a layer higher in the hierarchy a justification exists. The evaluator uses the information garnered from the previous work unit to determine the set of initiated interactions of interest.

ADV_INT.4-19 The evaluator ***shall examine*** the TSF internals description to determine the justifications for lower layers initiating an interaction to a higher layer are acceptable.

955 The are instances where it may be acceptable for layering violations (for a lower layer to have a dependency on a higher layer). Typically, these exceptions are for efficiency or performance reasons. The evaluator assesses the developer's justification and exercises judgement to the acceptability of the argument. The evaluator considers how pervasive are the layering violations, does the justification for the laying violation make sense in the context of the developer's architecture, or does it appear that modest alterations could be made to the architecture resulting in the elimination of the violation without impacting performance too much. The evaluator considers the trade-off between understandable (and maintainable) code and the need for efficiency and performance.

ADV_INT.4.13C ***The TSF internals description shall justify all modules of the TSF that contain unused or redundant code.***

ADV_INT.4-20 The evaluator ***shall examine*** the TSF internals description to determine if the justifications for unused or redundant code are reasonable.

956 The evaluator ensures that the justifications provided for unused or redundant code provide an argument that justified its existence, and describes why the inclusion of this code provides no concern with respect to security.

The justification for unused code could include planned future functionality. A suitable justification for redundant code may be for performance reasons, while the pervasiveness of redundant or unused code may influence the decision of whether the inclusion of this code is suitable.

ADV_INT.4-21 The evaluator ***shall examine*** implementation representation to determine that the developer has justified all areas of unused or redundant code.

957 The evaluator reviews the implementation representation (manually and/or with automated tools) to ensure that the developer has justified all instances of unused or redundant code.

958 In some cases, the inclusion of unused or redundant code will be obvious. For example, functions in a module that are never called, or macros that are never used. Cases of redundant code may be sections of code (within a function, a module, or in multiple modules) that are repeated rather than creating a function that performs the required task and is called when needed.

959 In other cases the evaluator will have to use judgement whether unused or redundant code is a source of concern. There may be a case where it is determined that a line of code, or section of code, is unreachable, but the developer put the code in as a form of defensive programming. This would be an example of where the unreachable code (at least in "normal" operation) is acceptable. Sections of code may appear to be redundant code, but there are very slight differences between the sections. Judgement should be exercised to determine if the code should be collected in one place (e.g., a function) and called, or if due to the tasks being performed, the pervasiveness of the code, and the differences between the apparent redundant code warrant restructuring. The evaluator should keep in mind, the purpose of these requirements is to facilitate an understanding of the software that ensures maintenance activities and future software enhancements will not adversely impact the TSF's ability to enforce the SFRs, or the architectural requirements of domain isolation and non-bypassability.

960 The evaluator is only concerned with unused code that will be transformed (e.g., compiled) into the resulting implementation. Sections of code that will not be transformed (e.g., due to compiler directives) are not an issue for this work unit, however, this type of code could be an issue if improper directives are supplied to the compiler, or an ifdef statement is deleted or commented out.

ADV_INT.4.14C ***The TSF internals description shall show that mutual dependencies have been minimised, and justify those that remain.***

ADV_INT.4-22 The evaluator ***shall check*** the TSF internals description to determine mutual dependencies are identified.

961 The evaluator ensures the mutual dependencies have been identified in the TSF internals description. Mutual dependencies between modules, as well as between layers are identified.

ADV_INT.4-23 The evaluator ***shall examine*** the TSF internals description to determine the mutual dependencies are minimized.

962 The evaluator uses the understanding gained throughout the analysis performed in TSF internals (ADV_INT) to determine that the mutual dependencies that exist in the TSF are minimal. The evaluator uses information provided in the developer's coupling analysis to understand the interactions between modules, and the modular design description, which provides an algorithmic description of the processing performed by modules to determine how the modules interact and how the modules processing is dependent on those involved in the interaction.

963 Minimising mutual dependencies helps reduce complexity associated with circular interaction and results in a more easily understood and maintainable software base.

ADV_INT.4-24 The evaluator ***shall examine*** the TSF internals description to determine the justifications for any mutual dependencies are acceptable.

964 The evaluator ensures the justifications for any mutual dependencies that exist are reasonable. The purpose of reducing mutual or circular dependencies is to simply the software, which facilitates understanding, value of code walk-throughs, and maintainability. Mutual dependencies can create problems with deadlocks or race conditions that may have an adverse impact on the TSF's ability to enforce the SFRs or provide for domain isolation or non-bypassability. Any justification provided by the developer addresses these concerns.

ADV_INT.4.15C ***The TSF internals description shall describe how the entire TSF has been structured to minimise complexity.***

ADV_INT.4-25 The evaluator ***shall examine*** the TSF internals description to determine the developer's description for how the TSF is designed and structured minimises complexity is acceptable.

965 The evaluator uses a wealth of information provided in the TSF internals description to make an assessment of the developer's claim to have structured the TSF to minimise complexity. Here the developer provides a description that ties all of the developer's activities and analyses together to provide a "big picture" view of how the TSF has been designed and structured. This argument should provide specific references to the detailed analyses performed, the methodologies employed, and provide the "glue" that "tells a story" of how all of the performed activities resulted in a well structured TSF that has minimised complexity and is simple enough to be analysed.

966 The evaluator will also confirm or disprove the developer's argument in the performance of work unit ADV_INT.4-29.

ADV_INT.4.16C ***The TSF internals description shall justify the inclusion of any SFR non-interfering modules in the TSF.***

ADV_INT.4-26 The evaluator ***shall examine*** the TSF internals description to determine the justifications for including SFR non-interfering modules are acceptable.

967 The evaluator considers the justifications made by the developer as to why SFR non-interfering modules are included in the TSF. At this level of assurance, it is difficult to provide examples of where a module that this not SFR-supporting or SFR-enforcing should be allowed in the TSF. The evaluator uses judgement in determining the appropriateness of the developer's argument while taking into account that the TSF's complexity is to be minimal at this level of assurance.

11.6.4.5 Action ADV_INT.4.2E

ADV_INT.4-27 The evaluator ***shall examine*** the implementation representation to determine that the type of cohesion exhibited by a subset of SFR-enforcing, SFR-supporting, and SFR non-interfering TSF modules is consistent with that claimed in the TSF internals description.

968 Cohesion is a measure of the strength of relationship between the activities performed by a module. Cohesion ranges from the most desirable form, functional cohesion, to the least desirable form, coincidental cohesion. Allowable forms of cohesion include functional, sequential, and communicational cohesion; coincidental cohesion is unacceptable. Temporal cohesion is acceptable only for special-purpose use, such as initialisation, shutdown, and error recovery. Logical cohesion may be acceptable, based on the argument for its justification.

969 The evaluator uses the implementation representation and modular design description to determine the purpose of the modules and the relationship the functions have within a module to achieve that purpose. The processing performed by the functions in a module and the service they provide are an indication of the design in the grouping of functions and what type of cohesion is exhibited.

970 The evaluator performs an analysis to determine the type of cohesion exhibited by the modules and ensures that the evaluator's findings are consistent with the developer's claim in the TSF internals description. It is important to note that there may be some subjectivity involved in making a determination of the type of cohesion exhibited (e.g., sequential vs. temporal), but the type of cohesion that is deemed unacceptable (e.g., coincidental, logical) is more clear cut.

971 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.4.6 Action ADV_INT.4.3E

ADV_INT.4-28 The evaluator ***shall examine*** the implementation representation to determine that a subset of the SFR-enforcing, SFR-supporting, and SFR non-interfering TSF modules are not overly complex.

972 A metric that simply relies on the numbers of lines of code, or the percentage of comments and statements is considered a weak complexity measure and does provide a good metric of software complexity. More meaningful metrics consider the number of unique operators and operands in the program, or which measures the amount of unstructured decision logic in software by examining cyclomatic complexity.

973 The evaluator determines the complexity of the implementation representation by measuring the length of time it takes the evaluator to understand the logic in a module and the module's relationship with other modules. When using the design documentation and TSF internals description, the evaluator should understand a module's implementation and interactions in a reasonable amount of time. The measurement of time is subjective and is based on the evaluator's experience and skill in reading and understanding source code. It is assumed that the evaluator has knowledge of computer programming, and is familiar with the programming language used by the developer. Given this minimum level of expertise, the amount of time it takes to understand a module's implementation should not exceed eight hours.

974 The evaluator needs to also consider how a module incorporates access to "global entities", such as functions, macros or data. One example of this is the use of header files in the C programming language. Often times header files are nested and a programmer does not understand all of the contents that are potentially accessible by their software. Another poor example of programming is the incorporation of header files without knowing whether the header file is actually needed. The problem arises when a programmer makes a mistake (e.g., mis-types a variable or macro name). The likelihood of the improper variable being referenced or macro being invoked is less if only the necessary header files are included. Having several or more unnecessary header files included increases the likelihood that the mis-typed variable or macro exists in one of those unnecessary header files which could lead to serious consequences.

975 This work unit (as does the previous work unit) specifies that the subset of code the evaluator examines contains all three types of modules (i.e., SFR-enforcing, SFR-supporting, and SFR non-interfering). One might argue that SFR non-interfering modules are unimportant and therefore the evaluator should not be spending any time analysing these types of modules. However, for this very reason it is not clear that developers spend much time analysing these types of modules and therefore these types of modules may be more susceptible to containing flaws. When determining the subset of the SFR non-interfering modules, the evaluator should ensure they include any SFR non-interfering modules that are accessible by untrusted entities.

976 As with all areas where the evaluator performs activities on a subset the evaluator provides a justification of their sample size and scope.

11.6.4.7 Action ADV_INT.4.4E

ADV_INT.4-29 The evaluator ***shall examine*** the implementation representation to determine the modules are simple enough to be analysed.

977 During the course of the evaluator's examination of the implementation representation the evaluator makes a judgement as to the level of complexity presented in the modules. Another requirement and work unit requires the evaluator to perform a complexity analysis and for this work unit the evaluator can build off of that work. While there is no pre-defined metric that provides a clear indication of whether a module is simple enough to be analysed, the evaluator judges whether the modules are constructed such that internally, and there interaction/dependencies on other modules can be well understood in a relatively short period of time.

978 The goal is to ensure that intricate programming techniques, or subtle dependencies that may have been understood by the original programmer, but not easily understood by very few (if any) other programmers do not exist in the implementation representation. It should not be the case where an evaluator runs across a comment in the code that says “I'm not sure what this section of code is doing, but it works and don't mess with it."

11.7 Security policy modelling (ADV_SPM)

11.7.1 Evaluation of sub-activity (ADV_SPM.1)

11.7.1.1 Objectives

979 The objectives of this sub-activity are to determine whether the formal security policy model unambiguously defines the rules and characteristics of the security policies and whether this definition and its associated description correspond with the security functions provided by the interfaces described in the functional specification.

980 The model describes the policies enforced by the TSF whose interfaces are described in the functional specification. For example, access control policies would describe the externally-visible resources being protected and the conditions that must be met for access to be granted; audit policies would describe the TOE's auditable events, identifying both those that are selectable by the administrator and those that are always audited; identification and authentication policies would describe how users are identified, how those claimed identities are authenticated, and any rules affecting how identities are authenticated (e.g. users on the corporate intranet need no authentication, while external users are authenticated with one-time passwords). A model is simply a description of the security policies enforced by services or functions available at the external interface.

11.7.1.2 Input

981 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE security policy model;

d) the operational user guidance;

11.7.1.3 Action ADV_SPM.1.1E

ADV_SPM.1.1C ***The TSP model shall be in a formal style, supported by explanatory text as required, and identify the security policies of the TSF that are modelled.***

ADV_SPM.1-1 The evaluator ***shall check*** that the TSP model is in a formal style and, where required, is supported by explanatory text.

982 A formal security model is expressed in a restricted syntax language with defined semantics based upon well-established mathematical concepts. It precisely describes important aspects (see ADV_SPM.1-2) of security and their relationship to the behaviour of the TSF. Supporting narrative description is required to aid understanding of the formal model.

ADV_SPM.1-2 The evaluator ***shall check*** that that all security policies are included within the TSP model.

983 The assignments in ADV_SPM.1.2D and ADV_SPM.1.3D list the policies to be modelled either formally or semi-formally (respectively). The evaluator checks that the list of policies completing the assignment in ADV_SPM.1.2D is accurate and that all of these identified policies are formally modelled; and that the list of policies completing the assignment in ADV_SPM.1.3D is accurate and that all of these identified policies are semiformally modelled.

984 To judge the completeness of this list, the evaluator determines the policies that are reflected by the SFRs being claimed in the ST/PP. Note that there may be security policies for non-disclosure, audit, identification and authentication, encryption, and management.

ADV_SPM.1.2C ***The TSP model shall describe the rules and characteristics of all policies of the TSF that are modelled.***

ADV_SPM.1-3 The evaluator ***shall examine*** the TSP model to determine that it describes the rules and characteristics of all policies of the TSF.

985 The evaluator ensures that the security policy model contains a definition of security that captures key security functionality, and a definition of the rules of operation that shows how the definition of security is enforced. The rules of operation and security characteristics of the TSP model are intended to allow flexibility in the type of model that may be developed (e.g. state transition, non-interference). The security characteristics are the properties that must hold for security to be maintained (e.g., definitions of secure values for TSF data); the description of security rules are how the TOE enforces the values that the security-relevant attributes can take.

986 For example, an authentication policy that uses passwords would include a definition of the minimum acceptable password length in its description within the model. The security functionality would include the minimum password length; the security rules would describe the TOE's enforcement of that length (should the user attempt to set a password of lesser length, for example).

987 The model used might influence the manner in which these rules and characteristics are expressed. For example, in a state-transition model, security functionality might be represented as definitions of possible states (e.g. “initial state”, “secure state”, etc.) while rules of operation might identify the rules of transition as well as properties that must hold during such transitions (e.g. tranquillity).

ADV_SPM.1.3C ***The correspondence shall show that the functional specification is consistent and complete with respect to the TSP model.***

ADV_SPM.1-4 The evaluator ***shall examine*** the correspondence to determine that all of the TSFI in the functional specification and all of the security functionality identified in the TSP model are consistent and complete.

988 While the previous work unit is concerned with whether the TSF is modelled correctly, this work unit is concerned with whether the model reflects the functional specification. All aspects of the security functionality that are described in the model must correspond to at least one of the TSFI described in the functional specification. This ensures that the TSFI required to invoke the security functionality described in the model is identified in each instance.

ADV_SPM.1.4C ***The correspondence between the TSP model and the functional specification shall be at the correct level of formality.***

ADV_SPM.1-5 The evaluator ***shall examine*** the correspondence between the TSP model and the functional specification to determine that it is at the correct level of formality.

989 The degree of formality of the correspondence between the TSP model and the functional specification is determined by the lower level of formality of the two. That is:

a) the correspondence between the formally-stated portions of the TSP model and the formal functional specification must be formal.

b) the correspondence between the semiformally-stated portions of the TSP model and the formal functional specification may be semiformal.

c) the correspondence between the TSP model and the semiformal functional specification may be semiformal.

11.8 TOE design (ADV_TDS)

11.8.1 Evaluation of sub-activity (ADV_TDS.1)

11.8.1.1 Input

990 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design.

11.8.1.2 Action ADV_TDS.1.1E

ADV_TDS.1.1C ***The design shall describe the structure of the TOE in terms of components.***

ADV_TDS.1-1 The evaluator ***shall examine*** the TOE design to determine that the structure of the entire TOE is described in terms of components.

991 The evaluator ensures that all of the components of the TOE are identified. This description of the TOE will be used as input to work unit ADV_TDS.1-2, where the parts of the TOE that make up the TSF are identified. That is, this requirement is on the entire TOE rather than on only the TSF.

992 The TOE (and TSF) may be described in multiple layers of abstraction (i.e. components and modules) Depending upon the complexity of the TOE, its design may be described in terms of components and modules, as described in CC Part 3 Annex A.4, ADV_TDS: Components and Modules. At this level of assurance, the decomposition only need be at the "component" level.

993 In performing this activity, the evaluator examines other evidence presented for the TOE (e.g., ST, operator user guidance) to determine that the description of the TOE in such evidence is consistent with the description contained in the TOE design.

ADV_TDS.1.2C ***The design shall identify all components of the TSF.***

ADV_TDS.1-2 The evaluator ***shall examine*** the TOE design to determine that all components of the TSF are identified.

994 In work unit ADV_TDS.1-1 all of the components of the TOE were identified, and a determination made that the non-TSF components were correctly characterised. Building on that work, the components that were not characterised as non-TSF components should be precisely identified. The evaluator determines that, of the hardware and software installed and configured according to the Preparative user guidance (AGD_PRE) guidance, each component has been accounted for as either one that is part of the TSF, or one that is not.

ADV_TDS.1.3C ***The design shall describe each TSF component identified as non-SFR-enforcing in sufficient detail to determine that it is non-SFR-enforcing.***

ADV_TDS.1-3 The evaluator ***shall examine*** the TOE design to determine that each non-SFR-enforcing component of the TSF is described such that the evaluator can determine that the component is non-SFR-enforcing.

995 non-SFR-enforcing components do not need to be described in detail as to how they function in the system. However, the evaluator makes a determination, based on the evidence provided by the developer, that the components that do not have high-level descriptions are non-SFR-enforcing (that is, either SFR-supporting or SFR-non-interfering). Note that if the developer provides a uniform level of detailed documentation then this work unit will be largely satisfied, since the point of categorising the components is to allow the developer to provide less information for non-SFR-enforcing components than for SFR-enforcing components.

996 An SFR-supporting component is one that is depended on by an SFR-enforcing component in order to implement an SFR, but does not play as direct a role as an SFR-supporting requirement. An SFR-non-interfering component is one that is not depended upon, in either a supporting or enforcing role, to implement an SFR.

ADV_TDS.1.4C ***The design shall provide a high-level description of the SFR-enforcing behaviour of the SFR-enforcing components.***

ADV_TDS.1-4 The evaluator ***shall examine*** the TOE design to determine that it provides a complete, accurate, and high-level description of the SFR-enforcing behaviour of the SFR-enforcing components.

997 The evaluator should note that the developer is not required to explicitly designate components as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These “tags” are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the components have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular component.

998 SFR-enforcing behaviour refers to *how* a component 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 component. The goal of these descriptions is to give the evaluator enough information to understand *how* the SFR-enforcing behaviour is achieved. Note that the evaluator should find unacceptable asserts of SFR-enforcement in the TOE design documentation for this work unit. It should be noted that it is the evaluator's determination with respect to what “high-level” means for a

particular TOE, and the evaluator obtains enough information from the developer to make a sound verdict for this work unit.

999 The evaluator is cautioned, however, that "perfect" assurance is not a goal nor required by this work unit, so judgement will have to be exercised in determine the amount and composition of the evidence required to make a verdict on this work unit.

1000 To determine completeness and accuracy, the evaluator examines other information available (e.g., functional specification, architectural design, implementation representation). Descriptions of functionality in these documents should be consistent with what is provided for evidence for this work unit

ADV_TDS.1.5C ***The design shall provide a description of the interactions between the SFR-enforcing components of the TSF and other components of the TSF.***

ADV_TDS.1-5 The evaluator ***shall examine*** the TOE design to determine that interactions between the components of the TSF are described.

1001 The goal of describing the interactions between the SFR-enforcing components and other components is to help provide the reader a better understanding of how the TSF performs it functions. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a component to a routine in a different component; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular component that are going to be used by another component should be covered in this discussion. Any control relationships between components (e.g., a component responsible for configuring a rule base for a firewall system and the component that actually implements these rules) should also be described.

1002 The evaluator should use their own judgement in assessing the completeness of the description. If the reason for an interaction is unclear, or if there are SFR-related interactions (discovered, for instance, in examining the descriptions of component behaviour) that do not appear to be described, the evaluator ensures that this information is provided by the developer. However, if the evaluator can determine that interactions among a particular set of components, while incompletely described by the developer, will not aid in understanding the overall functionality nor security functionality provided by the TSF, then the evaluator may choose to consider the description sufficient, and not pursue completeness for its own sake.

11.8.1.3 Action ADV_TDS.1.2E

ADV_TDS.1-6 The evaluator ***shall examine*** the TOE security functional requirements and the TOE design, to determine that all ST security functional requirements are covered by the TOE design.

1003 The evaluator may construct a map between the TOE security functional requirements and the TOE design. This map will likely be from a functional requirement to a set of components. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

1004 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were implemented in specific places within fifteen modules, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to one component and claim the work unit had been completed. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to Component A, behaviours x, y, and z; FDP_ACC.1 Access control (rule 2) to Component A, behaviours x, p, and q; etc.

ADV_TDS.1-7 The evaluator ***shall examine*** the TOE design to determine that it is an accurate instantiation of all security functional requirements.

1005 The evaluator ensures that each security requirement listed in the TOE security functional requirements section of the ST has a corresponding design description in the TOE design that accurately details how the TSF meets that requirement. This requires that the evaluator identify a collection of components that are responsible for implementing a given functional requirement, and then examine those components to understand how the requirement is implemented. Finally, the evaluator would assess whether the requirement was accurately implemented.

1006 As an example, if the ST requirements specified a role-based access control mechanism, the evaluator would first identify the components that contribute to this mechanism's implementation. This could be done by in-depth knowledge or understanding of the TOE design or by work done in the previous work unit. Note that this trace is only to identify the components, and is not the complete analysis.

1007 The next step would be to understand what mechanism the components implemented. For instance, if the design described an implementation of access control based on UNIX-style protection bits, the design would not be an accurate instantiation of those access control requirements present in the ST example used above. If the evaluator could not determine that the mechanism was accurately implemented because of a lack of detail, the evaluator would have to assess whether all of the SFR-enforcing components have been identified, or if adequate detail had been provided for those components.

11.8.1.4 Implied evaluator action

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

ADV_TDS.1-8 The evaluator ***shall examine*** the TOE design to determine that it contains a complete and accurate mapping from the TSFI described in the functional specification to the components of the TSF described in the TOE design.

1008 The components 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 component that is initially involved when an operation is requested at the TSFI, and identify the various components that are primarily responsible for implementing the functionality. Note that a complete "call tree" for each TSFI is not required for this work unit.

1009 The evaluator assesses the completeness of the mapping by ensuring that all of the TSFI map to at least one component. The verification of accuracy is more complex.

1010 The first aspect of accuracy is that each TSFI is mapped to a component at the TSF boundary. This determination can be made by reviewing the component description and interactions, and from this information determining its place in the architecture. The next aspect of accuracy is that the mapping makes sense. For instance, mapping a TSFI dealing with access control to a component that checks passwords is not accurate. The evaluator should again use judgement in making this determination. The goal is that this information aids the evaluator in understanding the system and implementation of the SFRs, and ways in which entities at the TSF boundary can interact with the TSF. The bulk of the assessment of whether the SFRs are described accurately by the components is performed in other work units.

11.8.2 Evaluation of sub-activity (ADV_TDS.2)

11.8.2.1 Input

1011 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design.

11.8.2.2 Action ADV_TDS.2.1E

ADV_TDS.2.1C ***The design shall describe the structure of the TOE in terms of components.***

ADV_TDS.2-1 The evaluator ***shall examine*** the TOE design to determine that the structure of the entire TOE is described in terms of components.

1012 The evaluator ensures that all of the components of the TOE are identified. This description of the TOE will be used as input to work unit ADV_TDS.2-2, where the parts of the TOE that make up the TSF are identified. That is, this requirement is on the entire TOE rather than on only the TSF.

1013 The TOE (and TSF) may be described in multiple layers of abstraction (i.e. components and modules) Depending upon the complexity of the TOE, its design may be described in terms of components and modules, as described in CC Part 3 Annex A.4, ADV_TDS: Components and Modules. At this level of assurance, the decomposition only need be at the "component" level.

1014 In performing this activity, the evaluator examines other evidence presented for the TOE (e.g., ST, operator user guidance) to determine that the description of the TOE in such evidence is consistent with the description contained in the TOE design.

ADV_TDS.2.2C ***The design shall identify all components of the TSF.***

ADV_TDS.2-2 The evaluator ***shall examine*** the TOE design to determine that all components of the TSF are identified.

1015 In work unit ADV_TDS.2-1 all of the components of the TOE were identified, and a determination made that the non-TSF components were correctly characterised. Building on that work, the components that were not characterised as non-TSF components should be precisely identified. The evaluator determines that, of the hardware and software installed and configured according to the Preparative user guidance (AGD_PRE) guidance, each component has been accounted for as either one that is part of the TSF, or one that is not.

ADV_TDS.2.3C ***The design shall describe each SFR non-interfering component of the TSF in detail sufficient to determine that it is SFR non-interfering.***

ADV_TDS.2-3 The evaluator ***shall examine*** the TOE design to determine that each SFR-non-interfering component of the TSF is described such that the evaluator can determine that the component is SFR-non-interfering.

1016 SFR-non-interfering components do not need to be described in detail as to how they function in the system. However, the evaluator makes a determination, based on the evidence provided by the developer, that the components that do not have detailed descriptions are SFR-non-interfering. Note that if the developer provides a uniform level of detailed documentation then this work unit will be largely satisfied, since the point of categorising the components is to allow the developer to provide less information for SFR-non-interfering components than for SFR-enforcing and SFR-supporting components.

1017 An SFR-non-interfering component is one on which the SFR-enforcing and SFR-supporting components have no dependence; that is, they play no role in implementing SFR functionality.

ADV_TDS.2.4C ***The design shall provide a detailed description of the SFR-enforcing behaviour of the SFR-enforcing components.***

ADV_TDS.2-4 The evaluator ***shall examine*** the TOE design to determine that it provides a complete, accurate, and detailed description of the SFR-enforcing behaviour of the SFR-enforcing components.

1018 The evaluator should note that the developer is not required to explicitly designate components as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the components have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular component.

1019 SFR-enforcing behaviour refers to *how* a component 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 component, and how these elements work together to provide the SFR-enforcing behaviour. Such a description also references SFR-supporting behaviour, which the evaluator should consider in performing subsequent work units.

1020 To determine completeness and accuracy, the evaluator examines other information available (e.g., functional specification, architectural design, implementation representation). Descriptions of functionality in these documents should be consistent with what is provided for evidence for this work unit

ADV_TDS.2.5C ***The design shall provide a high-level description of the non-SFR-enforcing behaviour of the SFR-enforcing components.***

ADV_TDS.2-5 The evaluator ***shall examine*** 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 components.

1021 The evaluator should note that the developer is not required to explicitly designate components as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the components have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular component.

1022 In contrast to the previous work unit, this work unit calls for the evaluator to assess the information provided for SFR-enforcing components that is non-SFR-enforcing. The goal of this assessment is two-fold. First, it should provide the evaluator greater understanding of the way each component works. Second, the evaluator determines that all SFR-enforcing behaviour exhibited by a component has been described. Unlike the previous work unit, the information provided for non-SFR-enforcing behaviour does not have to be as detailed as that provided by the SFR-enforcing behaviour. For example, data structures or data items that do not pertain to SFR-enforcing functionality will likely not need to be described in detail, if at all. It is the evaluator's determination, however, with respect to what “high-level” means for a particular TOE, and the evaluator obtains enough information from the developer (even if it turns out to be equivalent to information provided for the parts of the component that are SFR-enforcing) to make a sound verdict for this work unit.

1023 The evaluator is cautioned, however, that “perfect" assurance is not a goal nor required by this work unit, so judgement will have to be exercised in determine the amount and composition of the evidence required to make a verdict on this work unit.

1024 To determine completeness and accuracy, the evaluator examines other information available (e.g., functional specification, architectural design, implementation representation). Descriptions of functionality in these documents should be consistent with what is provided for evidence for this work unit. In particular, the functional specification should be used to determine that the behaviour required to implement the TSF Interfaces described by the functional specification are completely described by the component, since the behaviour will either be SFR-enforcing or non-SFR-enforcing.

ADV_TDS.2.6C ***The design shall provide a high-level description of the behaviour of the SFR-supporting components.***

ADV_TDS.2-6 The evaluator ***shall examine*** the TOE design to determine that it provides a complete and accurate high-level description of the behaviour of the SFR-supporting components.

1025 The evaluator should note that the developer is not required to explicitly designate components as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These “tags” are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the components have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular component.

1026 In contrast to the previous two work units, this work unit calls for the developer to provide (and the evaluator to assess) information about SFR supporting components. Such components should be referenced by the descriptions of the SFR-enforcing components, as well as by the descriptions of interactions in work unit ADV_TDS.2-7. The goal of evaluator's assessment, like that for the previous work unit, is two-fold. First, it should provide the evaluator with an understanding of the way each SFR-supporting component works. Second, the evaluator determines that the behaviour is described in enough detail so that the way in which the component supports the SFR-enforcing behaviour is clear, and that the behaviour is not itself SFR-enforcing. The information provided for SFR-supporting component's behaviour does not have to be as detailed as that provided by the SFR-enforcing behaviour. For example, data structures or data items that do not pertain to SFR-enforcing functionality will likely not need to be described in detail, if at all. It is the evaluator's determination, however, with respect to what “high-level” means for a particular TOE, and the evaluator obtains enough information from the developer (even if it turns out to be equivalent to information provided for the parts of the component that are SFR-enforcing) to make a sound verdict for this work unit.

1027 The evaluator is cautions, however, that “perfect" assurance is not a goal nor required by this work unit, so judgement will have to be exercised in determine the amount and composition of the evidence required to make a verdict on this work unit.

1028 To determine completeness and accuracy, the evaluator examines other information available (e.g., functional specification, architectural design, implementation representation). Descriptions of functionality in these documents should be consistent with what is provided for evidence for this work unit. In particular, the functional specification should be used to determine that the behaviour required to implement the TSF Interfaces described by the functional specification are completely described by the component.

ADV_TDS.2.7C ***The design shall provide a description of the interactions between the components of the TSF.***

ADV_TDS.2-7 The evaluator ***shall examine*** the TOE design to determine that interactions between the components of the TSF are described.

1029 The goal of describing the interactions between the components is to help provide the reader a better understanding of how the TSF performs it functions. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a component to a routine in a different component; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular component that are going to be used by another component should be covered in this discussion. Any control relationships between components (e.g., a component responsible for configuring a rule base for a firewall system and

the component that actually implements these rules) should also be described.

1030 It should be noted while the developer should characterise all interactions between components, the evaluator should use their own judgement in assessing the completeness of the description. If the reason for an interaction is unclear, or if there are SFR-related interactions (discovered, for instance, in examining the descriptions of component behaviour) that do not appear to be described, the evaluator ensures that this information is provided by the developer. However, if the evaluator can determine that interactions among a particular set of components, while incompletely described by the developer, will not aid in understanding the overall functionality nor security functionality provided by the TSF, then the evaluator may choose to consider the description sufficient, and not pursue completeness for its own sake.

11.8.2.3 Action ADV_TDS.2.2E

ADV_TDS.2-8 The evaluator ***shall examine*** the TOE security functional requirements and the TOE design, to determine that all ST security functional requirements are covered by the TOE design.

1031 The evaluator may construct a map between the TOE security functional requirements and the TOE design. This map will likely be from a functional requirement to a set of components. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

1032 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were implemented in specific places within fifteen modules, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to one component and claim the work unit had been completed. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to Component A, behaviours x, y, and z; FDP_ACC.1 Access control (rule 2) to Component A, behaviours x, p, and q; etc.

ADV_TDS.2-9 The evaluator ***shall examine*** the TOE design to determine that it is an accurate instantiation of all security functional requirements.

1033 The evaluator ensures that each security requirement listed in the TOE security functional requirements section of the ST has a corresponding design description in the TOE design that accurately details how the TSF meets that requirement. This requires that the evaluator identify a collection of components that are responsible for implementing a given functional requirement, and then examine those components to understand how the requirement is implemented. Finally, the evaluator would assess whether the requirement was accurately implemented.

1034 As an example, if the ST requirements specified a role-based access control mechanism, the evaluator would first identify the components that contribute to this mechanism's implementation. This could be done by in-depth knowledge or understanding of the TOE design or by work done in the previous work unit. Note that this trace is only to identify the components, and is not the complete analysis.

1035 The next step would be to understand what mechanism the components implemented. For instance, if the design described an implementation of access control based on UNIX-style protection bits, the design would not be an accurate instantiation of those access control requirements present in the ST example used above. If the evaluator could not determine that the mechanism was accurately implemented because of a lack of detail, the evaluator would have to assess whether all of the SFR-enforcing components have been identified, or if adequate detail had been provided for those components.

11.8.2.4 Implied evaluator action

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

ADV_TDS.2-10 The evaluator ***shall examine*** the TOE design to determine that it contains a complete and accurate mapping from the TSFI described in the functional specification to the components of the TSF described in the TOE design.

1036 The components 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 component that is initially involved when an operation is requested at the TSFI, and identify the various components that are primarily responsible for implementing the functionality. Note that a complete “call tree” for each TSFI is not required for this work unit.

1037 The evaluator assesses the completeness of the mapping by ensuring that all of the TSFI map to at least one component. The verification of accuracy is more complex.

1038 The first aspect of accuracy is that each TSFI is mapped to a component at the TSF boundary. This determination can be made by reviewing the component description and interactions, and from this information determining its place in the architecture. The next aspect of accuracy is that the mapping makes sense. For instance, mapping a TSFI dealing with access control to a component that checks passwords is not accurate. The evaluator should again use judgement in making this determination. The goal is that this information aids the evaluator in understanding the system and implementation of the SFRs, and ways in which entities at the TSF boundary can interact with the TSF. The bulk of the assessment of whether the SFRs are described accurately by the components is performed in other work units.

11.8.3 Evaluation of sub-activity (ADV_TDS.3)

11.8.3.1 Objectives

1039 The objective of this sub-activity is to determine whether the TOE design provides a description of the TOE in terms of components sufficient to determine the TSF boundary, and provides a description of the TSF internals in terms of modules (and optionally higher-level abstractions). It provides a detailed description of the SFR-enforcing modules and enough information about the non-SFR-supporting modules for the evaluator to determine that the SFRs are completely and accurately implemented.

11.8.3.2 Input

1040 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the TSF internals document; and

e) the implementation representation.

11.8.3.3 Application notes

1041 There are three types of activity that the evaluator must undertake with respect to the TOE design. First, the evaluator determines that the TSF boundary has been adequately described. Second, the evaluator determines that the developer has provided documentation that conforms to the content and presentation requirements this component, and that is consistent with other documentation provided for the TOE. Finally, the evaluator must analyse the design information provided for the SFR-enforcing modules (at a detailed level) and the non-SFR-enforcing modules (at a less detailed level) to understand how the system is implemented, and with that knowledge ensure that the TSFI in the functional specification are adequately described, and that the test information adequately tests the TSF (done in the Class ATE: Tests work units).

1042 It is important to note that while the developer is obligated to provide a complete description of the TSF (although SFR-enforcing modules will have more detail than the non-SFR-enforcing modules), the evaluator is expected to use their judgement in performing their analysis. While the evaluator is expected to look at every module, the detail to which they examine each module may vary. The evaluator analyses each module in order to gain enough understanding to determine the effect of the functionality of the module on the security of the system, and the depth to which they need to analyse the module may vary depending on the module's role in the system. An important aspect of this analysis is that the evaluator should use the other documentation provided (TSS, functional specification, architectural design,

TSF internal document, and the implementation representation) in order to determine that the functionality that is described is correct, and that the implicit designation of non-SFR-enforcing modules (see below) is supported by their role in the system architecture.

1043 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

11.8.3.4 Action ADV_TDS.3.1E

ADV_TDS.3.1C ***The design shall describe the structure of the TOE in terms of components.***

ADV_TDS.3-1 The evaluator ***shall examine*** the TOE design to determine that the structure of the entire TOE is described in terms of components.

1044 The evaluator ensures that all of the components of the TOE are identified. This description of the TOE will be used as input to work unit ADV_TDS.3-2, where the parts of the TOE that make up the TSF are identified. That is, this requirement is on the entire TOE rather than on only the TSF.

1045 The TOE (and TSF) may be described in multiple layers of abstraction (i.e. components and modules) Depending upon the complexity of the TOE, its design may be described in terms of components and modules, as described in CC Part 3 Annex A.4, ADV_TDS: Components and Modules. For a very simple TOE that can be described solely at the "module" level (see ADV_TDS.3-2), this work unit is not applicable and therefore considered to be satisfied.

1046 In performing this activity, the evaluator examines other evidence presented for the TOE (e.g., ST, operator user guidance) to determine that the description of the TOE in such evidence is consistent with the description contained in the TOE design.

ADV_TDS.3.2C ***The design shall describe the TSF in terms of modules.***

ADV_TDS.3-2 The evaluator ***shall examine*** the TOE design to determine that the entire TSF is described in terms of modules.

1047 The evaluator will examine the modules for specific properties in other work units; in this work unit the evaluator determines that the modular description covers the entire TSF, and not just a portion of the TSF. The evaluator uses other evidence provided for the evaluation (e.g., functional specification,

TSF internals description, architectural description, implementation representation) in making this determination. For example, if the TSF internals description refers to modules not contained in the TOE design description of the TSF, then the evaluator should understand why there is this inconsistency. Likewise, if the functional specification contains interfaces to functionality that does not appear to be described in the TOE design description, it may be the case that a portion of the TSF has not been included appropriately. Making this determination will likely be an iterative process, where as more analysis is done on the other evidence, more confidence can be gained with respect to the completeness of the documentation.

1048 A module is generally a relatively small architectural unit that can be characterised in terms of the properties discussed in TSF internals (ADV_INT). When TSF internals (ADV_INT) requirements are also present in an ST, a "module" in terms in of the TOE design (ADV_TDS) requirements refers to the same entity as a "module" for the TSF internals (ADV_INT) requirements. Unlike components, modules describe the implementation in a level of detail that can serve as a guide to reviewing the implementation representation.

1049 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. 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).

ADV_TDS.3.3C ***The design shall identify all components of the TSF.***

ADV_TDS.3-3 The evaluator ***shall examine*** the TOE design to determine that all components of the TSF are identified.

1050 If the design is presented solely in terms of modules, then components in these requirements are equivalent to modules and the activity should be performed at the module level.

1051 In work unit ADV_TDS.3-1 all of the components of the TOE were identified, and a determination made that the non-TSF components were correctly characterised. Building on that work, the components that were not characterised as non-TSF components should be precisely identified. The evaluator determines that, of the hardware and software installed and

configured according to the Preparative user guidance (AGD_PRE) guidance, each component has been accounted for as either one that is part of the TSF, or one that is not.

ADV_TDS.3.4C ***The design shall provide a description of each component of the TSF.***

ADV_TDS.3-4 The evaluator ***shall examine*** the TOE design to determine that each components of the TSF describes its role in the enforcement of SFRs described in the ST.

1052 If the design is presented solely in terms of modules, then this work unit will be considered satisfied by the assessment done in subsequent work units; no explicit action on the part of the evaluator is necessary in this case.

1053 On systems that are complex enough to warrant a component-level description of the TSF in addition to the modular description, the goal of the component-level description is to give the evaluator context for the modular description that follows. Therefore, the evaluator ensures that the component-level description contains a description of how the security functional requirements are achieved in the design, but at a level of abstraction above the modular description. This description should discuss the mechanisms used at a level that is aligned with the module description; this will provide the evaluators the road map needed to intelligently assess the information contained in the module description. A well-written set of component descriptions will help guide the evaluator in determining the modules that are most important to examine, thus focusing the evaluation activity on the portions of the TSF that have the most relevance with respect to the enforcement of the SFRs.

1054 The evaluator ensures that all components of the TSF have a description. While the description should focus on the role that the component plays in enforcing or supporting the implementation of the SFRs, enough information must be present so that a context for understanding the SFR-related functionality is provided.

ADV_TDS.3.5C ***The design shall provide a description of the interactions between the components of the TSF.***

ADV_TDS.3-5 The evaluator ***shall examine*** the TOE design to determine that interactions between the components of the TSF are described.

1055 If the design is presented solely in terms of modules, then this work unit will be considered satisfied by the assessment done in subsequent work units; no explicit action on the part of the evaluator is necessary in this case.

1056 On systems that are complex enough to warrant a component-level description of the TSF in addition to the modular description, the goal of describing the interactions between the components is to help provide the reader a better understanding of how the TSF performs it functions. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a component to a routine in a different

component; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular component that are going to be used by another component should be covered in this discussion. Any control relationships between components (e.g., a component responsible for configuring a rule base for a firewall system and the component that actually implements these rules) should also be described.

1057 It should be noted while the developer should characterise all interactions between components, the evaluator should use their own judgement in assessing the completeness of the description. If the reason for an interaction is unclear, or if there are SFR-related interactions (discovered, for instance, in examining the module-level documentation) that do not appear to be described, the evaluator ensures that this information is provided by the developer. However, if the evaluator can determine that interactions among a particular set of components, while incompletely described by the developer, will not aid in understanding the overall functionality nor security functionality provided by the TSF, then the evaluator may choose to consider the description sufficient, and not pursue completeness for its own sake.

ADV_TDS.3.6C ***The design shall provide a mapping from the components of the TSF to the modules of the TSF.***

ADV_TDS.3-6 The evaluator ***shall examine*** the TOE design to determine that the mapping between the components of the TSF to the modules of the TSF is complete.

1058 If the design is presented solely in terms of modules, then this work unit is considered satisfied.

1059 For TOEs that are complex enough to warrant a component-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 components. This will provide the evaluator a guide in performing their module-level assessment. To determine completeness, the evaluator examines each mapping and determines that all components map to at least one module, and that all modules map to exactly one component.

ADV_TDS.3-7 The evaluator ***shall examine*** the TOE design to determine that the mapping between the components of the TSF to the modules of the TSF is accurate.

1060 If the design is presented solely in terms of modules, then this work unit is considered satisfied.

1061 For TOEs that are complex enough to warrant a component-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 components. This will provide the evaluator a guide in performing their module-level assessment. The goal of the accuracy determination is to ensure that modules are mapped to components in a manner consistent with the actual implementation. The evaluator may choose to check the accuracy of the mapping in conjunction with performing other work units. An

"inaccurate" mapping is one where the module is mistakenly associated with a component where its functions are not used within the component. Because the mapping is intended to be a guide supporting more detailed analysis, the evaluator is cautioned to apply appropriate effort to this work unit. Expending extensive evaluator resources verifying the accuracy of the mapping is not necessary. Inaccuracies that lead to mis-understandings related to the design that are uncovered as part of this or other work units are the ones that should be associated with this work unit and corrected.

ADV_TDS.3.7C ***The design shall identify and describe data that are common to more than one module, where any of the modules is an SFR-enforcing module.***

ADV_TDS.3-8 The evaluator ***shall examine*** the TOE design to determine that it identifies and describes the global data that are common to a SFR-enforcing module and at least one other module.

1062 If the implementation does not make use of global data, then this work unit is not applicable and considered to be satisfied.

1063 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1064 Global data are those data that are used by more than one module, yet are not passed as formal parameters or messages to a module. For the purposes of this work unit, the criterion is that the data are used, as opposed to just "being available." The intent is that the evaluator performs this work unit for global data that are used (read or written) by the SFR-enforcing modules, even if all of the other users of those data are SFR-supporting modules. Note that if there are SFR-non-interfering modules that use those data, the evaluator should consider whether the SFR-non-interfering modules are correctly characterised.

1065 The evaluator determines that global data are identified and described much like parameters of an explicit "call" interface. The evaluator should be able to tell from the description of the global data what their role in the system design and implementation is. The evaluator also refers to the global variable analysis performed as part of the coupling analysis in the TSF Internals document (TSF internals (ADV_INT)) to determine whether the identification and description of the global data is complete.

1066 The evaluator also examines other information available (e.g., functional specification, architectural design, implementation representation) to determine that the global data described seem complete.

ADV_TDS.3.8C ***The design shall describe each SFR-enforcing module in terms of its purpose, interfaces, return values from those interfaces, and called interfaces to other modules.***

ADV_TDS.3-9 The evaluator ***shall examine*** the TOE design to determine that the description of the purpose of each SFR-enforcing module is complete and accurate.

1067 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1068 The purpose for a module provides is a short description indicating what function the module is fulfilling. The description should be written such that the evaluator is able to obtain a general idea of what the module's function is in the overall TSF architecture. In order to assure the description is complete, the evaluator uses other information provided about the module in the TOE design (interfaces, algorithmic description, etc.) to ensure that the major functionality is reflected in the description. The evaluator determines accuracy by ensuring that the description of the functionality matches the information contained in other information available for the module (interfaces, interactions, algorithmic description, etc.).

1069 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the TSF internals, or the architectural design. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the purpose is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.3.2D (work unit ADV_TDS.3-20), which maps the TSFI in the functional specification to the modules of the TSF.

ADV_TDS.3-10 The evaluator ***shall examine*** the TOE design to determine that the description of the interfaces presented by each SFR-enforcing module contain an accurate and complete description of the parameters, the invocation conventions for each interface, and any values returned directly by the interface.

1070 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These “tags” are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1071 The interfaces of a module are those interfaces used by other modules as a means to invoke the operations provided, and to provide inputs to or receive outputs from the module. Interfaces are described in terms of how they are invoked, and any values that are returned. This description would include a list of parameters, and descriptions of these parameters. 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.

1072 In terms of the assessment of parameters (inputs and outputs) to a module, any use of global data must also be considered. A module “uses” global data if it either reads or writes the data. In order to assure the description of such parameters (if used) is complete, the evaluator uses other information provided about the module in the TOE design (interfaces, algorithmic description, etc.), as well as the description of the particular set of global data assessed in work unit ADV_TDS.3-8. For instance, the evaluator could first determine the processing the module performs by examining its function and interfaces presented (particularly the parameters of the interfaces). They could then check to see if the processing appears to “touch” any of the global data areas identified in the TDS design. The evaluator then determines that, for each global data area that appears to be “touched”, that global data area is listed as a means of input or output by the module the evaluator is examining.

1073 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.

1074 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).

1075 In order to assure the description is complete, the evaluator uses other information provided about the module in the TOE design (e.g., algorithmic description, global data used) to ensure that it appears all data necessary for performing the functions of the module is presented to the module, and that any values that other modules expect the module under examination to provide are identified as being returned by the module. The evaluator determines accuracy by ensuring that the description of the processing matches the information listed as being passed to or from an interface.

1076 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the TSF internals, or the architectural design. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the purpose is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.3.2D (work unit ADV_TDS.3-20), which maps the TSFI in the functional specification to the modules of the TSF.

1077 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is a complete and accurate reflection of the implementation.

ADV_TDS.3-11 The evaluator ***shall examine*** the TOE design to determine that the description of the interfaces used by each SFR-enforcing module are uniquely and completely identified.

1078 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These “tags” are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1079 Interfaces used by each SFR-enforcing module must be identified in a unique manner such that it can be determined by the evaluator which module is

being invoked by the module being described. Note that this applies to all interfaces used by a SFR-enforcing module, not just those interfaces presented by other SFR-enforcing modules.

1080 In order to assess completeness, the evaluator uses other information provided about the module in the TOE design (e.g., algorithmic description) to ensure that it appears all functionality that is not supplied directly by the code associated with that module is provided by an identified (external) module. For instance, if the algorithmic description of a module is discussing the manipulation of certain data, and from one step to the next it appears that the data are "suddenly" sorted, the evaluator might question whether a "sort module" interface was invoked to accomplish this.

1081 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is a complete and accurate reflection of the implementation.

ADV_TDS.3-12 The evaluator ***shall examine*** the TOE design to determine that the description of the interfaces used by each SFR-enforcing module are of sufficient detail to determine the algorithmic purpose of invoking the interface, and the expected results of invoking the interface.

1082 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1083 It must also be clear from the TOE design the algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be "Module A invokes the *double_bubble()* interface in Module B to perform a bubble sort." An adequate algorithmic description would be "Module A invokes the *double_bubble* routine with the list of access control entries; *double_bubble()* will return the entries sorted first on the username, then on the *access_allowed* field according the following rules..." The TOE design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting these called interfaces is via a call tree, and then the algorithmic purpose for invoking the interfaces can be included in

the algorithmic description of the module. If the call-tree approach is used, the analysis of the work associated with the previous work unit might be made easier.

1084 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is complete and accurate with respect to the algorithmic reason for calling an interface.

ADV_TDS.3.9C ***For each SFR-enforcing module, the design shall provide an algorithmic description detailed enough to represent the TSF implementation.***

ADV_TDS.3-13 The evaluator ***shall examine*** the TOE design to determine that the algorithmic description of each SFR-enforcing module is complete and accurate.

1085 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These “tags” are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1086 This work unit should be performed in association with ADV_TDS.3-14. This work unit deals with quality of the algorithmic description, while ADV_TDS.3-14 deals with the level of detail associated with the algorithmic description.

1087 A word of caution to evaluator is in order. The focus of this work unit should be to provide the evaluator an understanding of how the module works so that determinations can be made about the soundness of the implementation of the SFRs, as well as to support architectural analysis performed for ADV_ARC components. Given that infinite and perfect evaluations are unattainable and impractical, the evaluator must exercise judgement in performing this work unit (they should do so for other work units as well, but because of the time it will take to perform this work unit, it is especially critical). As long as the evaluator has a sound understanding of the module's operation, and its relationship to other modules and the TOE as a whole, the evaluator should consider the objective of the work achieved and not engage in a documentation exercise for the developer (by requiring, for example, a more complete algorithmic description that is present). Conversely, it is the evaluator's decision about what constitutes a sufficient algorithmic

description, not the developer's. The requirement is that the developer must provide a full algorithmic description, and so the evaluator is not obligated to accept anything less, and is indeed not obligated to justify accepting anything less; the requirement provides the necessary justification.

1088 The algorithmic description is complete if it describes (in an algorithmic fashion) all of the functionality performed by the module. This description should detail how the module inputs are manipulated to produce the desired system effect, using the information provided about the parameters to the interface being invoked, as well as any calls made to other modules. The evaluator should examine the other evidence presented about the module (global data use, interfaces, function) to ensure that the description appears to be complete.

1089 Interfaces used by each SFR-enforcing module are identified in a unique manner such that it can be determined by the evaluator which module is being invoked by the module being described. Note that this applies to all interfaces used by a SFR-enforcing module, not just those interfaces presented by other SFR-enforcing modules.

1090 It must also be clear from the TOE design the algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be “Module A invokes the *double_bubble()* interface in Module B to perform a bubble sort.” An adequate algorithmic description would be “Module A invokes the *double_bubble* routine with the list of access control entries; *double_bubble()* will return the entries sorted first on the username, then on the *access_allowed* field according the following rules...” The TOE design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting these called interfaces is via a call tree, and then the algorithmic description can be included in the algorithmic description of the module.

1091 Because the algorithmic description is at such a low level of detail, it may be difficult determine completeness and accuracy impacts from higher-level documentation, such as administrative guidance, the functional specification, the architectural design, or the TSF internal design documents. However, the evaluator uses the information present in those documents to help ensure that the algorithmic description is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.3.2D (work unit ADV_TDS.3-20), which maps the TSFI in the functional specification to the modules of the TSF.

1092 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the

case that the TOE design is a complete and accurate description of the implemented algorithm.

ADV_TDS.3-14 The evaluator ***shall examine*** the TOE design to determine that the algorithmic detail for each SFR-enforcing module is representative of the actual TSF implementation.

1093 The evaluator should note that the developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. It is the evaluator's responsibility to determine that the modules have the appropriate information for their role (SFR-enforcing, etc.) in the TOE, and to obtain the appropriate information from the developer should the developer fail to provide the required information for a particular module.

1094 The focus of this work unit should be to provide the evaluator with confidence that the module descriptions for SFR-enforcing modules sufficiently describe the implementation representation, where sufficiently is determined by the evaluator. The same caution to evaluator (and caveats to developer) as were discussed for the previous work unit apply to this work unit; the evaluator must exercise judgement in the degree of detail they need to be present in the algorithmic description to make a determination that it is sufficiently reflective of the underlying implementation. Requiring the developer to provide additional detail in the presentation of the algorithmic description of the module should be weighed by the evaluator in terms of the benefit provided by such additional information.

1095 The algorithmic description of an SFR-enforcing module describes in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or other presentation methods. It discusses how the inputs to the modules and operations obtained from called modules are used to accomplish the function of the module. It notes changes to global data, system state, and other outputs produced by the module. A description of a module in the TOE design should be of sufficient detail so that one could create an implementation of the module from the TOE design, and that implementation would be identical to the actual TSF implementation in terms of its inputs, outputs, and operations accomplished; and the internals of the module would be algorithmically identical. 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 they were part of the TSF).

1096 As previously indicated, the evaluator should sample the implementation representation for modules to ensure the description in the TOE design reflects the implementation in the appropriate amount of detail. It should be fairly easy for the evaluator to understand all portions of a module's implementation representation if the TOE design is written in sufficient detail.

ADV_TDS.3.10C ***The design shall describe each non-SFR-enforcing module in terms of its purpose and interaction with other modules.***

ADV_TDS.3-15 The evaluator ***shall examine*** the TOE design to determine that non-SFR-supporting modules are correctly categorized.

1097 The developer is not required to explicitly designate modules as SFR-enforcing, SFR-supporting, and SFR non-interfering (although they are free to do so if they wish). These "tags" are used only to describe the amount and type of information the developer must provide, and can be used to limit the amount of information the developer has to develop if their engineering process does not produce the documentation required. If the developer has provided a uniform level of documentation for all modules in the TSF, then the evaluator is expected to determine which are SFR-enforcing and which are not (see work unit ADV_TDS.3-18 for the reporting of this information) and apply the appropriate work units.

1098 In the cases where the developer has provided different amounts of information for different modules, an implicit categorisation has been done. That is, modules (for instance) with detail presented on their interfaces and accompanied by an algorithmic description are candidate SFR-enforcing modules (although examination by the evaluator may lead to a determination that some set of them are non-SFR-enforcing). Those with only a description of their purpose and interaction with other modules (for instance) are "implicitly categorised" as non-SFR-enforcing.

1099 In these cases, a key focus of the evaluator for this work unit is attempting to determine from the evidence provided for each module implicitly categorised as non-SFR-enforcing and the evaluation (information about other modules in the TOE design, the functional specification, the architectural design, the administrative guidance, the TSF internals document, and perhaps even the implementation representation) whether the module is indeed non-SFR-enforcing. At this level of assurance some error should be tolerated; the evaluator does not have to be absolutely sure that a given module is non-SFR-enforcing, even though it is labelled as such. However, if the evidence provided indicates that a non-SFR-enforcing module is SFR-enforcing, the evaluator requests additional information from the developer in order to resolve the apparent inconsistency. For instance, suppose the documentation for Module A (an SFR-enforcing module) indicates that it calls Module B to perform an access check on a certain type of construct. When the evaluator examines the information associated with Module B, they find that all the

developer has provided is a purpose and a set of interactions (thus implicitly categorising Module B as non-SFR-enforcing). On examining the purpose and interactions from Module A, the evaluator finds no mention of Module B performing any access checks, and Module A is not listed as a module with which Module B interacts. At this point the evaluator should approach the developer to resolve the discrepancies between the information provided in Module A and that in Module B.

1100 Another example would be where the evaluator examines the mapping of the TSFI to the modules as provided by ADV_TDS.3.2D. This examination shows that Module C is associated with an SFR requiring identification of the user. Again, when the evaluator examines the information associated with Module C, they find that all the developer has provided is a purpose and a set of interactions (thus implicitly categorising Module C as non-SFR-enforcing). Examining the purpose and interactions presented for Module C, the evaluator is unable to determine why Module C, listed as mapping to a TSFI concerned with user identification, would not be classified as SFR-enforcing. Again, the evaluator should approach the developer to resolve this discrepancy.

1101 A final example from the opposite point of view. As before, the developer has provided information associated with Module D consisting of a purpose and a set of interactions (thus implicitly categorising Module D as non-SFR-enforcing). The evaluator examines all of the evidence provided, including the purpose and interactions for Module D. The purpose appears to give a meaningful description of Module D's function in the TOE, the interactions are consistent with that description, and there is nothing to indicate that Module D is SFR-enforcing. In this case, the evaluator should not demand more information about Module D “just be to sure” it is correctly categorised. The developer has met their obligations and the resulting assurance the evaluator has in the implicit categorisation of Module D is (by definition) appropriate for this assurance level.

ADV_TDS.3-16 The evaluator ***shall examine*** the TOE design to determine that the description of the purpose of each non-SFR-supporting module is complete and accurate.

1102 The purpose a module provides is a short description indicating what function the module is fulfilling. The evaluator should be able to obtain a general idea of what the module's function is in the architecture from the description. In order to assure the description is complete, the evaluator uses the information provided about the module's interactions with other modules to assess whether the reasons for the module being called are consistent with the module's purpose. If the interaction description contains functionality that is not apparent from, or in conflict with, the module's purpose, the evaluator needs to determine whether the problem is one of accuracy or of completeness. The evaluator should be wary of purposes that are too short, since meaningful analysis based on a one-sentence purpose is likely impossible.

1103 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the architectural design, or the TSF internals document. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the function is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.3.2D (work unit ADV_TDS.3-20), which maps the TSFI in the functional specification to the modules of the TSF.

ADV_TDS.3-17 The evaluator ***shall examine*** the TOE design to determine that the description of a non-SFR-supporting module's interaction with other modules is complete and accurate.

1104 It is important to note that, in terms of the Part 3 requirement and this work unit, the term *interaction* is intended to convey less rigour than *interface*. An interaction does not need to be characterised at the implementation level (e.g., parameters passed from one routine in a module to a routine in a different module; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular module that are going to be used by another module should be covered in this discussion. Any control relationships between modules (e.g., a module responsible for configuring a rule base for a firewall system and the module that actually implements these rules) should also be described.

1105 A module's interaction with other modules can be captured in many ways. The intent for the TOE design is to allow the evaluator to understand (in part through analysis of module interactions) the role of the non-SFR-enforcing modules in the overall TOE design. Understanding of this role will aid the evaluator in performing work unit ADV_TDS.3-15.

1106 A module's interaction with other modules goes beyond just a call-tree-type document. The interaction is described from a functional perspective of why a module interacts with other modules. The module's purpose describes what functions the module provides to other modules; the interactions should describe what the module depends on from other modules in order to accomplish this function.

1107 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the architectural design, or the TSF internals document. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the interactions are accurately and completely described.

11.8.3.5 Action ADV_TDS.3.2E

ADV_TDS.3-18 The evaluator ***shall examine*** the TOE security functional requirements and the TOE design, to determine that all ST security functional requirements are covered by the TOE design.

1108 The evaluator may construct a map between the TOE security functional requirements and the TOE design. This map will likely be from a functional requirement to a set of components. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

1109 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were implemented in specific places within fifteen modules, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to one component and claim the work unit had been completed. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to Component A, behaviours x, y, and z; FDP_ACC.1 Access control (rule 2) to Component A, behaviours x, p, and q; etc.

ADV_TDS.3-19 The evaluator ***shall examine*** the TOE design to determine that it is an accurate instantiation of all security functional requirements.

1110 The evaluator ensures that each security requirement listed in the TOE security functional requirements section of the ST has a corresponding design description in the TOE design that accurately details how the TSF meets that requirement. This requires that the evaluator identify a collection of components that are responsible for implementing a given functional requirement, and then examine those components to understand how the requirement is implemented. Finally, the evaluator would assess whether the requirement was accurately implemented.

1111 As an example, if the ST requirements specified a role-based access control mechanism, the evaluator would first identify the components that contribute to this mechanism's implementation. This could be done by in-depth knowledge or understanding of the TOE design or by work done in the previous work unit. Note that this trace is only to identify the components, and is not the complete analysis.

1112 The next step would be to understand what mechanism the components implemented. For instance, if the design described an implementation of access control based on UNIX-style protection bits, the design would not be an accurate instantiation of those access control requirements present in the ST example used above. If the evaluator could not determine that the mechanism was accurately implemented because of a lack of detail, the evaluator would have to assess whether all of the SFR-enforcing components have been identified, or if adequate detail had been provided for those components.

11.8.3.6 Implied evaluator action

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

ADV_TDS.3-20 The evaluator ***shall examine*** the TOE design to determine that it contains a complete and accurate mapping from the TSFI described in the functional specification to the modules of the TSF described in the TOE design.

1113 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. However, a complete call tree for each TSFI is not required for this work unit. The cases in which more than one module would have to be identified are where there are "entry point" modules or wrapper modules that have no functionality other than conditioning inputs or de-multiplexing an input. Mapping to one of these modules would not provide any useful information to the evaluator.

1114 The evaluator assesses the completeness of the mapping by ensuring that all of the TSFI map to at least one module. The verification of accuracy is more complex.

1115 The first aspect of accuracy is that each TSFI is mapped to a module at the TSF boundary. This determination can be made by reviewing the module description and its interfaces/interactions. The next aspect of accuracy is that each TSFI identifies a chain of modules between the initial module identified and a module that is primarily responsible for implementing the function presented at the TSF. Note that this may be the initial module, or there may be several modules, depending on how much pre-conditioning of the inputs is done. It should be noted that one indicator of a pre-conditioning module is that it is invoked for a large number of the TSFI, where the TSFI are all of similar type (e.g., system call). The final aspect of accuracy is that the mapping makes sense. For instance, mapping a TSFI dealing with access control to a module that checks passwords is not accurate. The evaluator should again use judgement in making this determination. The goal is that this information aids the evaluator in understanding the system and implementation of the SFRs, and ways in which entities at the TSF boundary can interact with the TSF. The bulk of the assessment of whether the SFRs are described accurately by the modules is performed in other work units.

11.8.4 Evaluation of sub-activity (ADV_TDS.4)

11.8.4.1 Input

1116 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the TSF internals document; and

e) the implementation representation.

11.8.4.2 Action ADV_TDS.4.1E

ADV_TDS.4.1C ***The design shall describe the structure of the TOE in terms of components.***

ADV_TDS.4-1 The evaluator ***shall examine*** the TOE design to determine that the structure of the entire TOE is described in terms of components.

1117 The evaluator ensures that all of the components of the TOE are identified. This description of the TOE will be used as input to work unit ADV_TDS.4-2, where the parts of the TOE that make up the TSF are identified. That is, this requirement is on the entire TOE rather than on only the TSF.

1118 The TOE (and TSF) may be described in multiple layers of abstraction (i.e. components and modules) Depending upon the complexity of the TOE, its design may be described in terms of components and modules, as described in CC Part 3 Annex A.4, ADV_TDS: Components and Modules. For a very simple TOE that can be described solely at the "module" level (see ADV_TDS.4-2), this work unit is not applicable and therefore considered to be satisfied.

1119 In performing this activity, the evaluator examines other evidence presented for the TOE (e.g., ST, operator user guidance) to determine that the description of the TOE in such evidence is consistent with the description contained in the TOE design.

ADV_TDS.4.2C ***The design shall describe the TSF in terms of modules, designating each module as either as SFR-enforcing, SFR-supporting, or SFR-non-interfering.***

ADV_TDS.4-2 The evaluator ***shall examine*** the TOE design to determine that the entire TSF is described in terms of modules.

1120 The evaluator will examine the modules for specific properties in other work units; in this work unit the evaluator determines that the modular description covers the entire TSF, and not just a portion of the TSF. The evaluator uses other evidence provided for the evaluation (e.g., functional specification, TSF internals description, architectural description, implementation representation) in making this determination. For example, if the TSF internals description refers to modules not contained in the TOE design description of the TSF, then the evaluator should understand why there is this inconsistency. Likewise, if the functional specification contains interfaces to functionality that does not appear to be described in the TOE design description, it may be the case that a portion of the TSF has not been included appropriately. Making this determination will likely be an iterative process, where as more analysis is done on the other evidence, more confidence can be gained with respect to the completeness of the documentation.

1121 A module is generally a relatively small architectural unit that can be characterised in terms of the properties discussed in TSF internals (ADV_INT). When TSF internals (ADV_INT) requirements are also present in an ST, a "module" in terms in of the TOE design (ADV_TDS) requirements refers to the same entity as a "module" for the TSF internals (ADV_INT) requirements. Unlike components, modules describe the implementation in a level of detail that can serve as a guide to reviewing the implementation representation.

1122 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. 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).

ADV_TDS.4-3 The evaluator ***shall check*** the TOE design to determine that the TSF modules are identified as either SFR-enforcing, SFR-supporting, or SFR-non-interfering.

1123 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. It is always permissible for the developer to provide more information or detail than the requirements demand, as might occur when the information has been gathered outside the evaluation context). In such cases the developer must still designate the modules as either SFR-enforcing, SFR-supporting, or SFR-non-interfering.

1124 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. While blatant mis-designations may be immediately apparent (e.g., designating an authentication module as anything but SFR-enforcing when User authentication (FIA_UAU) is one of the SFRs being claimed), other mis-designations might not be discovered until the TSF is better understood. The evaluator must therefore keep in mind that these designations are the developer's initial best effort, but are subject to change. Further guidance is provided under workunit ADV_TDS.4-16, which examines the accuracy of these designations.

ADV_TDS.4.3C ***The design shall identify all components of the TSF.***

ADV_TDS.4-4 The evaluator ***shall examine*** the TOE design to determine that all components of the TSF are identified.

1125 If the design is presented solely in terms of modules, then components in these requirements are equivalent to modules and the activity should be performed at the module level.

1126 In work unit ADV_TDS.4-1 all of the components of the TOE were identified, and a determination made that the non-TSF components were correctly characterised. Building on that work, the components that were not characterised as non-TSF components should be precisely identified. The evaluator determines that, of the hardware and software installed and configured according to the Preparative user guidance (AGD_PRE) guidance, each component has been accounted for as either one that is part of the TSF, or one that is not.

ADV_TDS.4.4C ***The design shall provide a description of each component of the TSF.***

ADV_TDS.4-5 The evaluator ***shall examine*** the TOE design to determine that each components of the TSF describes its role in the enforcement of SFRs described in the ST.

1127 If the design is presented solely in terms of modules, then this work unit will be considered satisfied by the assessment done in subsequent work units; no explicit action on the part of the evaluator is necessary in this case.

1128 On systems that are complex enough to warrant a component-level description of the TSF in addition to the modular description, the goal of the component-level description is to give the evaluator context for the modular description that follows. Therefore, the evaluator ensures that the component-level description contains a description of how the security functional requirements are achieved in the design, but at a level of abstraction above the modular description. This description should discuss the mechanisms used at a level that is aligned with the module description; this will provide the evaluators the road map needed to intelligently assess the information contained in the module description. A well-written set of component descriptions will help guide the evaluator in determining the modules that are most important to examine, thus focusing the evaluation activity on the portions of the TSF that have the most relevance with respect to the enforcement of the SFRs.

1129 The evaluator ensures that all components of the TSF have a description. While the description should focus on the role that the component plays in enforcing or supporting the implementation of the SFRs, enough information must be present so that a context for understanding the SFR-related functionality is provided.

ADV_TDS.4.5C ***The design shall provide a description of the interactions between the components of the TSF.***

ADV_TDS.4-6 The evaluator ***shall examine*** the TOE design to determine that interactions between the components of the TSF are described.

1130 If the design is presented solely in terms of modules, then this work unit will be considered satisfied by the assessment done in subsequent work units; no explicit action on the part of the evaluator is necessary in this case.

1131 On systems that are complex enough to warrant a component-level description of the TSF in addition to the modular description, the goal of describing the interactions between the components is to help provide the reader a better understanding of how the TSF performs it functions. These interactions do not need to be characterised at the implementation level (e.g., parameters passed from one routine in a component to a routine in a different component; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular component that are going to be used by another component should be covered in this discussion. Any control relationships between components (e.g., a component responsible for configuring a rule base for a firewall system and the component that actually implements these rules) should also be described.

1132 It should be noted while the developer should characterise all interactions between components, the evaluator should use their own judgement in assessing the completeness of the description. If the reason for an interaction is unclear, or if there are SFR-related interactions (discovered, for instance, in examining the module-level documentation) that do not appear to be described, the evaluator ensures that this information is provided by the developer. However, if the evaluator can determine that interactions among a particular set of components, while incompletely described by the developer, will not aid in understanding the overall functionality nor security functionality provided by the TSF, then the evaluator may choose to consider the description sufficient, and not pursue completeness for its own sake.

ADV_TDS.4.6C ***The design shall provide a mapping from the components of the TSF to the modules of the TSF.***

ADV_TDS.4-7 The evaluator ***shall examine*** the TOE design to determine that the mapping between the components of the TSF to the modules of the TSF is complete.

1133 If the design is presented solely in terms of modules, then this work unit is considered satisfied.

1134 For TOEs that are complex enough to warrant a component-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 components. This will provide the evaluator a guide in performing their module-level assessment. To determine completeness, the evaluator examines each mapping and determines that all components map to at least one module, and that all modules map to exactly one component.

ADV_TDS.4-8 The evaluator ***shall examine*** the TOE design to determine that the mapping between the components of the TSF to the modules of the TSF is accurate.

1135 If the design is presented solely in terms of modules, then this work unit is considered satisfied.

1136 For TOEs that are complex enough to warrant a component-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 components. This will provide the evaluator a guide in performing their module-level assessment. The goal of the accuracy determination is to ensure that modules are mapped to components in a manner consistent with the actual implementation. The evaluator may choose to check the accuracy of the mapping in conjunction with performing other work units. An "inaccurate" mapping is one where the module is mistakenly associated with a component where its functions are not used within the component. Because the mapping is intended to be a guide supporting more detailed analysis, the evaluator is cautioned to apply appropriate effort to this work unit. Expending extensive evaluator resources verifying the accuracy of the mapping is not necessary. Inaccuracies that lead to mis-understandings related to the design that are uncovered as part of this or other work units are the ones that should be associated with this work unit and corrected.

ADV_TDS.4.7C ***The design shall identify and describe data that are common to more than one module, where any of the modules is an SFR-enforcing or SFR-supporting module.***

ADV_TDS.4-9 The evaluator ***shall examine*** the TOE design to determine that it identifies and describes the global data that are used by SFR-enforcing and/or SFR-supporting modules.

1137 If the implementation does not make use of global data, then this work unit is not applicable and considered to be satisfied.

1138 Global data are those data that are used by more than one module, yet are not passed as formal parameters or messages to a module. For the purposes of this work unit, the criterion is that the data are used, as opposed to just "being available." The intent is that the evaluator performs this work unit for global data that are used (read or written) by the SFR-enforcing modules, SFR-supporting modules or both. Note that if there are SFR-non-interfering modules that use those data, the evaluator should consider whether the SFR-non-interfering modules are correctly characterised.

1139 The evaluator determines that global data are identified and described much like parameters of an explicit "call" interface. The evaluator should be able to tell from the description of the global data what their role in the system design and implementation is. The evaluator also refers to the global variable analysis performed as part of the coupling analysis in the TSF Internals document (TSF internals (ADV_INT)) if available, to determine whether the identification and description of the global data is complete.

1140 The evaluator also examines other information available (e.g., functional specification, architectural design, implementation representation) to determine that the global data described seem complete.

ADV_TDS.4.8C ***The design shall describe each SFR-enforcing and SFR-supporting module in terms of its purpose, interfaces, return values from those interfaces, and called interfaces to other modules.***

ADV_TDS.4-10 The evaluator ***shall examine*** the TOE design to determine that the description of the purpose of each SFR-enforcing and SFR-supporting module is complete and accurate.

1141 The purpose for a module provides is a short description indicating what function the module is fulfilling. The description should be written such that the evaluator is able to obtain a general idea of what the module's function is in the overall TSF architecture. In order to assure the description is complete, the evaluator uses other information provided about the module in the TOE design (interfaces, algorithmic description, etc.) to ensure that the major functionality is reflected in the description. The evaluator determines accuracy by ensuring that the description of the functionality matches the information contained in other information available for the module (interfaces, interactions, algorithmic description, etc.).

1142 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the TSF internals, or the architectural design. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the purpose is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.4.2D (work unit ADV_TDS.4-21), which maps the TSFI in the functional specification to the modules of the TSF.

ADV_TDS.4-11 The evaluator ***shall examine*** the TOE design to determine that the description of the interfaces presented by each SFR-enforcing and SFR-supporting module contain an accurate and complete description of the parameters, the invocation conventions for each interface, and any values returned directly by the interface.

1143 The interfaces of a module are those interfaces used by other modules as a means to invoke the operations provided, and to provide inputs to or receive outputs from the module. Interfaces are described in terms of how they are invoked, and any values that are returned. This description would include a list of parameters, and descriptions of these parameters. 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.

1144 In terms of the assessment of parameters (inputs and outputs) to a module, any use of global data must also be considered. A module “uses” global data if it either reads or writes the data. In order to assure the description of such parameters (if used) is complete, the evaluator uses other information provided about the module in the TOE design (interfaces, algorithmic description, etc.), as well as the description of the particular set of global data assessed in work unit ADV_TDS.4-9. For instance, the evaluator could first determine the processing the module performs by examining its function and interfaces presented (particularly the parameters of the interfaces). They could then check to see if the processing appears to “touch” any of the global data areas identified in the TDS design. The evaluator then determines that, for each global data area that appears to be “touched”, that global data area is listed as a means of input or output by the module the evaluator is examining.

1145 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.

1146 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).

1147 In order to assure the description is complete, the evaluator uses other information provided about the module in the TOE design (e.g., algorithmic description, global data used) to ensure that it appears all data necessary for performing the functions of the module is presented to the module, and that any values that other modules expect the module under examination to provide are identified as being returned by the module. The evaluator determines accuracy by ensuring that the description of the processing matches the information listed as being passed to or from an interface.

1148 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the TSF internals, or the architectural design. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the purpose is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.4.2D (work unit ADV_TDS.4-21), which maps the TSFI in the functional specification to the modules of the TSF.

1149 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is a complete and accurate reflection of the implementation.

ADV_TDS.4-12 The evaluator ***shall examine*** the TOE design to determine that the description of the interfaces used by each SFR-enforcing and/or SFR-supporting module are uniquely and completely identified.

1150 Interfaces used by each SFR-enforcing and SFR-supporting module module must be identified in a unique manner such that it can be determined by the evaluator which module is being invoked by the module being described.

1151 In order to assess completeness, the evaluator uses other information provided about the module in the TOE design (e.g., algorithmic description) to ensure that it appears all functionality that is not supplied directly by the code associated with that module is provided by an identified (external) module. For instance, if the algorithmic description of a module is discussing the manipulation of certain data, and from one step to the next it appears that the data are "suddenly" sorted, the evaluator might question whether a "sort module" interface was invoked to accomplish this.

1152 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is a complete and accurate reflection of the implementation.

ADV_TDS.4-13 The evaluator ***shall examine*** the TOE design to determine that the description of the interfaces used by each SFR-enforcing and/or SFR-supporting module are of sufficient detail to determine the algorithmic purpose of invoking the interface, and the expected results of invoking the interface.

1153 It must be clear from the TOE design the algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be "Module A invokes the *double_bubble()* interface in Module B to perform a bubble sort." An adequate algorithmic description would be "Module A invokes the *double_bubble* routine with the list of access control entries; *double_bubble()* will return the entries sorted first on the username, then on the *access_allowed* field according the following rules..." The TOE design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting

these called interfaces is via a call tree, and then the algorithmic purpose for invoking the interfaces can be included in the algorithmic description of the module. If the call-tree approach is used, the analysis of the work associated with the previous work unit might be made easier.

1154 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is complete and accurate with respect to the algorithmic reason for calling an interface.

ADV_TDS.4.9C ***For each SFR-enforcing and SFR-supporting module, the design shall provide an algorithmic description detailed enough to represent the TSF implementation.***

ADV_TDS.4-14 The evaluator ***shall examine*** the TOE design to determine that the algorithmic description of each SFR-enforcing and SFR-supporting module is complete and accurate.

1155 This work unit should be performed in association with ADV_TDS.4-15. This work unit deals with quality of the algorithmic description, while ADV_TDS.4-15 deals with the level of detail associated with the algorithmic description.

1156 A word of caution to evaluator is in order. The focus of this work unit should be to provide the evaluator an understanding of how the module works so that determinations can be made about the soundness of the implementation of the SFRs, as well as to support architectural analysis performed for ADV_ARC components. Given that infinite and perfect evaluations are unattainable and impractical, the evaluator must exercise judgement in performing this work unit (they should do so for other work units as well, but because of the time it will take to perform this work unit, it is especially critical). As long as the evaluator has a sound understanding of the module's operation, and its relationship to other modules and the TOE as a whole, the evaluator should consider the objective of the work achieved and not engage in a documentation exercise for the developer (by requiring, for example, a more complete algorithmic description that is present). Conversely, it is the evaluator's decision about what constitutes a sufficient algorithmic description, not the developer's. The requirement is that the developer must provide a full algorithmic description, and so the evaluator is not obligated to accept anything less, and is indeed not obligated to justify accepting anything less; the requirement provides the necessary justification.

1157 The algorithmic description is complete if it describes (in an algorithmic fashion) all of the functionality performed by the module. This description should detail how the module inputs are manipulated to produce the desired system effect, using the information provided about the parameters to the interface being invoked, as well as any calls made to other modules. The

evaluator should examine the other evidence presented about the module (global data use, interfaces, function) to ensure that the description appears to be complete.

1158 Interfaces used by each SFR-enforcing module are identified in a unique manner such that it can be determined by the evaluator which module is being invoked by the module being described.

1159 It must also be clear from the TOE design the algorithmic reason the invoking module is being called. For instance, if Module A is being described, and it uses Module B's bubble sort routine, an inadequate algorithmic description would be "Module A invokes the *double_bubble()* interface in Module B to perform a bubble sort." An adequate algorithmic description would be "Module A invokes the *double_bubble* routine with the list of access control entries; *double_bubble()* will return the entries sorted first on the username, then on the *access_allowed* field according the following rules..." The TOE design must provide enough detail so that it is clear what effects Module A is expecting from the bubble sort interface. Note that one method of presenting these called interfaces is via a call tree, and then the algorithmic description can be included in the algorithmic description of the module.

1160 Because the algorithmic description is at such a low level of detail, it may be difficult determine completeness and accuracy impacts from higher-level documentation, such as administrative guidance, the functional specification, the architectural design, or the TSF internal design documents. However, the evaluator uses the information present in those documents to help ensure that the algorithmic description is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.3.2D (work unit ADV_TDS.3-1), which maps the TSFI in the functional specification to the modules of the TSF.

1161 The evaluator should also examine the implementation representation provided for the modules described by the TOE design. While it is not expected that the evaluator necessarily examine the implementation representation for each SFR-enforcing module, the evaluator should sample the implementation representation related to modules that describe different portions of the security architecture (auditing, I&A, etc.) to help make the case that the TOE design is a complete and accurate description of the implemented algorithm.

ADV_TDS.4-15 The evaluator ***shall examine*** the TOE design to determine that the algorithmic detail for each SFR-enforcing and SFR-supporting module is representative of the actual TSF implementation.

1162 The focus of this work unit should be to provide the evaluator with confidence that the module descriptions for SFR-enforcing and SFR-supporting modules sufficiently describe the implementation representation, where sufficiently is determined by the evaluator. The same caution to evaluator (and caveats to developer) as were discussed for the previous work unit apply to this work unit; the evaluator must exercise judgement in the

degree of detail they need to be present in the algorithmic description to make a determination that it is sufficiently reflective of the underlying implementation. Requiring the developer to provide additional detail in the presentation of the algorithmic description of the module should be weighed by the evaluator in terms of the benefit provided by such additional information.

1163 The algorithmic description of a module describes in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or other presentation methods. It discusses how the inputs to the modules and operations obtained from called modules are used to accomplish the function of the module. It notes changes to global data, system state, and other outputs produced by the module. A description of a module in the TOE design should be of sufficient detail so that one could create an implementation of the module from the TOE design, and that implementation would be identical to the actual TSF implementation in terms of its inputs, outputs, and operations accomplished; and the internals of the module would be algorithmically identical. 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 they were part of the TSF).

1164 As previously indicated, the evaluator should sample the implementation representation for modules to ensure the description in the TOE design reflects the implementation in the appropriate amount of detail. It should be fairly easy for the evaluator to understand all portions of a module's implementation representation if the TOE design is written in sufficient detail.

ADV_TDS.4.10C ***The design shall describe each SFR-non-interfering module in terms of its purpose and interaction with other modules.***

ADV_TDS.4-16 The evaluator ***shall examine*** the TOE design to determine that SFR-non-interfering modules are correctly designated.

1165 As mentioned in work unit ADV_TDS.4-3, less information is required about modules that are SFR-non-interfering. A key focus of the evaluator for this work unit is attempting to determine from the evidence provided for each module implicitly categorised as SFR-non-interfering and the evaluation (information about other modules in the TOE design, the functional specification, the architectural design, the administrative guidance, the TSF internals document, and perhaps even the implementation representation) whether the module is indeed SFR-non-interfering. At this level of assurance some error should be tolerated; the evaluator does not

have to be absolutely sure that a given module is SFR-non-interfering, even though it is labelled as such. However, if the evidence provided indicates that a SFR-non-interfering module is SFR-enforcing or SFR-supporting, the evaluator requests additional information from the developer in order to resolve the apparent inconsistency. For example, suppose the documentation for Module A (an SFR-enforcing module) indicates that it calls Module B to perform an access check on a certain type of construct. When the evaluator examines the information associated with Module B, it is discovered that the only information the developer has provided is a purpose and a set of interactions (thus implicitly categorising Module B as non-SFR-enforcing). On examining the purpose and interactions from Module A, the evaluator finds no mention of Module B performing any access checks, and Module A is not listed as a module with which Module B interacts. At this point the evaluator should approach the developer to resolve the discrepancies between the information provided in Module A and that in Module B.

1166 Another example would be where the evaluator examines the mapping of the TSFI to the modules as provided by ADV_TDS.4.2D. This examination shows that Module C is associated with an SFR requiring identification of the user. Again, when the evaluator examines the information associated with Module C, they find that all the developer has provided is a purpose and a set of interactions (thus implicitly categorising Module C as SFR-non-interfering). Examining the purpose and interactions presented for Module C, the evaluator is unable to determine why Module C, listed as mapping to a TSFI concerned with user identification, would not be classified as SFR-enforcing or SFR-supporting. Again, the evaluator should approach the developer to resolve this discrepancy.

1167 A final example illustrates the opposite situation. As before, the developer has provided information associated with Module D consisting of a purpose and a set of interactions (thus implicitly categorising Module D as SFR-non-interfering). The evaluator examines all of the evidence provided, including the purpose and interactions for Module D. The purpose appears to give a meaningful description of Module D's function in the TOE, the interactions are consistent with that description, and there is nothing to indicate that Module D is SFR-enforcing or SFR-supporting. In this case, the evaluator should not demand more information about Module D “just be to sure” it is correctly categorised. The developer has met the obligations and the resulting assurance the evaluator has in the implicit categorisation of Module D is (by definition) appropriate for this assurance level.

ADV_TDS.4-17 The evaluator ***shall examine*** the TOE design to determine that the description of the purpose of each SFR-non-interfering module is complete and accurate.

1168 The purpose a module provides is a short description indicating what function the module is fulfilling. The evaluator should be able to obtain a general idea of what the module's function is in the architecture from the description. In order to assure the description is complete, the evaluator uses the information provided about the module's interactions with other modules to assess whether the reasons for the module being called are consistent with

the module's purpose. If the interaction description contains functionality that is not apparent from, or in conflict with, the module's purpose, the evaluator needs to determine whether the problem is one of accuracy or of completeness. The evaluator should be wary of purposes that are too short, since meaningful analysis based on a one-sentence purpose is likely impossible.

1169 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the architectural design, or the TSF internals document. However, the evaluator uses the information present in those documents to the extent possible to help ensure that the function is accurately and completely described. This analysis can be aided by the analysis performed for the implied evaluator action ADV_TDS.4.2D (work unit ADV_TDS.4-21), which maps the TSFI in the functional specification to the modules of the TSF.

ADV_TDS.4-18 The evaluator ***shall examine*** the TOE design to determine that the description of a SFR-non-interfering module's interaction with other modules is complete and accurate.

1170 It is important to note that, in terms of the Part 3 requirement and this work unit, the term *interaction* is intended to convey less rigour than *interface*. An interaction does not need to be characterised at the implementation level (e.g., parameters passed from one routine in a module to a routine in a different module; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular module that are going to be used by another module should be covered in this discussion. Any control relationships between modules (e.g., a module responsible for configuring a rule base for a firewall system and the module that actually implements these rules) should also be described.

1171 A module's interaction with other modules can be captured in many ways. The intent for the TOE design is to allow the evaluator to understand (in part through analysis of module interactions) the role of the non-SFR-enforcing modules in the overall TOE design. Understanding of this role will aid the evaluator in performing work unit ADV_TDS.4-6.

1172 A module's interaction with other modules goes beyond just a call-tree-type document. The interaction is described from a functional perspective of why a module interacts with other modules. The module's purpose describes what functions the module provides to other modules; the interactions should describe what the module depends on from other modules in order to accomplish this function.

1173 Because the modules are at such a low level, it may be difficult determine completeness and accuracy impacts from other documentation, such as administrative guidance, the functional specification, the architectural design, or the TSF internals document. However, the evaluator uses the

information present in those documents to the extent possible to help ensure that the interactions are accurately and completely described.

11.8.4.3 Action ADV_TDS.4.2E

ADV_TDS.4-19 The evaluator ***shall examine*** the TOE security functional requirements and the TOE design, to determine that all ST security functional requirements are covered by the TOE design.

1174 The evaluator may construct a map between the TOE security functional requirements and the TOE design. This map will likely be from a functional requirement to a set of components. Note that this map may have to be at a level of detail below the component or even element level of the requirements, because of operations (assignments, refinements, selections) performed on the functional requirement by the ST author.

1175 For example, the FDP_ACC.1 Access control component contains an element with assignments. If the ST contained, for instance, ten rules in the FDP_ACC.1 Access control assignment, and these ten rules were implemented in specific places within fifteen modules, it would be inadequate for the evaluator to map FDP_ACC.1 Access control to one component and claim the work unit had been completed. Instead, the evaluator would map FDP_ACC.1 Access control (rule 1) to Component A, behaviours x, y, and z; FDP_ACC.1 Access control (rule 2) to Component A, behaviours x, p, and q; etc.

ADV_TDS.4-20 The evaluator ***shall examine*** the TOE design to determine that it is an accurate instantiation of all security functional requirements.

1176 The evaluator ensures that each security requirement listed in the TOE security functional requirements section of the ST has a corresponding design description in the TOE design that accurately details how the TSF meets that requirement. This requires that the evaluator identify a collection of components that are responsible for implementing a given functional requirement, and then examine those components to understand how the requirement is implemented. Finally, the evaluator would assess whether the requirement was accurately implemented.

1177 As an example, if the ST requirements specified a role-based access control mechanism, the evaluator would first identify the components that contribute to this mechanism's implementation. This could be done by in-depth knowledge or understanding of the TOE design or by work done in the previous work unit. Note that this trace is only to identify the components, and is not the complete analysis.

1178 The next step would be to understand what mechanism the components implemented. For instance, if the design described an implementation of access control based on UNIX-style protection bits, the design would not be an accurate instantiation of those access control requirements present in the ST example used above. If the evaluator could not determine that the mechanism was accurately implemented because of a lack of detail, the

evaluator would have to assess whether all of the SFR-enforcing components have been identified, or if adequate detail had been provided for those components.

11.8.4.4 Implied evaluator action

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

ADV_TDS.4-21 The evaluator ***shall examine*** the TOE design to determine that it contains a complete and accurate mapping from the TSFI described in the functional specification to the modules of the TSF described in the TOE design.

1179 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. However, a complete call tree for each TSFI is not required for this work unit. The cases in which more than one module would have to be identified are where there are "entry point" modules or wrapper modules that have no functionality other than conditioning inputs or de-multiplexing an input. Mapping to one of these modules would not provide any useful information to the evaluator.

1180 The evaluator assesses the completeness of the mapping by ensuring that all of the TSFI map to at least one module. The verification of accuracy is more complex.

1181 The first aspect of accuracy is that each TSFI is mapped to a module at the TSF boundary. This determination can be made by reviewing the module description and its interfaces/interactions. The next aspect of accuracy is that each TSFI identifies a chain of modules between the initial module identified and a module that is primarily responsible for implementing the function presented at the TSF. Note that this may be the initial module, or there may be several modules, depending on how much pre-conditioning of the inputs is done. It should be noted that one indicator of a pre-conditioning module is that it is invoked for a large number of the TSFI, where the TSFI are all of similar type (e.g., system call). The final aspect of accuracy is that the mapping makes sense. For instance, mapping a TSFI dealing with access control to a module that checks passwords is not accurate. The evaluator should again use judgement in making this determination. The goal is that this information aids the evaluator in understanding the system and implementation of the SFRs, and ways in which entities at the TSF boundary can interact with the TSF. The bulk of the assessment of whether the SFRs are described accurately by the modules is performed in other work units.

11.8.5 Evaluation of sub-activity (ADV_TDS.5)

1182 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

11.8.6 Evaluation of sub-activity (ADV_TDS.6)

1183 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

12 Class AGD: Guidance documents

12.1 Introduction

1184 The purpose of the guidance document activity is to judge the adequacy of the documentation describing how the user can handle the TOE in a secure manner. Such documentation should take into account the various types of users (e.g. those who accept, install, administrate or operate the TOE) whose incorrect actions could adversely affect the security of the TOE or of their own data.

1185 The guidance documents class is subdivided into two families which are concerned with the preparative user guidance (all that what has to be done to transform the delivered TOE into its evaluated configuration in the environment as described in the ST, i.e. accepting and installing the TOE) respectively with the operational user guidance (all that what has to be done during the operation of the TOE in its evaluated configuration, i.e. operation and administration).

12.2 Application notes

1186 The guidance documents activity applies to those functions and interfaces which are related to the security of the TOE. The secure configuration of the TOE is described in the ST.

12.3 Operational user guidance (AGD_OPE)

12.3.1 Evaluation of sub-activity (AGD_OPE.1)

12.3.1.1 Objectives

1187 The objectives of this sub-activity are to determine whether the user guidance describes for each user role the security functionality and interfaces provided by the TSF, provides instructions and guidelines for the secure use of the TOE, addresses secure procedures for all modes of operation, facilitates prevention and detection of insecure TOE states, or whether it is misleading or unreasonable.

12.3.1.2 Input

1188 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the high-level design, if applicable;

d) the user guidance;

12.3.1.3 Action AGD_OPE.1.1E

AGD_OPE.1.1C ***The user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.***

AGD_OPE.1-1 The evaluator ***shall examine*** the user guidance to determine that it describes, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

1189 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.

1190 The user guidance identifies, for each user role, the functions and privileges that must be controlled, 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.

AGD_OPE.1.2C ***The user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.***

AGD_OPE.1-2 The evaluator ***shall examine*** the user guidance to determine that it describes, for each user role, the secure use of the available interfaces provided by the TOE.

1191 The user guidance should provide advice regarding effective use of the TSF (e.g. reviewing password composition practises, suggested frequency of user file backups, discussion on the effects of changing user access privileges).

AGD_OPE.1.3C ***The user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.***

AGD_OPE.1-3 The evaluator ***shall examine*** the user guidance to determine that it describes, for each user role, the available security functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

1192 The user guidance should contain an overview of the security functionality that is visible at the user interfaces.

1193 The user guidance should identify and describe the purpose, behaviour, and interrelationships of the security interfaces and functions.

1194 For each user-accessible interface, the user guidance should:

a) describe the method(s) by which the interface is invoked (e.g. command-line, programming-language system call, menu selection, command button);

b) 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.

c) describe the immediate TSF response, message, or code returned.

1195 The evaluator should consider the functional specification and the ST to determine that the TSF described in these documents is described in the guidance as required and completely to permit secure use through the TSFI available to all types of human users. The evaluator may, as an aid, prepare an informal mapping between the guidance and these documents. Any omissions in this mapping may indicate incompleteness.

AGD_OPE.1.4C ***The user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.***

AGD_OPE.1-4 The evaluator ***shall examine*** the user guidance to determine that it describes, for each user role, each type of security-relevant event relative to the user functions that need to be performed, including changing the security characteristics of entities under the control of the TSF and operation following failure or operational error.

1196 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 (e.g. audit trail overflow, system crash, updates to user records, such as when a user account is removed when the user leaves the organisation) are adequately defined to allow user intervention to maintain secure operation.

AGD_OPE.1.5C ***The user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.***

AGD_OPE.1-5 The evaluator ***shall examine*** the guidance and other evaluation evidence to determine that the guidance identifies all possible modes of operation of the TOE (including, if applicable, operation following failure or operational error), their consequences and implications for maintaining secure operation.

1197 Other evaluation evidence, particularly the functional specification, provide an information source that the evaluator should use to determine that the guidance contains sufficient guidance information.

1198 If test documentation is included in the assurance package, then the information provided in this evidence can also be used to determine that the guidance contains sufficient guidance documentation. The detail provided in the test steps can be used to confirm that the guidance provided is sufficient for the use and administration of the TOE.

1199 The evaluator should focus on a single human visible TSFI at a time, comparing the guidance for securely using the TSFI with other evaluation evidence, to determine that the guidance related to the TSFI is sufficient for the secure usage (i.e. consistent with the TSP) of that TSFI. The evaluator should also consider the relationships between interfaces, searching for potential conflicts.

AGD_OPE.1.6C ***The 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.***

AGD_OPE.1-6 The evaluator ***shall examine*** the user guidance to determine that it describes, for each user role, the security measures to be followed in order to fulfil the security objectives for the operational environment as described in the ST.

1200 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.

1201 The security measures described in the user guidance should include all relevant external procedural, physical, personnel and connectivity measures.

1202 Note that those measures relevant for secure installation of the TOE are examined in Preparative user guidance (AGD_PRE).

AGD_OPE.1.7C ***The user guidance shall be clear and reasonable.***

AGD_OPE.1-7 The evaluator ***shall examine*** the operational user guidance to determine that it is clear.

1203 The guidance is unclear if it can reasonably be misconstrued by an administrator or user, and used in a way detrimental to the TOE, or to the security provided by the TOE.

AGD_OPE.1-8 The evaluator ***shall examine*** the operational user guidance to determine that it is reasonable.

1204 The guidance is unreasonable if it makes demands on the TOE's usage or operational environment that are inconsistent with the ST or unduly onerous to maintain security.

12.4 Preparative user guidance (AGD_PRE)

12.4.1 Evaluation of sub-activity (AGD_PRE.1)

12.4.1.1 Objectives

1205 The objective of this sub-activity is to determine whether the procedures and steps for the secure preparation of the TOE have been documented and result in a secure configuration.

12.4.1.2 Input

1206 The evaluation evidence for this sub-activity is:

a) the ST;

b) the user guidance;

c) the description of developer's delivery procedures, if applicable;

d) the TOE suitable for testing.

12.4.1.3 Application notes

1207 The preparative procedures refer to all acceptance and installation procedures, that are necessary to progress the TOE to the secure configuration as described in the ST.

12.4.1.4 Action AGD_PRE.1.1E

AGD_PRE.1.1C ***The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with developer's delivery procedures.***

AGD_PRE.1-1 The evaluator ***shall check*** that the procedures necessary for the secure acceptance of the delivered TOE have been provided.

1208 If it is not anticipated by developer's delivery procedures that acceptance procedures will or can be applied, this work unit is not applicable, and is therefore considered to be satisfied.

AGD_PRE.1-2 The evaluator ***shall examine*** the provided acceptance procedures to determine that they describe the steps necessary for secure acceptance of the TOE in accordance with developer's delivery procedures.

1209 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.

1210 The acceptance procedures should reflect the steps the user has to perform in order to accept the delivered TOE that are implied by developer's delivery procedures.

1211 The acceptance procedures should provide detailed information about the following, if applicable:

a) making sure that the delivered TOE is the complete evaluated instance;

b) detecting modification/masquerading of the delivered TOE.

AGD_PRE.1.2C ***The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.***

AGD_PRE.1-3 The evaluator ***shall check*** that the procedures necessary for the secure installation of the TOE have been provided.

1212 If it is not anticipated that installation procedures will or can be applied for the TOE and the operational environment (e.g. because the TOE may already be delivered in an operational state and there are no requirements for the environment) this work unit is not applicable, and is therefore considered to be satisfied.

AGD_PRE.1-4 The evaluator ***shall examine*** the provided installation procedures to determine that they describe the steps necessary for secure installation of the TOE and the secure preparation of the operational environment in accordance with the security objectives in the ST.

1213 If it is not anticipated that installation procedures will or can be applied (e.g. because the TOE may already be delivered in an operational state), this work unit is not applicable, and is therefore considered to be satisfied.

1214 The installation procedures should provide detailed information about the following, if applicable:

a) minimum system requirements for secure installation;

b) requirements for the operational environment in accordance with the security objectives provided by the ST;

c) changing the installation specific security characteristics of entities under the control of the TSF;

d) handling exceptions and problems.

12.4.1.5 Action AGD_PRE.1.2E

AGD_PRE.1-5 The evaluator ***shall perform*** all user procedures necessary to prepare the TOE to determine that the TOE and its operational environment can be prepared securely using only the supplied preparative user guidance.

1215 Preparation requires the evaluator to advance the TOE from a deliverable state to the state in which it is operational, including acceptance and

installation of the TOE, and enforcing a TSP consistent with the security objectives for the TOE specified in the ST.

1216 The evaluator should follow only the developer's procedures and may perform the activities that customers are usually expected to perform to accept and install the TOE, using the supplied preparative guidance documentation only. Any difficulties encountered during such an exercise may be indicative of incomplete, unclear or unreasonable guidance.

1217 This work unit may be performed in conjunction with the evaluation activities under Independent testing (ATE_IND).

1218 If it is known that the TOE will be used as a dependent component for a composed TOE evaluation, then the evaluator should ensure that the operational environment is satisfied by the base component used in the composed TOE.

13 Class ALC: Life-cycle support

13.1 Introduction

1219 The purpose of the life-cycle support activity is to determine the adequacy of the security procedures the developer uses during the development and maintenance of the TOE. These procedures include the life-cycle model used by the developer, the configuration management, the security measures used throughout TOE development, the tools used by the developer throughout the life-cycle of the TOE, the handling of security flaws, and the delivery activity.

1220 Poorly controlled development and maintenance of the TOE can result in vulnerabilities in the implementation. Conformance to a defined life-cycle model can help to improve controls in this area. A standardised life-cycle model used for the TOE can avoid the use of proprietary possibly inexperienced methods. A measurable life-cycle model used for the TOE can remove ambiguity in assessing the development progress of the TOE.

1221 The purpose of the configuration management activity is to assist the consumer in identifying the evaluated TOE, to ensure that configuration items are uniquely identified, and the adequacy of the procedures that are used by the developer to control and track changes that are made to the TOE. This includes details on what changes are tracked, how potential changes are incorporated, and the degree to which automation is used to reduce the scope for error.

1222 Developer security procedures are intended to protect the TOE and its associated design information from interference or disclosure. Interference in the development process may allow the deliberate introduction of vulnerabilities. Disclosure of design information may allow vulnerabilities to be more easily exploited. The adequacy of the procedures will depend on the nature of the TOE and the development process.

1223 The use of well-defined development tools and the application of implementation standards by the developer and by third parties involved in the development process help to ensure that vulnerabilities are not inadvertently introduced during refinement.

1224 The flaw remediation activity is intended to track security flaws, to identify corrective actions, and to distribute the corrective action information to TOE users.

1225 The purpose of the delivery activity is to judge the adequacy of the documentation of the procedures used to ensure that the TOE is delivered to the consumer without modification.

13.2 CM capabilities (ALC_CMC)

13.2.1 Evaluation of sub-activity (ALC_CMC.1)

13.2.1.1 Objectives

1226 The objectives of this sub-activity are to determine whether the developer has clearly identified the TOE.

13.2.1.2 Input

1227 The evaluation evidence for this sub-activity is:

a) the ST;

b) the TOE suitable for testing.

13.2.1.3 Action ALC_CMC.1.1E

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

ALC_CMC.1-1 The evaluator ***shall check*** that the TOE provided for evaluation is labelled with its reference.

1228 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

1229 The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

1230 If this work unit is applied to a composed TOE, the following will apply. The unique reference provided for the composed TOE may be the combination of the unique reference of each component, or it may be an unrelated unique reference (unlikely except in the case where the composed TOE is delivered as an integrated product, rather than as individual components).

ALC_CMC.1-2 The evaluator ***shall check*** that the TOE references used are consistent.

1231 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.

1232 The evaluator also verifies that the TOE reference is consistent with the ST.

1233 If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but that only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumer will be able to determine whether they have the appropriate items.

13.2.2 Evaluation of sub-activity (ALC_CMC.2)

13.2.2.1 Objectives

1234 The objectives of this sub-activity are to determine whether the developer uses a CM system that uniquely identifies all configuration items.

13.2.2.2 Input

1235 The evaluation evidence for this sub-activity is:

a) the ST;

b) the TOE suitable for testing;

c) the configuration management documentation.

13.2.2.3 Application notes

1236 This component contains an implicit evaluator action to determine that the CM system is being used. As the requirements here are limited to identification of the TOE and provision of a configuration list, this action is already covered by, and limited to, the existing work units. At Evaluation of sub-activity (ALC_CMC.3) the requirements are expanded beyond these two items, and more explicit evidence of operation is required.

13.2.2.4 Action ALC_CMC.2.1E

ALC_CMC.2.1C ***The TOE shall be labelled with its reference.***

ALC_CMC.2-1 The evaluator ***shall check*** that the TOE provided for evaluation is labelled with its reference.

1237 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

1238 The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

1239 If this work unit is applied to a composed TOE, the following will apply. The unique reference provided for the composed TOE may be the combination of the unique reference of each component, or it may be an unrelated unique reference (unlikely expect in the case where the composed TOE is delivered as an integrated product, rather than as individual components).

ALC_CMC.2-2 The evaluator ***shall check*** that the TOE references used are consistent.

1240 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.

1241 The evaluator also verifies that the TOE reference is consistent with the ST.

1242 If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but that only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumer will be able to determine whether they have the appropriate items.

ALC_CMC.2.2C ***The CM documentation shall describe the method used to uniquely identify the configuration items.***

ALC_CMC.2-3 The evaluator ***shall examine*** the method of identifying configuration items to determine that it describes how configuration items are uniquely identified.

1243 Procedures should describe 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:

a) the method how each configuration item is uniquely identified, such that it is possible to track versions of the same configuration item;

b) the method how configuration items are assigned unique identifiers and how they are entered into the CM system;

c) the method to be used to identify superseded versions of a configuration item.

ALC_CMC.2.3C ***The CM system shall uniquely identify all configuration items.***

ALC_CMC.2-4 The evaluator ***shall examine*** the configuration items to determine that they are identified in a way that is consistent with the CM documentation.

1244 Assurance that the CM system uniquely identifies all configuration items is gained by examining the identifiers for the configuration items. For both configuration items that comprise the TOE, and drafts of configuration items that are submitted by the developer as evaluation evidence, the evaluator confirms that each configuration item possesses a unique identifier in a manner consistent with the unique identification method that is described in the CM documentation.

13.2.3 Evaluation of sub-activity (ALC_CMC.3)

13.2.3.1 Objectives

1245 The objectives of this sub-activity are to determine whether the developer uses a CM system that uniquely identifies all configuration items, and whether the ability to modify these items is properly controlled.

13.2.3.2 Input

1246 The evaluation evidence for this sub-activity is:

a) the ST;

b) the TOE suitable for testing;

c) the configuration management documentation.

13.2.3.3 Action ALC_CMC.3.1E

ALC_CMC.3.1C ***The TOE shall be labelled with its reference.***

ALC_CMC.3-1 The evaluator ***shall check*** that the TOE provided for evaluation is labelled with its reference.

1247 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

1248 The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during

the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

1249 If this work unit is applied to a composed TOE, the following will apply. The unique reference provided for the composed TOE may be the combination of the unique reference of each component, or it may be an unrelated unique reference (unlikely expect in the case where the composed TOE is delivered as an integrated product, rather than as individual components).

ALC_CMC.3-2 The evaluator ***shall check*** that the TOE references used are consistent.

1250 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.

1251 The evaluator also verifies that the TOE reference is consistent with the ST.

1252 If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but that only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumer will be able to determine whether they have the appropriate items.

ALC_CMC.3.2C ***The CM documentation shall describe the method used to uniquely identify the configuration items.***

ALC_CMC.3-3 The evaluator ***shall examine*** the method of identifying configuration items to determine that it describes how configuration items are uniquely identified.

1253 Procedures should describe 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:

a) the method how each configuration item is uniquely identified, such that it is possible to track versions of the same configuration item;

b) the method how configuration items are assigned unique identifiers and how they are entered into the CM system;

c) the method to be used to identify superseded versions of a configuration item.

ALC_CMC.3.3C ***The CM system shall uniquely identify all configuration items.***

ALC_CMC.3-4 The evaluator ***shall examine*** the configuration items to determine that they are identified in a way that is consistent with the CM documentation.

1254 Assurance that the CM system uniquely identifies all configuration items is gained by examining the identifiers for the configuration items. For both configuration items that comprise the TOE, and drafts of configuration items that are submitted by the developer as evaluation evidence, the evaluator confirms that each configuration item possesses a unique identifier in a manner consistent with the unique identification method that is described in the CM documentation.

ALC_CMC.3.4C ***The CM system shall provide measures such that only authorised changes are made to the configuration items.***

ALC_CMC.3-5 The evaluator ***shall examine*** the CM access control measures described in the CM plan to determine that they are effective in preventing unauthorised access to the configuration items.

1255 The evaluator may use a number of methods to determine that the CM access control measures are effective. For example, the evaluator may exercise the access control measures to ensure that the procedures could not be bypassed. The evaluator may use the outputs generated by the CM system procedures required by ALC_CMC.3.7C. The evaluator may also witness a demonstration of the CM system to ensure that the access control measures employed are operating effectively.

ALC_CMC.3.5C ***The CM documentation shall include a CM plan.***

ALC_CMC.3-6 The evaluator ***shall check*** that the CM documentation provided includes a CM plan. The CM plan needs not to be a connected document, but it is recommended that there is a single document that describes where the various parts of the CM plan can be found.

ALC_CMC.3.6C ***The CM plan shall describe how the CM system is used for the development of the TOE.***

ALC_CMC.3-7 The evaluator ***shall examine*** the CM plan to determine that it describes how the CM system is used to maintain the integrity of the TOE configuration items.

1256 The descriptions contained in a CM plan include, if applicable:

a) 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);

b) which means (e.g. CM tools, forms) have to be made available;

c) the usage of the CM tools: the necessary details for a user of the CM system to be able to operate the CM tools correctly in order to maintain the integrity of the TOE;

d) which other objects (development components, tools, assessment environments, ...) are taken under CM control;

e) the roles and responsibilities of individuals required to perform operations on individual configuration items (different roles may be identified for different types of configuration items (e.g. design documentation or source code));

f) how CM instances (e.g. change control boards, interface control working groups) are introduced and staffed;

g) the description of the change management;

h) the procedures that are used to ensure that only authorised individuals can make changes to configuration items;

i) the procedures that are used to ensure that concurrency problems do not occur as a result of simultaneous changes to configuration items;

j) the evidence that is generated as a result of application of the procedures. For example, for a change to a configuration item, the CM system might record a description of the change, accountability for the change, identification of all configuration items affected, status (e.g. pending or completed), and date and time of the change. This might be recorded in an audit trail of changes made or change control records;

k) the approach to version control and unique referencing of TOE versions (e.g. covering the release of patches in operating systems, and the subsequent detection of their application).

ALC_CMC.3.7C ***The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.***

ALC_CMC.3-8 The evaluator ***shall check*** the CM documentation to ascertain that it includes the CM system records identified by the CM plan.

1257 The output produced by the CM system should provide the evidence that the evaluator needs to be confident that the CM plan is being applied, and also that all configuration items are being maintained by the CM system as required by ALC_CMC.3.8C. Example output could include change control forms, or configuration item access approval forms.

ALC_CMC.3-9 The evaluator ***shall examine*** the evidence to determine that the CM system is being used as it is described in the CM plan.

1258 The evaluator should select and examine a sample of evidence covering each type of CM-relevant operation that has been performed on a configuration

item (e.g. creation, modification, deletion, reversion to an earlier version) to confirm that all operations of the CM system have been carried out in line with documented procedures. The evaluator confirms that the evidence includes all the information identified for that operation in the CM plan. Examination of the evidence may require access to a CM tool that is used. The evaluator may choose to sample the evidence.

1259 For guidance on sampling see A.2, Sampling.

1260 Further confidence in the correct operation of the CM system and the effective maintenance of configuration items may be established by means of interview with selected development staff. In conducting such interviews, the evaluator should aim to gain a deeper understanding of how the CM system is used in practise as well as to confirm that the CM procedures are being applied as described in the CM documentation. Note that such interviews should complement rather than replace the examination of documentary evidence, and may not be necessary if the documentary evidence alone satisfies the requirement. However, given the wide scope of the CM plan it is possible that some aspects (e.g. roles and responsibilities) may not be clear from the CM plan and records alone. This is one case where clarification may be necessary through interviews.

1261 It is expected that the evaluator will visit the development site in support of this activity.

1262 For guidance on site visits see A.4, Site Visits.

ALC_CMC.3.8C ***The evidence shall demonstrate that all configuration items have been and are being maintained under the CM system.***

ALC_CMC.3-10 The evaluator ***shall check*** that the configuration items identified in the configuration list are being maintained by the CM system.

1263 The CM system employed by the developer should maintain the integrity of the TOE. The evaluator should check that for each type of configuration item (e.g. high-level design or source code modules) contained in the configuration list there are examples of the evidence generated by the procedures described in the CM plan. In this case, the approach to sampling will depend upon the level of granularity used in the CM system to control CM items. Where, for example, 10,000 source code modules are identified in the configuration list, a different sampling strategy should be applied compared to the case in which there are only 5, or even 1. The emphasis of this activity should be on ensuring that the CM system is being operated correctly, rather than on the detection of any minor error.

1264 For guidance on sampling see A.2, Sampling.

13.2.4 Evaluation of sub-activity (ALC_CMC.4)

13.2.4.1 Objectives

1265 The objectives of this sub-activity are to determine whether the developer has clearly identified the TOE and its associated configuration items, and whether the ability to modify these items is properly controlled by automated tools, thus making the CM system less susceptible to human error or negligence.

13.2.4.2 Input

1266 The evaluation evidence for this sub-activity is:

a) the ST;

b) the TOE suitable for testing;

c) the configuration management documentation.

13.2.4.3 Action ALC_CMC.4.1E

ALC_CMC.4.1C ***The TOE shall be labelled with its reference.***

ALC_CMC.4-1 The evaluator ***shall check*** that the TOE provided for evaluation is labelled with its reference.

1267 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

1268 The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

1269 If this work unit is applied to a composed TOE, the following will apply. The unique reference provided for the composed TOE may be the combination of the unique reference of each component, or it may be an unrelated unique reference (unlikely expect in the case where the composed TOE is delivered as an integrated product, rather than as individual components).

ALC_CMC.4-2 The evaluator ***shall check*** that the TOE references used are consistent.

1270 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.

1271 The evaluator also verifies that the TOE reference is consistent with the ST.

1272 If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but that only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumer will be able to determine whether they have the appropriate items.

ALC_CMC.4.2C ***The CM documentation shall describe the method used to uniquely identify the configuration items.***

ALC_CMC.4-3 The evaluator ***shall examine*** the method of identifying configuration items to determine that it describes how configuration items are uniquely identified.

1273 Procedures should describe 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:

a) the method how each configuration item is uniquely identified, such that it is possible to track versions of the same configuration item;

b) the method how configuration items are assigned unique identifiers and how they are entered into the CM system;

c) the method to be used to identify superseded versions of a configuration item.

ALC_CMC.4.3C ***The CM system shall uniquely identify all configuration items.***

ALC_CMC.4-4 The evaluator ***shall examine*** the configuration items to determine that they are identified in a way that is consistent with the CM documentation.

1274 Assurance that the CM system uniquely identifies all configuration items is gained by examining the identifiers for the configuration items. For both configuration items that comprise the TOE, and drafts of configuration items that are submitted by the developer as evaluation evidence, the evaluator confirms that each configuration item possesses a unique identifier in a manner consistent with the unique identification method that is described in the CM documentation.

ALC_CMC.4.4C ***The CM system shall provide automated measures such that only authorised changes are made to the configuration items.***

ALC_CMC.4-5 The evaluator ***shall examine*** the CM access control measures described in the CM plan (cf. ALC_CMC.4.6C) to determine that they are automated and effective in preventing unauthorised access to the configuration items.

1275 The evaluator may use a number of methods to determine that the CM access control measures are effective. For example, the evaluator may exercise the access control measures to ensure that the procedures could not be bypassed. The evaluator may use the outputs generated by the CM system procedures required by ALC_CMC.4.9C. The evaluator may also witness a demonstration of the CM system to ensure that the access control measures employed are operating effectively.

ALC_CMC.4.5C ***The CM system shall support the production of the TOE by automated means.***

ALC_CMC.4-6 The evaluator ***shall check*** the CM plan (cf. ALC_CMC.4.6C) for automated procedures for supporting the production of the TOE.

1276 The term “production” applies to those processes adopted by the developer to progress the TOE from the implementation representation to a state acceptable for delivery to the end customer.

1277 The evaluator verifies the existence of automated production support procedures within the CM plan.

1278 In the case of a software TOE, a make tool is an example for an automated means supporting the production.

ALC_CMC.4-7 The evaluator ***shall examine*** the TOE production support procedures to determine that they are effective in ensuring that a TOE is generated that reflects its implementation representation.

1279 The production support procedures should describe which tools have to be used to produce the final TOE from the implementation representation in a clearly defined way. The conventions, directives, or other necessary constructs are described under ALC_TAT.

1280 The evaluator determines that by following the production support procedures the correct configuration items would be used to generate the TOE. For example, in a software TOE this may include checking that the automated production procedures ensure that all source files and related libraries are included in the compiled object code. Moreover, the procedures should ensure that compiler options and comparable other options are defined uniquely.

1281 The customer can then be confident that the version of the TOE delivered for installation is derived from the implementation representation in an unambiguous way and implements the SFRs as described in the ST.

1282 The evaluator should bear in mind that the CM system need not necessarily possess the capability to produce the TOE, but should provide support for the process that will help reduce the probability of human error.

ALC_CMC.4.6C ***The CM documentation shall include a CM plan.***

ALC_CMC.4-8 The evaluator ***shall check*** that the CM documentation provided includes a CM plan. The CM plan needs not to be a connected document, but it is recommended that there is a single document that describes where the various parts of the CM plan can be found.

ALC_CMC.4.7C ***The CM plan shall describe how the CM system is used for the development of the TOE.***

ALC_CMC.4-9 The evaluator ***shall examine*** the CM plan to determine that it describes how the CM system is used to maintain the integrity of the TOE configuration items.

1283 The descriptions contained in a CM plan include, if applicable:

a) 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);

b) which means (e.g. CM tools, forms) have to be made available;

c) the usage of the CM tools: the necessary details for a user of the CM system to be able to operate the CM tools correctly in order to maintain the integrity of the TOE;

d) the production support procedures;

e) which other objects (development components, tools, assessment environments, ...) are taken under CM control;

f) the roles and responsibilities of individuals required to perform operations on individual configuration items (different roles may be identified for different types of configuration items (e.g. design documentation or source code));

g) how CM instances (e.g. change control boards, interface control working groups) are introduced and staffed;

h) the description of the change management;

i) the procedures that are used to ensure that only authorised individuals can make changes to configuration items;

j) the procedures that are used to ensure that concurrency problems do not occur as a result of simultaneous changes to configuration items;

k) the evidence that is generated as a result of application of the procedures. For example, for a change to a configuration item, the CM system might record a description of the change, accountability for the change, identification of all configuration items affected, status (e.g. pending or completed), and date and time of the change. This might be recorded in an audit trail of changes made or change control records;

l) the approach to version control and unique referencing of TOE versions (e.g. covering the release of patches in operating systems, and the subsequent detection of their application).

ALC_CMC.4.8C ***The CM plan shall describe the acceptance procedures.***

ALC_CMC.4-10 The evaluator ***shall examine*** the CM plan to determine that it includes the procedures used to accept modified or newly created configuration items as parts of the TOE.

1284 The descriptions of the acceptance procedures in the CM plan should include the developer roles or individuals responsible for the acceptance and the criteria to be used for acceptance. They should take into account all acceptance situations that may occur, in particular:

a) 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”);

b) moving configuration items to the next life-cycle phase at each stage of the construction of the TOE (e.g. module, subsystem, system);

c) subsequent to transports between different development sites.

1285 If this work unit is applied to a dependent component that is going to be integrated in a composed TOE, the CM plan should consider the control of base components obtained by the dependent TOE developer.

1286 When obtaining the components the evaluators are to verify the following:

a) Transfer of each base component from the base component developer to the integrator (dependent TOE developer) was performed in accordance with the base component TOE's secure delivery procedures, as reported in the base component TOE certification report.

b) The component received has the same identifiers as those stated in the ST and Certification Report for the component TOE.

c) All additional material required by a developer for composition (integration) is provided. This is to include the necessary extract of the component TOE's functional specification.

ALC_CMC.4.9C ***The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.***

ALC_CMC.4-11 The evaluator ***shall check*** the CM documentation to ascertain that it includes the CM system records identified by the CM plan.

1287 The output produced by the CM system should provide the evidence that the evaluator needs to be confident that the CM plan is being applied, and also that all configuration items are being maintained by the CM system as required by ALC_CMC.4.10C. Example output could include change control forms, or configuration item access approval forms.

ALC_CMC.4-12 The evaluator ***shall examine*** the evidence to determine that the CM system is being used as it is described in the CM plan.

1288 The evaluator should select and examine a sample of evidence covering each type of CM-relevant operation that has been performed on a configuration item (e.g. creation, modification, deletion, reversion to an earlier version) to confirm that all operations of the CM system have been carried out in line with documented procedures. The evaluator confirms that the evidence includes all the information identified for that operation in the CM plan. Examination of the evidence may require access to a CM tool that is used. The evaluator may choose to sample the evidence.

1289 For guidance on sampling see A.2, Sampling.

1290 Further confidence in the correct operation of the CM system and the effective maintenance of configuration items may be established by means of interview with selected development staff. In conducting such interviews, the evaluator should aim to gain a deeper understanding of how the CM system is used in practise as well as to confirm that the CM procedures are being applied as described in the CM documentation. Note that such interviews should complement rather than replace the examination of documentary evidence, and may not be necessary if the documentary evidence alone satisfies the requirement. However, given the wide scope of the CM plan it is possible that some aspects (e.g. roles and responsibilities) may not be clear from the CM plan and records alone. This is one case where clarification may be necessary through interviews.

1291 It is expected that the evaluator will visit the development site in support of this activity.

1292 For guidance on site visits see A.4, Site Visits.

ALC_CMC.4.10C ***The evidence shall demonstrate that all configuration items have been and are being maintained under the CM system.***

ALC_CMC.4-13 The evaluator ***shall check*** that the configuration items identified in the configuration list are being maintained by the CM system.

1293 The CM system employed by the developer should maintain the integrity of the TOE. The evaluator should check that for each type of configuration item (e.g. high-level design or source code modules) contained in the configuration list there are examples of the evidence generated by the procedures described in the CM plan. In this case, the approach to sampling will depend upon the level of granularity used in the CM system to control CM items. Where, for example, 10,000 source code modules are identified in the configuration list, a different sampling strategy should be applied compared to the case in which there are only 5, or even 1. The emphasis of this activity should be on ensuring that the CM system is being operated correctly, rather than on the detection of any minor error.

1294 For guidance on sampling see A.2, Sampling.

13.2.5 Evaluation of sub-activity (ALC_CMC.5)

13.2.5.1 Objectives

1295 The objectives of this sub-activity are to determine whether the developer has clearly identified the TOE and its associated configuration items, and whether the ability to modify these items is properly controlled by automated tools, thus making the CM system less susceptible to human error or negligence.

13.2.5.2 Input

1296 The evaluation evidence for this sub-activity is:

a) the ST;

b) the TOE suitable for testing;

c) the configuration management documentation.

13.2.5.3 Action ALC_CMC.5.1E

ALC_CMC.5.1C ***The TOE shall be labelled with its reference.***

ALC_CMC.5-1 The evaluator ***shall check*** that the TOE provided for evaluation is labelled with its reference.

1297 The evaluator should ensure that the TOE contains the unique reference which is stated in the ST. This could be achieved through labelled packaging or media, or by a label displayed by the operational TOE. This is to ensure that it would be possible for consumers to identify the TOE (e.g. at the point of purchase or use).

1298 The TOE may provide a method by which it can be easily identified. For example, a software TOE may display its name and version number during the start up routine, or in response to a command line entry. A hardware or firmware TOE may be identified by a part number physically stamped on the TOE.

1299 If this work unit is applied to a composed TOE, the following will apply. The unique reference provided for the composed TOE may be the combination of the unique reference of each component, or it may be an unrelated unique reference (unlikely expect in the case where the composed TOE is delivered as an integrated product, rather than as individual components).

ALC_CMC.5-2 The evaluator ***shall check*** that the TOE references used are consistent.

1300 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.

1301 The evaluator also verifies that the TOE reference is consistent with the ST.

1302 If this work unit is applied to a composed TOE, the following will apply. The composed IT TOE will not be labelled with its unique (composite) reference, but that only the individual components will be labelled with their appropriate TOE reference. It would require further development for the IT TOE to be labelled, i.e. during start-up and/or operation, with the composite reference. If the composed TOE is delivered as the constituent component TOEs, then the TOE items delivered will not contain the composite reference. However, the composed TOE ST will include the unique reference for the composed TOE and will identify the components comprising the composed TOE through which the consumer will be able to determine whether they have the appropriate items.

ALC_CMC.5.2C ***The CM documentation shall describe the method used to uniquely identify the configuration items.***

ALC_CMC.5-3 The evaluator ***shall examine*** the method of identifying configuration items to determine that it describes how configuration items are uniquely identified.

1303 Procedures should describe 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:

a) the method how each configuration item is uniquely identified, such that it is possible to track versions of the same configuration item;

b) the method how configuration items are assigned unique identifiers and how they are entered into the CM system;

c) the method to be used to identify superseded versions of a configuration item.

ALC_CMC.5.3C ***The CM documentation shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to all configuration items.***

ALC_CMC.5-4 The evaluator ***shall examine*** the CM documentation to determine that it justifies that the acceptance procedures provide for an adequate and appropriate review of changes to all configuration items.

1304 The CM documentation should make it sufficiently clear that by following the acceptance procedures only parts of adequate quality are incorporated into the TOE.

ALC_CMC.5.4C ***The CM system shall uniquely identify all configuration items.***

ALC_CMC.5-5 The evaluator ***shall examine*** the configuration items to determine that they are identified in a way that is consistent with the CM documentation.

1305 Assurance that the CM system uniquely identifies all configuration items is gained by examining the identifiers for the configuration items. For both configuration items that comprise the TOE, and drafts of configuration items that are submitted by the developer as evaluation evidence, the evaluator confirms that each configuration item possesses a unique identifier in a manner consistent with the unique identification method that is described in the CM documentation.

ALC_CMC.5.5C ***The CM system shall provide automated measures such that only authorised changes are made to the configuration items.***

ALC_CMC.5-6 The evaluator ***shall examine*** the CM access control measures described in the CM plan (cf. ALC_CMC.5.12C) to determine that they are automated and effective in preventing unauthorised access to the configuration items.

1306 The evaluator may use a number of methods to determine that the CM access control measures are effective. For example, the evaluator may exercise the access control measures to ensure that the procedures could not be bypassed. The evaluator may use the outputs generated by the CM system procedures required by ALC_CMC.5.15C. The evaluator may also witness a demonstration of the CM system to ensure that the access control measures employed are operating effectively.

ALC_CMC.5.6C ***The CM system shall support the production of the TOE by automated means.***

ALC_CMC.5-7 The evaluator ***shall check*** the CM plan (cf. ALC_CMC.5.12C) for automated procedures for supporting the production of the TOE.

1307 The term “production” applies to those processes adopted by the developer to progress the TOE from the implementation representation to a state acceptable for delivery to the end customer.

1308 The evaluator verifies the existence of automated production support procedures within the CM plan.

1309 In the case of a software TOE, a make tool is an example for an automated means supporting the production.

ALC_CMC.5-8 The evaluator ***shall examine*** the TOE production support procedures to determine that they are effective in ensuring that a TOE is generated that reflects its implementation representation.

1310 The production support procedures should describe which tools have to be used to produce the final TOE from the implementation representation in a clearly defined way. The conventions, directives, or other necessary constructs are described under ALC_TAT.

1311 The evaluator determines that by following the production support procedures the correct configuration items would be used to generate the TOE. For example, in a software TOE this may include checking that the automated production procedures ensure that all source files and related libraries are included in the compiled object code. Moreover, the procedures should ensure that compiler options and comparable other options are defined uniquely.

1312 The customer can then be confident that the version of the TOE delivered for installation is derived from the implementation representation in an unambiguous way and implements the SFRs as described in the ST.

1313 The evaluator should bear in mind that the CM system need not necessarily possess the capability to produce the TOE, but should provide support for the process that will help reduce the probability of human error.

ALC_CMC.5.7C ***The CM system shall ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.***

ALC_CMC.5-9 The evaluator ***shall examine*** the CM system to determine that it ensures that the person responsible for accepting a configuration item is not the person who developed it.

1314 The acceptance procedures describe who is responsible for accepting a configuration item. From these descriptions, the evaluator should be able to determine that the person who developed a configuration item is in no case responsible for its acceptance.

ALC_CMC.5.8C ***The CM system shall clearly identify the configuration items that comprise the TSF.***

ALC_CMC.5-10 The evaluator ***shall examine*** the CM system to determine that it clearly identifies the configuration items that comprise the TSF.

1315 The CM documentation should describe how the CM system identifies the configuration items that comprise the TSF. The evaluator should select a sample of configuration items covering each type of items, particularly containing TSF and non-TSF items, and check that they are correctly classified by the CM system.

1316 For guidance on sampling see A.2, Sampling.

ALC_CMC.5.9C ***The CM system shall support the audit of all changes to the TOE by automated means, including as a minimum the originator, date, and time in the audit trail.***

ALC_CMC.5-11 The evaluator ***shall examine*** the CM system to determine that it supports the audit of all changes to the TOE by automated means, including as a minimum the originator, date, and time in the audit trail.

1317 The evaluator should inspect a sample of audit trails and check, if they contain the minimum information.

ALC_CMC.5.10C ***The CM system shall provide an automated means to identify all other configuration items that are affected by the change of a given configuration item.***

ALC_CMC.5-12 The evaluator ***shall examine*** the CM system to determine that it provides an automated means to identify all other configuration items that are affected by the change of a given configuration item.

1318 The CM documentation should describe how the CM system identifies all other configuration items that are affected by the change of a given configuration item. The evaluator should select a sample of configuration items, covering all types of items, and exercise the automated means to determine that it identifies all items that are affected by the change of the selected item.

1319 For guidance on sampling see A.2, Sampling.

ALC_CMC.5.11C ***The CM system shall be able to identify the version of the implementation representation from which the TOE is generated.***

ALC_CMC.5-13 The evaluator ***shall examine*** the CM system to determine that it is able to identify the version of the implementation representation from which the TOE is generated.

1320 The CM documentation should describe how the CM system identifies the version of the implementation representation from which the TOE is generated. The evaluator should select a sample of the parts used to produce the TOE and should apply the CM system to verify that it identifies the corresponding implementation representations in the correct version.

1321 For guidance on sampling see A.2, Sampling.

ALC_CMC.5.12C ***The CM documentation shall include a CM plan.***

ALC_CMC.5-14 The evaluator ***shall check*** that the CM documentation provided includes a CM plan. The CM plan needs not to be a connected document, but it is recommended that there is a single document that describes where the various parts of the CM plan can be found.

ALC_CMC.5.13C ***The CM plan shall describe how the CM system is used for the development of the TOE.***

ALC_CMC.5-15 The evaluator ***shall examine*** the CM plan to determine that it describes how the CM system is used to maintain the integrity of the TOE configuration items.

1322 The descriptions contained in a CM plan include, if applicable:

a) 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);

b) which means (e.g. CM tools, forms) have to be made available;

c) the usage of the CM tools: the necessary details for a user of the CM system to be able to operate the CM tools correctly in order to maintain the integrity of the TOE;

d) the production support procedures;

e) which other objects (development components, tools, assessment environments, ...) are taken under CM control;

f) the roles and responsibilities of individuals required to perform operations on individual configuration items (different roles may be identified for different types of configuration items (e.g. design documentation or source code));

g) how CM instances (e.g. change control boards, interface control working groups) are introduced and staffed;

h) the description of the change management;

i) the procedures that are used to ensure that only authorised individuals can make changes to configuration items;

j) the procedures that are used to ensure that concurrency problems do not occur as a result of simultaneous changes to configuration items;

k) the evidence that is generated as a result of application of the procedures. For example, for a change to a configuration item, the CM system might record a description of the change, accountability for the change, identification of all configuration items affected, status (e.g. pending or completed), and date and time of the change. This might be recorded in an audit trail of changes made or change control records;

l) the approach to version control and unique referencing of TOE versions (e.g. covering the release of patches in operating systems, and the subsequent detection of their application).

ALC_CMC.5.14C ***The CM plan shall describe the acceptance procedures.***

ALC_CMC.5-16 The evaluator ***shall examine*** the CM plan to determine that it includes the procedures used to accept modified or newly created configuration items as parts of the TOE.

1323 The descriptions of the acceptance procedures in the CM plan should include the developer roles or individuals responsible for the acceptance and the criteria to be used for acceptance. They should take into account all acceptance situations that may occur, in particular:

a) 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");

b) moving configuration items to the next life-cycle phase at each stage of the construction of the TOE (e.g. module, subsystem, system);

c) subsequent to transports between different development sites.

ALC_CMC.5.15C ***The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.***

ALC_CMC.5-17 The evaluator ***shall check*** the CM documentation to ascertain that it includes the CM system records identified by the CM plan.

1324 The output produced by the CM system should provide the evidence that the evaluator needs to be confident that the CM plan is being applied, and also that all configuration items are being maintained by the CM system as required by ALC_CMC.5.16C. Example output could include change control forms, or configuration item access approval forms.

ALC_CMC.5-18 The evaluator ***shall examine*** the evidence to determine that the CM system is being used as it is described in the CM plan.

1325 The evaluator should select and examine a sample of evidence covering each type of CM-relevant operation that has been performed on a configuration item (e.g. creation, modification, deletion, reversion to an earlier version) to confirm that all operations of the CM system have been carried out in line with documented procedures. The evaluator confirms that the evidence includes all the information identified for that operation in the CM plan. Examination of the evidence may require access to a CM tool that is used. The evaluator may choose to sample the evidence.

1326 For guidance on sampling see A.2, Sampling.

1327 Further confidence in the correct operation of the CM system and the effective maintenance of configuration items may be established by means of interview with selected development staff. In conducting such interviews, the evaluator should aim to gain a deeper understanding of how the CM system is used in practise as well as to confirm that the CM procedures are being applied as described in the CM documentation. Note that such interviews

should complement rather than replace the examination of documentary evidence, and may not be necessary if the documentary evidence alone satisfies the requirement. However, given the wide scope of the CM plan it is possible that some aspects (e.g. roles and responsibilities) may not be clear from the CM plan and records alone. This is one case where clarification may be necessary through interviews.

1328 It is expected that the evaluator will visit the development site in support of this activity.

1329 For guidance on site visits see A.4, Site Visits.

ALC_CMC.5.16C ***The evidence shall demonstrate that all configuration items have been and are being maintained under the CM system.***

ALC_CMC.5-19 The evaluator ***shall check*** that the configuration items identified in the configuration list are being maintained by the CM system.

1330 The CM system employed by the developer should maintain the integrity of the TOE. The evaluator should check that for each type of configuration item (e.g. high-level design or source code modules) contained in the configuration list there are examples of the evidence generated by the procedures described in the CM plan. In this case, the approach to sampling will depend upon the level of granularity used in the CM system to control CM items. Where, for example, 10,000 source code modules are identified in the configuration list, a different sampling strategy should be applied compared to the case in which there are only 5, or even 1. The emphasis of this activity should be on ensuring that the CM system is being operated correctly, rather than on the detection of any minor error.

1331 For guidance on sampling see A.2, Sampling.

13.2.5.4 Action ALC_CMC.5.2E

ALC_CMC.5-20 The evaluator ***shall examine*** the production support procedures to determine that by following these procedures a TOE would be produced like that one provided by the developer for testing activities.

1332 If the TOE is a small software TOE and production consists of compiling and linking, the evaluator might confirm the adequacy of the production support procedures by reapplying them himself.

1333 If the production process of the TOE is more complicated (as for example in the case of a smartcard), but has already started, the evaluator should inspect the application of the production support procedures during a visit of the development site. He might compare a copy of the TOE produced in his presence with the samples used for his testing activities.

1334 For guidance on site visits see A.4, Site Visits.

1335 Otherwise the evaluator's determination should be based on the documentary evidence provided by the developer.

1336 This work unit may be performed in conjunction with the evaluation activities under ADV_IMP.

13.3 CM scope (ALC_CMS)

13.3.1 Evaluation of sub-activity (ALC_CMS.1)

13.3.1.1 Objectives

1337 The objective of this sub-activity is to determine whether the developer performs configuration management on the TOE and the evaluation evidence.

13.3.1.2 Input

1338 The evaluation evidence for this sub-activity is:

a) the ST;

b) the configuration list.

13.3.1.3 Action ALC_CMS.1.1E

ALC_CMS.1.1C ***The configuration list includes the following: the TOE itself; and the evaluation evidence required by the SARs in the ST.***

ALC_CMS.1-1 The evaluator ***shall check*** that the configuration list includes the following set of items:

a) the TOE itself;

b) the evaluation evidence required by the SARs in the ST.

ALC_CMS.1.2C ***The configuration list shall uniquely identify the configuration items.***

ALC_CMS.1-2 The evaluator ***shall examine*** the configuration list to determine that it uniquely identifies each configuration item.

1339 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

13.3.2 Evaluation of sub-activity (ALC_CMS.2)

13.3.2.1 Objectives

1340 The objective of this sub-activity is to determine whether the configuration list includes the TOE, the parts that comprise the TOE, and the evaluation evidence.

13.3.2.2 Input

1341 The evaluation evidence for this sub-activity is:

a) the ST;

b) the configuration list.

13.3.2.3 Action ALC_CMS.2.1E

ALC_CMS.2.1C ***The configuration list includes the following: the TOE itself; the evaluation evidence required by the SARs in the ST; and the parts that comprise the TOE.***

ALC_CMS.2-1 The evaluator ***shall check*** that the configuration item list includes the set of items required by the CC.

1342 The list includes at least the following:

a) the TOE itself;

b) the parts that comprise the TOE;

c) the evaluation evidence required by the SARs in the ST.

ALC_CMS.2.2C ***The configuration list shall uniquely identify the configuration items.***

ALC_CMS.2-2 The evaluator ***shall examine*** the configuration list to determine that it uniquely identifies each configuration item.

1343 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

ALC_CMS.2.3C ***For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.***

ALC_CMS.2-3 The evaluator ***shall check*** that the configuration list indicates the developer of each TSF relevant configuration item.

1344 If only one developer is involved in the development of the TOE, this work unit is not applicable, and is therefore considered to be satisfied.

13.3.3 Evaluation of sub-activity (ALC_CMS.3)

13.3.3.1 Objectives

1345 The objective of this sub-activity is to determine whether the configuration list includes the TOE, the parts that comprise the TOE, the TOE implementation representation, and the evaluation evidence.

13.3.3.2 Input

1346 The evaluation evidence for this sub-activity is:

a) the ST;

b) the configuration list.

13.3.3.3 Action ALC_CMS.3.1E

ALC_CMS.3.1C ***The configuration list includes the following: the TOE itself; the evaluation evidence required by the SARs in the ST; the parts that comprise the TOE; and the implementation representation.***

ALC_CMS.3-1 The evaluator ***shall check*** that the configuration list includes the following set of items:

a) the TOE itself;

b) the parts that comprise the TOE;

c) the TOE implementation representation;

d) the evaluation evidence required by the SARs in the ST.

ALC_CMS.3.2C ***The configuration list shall uniquely identify the configuration items.***

ALC_CMS.3-2 The evaluator ***shall examine*** the configuration list to determine that it uniquely identifies each configuration item.

1347 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

ALC_CMS.3.3C ***For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.***

ALC_CMS.3-3 The evaluator ***shall check*** that the configuration list indicates the developer of each TSF relevant configuration item.

1348 If only one developer is involved in the development of the TOE, this work unit is not applicable, and is therefore considered to be satisfied.

13.3.4 Evaluation of sub-activity (ALC_CMS.4)

13.3.4.1 Objectives

1349 The objective of this sub-activity is to determine whether the configuration list includes the TOE, the parts that comprise the TOE, the TOE implementation representation, security flaws, and the evaluation evidence.

13.3.4.2 Input

1350 The evaluation evidence for this sub-activity is:

a) the ST;

b) the configuration list.

13.3.4.3 Action ALC_CMS.4.1E

ALC_CMS.4.1C ***The configuration list includes the following: the TOE itself; the evaluation evidence required by the SARs in the ST; the parts that comprise the TOE; the implementation representation; and security flaws.***

ALC_CMS.4-1 The evaluator ***shall check*** that the configuration list includes the following set of items:

a) the TOE itself;

b) the parts that comprise the TOE;

c) the TOE implementation representation;

d) the evaluation evidence required by the SARs in the ST;

e) the documentation used to record details of reported security flaws associated with the implementation (e.g., problem status reports derived from a developer's problem database).

ALC_CMS.4.2C ***The configuration list shall uniquely identify the configuration items.***

ALC_CMS.4-2 The evaluator ***shall examine*** the configuration list to determine that it uniquely identifies each configuration item.

1351 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

ALC_CMS.4.3C ***For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.***

ALC_CMS.4-3 The evaluator ***shall check*** that the configuration list indicates the developer of each TSF relevant configuration item.

1352 If only one developer is involved in the development of the TOE, this work unit is not applicable, and is therefore considered to be satisfied.

13.3.5 Evaluation of sub-activity (ALC_CMS.5)

13.3.5.1 Objectives

1353 The objective of this sub-activity is to determine whether the configuration list includes the TOE, the parts that comprise the TOE, the TOE

implementation representation, security flaws, development tools and related information, and the evaluation evidence.

13.3.5.2 Input

1354 The evaluation evidence for this sub-activity is:

a) the ST;

b) the configuration list.

13.3.5.3 Action ALC_CMS.5.1E

ALC_CMS.5.1C ***The configuration list includes the following: the TOE itself; the evaluation evidence required by the SARs in the ST; the parts that comprise the TOE; the implementation representation; security flaws; and development tools and related information.***

ALC_CMS.5-1 The evaluator ***shall check*** that the configuration list includes the following set of items:

a) the TOE itself;

b) the parts that comprise the TOE;

c) the TOE implementation representation;

d) the evaluation evidence required by the SARs in the ST;

e) the documentation used to record details of reported security flaws associated with the implementation (e.g., problem status reports derived from a developer's problem database);

f) all tools (incl. test software, if applicable) involved in the development and production of the TOE including the names, versions, configurations and roles of each development tool, and related documentation. E.g. for a software TOE: “development tools” are usually programming languages and compiler and “related documentation” comprises compiler and linker options.

ALC_CMS.5.2C ***The configuration list shall uniquely identify the configuration items.***

ALC_CMS.5-2 The evaluator ***shall examine*** the configuration list to determine that it uniquely identifies each configuration item.

1355 The configuration list contains sufficient information to uniquely identify which version of each item has been used (typically a version number). Use of this list will enable the evaluator to check that the correct configuration items, and the correct version of each item, have been used during the evaluation.

ALC_CMS.5.3C ***For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.***

ALC_CMS.5-3 The evaluator ***shall check*** that the configuration list indicates the developer of each TSF relevant configuration item.

1356 If only one developer is involved in the development of the TOE, this work unit is not applicable, and is therefore considered to be satisfied.

13.4 Delivery (ALC_DEL)

13.4.1 Evaluation of sub-activity (ALC_DEL.1)

13.4.1.1 Objectives

1357 The objective of this sub-activity is to determine whether the delivery documentation describes all procedures used to maintain security of the TOE when distributing the TOE to the user.

13.4.1.2 Input

1358 The evaluation evidence for this sub-activity is:

a) the ST;

b) the delivery documentation.

13.4.1.3 Action ALC_DEL.1.1E

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

ALC_DEL.1-1 The evaluator ***shall examine*** the delivery documentation to determine that it describes all procedures that are necessary to maintain security when distributing versions of the TOE or parts of it to the user.

1359 The delivery documentation describes proper procedures to maintain security of the TOE during transfer of the TOE or its component parts and to determine the identification of the TOE.

1360 The delivery documentation should cover the entire TOE, but may contain different procedures for different parts of the TOE. The evaluation should consider the totality of procedures.

1361 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). 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.

1362 Interpretation of the term “necessary to maintain security” will need to consider the nature of the TOE (e.g. whether it is software or hardware), the overall security level stated for the TOE, and the security objectives provided by the ST.

1363 The level of protection provided should be commensurate with the chosen component 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.

1364 The security aspects (integrity, confidentiality, availability) relevant for the actual TOE should be derived from the security objectives defined in the ST. The emphasis in the delivery documentation is likely to be on measures related to integrity, as integrity of the TOE is always important. However, confidentiality and availability of the delivery will be of concern in the delivery of some TOEs; procedures relating to these aspects of the secure delivery should also be discussed in the procedures.

1365 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.

13.4.1.4 Implied evaluator action

ALC_DEL.1.2D ***The developer shall use the delivery procedures.***

ALC_DEL.1-2 The evaluator ***shall examine*** aspects of the delivery process to determine that the delivery procedures are used.

1366 The approach taken by the evaluator to check the application of delivery procedures will depend on the nature of the TOE, and the delivery process itself. In addition to examination of the procedures themselves, the evaluator seeks some assurance that they are applied in practise. Some possible approaches are:

a) a visit to the distribution site(s) where practical application of the procedures may be observed;

b) examination of the TOE at some stage during delivery, or after the user has received it (e.g. checking for tamper proof seals);

c) observing that the process is applied in practise when the evaluator obtains the TOE through regular channels;

d) questioning end users as to how the TOE was delivered.

1367 For guidance on site visits see A.4, Site Visits.

1368 It may be the case of a newly developed TOE that the delivery procedures have yet to be exercised. In these cases, the evaluator has to be satisfied that appropriate procedures and facilities are in place for future deliveries and that all personnel involved are aware of their responsibilities. The evaluator may request a “dry run” of a delivery if this is practical. If the developer has produced other similar products, then an examination of procedures in their use may be useful in providing assurance.

13.5 Development security (ALC_DVS)

13.5.1 Evaluation of sub-activity (ALC_DVS.1)

13.5.1.1 Objectives

1369 The objective of this sub-activity is to determine whether the developer's security controls on the development environment are adequate to provide the confidentiality and integrity of the TOE design and implementation that is necessary to ensure that secure operation of the TOE is not compromised.

13.5.1.2 Input

1370 The evaluation evidence for this sub-activity is:

a) the ST;

b) the development security documentation.

1371 In addition, the evaluator may need to examine other deliverables to determine that the security controls are well-defined and followed. Specifically, the evaluator may need to examine the developer's configuration management documentation (the input for the Evaluation of sub-activity (ALC_CMC.4) “Production support and acceptance procedures” and the Evaluation of sub-activity (ALC_CMS.4) “Problem tracking CM coverage”). Evidence that the procedures are being applied is also required.

13.5.1.3 Action ALC_DVS.1.1E

ALC_DVS.1.1C ***The development security 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.***

ALC_DVS.1-1 The evaluator ***shall examine*** the development security documentation to determine that 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.

1372 The evaluator determines what is necessary by first referring to the ST for any information that may assist in the determination of necessary protection, especially the security objectives for the development environment.

1373 If no explicit information is available from the ST the evaluator will need to make a determination of the necessary measures. In cases where the developer's measures are considered less than what is necessary, a clear justification should be provided for the assessment, based on a potential exploitable vulnerability.

1374 The following types of security measures are considered by the evaluator when examining the documentation:

a) physical, for example physical access controls used to prevent unauthorised access to the TOE development environment (during normal working hours and at other times);

b) procedural, for example covering:

- granting of access to the development environment or to specific parts of the environment such as development machines
- revocation of access rights when a person leaves the development team
- transfer of protected material within and out of the development environment and between different development sites in accordance with defined acceptance procedures
- admitting and escorting visitors to the development environment
- roles and responsibilities in ensuring the continued application of security measures, and the detection of security breaches.

c) personnel, for example any controls or checks made to establish the trustworthiness of new development staff;

d) other security measures, for example the logical protections on any development machines.

1375 The development security documentation should identify the locations at which development occurs, and describe the aspects of development performed, along with the security measures applied at each location and for transports between different locations. For example, development could occur at multiple facilities within a single building, multiple buildings at the same site, or at multiple sites. Transports of parts of the TOE or the unfinished TOE between different development sites are to be covered by Development security (ALC_DVS), whereas the transport of the finished TOE to the user is dealt with in Delivery (ALC_DEL).

1376 Development includes such tasks as creating multiple copies (production) of the TOE, where applicable.

1377 Whereas the CM capabilities (CM capabilities (ALC_CMC)) requirements are fixed, those for Development security (Development security (ALC_DVS)), mandating only necessary measures, are dependent on the nature of the TOE, and on information that may be provided in the ST. For example, the ST may identify a security objective for the development environment that requires the TOE to be developed by staff who have security clearance. The evaluators would then determine that such a policy had been applied under this sub-activity.

ALC_DVS.1-2 The evaluator ***shall examine*** the development confidentiality and integrity policies in order to determine the sufficiency of the security measures employed.

1378 These include the policies governing:

a) 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;

b) 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.

1379 The evaluator should determine that these policies are described in the development security documentation, that the security measures employed are consistent with the policies, and that they are complete.

1380 It should be noted that configuration management procedures will help protect the integrity of the TOE and the evaluator should avoid overlap with the work-units conducted for the CM capabilities (ALC_CMC). For example, the CM documentation may describe the security procedures necessary for controlling the roles or individuals who should have access to the development environment and who may modify the TOE.

1381 Whereas the CM capabilities (ALC_CMC) requirements are fixed, those for the Development security (ALC_DVS), mandating only necessary measures, are dependent on the nature of the TOE, and on information that may be provided in the ST. For example, the ST may identify a security objective for the development environment that requires the TOE to be developed by staff who have security clearance. The evaluators would then determine that such a policy had been applied under this sub-activity.

ALC_DVS.1.2C ***The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.***

ALC_DVS.1-3 The evaluator ***shall check*** the development security documentation to determine that documentary evidence that would be produced as a result of application of the procedures has been generated.

1382 Where documentary evidence is produced the evaluator inspects it to ensure compliance with procedures. Examples of the evidence produced may include entry logs and audit trails. The evaluator may choose to sample the evidence.

1383 For guidance on sampling see A.2, Sampling.

13.5.1.4 Action ALC_DVS.1.2E

ALC_DVS.1-4 The evaluator ***shall examine*** the development security documentation and associated evidence to determine that the security measures are being applied.

1384 This work unit requires the evaluator to determine that the security measures described in the development security documentation are being followed, such that the integrity of the TOE and the confidentiality of associated documentation is being adequately protected. For example, this could be determined by examination of the documentary evidence provided. Documentary evidence should be supplemented by visiting the development environment. A visit to the development environment will allow the evaluator to:

a) observe the application of security measures (e.g. physical measures);

b) examine documentary evidence of application of procedures;

c) interview development staff to check awareness of the development security policies and procedures, and their responsibilities.

1385 A development site visit is a useful means of gaining confidence in the measures being used. Any decision not to make such a visit should be determined in consultation with the overseer.

1386 For guidance on site visits see A.4, Site Visits.

13.5.2 Evaluation of sub-activity (ALC_DVS.2)

13.5.2.1 Objectives

1387 The objective of this sub-activity is to determine whether the developer's security controls on the development environment are adequate to provide the confidentiality and integrity of the TOE design and implementation that is necessary to ensure that secure operation of the TOE is not compromised. Additionally, sufficiency of the measures as applied is intended be justified.

13.5.2.2 Input

1388 The evaluation evidence for this sub-activity is:

a) the ST;

b) the development security documentation.

1389 In addition, the evaluator may need to examine other deliverables to determine that the security controls are well-defined and followed. Specifically, the evaluator may need to examine the developer's configuration management documentation (the input for the Evaluation of sub-activity (ALC_CMC.4) “Production support and acceptance procedures”

and the Evaluation of sub-activity (ALC_CMS.4) “Problem tracking CM coverage”). Evidence that the procedures are being applied is also required.

13.5.2.3 Action ALC_DVS.2.1E

ALC_DVS.2.1C ***The development security 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.***

ALC_DVS.2-1 The evaluator ***shall examine*** the development security documentation to determine that 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.

1390 The evaluator determines what is necessary by first referring to the ST for any information that may assist in the determination of necessary protection, especially the security objectives for the development environment.

1391 If no explicit information is available from the ST the evaluator will need to make a determination of the necessary measures. In cases where the developer's measures are considered less than what is necessary, a clear justification should be provided for the assessment, based on a potential exploitable vulnerability.

1392 The following types of security measures are considered by the evaluator when examining the documentation:

a) physical, for example physical access controls used to prevent unauthorised access to the TOE development environment (during normal working hours and at other times);

b) procedural, for example covering:

- granting of access to the development environment or to specific parts of the environment such as development machines
- revocation of access rights when a person leaves the development team
- transfer of protected material out of the development environment and between different development sites in accordance with defined acceptance procedures
- admitting and escorting visitors to the development environment
- roles and responsibilities in ensuring the continued application of security measures, and the detection of security breaches.

c) personnel, for example any controls or checks made to establish the trustworthiness of new development staff;

d) other security measures, for example the logical protections on any development machines.

1393 The development security documentation should identify the locations at which development occurs, and describe the aspects of development performed, along with the security measures applied at each location and for transports between different locations. For example, development could occur at multiple facilities within a single building, multiple buildings at the same site, or at multiple sites. Transports of parts of the TOE or the unfinished TOE between different development sites are to be covered by the Development security (ALC_DVS), whereas the transport of the finished TOE to the user is dealt with in the Delivery (ALC_DEL).

1394 Development includes such tasks as creating multiple copies (production) of the TOE, where applicable.

1395 Whereas the CM capabilities (ALC_CMC) requirements are fixed, those for the Development security (ALC_DVS)), mandating only necessary measures, are dependent on the nature of the TOE, and on information that may be provided in the ST. For example, the ST may identify a security objective for the development environment that requires the TOE to be developed by staff who have security clearance. The evaluators would then determine that such a policy had been applied under this sub-activity.

ALC_DVS.2-2 The evaluator ***shall examine*** the development confidentiality and integrity policies in order to determine the sufficiency of the security measures employed.

1396 These include the policies governing:

a) 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;

b) 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.

1397 The evaluator should determine that these policies are described in the development security documentation, that the security measures employed are consistent with the policies, and that they are complete.

1398 It should be noted that configuration management procedures will help protect the integrity of the TOE and the evaluator should avoid overlap with the work-units conducted for the CM capabilities (ALC_CMC). For example, the CM documentation may describe the security procedures necessary for controlling the roles or individuals who should have access to the development environment and who may modify the TOE.

1399 Whereas the CM capabilities (ALC_CMC) requirements are fixed, those for the Development security (ALC_DVS), mandating only necessary measures, are dependent on the nature of the TOE, and on information that may be provided in the ST. For example, the ST may identify a security objective for the development environment that requires the TOE to be developed by staff who have security clearance. The evaluators would then determine that such a policy had been applied under this sub-activity.

ALC_DVS.2.2C ***The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.***

ALC_DVS.2-3 The evaluator ***shall check*** the development security documentation to determine that documentary evidence that would be produced as a result of application of the procedures has been generated.

1400 Where documentary evidence is produced the evaluator inspects it to ensure compliance with procedures. Examples of the evidence produced may include entry logs and audit trails. The evaluator may choose to sample the evidence.

1401 For guidance on sampling see A.2, Sampling.

ALC_DVS.2.3C ***The evidence shall justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.***

ALC_DVS.2-4 The evaluator ***shall examine*** the development security documentation to determine that an appropriate justification is given why the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

1402 Since attacks on the TOE or its related information are assumed in different design and production stages, measures and procedures need to have an appropriate level necessary to prevent those attacks or to make them more difficult.

1403 Since this level depends on the overall attack potential claimed for the TOE (cf. the Vulnerability analysis (AVA_VAN) component chosen), the development security documentation should justify the necessary level of protection to maintain the confidentiality and integrity of the TOE. This level has to be achieved by the security measures applied.

1404 The concept of protection measures should be consistent, and the justification should include an analysis of how the measures are mutually supportive. All aspects of development and production on all the different sites with all roles involved up to delivery of the TOE should be analysed.

1405 Justification may include an analysis of potential vulnerabilities taking the applied security measures into account.

1406 There may be a convincing argument showing e.g.

– that the technical measures and mechanisms of the developer's infrastructure are sufficient for keeping the appropriate security level (e.g. cryptographic mechanisms as well as physical protection mechanisms, properties of the CM system (cf. ALC_CMC.4-5));

– that the system containing the implementation representation of the TOE (including concerning guidance documents) provides effective protection against logical attacks e.g. by “Trojan” code or viruses. It might be adequate, if the implementation representation is kept on an isolated system where only the software necessary to maintain it is installed and where no additional software is installed afterwards.

– Data brought into this system should be well considered to avoid that hidden functionality is installed on the system. The effectiveness of these measures should be tested, e.g. by independent trying to get access to the machine, install some additional executable (program, macro etc.) or manage to get some information out of the machine using logical attacks.

– The appropriate organisational (procedural and personal) measures are unconditionally enforced.

13.5.2.4 Action ALC_DVS.2.2E

ALC_DVS.2-5 The evaluator ***shall examine*** the development security documentation and associated evidence to determine that the security measures are being applied.

1407 This work unit requires the evaluator to determine that the security measures described in the development security documentation are being followed, such that the integrity of the TOE and the confidentiality of associated documentation is being adequately protected. For example, this could be determined by examination of the documentary evidence provided. Documentary evidence should be supplemented by visiting the development environment. A visit to the development environment will allow the evaluator to:

a) observe the application of security measures (e.g. physical measures);

b) examine documentary evidence of application of procedures;

c) interview development staff to check awareness of the development security policies and procedures, and their responsibilities.

1408 A development site visit is a useful means of gaining confidence in the measures being used. Any decision not to make such a visit should be determined in consultation with the overseer.

1409 For guidance on site visits see A.4, Site Visits.

13.6 Flaw remediation (ALC_FLR)

13.6.1 Evaluation of sub-activity (ALC_FLR.1)

13.6.1.1 Objectives

1410 The objective of this sub-activity is to determine whether the developer has established flaw remediation procedures that describe the tracking of security flaws, the identification of corrective actions, and the distribution of corrective action information to TOE users.

13.6.1.2 Input

1411 The evaluation evidence for this sub-activity is:

a) the flaw remediation procedures documentation.

13.6.1.3 Action ALC_FLR.1.1E

ALC_FLR.1.1C ***The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.***

ALC_FLR.1-1 The evaluator ***shall examine*** the flaw remediation procedures documentation to determine that it describes the procedures used to track all reported security flaws in each release of the TOE.

1412 The procedures describe the actions that are taken by the developer from the time each suspected security flaw is reported to the time that it is resolved. This includes the flaw's entire time frame, from initial detection through ascertaining the flaw is a security flaw, to resolution of the security flaw.

1413 If a flaw is discovered not to be security-relevant, there is no need (for the purposes of the Flaw remediation (ALC_FLR) requirements) for the flaw remediation procedures to track it further; only that there be an explanation of why the flaw is not security-relevant.

1414 While these requirements do not mandate that there be a publicised means for TOE users to report security flaws, they do mandate that all security flaws that are reported be tracked. That is, a reported security flaw cannot be ignored simply because it comes from outside the developer's organisation.

ALC_FLR.1.2C ***The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.***

ALC_FLR.1-2 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would produce a description of each security flaw in terms of its nature and effects.

1415 The procedures identify the actions that are taken by the developer to describe the nature and effects of each security flaw in sufficient detail to be

able to reproduce it. The description of the nature of a security flaw addresses whether it is an error in the documentation, a flaw in the design of the TSF, a flaw in the implementation of the TSF, etc. The description of the security flaw's effects identifies the portions of the TSF that are affected and how those portions are affected. For example, a security flaw in the implementation might be found that affects the identification and authentication enforced by the TSF by permitting authentication with the password “BACKDOOR”.

ALC_FLR.1-3 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would identify the status of finding a correction to each security flaw.

1416 The flaw remediation procedures identify the different stages of security flaws. This differentiation includes at least: suspected security flaws that have been reported, suspected security flaws that have been confirmed to be security flaws, and security flaws whose solutions have been implemented. It is permissible that additional stages (e.g. flaws that have been reported but not yet investigated, flaws that are under investigation, security flaws for which a solution has been found but not yet implemented) be included.

ALC_FLR.1.3C ***The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.***

ALC_FLR.1-4 The evaluator ***shall check*** the flaw remediation procedures to determine that the application of these procedures would identify the corrective action for each security flaw.

1417 *Corrective action* may consist of a repair to the hardware, firmware, or software portions of the TOE, a modification of TOE guidance, or both. Corrective action that constitutes modifications to TOE guidance (e.g. details of procedural measures to be taken to obviate the security flaw) includes both those measures serving as only an interim solution (until the repair is issued) as well as those serving as a permanent solution (where it is determined that the procedural measure is the best solution).

1418 If the source of the security flaw is a documentation error, the corrective action consists of an update of the affected TOE guidance. If the corrective action is a procedural measure, this measure will include an update made to the affected TOE guidance to reflect these corrective procedures.

ALC_FLR.1.4C ***The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.***

ALC_FLR.1-5 The evaluator ***shall examine*** the flaw remediation procedures documentation to determine that it describes a means of providing the TOE users with the necessary information on each security flaw.

1419 The *necessary information* about each security flaw consists of its description (not necessarily at the same level of detail as that provided as part

of work unit ALC_FLR.1-2), the prescribed corrective action, and any associated guidance on implementing the correction.

1420 TOE users may be provided such information, correction, and documentation updates in any of several ways, such as their posting to a website, their being sent to TOE users, or arrangements made for the developer to install the correction. In cases where the means of providing this information requires action to be initiated by the TOE user, the evaluator examines any TOE guidance to ensure that it contains instructions for retrieving the information.

1421 The only metric for assessing the adequacy of the method used for providing the information, corrections and guidance is that there be a reasonable expectation that TOE users can obtain or receive it. For example, consider the method of dissemination where the requisite data is posted to a website for one month, and the TOE users know that this will happen and when this will happen. This may not be especially reasonable or effective (as, say, a permanent posting to the website), yet it is feasible that the TOE user could obtain the necessary information. On the other hand, if the information were posted to the website for only one hour, yet TOE users had no way of knowing this or when it would be posted, it is infeasible that they would ever get the necessary information.

13.6.2 Evaluation of sub-activity (ALC_FLR.2)

13.6.2.1 Objectives

1422 The objective of this sub-activity is to determine whether the developer has established flaw remediation procedures that describe the tracking of security flaws, the identification of corrective actions, and the distribution of corrective action information to TOE users. Additionally, this sub-activity determines whether the developer's procedures provide for the corrections of security flaws, for the receipt of flaw reports from TOE users, and for assurance that the corrections introduce no new security flaws.

1423 In order for the developer to be able to act appropriately upon security flaw reports from TOE users, TOE users need to understand how to submit security flaw reports to the developer, and developers need to know how to receive these reports. Flaw remediation guidance addressed to the TOE user ensures that TOE users are aware of how to communicate with the developer; flaw remediation procedures describe the developer's role is such communication

13.6.2.2 Input

1424 The evaluation evidence for this sub-activity is:

a) the flaw remediation procedures documentation;

b) flaw remediation guidance documentation.

13.6.2.3 Action ALC_FLR.2.1E

ALC_FLR.2.1C ***The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.***

ALC_FLR.2-1 The evaluator ***shall examine*** the flaw remediation procedures documentation to determine that it describes the procedures used to track all reported security flaws in each release of the TOE.

1425 The procedures describe the actions that are taken by the developer from the time each suspected security flaw is reported to the time that it is resolved. This includes the flaw's entire time frame, from initial detection through ascertaining the flaw is a security flaw, to resolution of the security flaw.

1426 If a flaw is discovered not to be security-relevant, there is no need (for the purposes of the Flaw remediation (ALC_FLR) requirements) for the flaw remediation procedures to track it further; only that there be an explanation of why the flaw is not security-relevant.

ALC_FLR.2.2C ***The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.***

ALC_FLR.2-2 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would produce a description of each security flaw in terms of its nature and effects.

1427 The procedures identify the actions that are taken by the developer to describe the nature and effects of each security flaw in sufficient detail to be able to reproduce it. The description of the nature of a security flaw addresses whether it is an error in the documentation, a flaw in the design of the TSF, a flaw in the implementation of the TSF, etc. The description of the security flaw's effects identifies the portions of the TSF that are affected and how those portions are affected. For example, a security flaw in the implementation might be found that affects the identification and authentication enforced by the TSF by permitting authentication with the password “BACKDOOR”.

ALC_FLR.2-3 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would identify the status of finding a correction to each security flaw.

1428 The flaw remediation procedures identify the different stages of security flaws. This differentiation includes at least: suspected security flaws that have been reported, suspected security flaws that have been confirmed to be security flaws, and security flaws whose solutions have been implemented. It is permissible that additional stages (e.g. flaws that have been reported but not yet investigated, flaws that are under investigation, security flaws for which a solution has been found but not yet implemented) be included.

ALC_FLR.2.3C ***The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.***

ALC_FLR.2-4 The evaluator ***shall check*** the flaw remediation procedures to determine that the application of these procedures would identify the corrective action for each security flaw.

1429 *Corrective action* may consist of a repair to the hardware, firmware, or software portions of the TOE, a modification of TOE guidance, or both. Corrective action that constitutes modifications to TOE guidance (e.g. details of procedural measures to be taken to obviate the security flaw) includes both those measures serving as only an interim solution (until the repair is issued) as well as those serving as a permanent solution (where it is determined that the procedural measure is the best solution).

1430 If the source of the security flaw is a documentation error, the corrective action consists of an update of the affected TOE guidance. If the corrective action is a procedural measure, this measure will include an update made to the affected TOE guidance to reflect these corrective procedures.

ALC_FLR.2.4C ***The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.***

ALC_FLR.2-5 The evaluator ***shall examine*** the flaw remediation procedures documentation to determine that it describes a means of providing the TOE users with the necessary information on each security flaw.

1431 *The necessary information* about each security flaw consists of its description (not necessarily at the same level of detail as that provided as part of work unit ALC_FLR.2-2), the prescribed corrective action, and any associated guidance on implementing the correction.

1432 TOE users may be provided such information, correction, and documentation updates in any of several ways, such as their posting to a website, their being sent to TOE users, or arrangements made for the developer to install the correction. In cases where the means of providing this information requires action to be initiated by the TOE user, the evaluator examines any TOE guidance to ensure that it contains instructions for retrieving the information.

1433 The only metric for assessing the adequacy of the method used for providing the information, corrections and guidance is that there be a reasonable expectation that TOE users can obtain or receive it. For example, consider the method of dissemination where the requisite data is posted to a website for one month, and the TOE users know that this will happen and when this will happen. This may not be especially reasonable or effective (as, say, a permanent posting to the website), yet it is feasible that the TOE user could obtain the necessary information. On the other hand, if the information were posted to the website for only one hour, yet TOE users had no way of knowing this or when it would be posted, it is infeasible that they would ever get the necessary information.

ALC_FLR.2.5C ***The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.***

ALC_FLR.2-6 The evaluator ***shall examine*** the flaw remediation procedures to determine that they describe procedures for the developer to accept reports of security flaws or requests for corrections to such flaws.

1434 The procedures ensure that TOE users have a means by which they can communicate with the TOE developer. By having a means of contact with the developer, the user can report security flaws, enquire about the status of security flaws, or request corrections to flaws. This means of contact may be part of a more general contact facility for reporting non-security related problems.

1435 The use of these procedures is not restricted to TOE users; however, only the TOE users are actively supplied with the details of these procedures. Others who might have access to or familiarity with the TOE can use the same procedures to submit reports to the developer, who is then expected to process them. Any means of submitting reports to the developer, other than those identified by the developer, are beyond the scope of this work unit; reports generated by other means need not be addressed.

ALC_FLR.2.6C ***The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.***

ALC_FLR.2-7 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would help to ensure every reported flaw is corrected.

1436 The flaw remediation procedures cover not only those security flaws discovered and reported by developer personnel, but also those reported by TOE users. The procedures are sufficiently detailed so that they describe how it is ensured that each reported security flaw is corrected. The procedures contain reasonable steps that show progress leading to the eventual, inevitable resolution.

1437 The procedures 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.

ALC_FLR.2-8 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would help to ensure that the TOE users are issued corrective actions for each security flaw.

1438 The procedures describe the process that is taken from the point at which a security flaw is resolved to the point at which the corrective action is provided. The procedures for delivering corrective actions should be consistent with the security objectives; they need not necessarily be identical to the procedures used for delivering the TOE, as documented to meet ALC_DEL, if included in the assurance requirements. For example, if the

hardware portion of a TOE were originally delivered by bonded courier, updates to hardware resulting from flaw remediation would likewise be expected to be distributed by bonded courier. Updates unrelated to flaw remediation would follow the procedures set forth in the documentation meeting the Delivery (ALC_DEL) requirements.

ALC_FLR.2.7C ***The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.***

ALC_FLR.2-9 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would result in safeguards that the potential correction contains no adverse effects.

1439 Through analysis, testing, or a combination of the two, the developer may reduce the likelihood that adverse effects will be introduced when a security flaw is corrected. The evaluator assesses whether the procedures provide detail in how the necessary mix of analysis and testing actions is to be determined for a given correction.

1440 The evaluator also determines that, for instances where the source of the security flaw is a documentation problem, the procedures include the means of safeguarding against the introduction of contradictions with other documentation.

ALC_FLR.2.8C ***The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.***

ALC_FLR.2-10 The evaluator ***shall examine*** the flaw remediation guidance to determine that the application of these procedures would result in a means for the TOE user to provide reports of suspected security flaws or requests for corrections to such flaws.

1441 The guidance ensures that TOE users have a means by which they can communicate with the TOE developer. By having a means of contact with the developer, the user can report security flaws, enquire about the status of security flaws, or request corrections to flaws.

13.6.3 Evaluation of sub-activity (ALC_FLR.3)

13.6.3.1 Objectives

1442 The objective of this sub-activity is to determine whether the developer has established flaw remediation procedures that describe the tracking of security flaws, the identification of corrective actions, and the distribution of corrective action information to TOE users. Additionally, this sub-activity determines whether the developer's procedures provide for the corrections of security flaws, for the receipt of flaw reports from TOE users, for assurance that the corrections introduce no new security flaws, for the establishment of a point of contact for each TOE user, and for the timely issue of corrective actions to TOE users.

1443 In order for the developer to be able to act appropriately upon security flaw reports from TOE users, TOE users need to understand how to submit security flaw reports to the developer, and developers need to know how to receive these reports. Flaw remediation guidance addressed to the TOE user ensures that TOE users are aware of how to communicate with the developer; flaw remediation procedures describe the developer's role is such communication.

13.6.3.2 Input

1444 The evaluation evidence for this sub-activity is:

a) the flaw remediation procedures documentation;

b) flaw remediation guidance documentation.

13.6.3.3 Action ALC_FLR.3.1E

ALC_FLR.3.1C ***The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.***

ALC_FLR.3-1 The evaluator ***shall examine*** the flaw remediation procedures documentation to determine that it describes the procedures used to track all reported security flaws in each release of the TOE.

1445 The procedures describe the actions that are taken by the developer from the time each suspected security flaw is reported to the time that it is resolved. This includes the flaw's entire time frame, from initial detection through ascertaining the flaw is a security flaw, to resolution of the security flaw.

1446 If a flaw is discovered not to be security-relevant, there is no need (for the purposes of the Flaw remediation (ALC_FLR) requirements) for the flaw remediation procedures to track it further; only that there be an explanation of why the flaw is not security-relevant.

ALC_FLR.3.2C ***The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.***

ALC_FLR.3-2 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would produce a description of each security flaw in terms of its nature and effects.

1447 The procedures identify the actions that are taken by the developer to describe the nature and effects of each security flaw in sufficient detail to be able to reproduce it. The description of the nature of a security flaw addresses whether it is an error in the documentation, a flaw in the design of the TSF, a flaw in the implementation of the TSF, etc. The description of the security flaw's effects identifies the portions of the TSF that are affected and how those portions are affected. For example, a security flaw in the implementation might be found that affects the identification and

authentication enforced by the TSF by permitting authentication with the password “BACKDOOR”.

ALC_FLR.3-3 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would identify the status of finding a correction to each security flaw.

1448 The flaw remediation procedures identify the different stages of security flaws. This differentiation includes at least: suspected security flaws that have been reported, suspected security flaws that have been confirmed to be security flaws, and security flaws whose solutions have been implemented. It is permissible that additional stages (e.g. flaws that have been reported but not yet investigated, flaws that are under investigation, security flaws for which a solution has been found but not yet implemented) be included.

ALC_FLR.3.3C ***The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.***

ALC_FLR.3-4 The evaluator ***shall check*** the flaw remediation procedures to determine that the application of these procedures would identify the corrective action for each security flaw.

1449 *Corrective action* may consist of a repair to the hardware, firmware, or software portions of the TOE, a modification of TOE guidance, or both. Corrective action that constitutes modifications to TOE guidance (e.g. details of procedural measures to be taken to obviate the security flaw) includes both those measures serving as only an interim solution (until the repair is issued) as well as those serving as a permanent solution (where it is determined that the procedural measure is the best solution).

1450 If the source of the security flaw is a documentation error, the corrective action consists of an update of the affected TOE guidance. If the corrective action is a procedural measure, this measure will include an update made to the affected TOE guidance to reflect these corrective procedures.

ALC_FLR.3.4C ***The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.***

ALC_FLR.3-5 The evaluator ***shall examine*** the flaw remediation procedures documentation to determine that it describes a means of providing the TOE users with the necessary information on each security flaw.

1451 *The necessary information* about each security flaw consists of its description (not necessarily at the same level of detail as that provided as part of work unit ALC_FLR.3-2), the prescribed corrective action, and any associated guidance on implementing the correction.

1452 TOE users may be provided such information, correction, and documentation updates in any of several ways, such as their posting to a website, their being sent to TOE users, or arrangements made for the developer to install the

correction. In cases where the means of providing this information requires action to be initiated by the TOE user, the evaluator examines any TOE guidance to ensure that it contains instructions for retrieving the information.

1453 The only metric for assessing the adequacy of the method used for providing the information, corrections and guidance is that there be a reasonable expectation that TOE users can obtain or receive it. For example, consider the method of dissemination where the requisite data is posted to a website for one month, and the TOE users know that this will happen and when this will happen. This may not be especially reasonable or effective (as, say, a permanent posting to the website), yet it is feasible that the TOE user could obtain the necessary information. On the other hand, if the information were posted to the website for only one hour, yet TOE users had no way of knowing this or when it would be posted, it is infeasible that they would ever get the necessary information.

1454 For TOE users who register with the developer (see work unit ALC_FLR.3-12), the passive availability of this information is not sufficient. Developers must actively send the information (or a notification of its availability) to registered TOE users.

ALC_FLR.3.5C ***The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.***

ALC_FLR.3-6 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would result in a means for the developer to receive from TOE user reports of suspected security flaws or requests for corrections to such flaws.

1455 The procedures ensure that TOE users have a means by which they can communicate with the TOE developer. By having a means of contact with the developer, the user can report security flaws, enquire about the status of security flaws, or request corrections to flaws. This means of contact may be part of a more general contact facility for reporting non-security related problems.

1456 The use of these procedures is not restricted to TOE users; however, only the TOE users are actively supplied with the details of these procedures. Others who might have access to or familiarity with the TOE can use the same procedures to submit reports to the developer, who is then expected to process them. Any means of submitting reports to the developer, other than those identified by the developer, are beyond the scope of this work unit; reports generated by other means need not be addressed.

ALC_FLR.3.6C ***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.***

ALC_FLR.3-7 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would result in a timely means of providing the registered TOE users who might be affected with reports about, and associated corrections to, each security flaw.

1457 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.

1458 It is unnecessary to restrict the recipients of the reports and associated corrections to only those TOE users who might be affected by the security flaw; it is permissible that all TOE users be given such reports and corrections for all security flaws, provided such is done in a timely manner.

ALC_FLR.3-8 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would result in automatic distribution of the reports and associated corrections to the registered TOE users who might be affected.

1459 *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.

1460 It is unnecessary to restrict the recipients of the reports and associated corrections to only those TOE users who might be affected by the security flaw; it is permissible that all TOE users be given such reports and corrections for all security flaws, provided such is done automatically.

ALC_FLR.3.7C ***The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.***

ALC_FLR.3-9 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would help to ensure that every reported flaw is corrected.

1461 The flaw remediation procedures cover not only those security flaws discovered and reported by developer personnel, but also those reported by TOE users. The procedures are sufficiently detailed so that they describe how it is ensured that each reported security flaw is corrected. The procedures contain reasonable steps that show progress leading to the eventual, inevitable resolution.

1462 The procedures 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.

ALC_FLR.3-10 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would help to ensure that the TOE users are issued corrective actions for each security flaw.

1463 The procedures describe the process that is taken from the point at which a security flaw is resolved to the point at which the corrective action is provided. The procedures for delivering corrective actions should be consistent with the security objectives; they need not necessarily be identical to the procedures used for delivering the TOE, as documented to meet Delivery (ALC_DEL), if included in the assurance requirements. For example, if the hardware portion of a TOE were originally delivered by bonded courier, updates to hardware resulting from flaw remediation would likewise be expected to be distributed by bonded courier. Updates unrelated to flaw remediation would follow the procedures set forth in the documentation meeting the Delivery (ALC_DEL) requirements.

ALC_FLR.3.8C ***The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.***

ALC_FLR.3-11 The evaluator ***shall examine*** the flaw remediation procedures to determine that the application of these procedures would result in safeguards that the potential correction contains no adverse effects.

1464 Through analysis, testing, or a combination of the two, the developer may reduce the likelihood that adverse effects will be introduced when a security flaw is corrected. The evaluator assesses whether the procedures provide detail in how the necessary mix of analysis and testing actions is to be determined for a given correction.

1465 The evaluator also determines that, for instances where the source of the security flaw is a documentation problem, the procedures include the means of safeguarding against the introduction of contradictions with other documentation.

ALC_FLR.3.9C ***The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.***

ALC_FLR.3-12 The evaluator ***shall examine*** the flaw remediation guidance to determine that the application of these procedures would result in a means for the TOE user to provide reports of suspected security flaws or requests for corrections to such flaws.

1466 The guidance ensures that TOE users have a means by which they can communicate with the TOE developer. By having a means of contact with the developer, the user can report security flaws, enquire about the status of security flaws, or request corrections to flaws.

ALC_FLR.3.10C ***The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections.***

ALC_FLR.3-13 The evaluator ***shall examine*** the flaw remediation guidance to determine that it describes a means of enabling the TOE users to register with the developer.

1467 *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.

1468 There need not be one registered TOE user per installation of the TOE; it would be sufficient if there were one registered TOE user for an organisation. For example, a corporate TOE user might have a centralised acquisition office for all of its sites. In this case, the acquisition office would be a sufficient point of contact for all of that TOE user's sites, so that all of the TOE user's installations of the TOE have a registered point of contact.

1469 In either case, it must be possible to associate each TOE that is delivered with an organisation in order to ensure that there is a registered user for each TOE. For organisations that have many different addresses, this assures that there will be no user who is erroneously presumed to be covered by a registered TOE user.

1470 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).

ALC_FLR.3.11C ***The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.***

ALC_FLR.3-14 The evaluator ***shall examine*** the flaw remediation guidance to determine that it identifies specific points of contact for user reports and enquiries about security issues involving the TOE.

1471 The guidance includes a means whereby registered TOE users can interact with the developer to report discovered security flaws in the TOE or to make enquiries regarding discovered security flaws in the TOE.

13.7 Life-cycle definition (ALC_LCD)

13.7.1 Evaluation of sub-activity (ALC_LCD.1)

13.7.1.1 Objectives

1472 The objective of this sub-activity is to determine whether the developer has used a documented model of the TOE life-cycle.

13.7.1.2 Input

1473 The evaluation evidence for this sub-activity is:

a) the ST;

b) the life-cycle definition documentation.

13.7.1.3 Action ALC_LCD.1.1E

ALC_LCD.1.1C ***The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.***

ALC_LCD.1-1 The evaluator ***shall examine*** the documented description of the life-cycle model used to determine that it covers the development and maintenance process.

1474 The description of the life-cycle model should include:

a) information on the life-cycle phases of the TOE and the boundaries between the subsequent phases;

b) information on the procedures, tools and techniques used by the developer (e.g. for design, coding, testing, bug-fixing);

c) 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);

d) information which parts of the TOE are delivered by subcontractors, if subcontractors are involved.

1475 Evaluation of sub-activity (ALC_LCD.1) does not require the model used to conform to any standard life-cycle model.

ALC_LCD.1.2C ***The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.***

ALC_LCD.1-2 The evaluator ***shall examine*** the life-cycle model to determine that 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.

1476 The information provided in the life-cycle model gives the evaluator assurance that the development and maintenance procedures adopted would minimise the likelihood of security flaws. For example, if the life-cycle model described the review process, but did not make provision for recording changes to components, then the evaluator may be less confident that errors will not be introduced into the TOE. The evaluator may gain further assurance by comparing the description of the model against an understanding of the development process gleaned from performing other evaluator actions relating to the TOE development (e.g. those covered under the CM capabilities (ALC_CMC)). Identified deficiencies in the life-cycle model will be of concern if they might reasonably be expected to give rise to the introduction of flaws into the TOE, either accidentally or deliberately.

1477 The CC does not mandate any particular development approach, and each should be judged on merit. For example, spiral, rapid-prototyping and waterfall approaches to design can all be used to produce a quality TOE if applied in a controlled environment.

13.7.2 Evaluation of sub-activity (ALC_LCD.2)

13.7.2.1 Objectives

1478 The objective of this sub-activity is to determine whether the developer has used a documented and standardised model of the TOE life-cycle.

13.7.2.2 Input

1479 The evaluation evidence for this sub-activity is:

a) the ST;

b) the life-cycle definition documentation.

c) information about the standard used.

13.7.2.3 Action ALC_LCD.2.1E

ALC_LCD.2.1C ***The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.***

ALC_LCD.2-1 The evaluator ***shall examine*** the documented description of the life-cycle model used to determine that it covers the development and maintenance process.

1480 The description of the life-cycle model should include:

a) information on the life-cycle phases of the TOE and the boundaries between the subsequent phases;

b) information on the procedures, tools and techniques used by the developer (e.g. for design, coding, testing, bug-fixing);

c) 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);

d) information which parts of the TOE are delivered by subcontractors, if subcontractors are involved.

ALC_LCD.2.2C ***The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.***

ALC_LCD.2-2 The evaluator ***shall examine*** the life-cycle definition documentation to determine that it explains how the model has been applied to the development and maintenance of the TOE.

1481 Whereas the requirements of the Evaluation of sub-activity (ALC_LCD.1) are confined to a description of the model used, this component requires the developer to explain how the model has been applied to the TOE under evaluation. This explanation should cover using the life-cycle model for development and maintenance of the TOE as well as compliance of the model with the configuration management (family ALC_CMC).

ALC_LCD.2.3C ***The life-cycle definition documentation shall demonstrate that the life-cycle model used by the developer is compliant with the standardised life-cycle model.***

ALC_LCD.2-3 The evaluator ***shall examine*** the life-cycle definition documentation to determine that it demonstrates that the life-cycle model used by the developer corresponds to the standardised model.

1482 The life-cycle definition documentation should relate aspects of the standardised model to the specific development and maintenance procedures in place for the TOE, such that conformance to the standardised model can be easily confirmed by the evaluator. The correspondence evidence may, for example, take the form of a mapping from detailed steps and organisation roles in the standardised model to individual development procedures and roles or personnel from the development environment.

1483 The life-cycle definition documentation should describe adaptations of the standardised model to meet specific TOE or organisational requirements.

1484 Through completion of this work unit, the evaluator should gain a clear understanding of how the standardised model has been applied, and that it has been applied correctly.

ALC_LCD.2.4C ***The life-cycle definition documentation shall explain why the model was chosen.***

ALC_LCD.2-4 The evaluator ***shall examine*** the life-cycle definition documentation to determine that it explains why the model was chosen.

1485 The life-cycle definition documentation should provide reasons for adoption of the chosen life-cycle model. Such reasons may include, for example, conformance to an organisational policy or to a security policy (also in the form of the Security Target), or may be in the form of benefits perceived to be attainable through use of the life-cycle model.

ALC_LCD.2.5C ***The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.***

ALC_LCD.2-5 The evaluator ***shall examine*** the life-cycle model to determine that 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.

1486 The information provided in the life-cycle model gives the evaluator assurance that the development and maintenance procedures adopted would minimise the likelihood of security flaws. For example, if the life-cycle model described the review process, but did not make provision for recording changes to components, then the evaluator may be less confident that errors will not be introduced into the TOE. The evaluator may gain further assurance by comparing the description of the model against an understanding of the development process gleaned from performing other evaluator actions relating to the TOE development (e.g. those covered under the CM capabilities (ALC_CMC)). Identified deficiencies in the life-cycle model will be of concern if they might reasonably be expected to give rise to the introduction of flaws into the TOE, either accidentally or deliberately.

1487 The CC does not mandate any particular development approach, and each should be judged on merit. For example, spiral, rapid-prototyping and waterfall approaches to design can all be used to produce a quality TOE if applied in a controlled environment.

13.7.3 Evaluation of sub-activity (ALC_LCD.3)

13.7.3.1 Objectives

1488 The objective of this sub-activity is to determine whether the developer has used a documented, standardised and measurable model of the TOE life-cycle.

13.7.3.2 Input

1489 The evaluation evidence for this sub-activity is:

a) the ST;

b) the life-cycle definition documentation;

c) information about the standard used;

d) the life-cycle output documentation.

13.7.3.3 Action ALC_LCD.3.1E

ALC_LCD.3.1C ***The life-cycle definition documentation shall describe the model used to develop and maintain the TOE, including the details of its arithmetic parameters and/or metrics used to measure the TOE development against the model.***

ALC_LCD.3-1 The evaluator ***shall examine*** the documented description of the life-cycle model used to determine that it covers the development and maintenance process, including the details of its arithmetic parameters and/or metrics used to measure the TOE development against the model.

1490 The description of the life-cycle model should include:

a) information on the life-cycle phases of the TOE and the boundaries between the subsequent phases;

b) information on the procedures, tools and techniques used by the developer (e.g. for design, coding, testing, bug-fixing);

c) 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);

d) information which parts of the TOE are delivered by subcontractors, if subcontractors are involved;

e) information on the parameters/metrics that are used to measure the TOE development against the model. Metrics standards typically include guides for measuring and producing reliable products and cover the aspects reliability, quality, performance, complexity and cost.

ALC_LCD.3.2C ***The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.***

ALC_LCD.3-2 The evaluator ***shall examine*** the life-cycle definition documentation to determine that it explains how the model has been applied to the development and maintenance of the TOE.

1491 Whereas the requirements of the Evaluation of sub-activity (ALC_LCD.1) are confined to a description of the model used, this component requires the developer to explain how the model has been applied to the TOE under evaluation. This explanation should cover using the life-cycle model for development and maintenance of the TOE as well as compliance of the model with the configuration management (family ALC_CMC).

ALC_LCD.3.3C ***The life-cycle definition documentation shall demonstrate that the life-cycle model used by the developer is compliant with the standardised and measurable life-cycle model.***

ALC_LCD.3-3 The evaluator ***shall examine*** the life-cycle definition documentation to determine that it demonstrates that the life-cycle model used by the developer corresponds to the standardised and measurable model.

1492 The life-cycle definition documentation should relate aspects of the standardised model to the specific development and maintenance procedures in place for the TOE, such that conformance to the standardised model can be easily confirmed by the evaluator. The correspondence evidence may, for example, take the form of a mapping from detailed steps and organisation roles in the standardised model to individual development procedures and roles or personnel from the development environment.

1493 The life-cycle definition documentation should describe adaptations of the standardised model to meet specific TOE or organisational requirements.

1494 Through completion of this work unit, the evaluator should gain a clear understanding of how the standardised model has been applied, and that it has been applied correctly.

ALC_LCD.3.4C ***The life-cycle definition documentation shall explain why the model was chosen.***

ALC_LCD.3-4 The evaluator ***shall examine*** the life-cycle definition documentation to determine that it explains why the model was chosen.

1495 The life-cycle definition documentation should provide reasons for adoption of the chosen life-cycle model. Such reasons may include, for example, conformance to an organisational policy or to a security policy (also in the form of the Security Target), or may be in the form of benefits perceived to be attainable through use of the life-cycle model.

ALC_LCD.3.5C ***The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.***

ALC_LCD.3-5 The evaluator ***shall examine*** the life-cycle model to determine that 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.

1496 The information provided in the life-cycle model gives the evaluator assurance that the development and maintenance procedures adopted would minimise the likelihood of security flaws. For example, if the life-cycle model described the review process, but did not make provision for recording changes to components, then the evaluator may be less confident that errors will not be introduced into the TOE. The evaluator may gain further assurance by comparing the description of the model against an understanding of the development process gleaned from performing other

evaluator actions relating to the TOE development (e.g. those covered under the CM capabilities (ALC_CMC)). Identified deficiencies in the life-cycle model will be of concern if they might reasonably be expected to give rise to the introduction of flaws into the TOE, either accidentally or deliberately.

1497 The CC does not mandate any particular development approach, and each should be judged on merit. For example, spiral, rapid-prototyping and waterfall approaches to design can all be used to produce a quality TOE if applied in a controlled environment.

ALC_LCD.3.6C ***The life-cycle output documentation shall provide the results of the measurements of the TOE development using the standardised and measurable life-cycle model.***

ALC_LCD.3-6 The evaluator ***shall examine*** the life-cycle output documentation to determine that it provides the results of the measurements of the TOE development using the standardised and measurable life-cycle model.

1498 The results of the measurements and the life-cycle progress of the TOE should be in accordance with the life-cycle model.

13.8 Tools and techniques (ALC_TAT)

13.8.1 Evaluation of sub-activity (ALC_TAT.1)

13.8.1.1 Objectives

1499 The objective of this sub-activity is to determine whether the developer has used well-defined development tools (e.g. programming languages or computer-aided design (CAD) systems) that yield consistent and predictable results.

13.8.1.2 Input

1500 The evaluation evidence for this sub-activity is:

a) the development tool documentation;

b) the subset of the implementation representation.

13.8.1.3 Application notes

1501 This work may be performed in parallel with the evaluation activities under ADV_IMP, specifically with regard to determining the use of features in the tools that will affect the object code (e.g. compilation options).

13.8.1.4 Action ALC_TAT.1.1E

ALC_TAT.1.1C ***All development tools used for implementation shall be well-defined.***

ALC_TAT.1-1 The evaluator ***shall examine*** the development tool documentation provided to determine that all development tools are well-defined.

1502 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.

ALC_TAT.1.2C ***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.***

ALC_TAT.1-2 The evaluator ***shall examine*** the documentation of development tools to determine that it unambiguously defines the meaning of all statements as well as all conventions and directives used in the implementation.

1503 The development tool documentation (e.g. programming language specifications and user manuals) should cover all statements used in the implementation representation of the TOE, and for each such statement provide a clear and unambiguous definition of the purpose and effect of that statement. This work may be performed in parallel with the evaluator's examination of the implementation representation performed during the

ADV_IMP sub-activity. The key test the evaluator should apply is whether or not the documentation is sufficiently clear for the evaluator to be able to understand the implementation representation. The documentation should not assume (for example) that the reader is an expert in the programming language used.

1504 Reference to the use of a documented standard is an acceptable approach to meet this requirement, provided that the standard is available to the evaluator. Any differences from the standard should be documented.

1505 The critical test is whether the evaluator can understand the TOE source code when performing source code analysis covered in the ADV_IMP sub-activity. However, the following checklist can additionally be used in searching for problem areas:

a) In the language definition, phrases such as "the effect of this construct is undefined" and terms such as "implementation dependent" or "erroneous" may indicate ill-defined areas.

b) Aliasing (allowing the same piece of memory to be referenced in different ways) is a common source of ambiguity problems.

c) Exception handling (e.g. what happens after memory exhaustion or stack overflow) is often poorly defined.

1506 Most languages in common use, however well designed, will have some problematic constructs. If the implementation language is mostly well defined, but some problematic constructs exist, then an inconclusive verdict should be assigned, pending examination of the source code.

1507 The evaluator should verify, during the examination of source code, that any use of the problematic constructs does not introduce vulnerabilities. The evaluator should also ensure that constructs precluded by the documented standard are not used.

1508 The development tool documentation should define all conventions and directives used in the implementation.

ALC_TAT.1.3C ***The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.***

ALC_TAT.1-3 The evaluator ***shall examine*** the development tool documentation to determine that it unambiguously defines the meaning of all implementation-dependent options.

1509 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.

1510 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).

13.8.2 Evaluation of sub-activity (ALC_TAT.2)

13.8.2.1 Objectives

1511 The objective of this sub-activity is to determine whether the developer has used well-defined development tools (e.g. programming languages or computer-aided design (CAD) systems) that yield consistent and predictable results, and whether implementation standards have been applied.

13.8.2.2 Input

1512 The evaluation evidence for this sub-activity is:

a) the development tool documentation;

b) the implementation standards description.

c) the provided implementation representation of the TSF.

13.8.2.3 Application notes

1513 This work may be performed in parallel with the evaluation activities under ADV_IMP, specifically with regard to determining the use of features in the tools that will affect the object code (e.g. compilation options).

13.8.2.4 Action ALC_TAT.2.1E

ALC_TAT.2.1C ***All development tools used for implementation shall be well-defined.***

ALC_TAT.2-1 The evaluator ***shall examine*** the development tool documentation provided to determine that all development tools are well-defined.

1514 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.

ALC_TAT.2.2C ***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.***

ALC_TAT.2-2 The evaluator ***shall examine*** the documentation of development tools to determine that it unambiguously defines the meaning of all statements as well as all conventions and directives used in the implementation.

1515 The development tool documentation (e.g. programming language specifications and user manuals) should cover all statements used in the

implementation representation of the TOE, and for each such statement provide a clear and unambiguous definition of the purpose and effect of that statement. This work may be performed in parallel with the evaluator's examination of the implementation representation performed during the ADV_IMP sub-activity. The key test the evaluator should apply is whether or not the documentation is sufficiently clear for the evaluator to be able to understand the implementation representation. The documentation should not assume (for example) that the reader is an expert in the programming language used.

1516 Reference to the use of a documented standard is an acceptable approach to meet this requirement, provided that the standard is available to the evaluator. Any differences from the standard should be documented.

1517 The critical test is whether the evaluator can understand the TOE source code when performing source code analysis covered in the ADV_IMP sub-activity. However, the following checklist can additionally be used in searching for problem areas:

a) In the language definition, phrases such as “the effect of this construct is undefined” and terms such as “implementation dependent” or “erroneous” may indicate ill-defined areas.

b) Aliasing (allowing the same piece of memory to be referenced in different ways) is a common source of ambiguity problems.

c) Exception handling (e.g. what happens after memory exhaustion or stack overflow) is often poorly defined.

1518 Most languages in common use, however well designed, will have some problematic constructs. If the implementation language is mostly well defined, but some problematic constructs exist, then an inconclusive verdict should be assigned, pending examination of the source code.

1519 The evaluator should verify, during the examination of source code, that any use of the problematic constructs does not introduce vulnerabilities. The evaluator should also ensure that constructs precluded by the documented standard are not used.

1520 The development tool documentation should define all conventions and directives used in the implementation.

ALC_TAT.2.3C ***The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.***

ALC_TAT.2-3 The evaluator ***shall examine*** the development tool documentation to determine that it unambiguously defines the meaning of all implementation-dependent options.

1521 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.

1522 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).

13.8.2.5 Action ALC_TAT.2.2E

ALC_TAT.2-4 The evaluator ***shall examine*** aspects of the implementation process to determine that documented implementation standards have been applied.

1523 This work unit requires the evaluator to analyse the provided implementation representation of the TOE to determine whether the documented implementation standards have been applied.

1524 The evaluator should verify that constructs excluded by the documented standard are not used.

1525 Additionally, the evaluator should verify the developer's procedures which ensure the application of the defined standards within the design and implementation process of the TOE. Therefore, documentary evidence should be supplemented by visiting the development environment. A visit to the development environment will allow the evaluator to:

a) observe the application of defined standards;

b) examine documentary evidence of application of procedures describing the use of defined standards;

c) interview development staff to check awareness of the application of defined standards and procedures.

1526 A development site visit is a useful means of gaining confidence in the procedures being used. Any decision not to make such a visit should be determined in consultation with the overseer.

1527 The evaluator compares the provided implementation representation with the description of the applied implementation standards and verifies their use.

1528 At this level it is not required that the complete provided implementation representation of the TSF is based on implementation standards, but only those parts that are developed by the TOE developer himself. The evaluator may consult the configuration list required by the CM scope (ALC_CMS) to get the information which parts are developed by the TOE developer, and which by third party developers.

1529 If the referenced implementation standards are not applied for at least parts of the provided implementation representation, this work unit fails.

1530 Note that parts of the TOE which are not TSF relevant need not to be examined.

1531 This work unit may be performed in conjunction with the evaluation activities under ADV_IMP.

13.8.3 Evaluation of sub-activity (ALC_TAT.3)

13.8.3.1 Objectives

1532 The objective of this sub-activity is to determine whether the developer and his subcontractors have used well-defined development tools (e.g. programming languages or computer-aided design (CAD) systems) that yield consistent and predictable results, and whether implementation standards have been applied.

13.8.3.2 Input

1533 The evaluation evidence for this sub-activity is:

a) the development tool documentation;

b) the implementation standards description.

c) the provided implementation representation of the TSF.

13.8.3.3 Application notes

1534 This work may be performed in parallel with the evaluation activities under ADV_IMP, specifically with regard to determining the use of features in the tools that will affect the object code (e.g. compilation options).

13.8.3.4 Action ALC_TAT.3.1E

ALC_TAT.3.1C ***All development tools used for implementation shall be well-defined.***

ALC_TAT.3-1 The evaluator ***shall examine*** the development tool documentation provided to determine that all development tools are well-defined.

1535 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.

1536 At this level, the documentation of development tools used by third party contributors to the TOE has to be included in the evaluator's examination.

ALC_TAT.3.2C ***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.***

ALC_TAT.3-2 The evaluator ***shall examine*** the documentation of development tools to determine that it unambiguously defines the meaning of all statements as well as all conventions and directives used in the implementation.

1537 The development tool documentation (e.g. programming language specifications and user manuals) should cover all statements used in the implementation representation of the TOE, and for each such statement provide a clear and unambiguous definition of the purpose and effect of that statement. This work may be performed in parallel with the evaluator's examination of the implementation representation performed during the ADV_IMP sub-activity. The key test the evaluator should apply is whether or not the documentation is sufficiently clear for the evaluator to be able to understand the implementation representation. The documentation should not assume (for example) that the reader is an expert in the programming language used.

1538 Reference to the use of a documented standard is an acceptable approach to meet this requirement, provided that the standard is available to the evaluator. Any differences from the standard should be documented.

1539 The critical test is whether the evaluator can understand the TOE source code when performing source code analysis covered in the ADV_IMP sub-activity. However, the following checklist can additionally be used in searching for problem areas:

a) In the language definition, phrases such as “the effect of this construct is undefined” and terms such as “implementation dependent” or “erroneous” may indicate ill-defined areas.

b) Aliasing (allowing the same piece of memory to be referenced in different ways) is a common source of ambiguity problems.

c) Exception handling (e.g. what happens after memory exhaustion or stack overflow) is often poorly defined.

1540 Most languages in common use, however well designed, will have some problematic constructs. If the implementation language is mostly well defined, but some problematic constructs exist, then an inconclusive verdict should be assigned, pending examination of the source code.

1541 The evaluator should verify, during the examination of source code, that any use of the problematic constructs does not introduce vulnerabilities. The evaluator should also ensure that constructs precluded by the documented standard are not used.

1542 The development tool documentation should define all conventions and directives used in the implementation.

1543 At this level, the documentation of development tools used by third party contributors to the TOE has to be included in the evaluator's examination.

ALC_TAT.3.3C ***The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.***

ALC_TAT.3-3 The evaluator ***shall examine*** the development tool documentation to determine that it unambiguously defines the meaning of all implementation-dependent options.

1544 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.

1545 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).

1546 At this level, the documentation of development tools used by third party contributors to the TOE has to be included in the evaluator's examination.

13.8.3.5 Action ALC_TAT.3.2E

ALC_TAT.3-4 The evaluator ***shall examine*** aspects of the implementation process to determine that documented implementation standards have been applied.

1547 This work unit requires the evaluator to analyse the provided implementation representation of the TOE to determine whether the documented implementation standards have been applied.

1548 The evaluator should verify that constructs excluded by the documented standard are not used.

1549 Additionally, the evaluator should verify the developer's procedures which ensure the application of the defined standards within the design and implementation process of the TOE. Therefore, documentary evidence should be supplemented by visiting the development environment. A visit to the development environment will allow the evaluator to:

a) observe the application of defined standards;

b) examine documentary evidence of application of procedures describing the use of defined standards;

c) interview development staff to check awareness of the application of defined standards and procedures.

1550 A development site visit is a useful means of gaining confidence in the procedures being used. Any decision not to make such a visit should be determined in consultation with the overseer.

1551 The evaluator compares the provided implementation representation with the description of the applied implementation standards and verifies their use.

1552 At this level it is required that the complete provided implementation representation of the TSF is based on implementation standards, including third party contributions. This may require the evaluator to visit the sites of contributors. The evaluator may consult the configuration list required by the CM scope (ALC_CMS) to see who has developed which part of the TOE.

1553 Note that parts of the TOE which are not TSF relevant need not to be examined.

1554 This work unit may be performed in conjunction with the evaluation activities under ADV_IMP.

14 Class ATE: Tests

14.1 Introduction

1555 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 (Functional tests (ATE_FUN)) and independent testing the TSF by the evaluator (Independent testing (ATE_IND)). At the lowest level of assurance, there is no requirement for developer involvement, so the only testing is conducted by the evaluator, using the limited available information about the TOE. Additional assurance is gained as the developer becomes increasingly involved both in testing and in providing additional information about the TOE, and as the evaluator increases the independent testing activities.

14.2 Application notes

1556 Testing of the TSF is conducted by the evaluator and, in most cases, by the developer. The evaluator's testing efforts consist not only of creating and running original tests, but also of assessing the adequacy of the developer's tests and re-running a set of them.

1557 The evaluator analyses the developer's tests to determine the extent to which they are sufficient to demonstrate that TSFI (see Functional specification (ADV_FSP)) perform as specified, and to understand the developer's approach to testing. The evaluator also executes a subset of the developer's tests as documented to gain confidence in the developer's test results: the evaluator will use the results of this analysis as an input to independently testing a subset of the TSF. With respect to this subset, the evaluator's tests take a testing approach that is different from that of the developer's tests, particularly if the developer's tests have shortcomings.

1558 To determine the adequacy of developer's test documentation or to create new tests, the evaluator needs to understand the desired expected behaviour of a TSFI in the context of the requirements it is to satisfy. The evaluator may choose to divide the TSFI into subsets according to functional areas of the ST (audit-related TSFI, authentication-related TSFI, etc.) and focus on one subset of the at a time, examining the ST requirement and the relevant parts of the development and guidance documentation to gain an understanding of the way the TOE is expected to behave. This reliance upon the development documentation underscores the need for the dependencies on ADV by Coverage (ATE_COV) and Depth (ATE_DPT).

1559 The CC has separated coverage and depth from functional tests to increase the flexibility when applying the components of the families. However, the requirements of the families are intended to be applied together to confirm that the TSF operates according to its specification. This tight coupling of families has led to some duplication of evaluator work effort across sub-

activities. These application notes are used to minimise duplication of text between sub-activities of the same activity and level of assurance.

14.2.1 Understanding the expected behaviour of the TOE

1560 Before the adequacy of test documentation can be accurately evaluated, or before new tests can be created, the evaluator has to understand the desired expected behaviour of a security function in the context of the requirements it is to satisfy.

1561 As mentioned earlier, the evaluator may choose to subset the TSFI according to functional requirements (audit, authentication, etc.) in the ST and focus on one subset of the at a time. The evaluator examines each ST requirement and the relevant parts of the functional specification and guidance documentation to gain an understanding of the way the related TSFI is expected to behave.

1562 With an understanding of the expected behaviour, the evaluator examines the test plan to gain an understanding of the testing approach. In most cases, the testing approach will entail a TSFI being stimulated at either the external or internal interfaces and its responses are observed. Externally-visible functionality can be tested directly; however, in cases where functionality is not visible external to the TOE (for example, testing the residual information protection functionality), other means will need to be employed.

14.2.2 Testing vs. alternate approaches to verify the expected behaviour of functionality

1563 In cases where it is impractical or inadequate to test specific functionality (where it provides no externally-visible TSFI), the test plan should identify the alternate approach to verify expected behaviour. It is the evaluator's responsibility to determine the suitability of the alternate approach. However, the following should be considered when assessing the suitability of alternate approaches:

a) an analysis of the implementation representation to determine that the required behaviour should be exhibited by the TOE is an acceptable alternate approach. This could mean a code inspection for a software TOE or perhaps a chip mask inspection for a hardware TOE.

b) it is acceptable to use evidence of developer integration or module testing, even if the claimed assurance requirements do not include availability of lower level descriptions of the TOE components (e.g. Evaluation of sub-activity (ADV_TDS.3)) or implementation (Implementation representation (ADV_IMP)). If evidence of developer integration or module testing is used in verifying the expected behaviour of a security function, care should be given to confirm that the testing evidence reflects the current implementation of the TOE. If the subsystem or modules have been changed since testing occurred, evidence that the changes were tracked and addressed by analysis or further testing will usually be required.

1564 It should be emphasised that supplementing the testing effort with alternate approaches should only be undertaken when both the developer and evaluator determine that there exists no other practical means to test the expected behaviour.

14.2.3 Verifying the adequacy of tests

1565 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. The evaluator must determine that the pre-requisites are complete and appropriate in that they will not bias the observed test results towards the expected test results.

1566 The test steps and expected results specify the actions and parameters to be applied to the TSFI as well as how the expected results should be verified and what they are. The evaluator must determine that the test steps and expected results are consistent with the descriptions of the TSFI in the functional specification. This means that each characteristic of the TSFI behaviour explicitly described in the functional specification should have tests and expected results to verify that behaviour.

1567 The overall aim of this testing activity is to determine that each TSFI has been sufficiently tested against the behavioural claims in the functional specification; at the higher assurance levels, this includes bounds testing and negative testing. The test procedures will provide insight as to how the TSFI have been exercised by the developer during testing. The evaluator will use this information when developing additional tests to independently test the TSF.

14.3 Coverage (ATE_COV)

14.3.1 Evaluation of sub-activity (ATE_COV.1)

14.3.1.1 Objectives

1568 The objective of this sub-activity is to determine whether the developer's test coverage evidence shows correspondence between the tests identified in the test documentation and the interfaces described in the functional specification.

14.3.1.2 Input

1569 The evaluation evidence for this sub-activity is:

a) the functional specification;

b) the test documentation;

c) the test coverage evidence.

14.3.1.3 Application notes

1570 The coverage analysis provided by the developer is required to show the correspondence between the tests provided as evaluation evidence and the functional specification. However, the coverage analysis need not demonstrate that all TSFI have been tested, or that all externally-visible interfaces to the TOE have been tested. Such shortcomings are considered by the evaluator during the independent testing (Evaluation of sub-activity (ATE_IND.2)) sub-activity.

14.3.1.4 Action ATE_COV.1.1E

ATE_COV.1.1C ***The evidence of the test coverage shall show the correspondence between the tests in the test documentation and the interfaces in the functional specification.***

ATE_COV.1-1 The evaluator ***shall examine*** the test coverage evidence to determine that the correspondence between the tests identified in the test documentation and the interfaces described in the functional specification is accurate.

1571 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. In cases where coverage is shown to be poor the evaluator should increase the level of independent testing to compensate.

14.3.2 Evaluation of sub-activity (ATE_COV.2)

14.3.2.1 Objectives

1572 The objective of this sub-activity is to determine whether the testing (as documented) is sufficient to establish that the TSF has been systematically tested against the functional specification.

14.3.2.2 Input

a) the ST;

b) the functional specification;

c) the test documentation;

d) the test coverage analysis.

14.3.2.3 Action ATE_COV.2.1E

ATE_COV.2.1C ***The analysis of the test coverage shall demonstrate the correspondence between the tests in the test documentation and the interfaces in the functional specification.***

ATE_COV.2-1 The evaluator ***shall examine*** the test coverage analysis to determine that the correspondence between the tests in the test documentation and the interfaces in the functional specification is accurate.

1573 A simple cross-table may be sufficient to show test correspondence. The identification of the tests and the interfaces presented in the test coverage analysis has to be unambiguous.

1574 The evaluator is reminded that this does not imply that all tests in the test documentation must map to interfaces in the functional specification.

ATE_COV.2-2 The evaluator ***shall examine*** the test plan to determine that the testing approach for each interface demonstrates the expected behaviour of that interface.

1575 Guidance on this work unit can be found in:

a) 14.2.1, Understanding the expected behaviour of the TOE

b) 14.2.2, Testing vs. alternate approaches to verify the expected behaviour of functionality

ATE_COV.2-3 The evaluator ***shall examine*** the test procedures to determine that the test prerequisites, test steps and expected result(s) adequately test each interface.

1576 Guidance on this work units, as it pertains to the functional specification, can be found in:

a) 14.2.3, Verifying the adequacy of tests

ATE_COV.2-4 The evaluator ***shall examine*** the test coverage analysis to determine that the correspondence between the interfaces in the functional specification and the tests in the test documentation is complete.

1577 All interfaces that are described in the functional specification have to be 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. Incomplete coverage would be evident if an interface was identified in the functional specification and no test was mapped to it.

1578 The evaluator is reminded that this does not imply that all tests in the test documentation must map to interfaces in the functional specification.

14.3.3 Evaluation of sub-activity (ATE_COV.3)

1579 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

14.4 Depth (ATE_DPT)

14.4.1 Evaluation of sub-activity (ATE_DPT.1)

14.4.1.1 Objectives

1580 The objective of this sub-activity is to determine whether the developer has tested the TSF components against the TOE design.

14.4.1.2 Input

a) the ST;

b) the functional specification;

c) the TOE design;

d) the test documentation;

e) the depth of testing analysis.

14.4.1.3 Action ATE_DPT.1.1E

ATE_DPT.1.1C ***The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the interfaces of TOE components in the TOE design.***

ATE_DPT.1-1 The evaluator ***shall examine*** the depth of testing analysis to determine that the correspondence between the tests in the test documentation and the component interfaces in the TOE design is accurate.

1581 A simple cross-table may be sufficient to show test correspondence. The identification of the tests and the interfaces presented in the depth-of coverage analysis has to be unambiguous.

1582 The evaluator is reminded that not all tests in the test documentation must map to an interface in the TOE design.

ATE_DPT.1-2 The evaluator ***shall examine*** the test plan to determine that the testing approach for each interface demonstrates the expected behaviour of that interface.

1583 Guidance on this work unit can be found in:

a) 14.2.1, Understanding the expected behaviour of the TOE

b) 14.2.2, Testing vs. alternate approaches to verify the expected behaviour of functionality

1584 Testing of an interface may be performed directly at that interface, or at the external interfaces, or a combination of both. Whatever strategy is used the evaluator will consider its appropriateness for adequately testing the

interfaces. Specifically the evaluator determines whether testing at the internal interfaces is necessary or whether these internal interfaces can be adequately tested (albeit implicitly) by exercising the external interfaces. This determination is left to the evaluator, as is its justification.

ATE_DPT.1-3 The evaluator ***shall examine*** the test procedures to determine that the test prerequisites, test steps and expected result(s) adequately test each interface.

1585 Guidance on this work units, as it pertains to the TOE design, can be found in:

a) 14.2.3, Verifying the adequacy of tests

ATE_DPT.1.2C ***The analysis of the depth of testing shall demonstrate that the correspondence between the component interfaces in the TOE design and the tests in the test documentation is complete.***

ATE_DPT.1-4 The evaluator ***shall examine*** the depth of testing analysis to determine that the correspondence between the component interfaces in the TOE design and the tests in the test documentation is complete.

1586 All component interfaces that are described in the TOE design have to be present in the depth of testing analysis and mapped to tests in order for completeness to be claimed, although exhaustive specification testing of component interfaces is not required. Incomplete depth of testing would be evident if a component interface was identified in the TOE design and no tests could be attributed to it.

1587 The evaluator is reminded that this does not imply that all tests in the test documentation must map to component interfaces in the TOE design.

14.4.2 Evaluation of sub-activity (ATE_DPT.2)

14.4.2.1 Objectives

1588 The objective of this sub-activity is to determine whether the developer has tested the TSF components and modules against the TOE design.

14.4.2.2 Input

a) the ST;

b) the functional specification;

c) the TOE design;

d) the test documentation;

e) the depth of testing analysis.

14.4.2.3 Action ATE_DPT.2.1E

ATE_DPT.2.1C ***The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test documentation and the interfaces of TOE components and modules in the TOE design.***

ATE_DPT.2-1 The evaluator ***shall examine*** the depth of testing analysis to determine that the correspondence between the tests in the test documentation and the component interfaces in the TOE design is accurate.

1589 A simple cross-table may be sufficient to show test correspondence. The identification of the tests and the interfaces presented in the depth-of coverage analysis has to be unambiguous.

1590 The evaluator is reminded that not all tests in the test documentation must map to an interface in the TOE design.

ATE_DPT.2-2 The evaluator ***shall examine*** the depth of testing analysis to determine that the correspondence between the tests in the test documentation and the modules interfaces in the TOE design is accurate.

1591 A simple cross-table may be sufficient to show test correspondence. The identification of the tests and the interfaces presented in the depth-of coverage analysis has to be unambiguous.

1592 The evaluator is reminded that not all tests in the test documentation must map to an interface in the TOE design.

ATE_DPT.2-3 The evaluator ***shall examine*** the test plan to determine that the testing approach for each interface demonstrates the expected behaviour of that interface.

1593 Guidance on this work unit can be found in:

a) 14.2.1, Understanding the expected behaviour of the TOE

b) 14.2.2, Testing vs. alternate approaches to verify the expected behaviour of functionality

1594 Testing of an interface may be performed directly at that interface, or at the external interfaces, or a combination of both. Whatever strategy is used the evaluator will consider its appropriateness for adequately testing the interfaces. Specifically the evaluator determines whether testing at the internal interfaces is necessary or whether these internal interfaces can be adequately tested (albeit implicitly) by exercising the external interfaces. This determination is left to the evaluator, as is its justification.

ATE_DPT.2-4 The evaluator ***shall examine*** the test procedures to determine that the test prerequisites, test steps and expected result(s) adequately test each interface.

1595 Guidance on this work units, as it pertains to the TOE design, can be found in:

a) 14.2.3, Verifying the adequacy of tests

ATE_DPT.2.2C ***The analysis of the depth of testing shall demonstrate that the correspondence between the component interfaces in the TOE design and the tests in the test documentation is complete.***

ATE_DPT.2-5 The evaluator ***shall examine*** the depth of testing analysis to determine that the correspondence between the component interfaces in the TOE design and the tests in the test documentation is complete.

1596 All component interfaces that are described in the TOE design have to be present in the depth of testing analysis and mapped to tests in order for completeness to be claimed, although exhaustive specification testing of component interfaces is not required. Incomplete depth of testing would be evident if a component interface was identified in the TOE design and no tests could be attributed to it.

1597 The evaluator is reminded that this does not imply that all tests in the test documentation must map to component interfaces in the TOE design.

ATE_DPT.2.3C ***The analysis of the depth of testing shall demonstrate that the correspondence between the module interfaces in the TOE design and the tests in the test documentation is complete.***

ATE_DPT.2-6 The evaluator ***shall examine*** the depth of testing analysis to determine that the correspondence between the module interfaces in the TOE design and the tests in the test documentation is complete.

1598 All module interfaces that are described in the TOE design have to be present in the depth of testing analysis and mapped to tests in order for completeness to be claimed, although exhaustive specification testing of module interfaces is not required. Incomplete depth of testing would be evident if a module interface was identified in the TOE design and no tests could be attributed to it.

1599 The evaluator is reminded that this does not imply that all tests in the test documentation must map to module interfaces in the TOE design.

14.4.3 Evaluation of sub-activity (ATE_DPT.3)

1600 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

14.5 Functional tests (ATE_FUN)

14.5.1 Evaluation of sub-activity (ATE_FUN.1)

14.5.1.1 Objectives

1601 The objective of this sub-activity is to determine whether the developer correctly performed and documented the tests in the test documentation.

14.5.1.2 Input

1602 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the test documentation;

14.5.1.3 Application notes

1603 The extent to which the test documentation is required to cover the TSF is dependent upon the coverage assurance component.

1604 For the developer tests provided, the evaluator determines whether the tests are repeatable, and the extent to which the developer's tests can be used for the evaluator's independent testing effort. Any TSFI for which the developer's test results indicate that it might not perform as specified should be tested independently by the evaluator to determine whether or not it does.

14.5.1.4 Action ATE_FUN.1.1E

ATE_FUN.1.1C ***The test documentation shall consist of test plans, expected test results and actual test results.***

ATE_FUN.1-1 The evaluator ***shall check*** that the test documentation includes test plans, expected test results and actual test results.

ATE_FUN.1.2C ***The test plans shall identify the tests to be performed and describe the scenarios for performing each test. These scenarios shall include any ordering dependencies on the results of other tests.***

ATE_FUN.1-2 The evaluator ***shall check*** that the test plan identifies the tests to be performed.

1605 The evaluator may wish to employ a sampling strategy when performing this work unit.

ATE_FUN.1-3 The evaluator ***shall examine*** the test plan to determine that it describes the scenarios for performing each test.

1606 The evaluator determines that the test plan provides 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.

1607 The evaluator also determines that the test plan provides information about how to execute the test: any necessary automated set-up procedures (and whether they require privilege to run), inputs to be applied, how these inputs are applied, how output is obtained, any automated clean-up procedures (and whether they require privilege to run), etc. This information should be detailed enough to ensure that the test is reproducible.

1608 The evaluator may wish to employ a sampling strategy when performing this work unit.

ATE_FUN.1-4 The evaluator ***shall examine*** the test plan to determine that the TOE test configuration is consistent with the ST.

1609 The TOE referred to in the developer's test plan should have the same unique reference as established by the CM capabilities (ALC_CMC) sub-activities and identified in the ST introduction.

1610 It is possible for the ST to specify more than one configuration for evaluation. The evaluator verifies that all test configurations identified in the developer test documentation are consistent with the ST. For example, the ST might define configuration options that must be set, which could have an impact upon the TOE boundary by including or excluding additional portions. The evaluator verifies that all such variations of the TOE boundary are considered.

1611 The evaluator should 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.

1612 The evaluator may wish to employ a sampling strategy when performing this work unit.

1613 If this work unit is applied to a composed TOE, the following will apply. In the instances that the component TOE under evaluation depends on other components in the operational environment to support their operation, the developer may wish to consider using the other component(s) that will be used in the composed TOE to fulfil the requirements of the operational environment as one of the test configurations. This will reduce the amount an additional testing that will be required for the composed TOE evaluation.

ATE_FUN.1-5 The evaluator ***shall examine*** the test plans to determine that sufficient instructions are provided for any ordering dependencies.

1614 Some steps may have to be performed to establish initial conditions. For example, user accounts need to be added before they can be deleted. An example of ordering dependencies on the results of other tests is the need to test an audit-related interface before relying on it to produce audit records for testing an access control-related interface. Another example of an ordering dependency would be where one test case generates a file of data to be used as input for another test case.

1615 The evaluator may wish to employ a sampling strategy when performing this work unit.

ATE_FUN.1.3C ***The expected test results shall show the anticipated outputs from a successful execution of the tests.***

ATE_FUN.1-6 The evaluator ***shall examine*** the test documentation to determine that all expected tests results are included.

1616 The expected test results are needed to determine whether or not a test has been successfully performed. Expected test results are sufficient if they are unambiguous and consistent with expected behaviour given the testing approach.

1617 The evaluator may wish to employ a sampling strategy when performing this work unit.

ATE_FUN.1.4C ***The actual test results shall be consistent with the expected test results.***

ATE_FUN.1-7 The evaluator ***shall check*** that the expected test results in the test documentation are consistent with the actual test results in the test documentation.

1618 A comparison of the actual and expected test results provided by the developer will reveal any inconsistencies between the results. It may be that a direct comparison of actual results cannot be made until some data reduction or synthesis has been first performed. In such cases, the developer's test documentation should describe the process to reduce or synthesise the actual data.

1619 For example, the developer may need to test the contents of a message buffer after a network connection has occurred to determine the contents of the buffer. The message buffer will contain a binary number. This binary number would have to be converted to another form of data representation in order to make the test more meaningful. The conversion of this binary representation of data into a higher-level representation will have to be described by the developer in enough detail to allow an evaluator to perform the conversion process (i.e. synchronous or asynchronous transmission, number of stop bits, parity, etc.).

1620 It should be noted that the description of the process used to reduce or synthesise the actual data is used by the evaluator not to actually perform the necessary modification but to assess whether this process is correct. It is up

to the developer to transform the expected test results into a format that allows an easy comparison with the actual test results.

1621 The evaluator may wish to employ a sampling strategy when performing this work unit.

ATE_FUN.1-8 The evaluator ***shall report*** the developer testing effort, outlining the testing approach, configuration, depth and results.

1622 The developer testing information recorded in the ETR allows the evaluator to convey the overall testing approach and effort expended on the testing of the TOE by the developer. The intent of providing this information is to give a meaningful overview of the developer testing effort. It is not intended that the information regarding developer testing in the ETR be an exact reproduction of specific test steps or results of individual tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the developer's testing approach, amount of testing performed, TOE test configurations, and the overall results of the developer testing.

1623 Information that would typically be found in the ETR section regarding the developer testing effort is:

a) TOE test configurations. The particular configurations of the TOE that were tested, including whether any privileged code was required to set up the test or clean up afterwards;

b) testing approach. An account of the overall developer testing strategy employed;

c) testing results. A description of the overall developer testing results.

1624 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the developer testing effort.

14.5.2 Evaluation of sub-activity (ATE_FUN.2)

1625 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

14.6 Independent testing (ATE_IND)

14.6.1 Evaluation of sub-activity (ATE_IND.1)

14.6.1.1 Objectives

1626 The goal of this activity is to determine, by independently testing a subset of the TSFI, whether the TOE behaves as specified in the functional specification and guidance documentation.

14.6.1.2 Input

1627 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the operational user guidance;

d) the preparative user guidance;

e) the TOE suitable for testing.

14.6.1.3 Action ATE_IND.1.1E

ATE_IND.1.1C ***The TOE shall be suitable for testing.***

ATE_IND.1-1 The evaluator ***shall examine*** the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

1628 The TOE provided by the developer and identified in the test plan should have the same unique reference as established by the CM capabilities (ALC_CMC) sub-activities and identified in the ST introduction.

1629 It is possible for the ST to specify more than one configuration for evaluation. The TOE may comprise a number of distinct hardware and software entities that need to be tested in accordance with the ST. The evaluator verifies that all test configurations are consistent with the ST.

1630 The evaluator should 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.

1631 If any test resources are used (e.g. meters, analysers) it will be the evaluator's responsibility to ensure that these resources are calibrated correctly.

ATE_IND.1-2 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state.

1632 It is possible for the evaluator to determine the state of the TOE in a number of ways. For example, previous successful completion of the Evaluation of sub-activity (AGD_PRE.1) sub-activity will satisfy this work unit if the evaluator still has confidence that the TOE being used for testing was installed properly and is in a known state. If this is not the case, then the evaluator should follow the developer's procedures to install and start up the TOE, using the supplied guidance only.

1633 If the evaluator has to perform the installation procedures because the TOE is in an unknown state, this work unit when successfully completed could satisfy work unit AGD_PRE.1-5.

14.6.1.4 Action ATE_IND.1.2E

ATE_IND.1-3 The evaluator ***shall devise*** a test subset.

1634 The evaluator selects a test subset and testing strategy that is appropriate for the TOE. One extreme testing strategy would be to have the test subset contain as many interfaces as possible tested with little rigour. Another testing strategy would be to have the test subset contain a few interfaces based on their perceived relevance and rigorously test these interfaces.

1635 Typically the testing approach taken by the evaluator should fall somewhere between these two extremes. The evaluator should exercise most of the interfaces using at least one test, but testing need not demonstrate exhaustive specification testing.

1636 The evaluator, when selecting the subset of the interfaces to be tested, should consider the following factors:

a) The number of interfaces from which to draw upon for the test subset. Where the TSF includes only a small number of relatively simple interfaces, it may be practical to rigorously test all of the interfaces. In other cases this may not be cost-effective, and sampling is required.

b) Maintaining a balance of evaluation activities. The evaluator effort expended on the test activity should be commensurate with that expended on any other evaluation activity.

1637 The evaluator selects the interfaces to compose the subset. This selection will depend on a number of factors, and consideration of these factors may also influence the choice of test subset size:

a) Significance of interfaces. Those interfaces more significant than others should be included in the test subset. One major factor of “significance” is the security-relevance (SFR-enforcing interfaces would be more significant than SFR-supporting interfaces, which are more significant than SFR-non-interfering interfaces; see CC Part 3 section 15.2.2.1, Security-Relevance of the Interfaces). The other major factors of “significance” is the number of SFRs mapping to

this interface (as determined when identifying the correspondence between levels of abstraction in 11).

b) Complexity of the interface. Complex interfaces may require complex tests that impose onerous requirements on the developer or evaluator, which will not be conducive to cost-effective evaluations. Conversely, they are a likely area to find errors and are good candidates for the subset. The evaluator will need to strike a balance between these considerations.

c) Implicit testing. Testing some interfaces may often implicitly test other interfaces, and their inclusion in the subset may maximise the number of interfaces tested (albeit implicitly). Certain interfaces will typically be used to provide a variety of security functionality, and will tend to be the target of an effective testing approach.

d) Types of interfaces (e.g. programmatic, command-line, protocol). The evaluator should consider including tests for all different types of interfaces that the TOE supports.

e) Interfaces that give rise to features that are innovative or unusual. Where the TOE contains innovative or unusual features, which may feature strongly in marketing literature and guidance documents, the corresponding interfaces should be strong candidates for testing.

1638 This guidance articulates factors to consider during the selection process of an appropriate test subset, but these are by no means exhaustive.

ATE_IND.1-4 The evaluator ***shall produce*** test documentation for the test subset that is sufficiently detailed to enable the tests to be reproducible.

1639 With an understanding of the expected behaviour of the TSF, from the ST and the functional specification, the evaluator has to determine the most feasible way to test the interface. Specifically the evaluator considers:

a) the approach that will be used, for instance, whether an external interface will be tested, or an internal interface using a test harness, or will an alternate test approach be employed (e.g. in exceptional circumstances, a code inspection, if the implementation representation is available);

b) the interface(s) that will be used to test and observe responses;

c) the initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

d) special test equipment that will be required to either stimulate an interface (e.g. packet generators) or make observations of an interface (e.g. network analysers).

1640 The evaluator may find it practical to test each interface using a series of test cases, where each test case will test a very specific aspect of expected behaviour.

1641 The evaluator's test documentation should specify the derivation of each test, tracing it back to the relevant interface(s).

ATE_IND.1-5 The evaluator ***shall conduct*** testing.

1642 The evaluator uses the test documentation developed as a basis for executing tests on the TOE. The test documentation is used as a basis for testing but this does not preclude the evaluator from performing additional ad hoc tests. The evaluator may devise new tests based on behaviour of the TOE discovered during testing. These new tests are recorded in the test documentation.

ATE_IND.1-6 The evaluator ***shall record*** the following information about the tests that compose the test subset:

a) identification of the interface behaviour to be tested;

b) instructions to connect and setup all required test equipment as required to conduct the test;

c) instructions to establish all prerequisite test conditions;

d) instructions to stimulate the interface;

e) instructions for observing the behaviour of the interface;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE;

h) actual test results.

1643 The level of detail should be such that another evaluator could repeat the tests and obtain an equivalent result. While some specific details of the test results may be different (e.g. time and date fields in an audit record) the overall result should be identical.

1644 There may be instances when it is unnecessary to provide all the information presented in this work unit (e.g. the actual test results of a test may not require any analysis before a comparison between the expected results can be made). The determination to omit this information is left to the evaluator, as is the justification.

ATE_IND.1-7 The evaluator ***shall check*** that all actual test results are consistent with the expected test results.

1645 Any differences in the actual and expected test results may indicate that the TOE does not perform as specified or that the evaluator test documentation may be incorrect. Unexpected actual results may require corrective maintenance to the TOE or test documentation and perhaps require re-running of impacted tests and modifying the test sample size and composition. This determination is left to the evaluator, as is its justification.

ATE_IND.1-8 The evaluator ***shall report*** in the ETR the evaluator testing effort, outlining the testing approach, configuration, depth and results.

1646 The evaluator testing information reported in the ETR allows the evaluator to convey the overall testing approach and effort expended on the testing activity during the evaluation. The intent of providing this information is to give a meaningful overview of the testing effort. It is not intended that the information regarding testing in the ETR be an exact reproduction of specific test instructions or results of individual tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the testing approach chosen, amount of testing performed, TOE test configurations, and the overall results of the testing activity.

1647 Information that would typically be found in the ETR section regarding the evaluator testing effort is:

a) TOE test configurations. The particular configurations of the TOE that were tested;

b) subset size chosen. The amount of interfaces that were tested during the evaluation and a justification for the size;

c) selection criteria for the interfaces that compose the subset. Brief statements about the factors considered when selecting interfaces for inclusion in the subset;

d) interfaces tested. A brief listing of the interfaces that merited inclusion in the subset;

e) verdict for the activity. The overall judgement on the results of testing during the evaluation.

1648 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the testing the evaluator performed during the evaluation.

14.6.2 Evaluation of sub-activity (ATE_IND.2)

14.6.2.1 Objectives

1649 The goal of this activity is to determine, by independently testing a subset of the TSF, whether the TOE behaves as specified in the design documentation, and to gain confidence in the developer's test results by performing a sample of the developer's tests.

14.6.2.2 Input

1650 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design description;

d) the operational user guidance;

e) the preparative user guidance;

f) the test documentation;

g) the TOE suitable for testing.

14.6.2.3 Action ATE_IND.2.1E

ATE_IND.2.1C ***The TOE shall be suitable for testing.***

ATE_IND.2-1 The evaluator ***shall examine*** the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

1651 The TOE provided by the developer and identified in the test plan should have the same unique reference as established by the CM capabilities (ALC_CMC) sub-activities and identified in the ST introduction.

1652 It is possible for the ST to specify more than one configuration for evaluation. The TOE may comprise a number of distinct hardware and software entities that need to be tested in accordance with the ST. The evaluator verifies that all test configurations are consistent with the ST.

1653 The evaluator should 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.

1654 If any test resources are used (e.g. meters, analysers) it will be the evaluator's responsibility to ensure that these resources are calibrated correctly.

ATE_IND.2-2 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state

1655 It is possible for the evaluator to determine the state of the TOE in a number of ways. For example, previous successful completion of the Evaluation of sub-activity (AGD_PRE.1) sub-activity will satisfy this work unit if the evaluator still has confidence that the TOE being used for testing was installed properly and is in a known state. If this is not the case, then the

evaluator should follow the developer's procedures to install and start up the TOE, using the supplied guidance only.

1656 If the evaluator has to perform the installation procedures because the TOE is in an unknown state, this work unit when successfully completed could satisfy work unit AGD_PRE.1-5.

ATE_IND.2.2C ***The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.***

ATE_IND.2-3 The evaluator ***shall examine*** the set of resources provided by the developer to determine that they are equivalent to the set of resources used by the developer to functionally test the TSF

1657 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.

14.6.2.4 Action ATE_IND.2.2E

ATE_IND.2-4 The evaluator ***shall conduct*** testing using a sample of tests found in the developer test plan and procedures.

1658 The overall aim of this work unit is to perform a sufficient number of the developer tests to confirm the validity of the developer's test results. The evaluator has to decide on the size of the sample, and the developer tests that will compose the sample (see A.2).

1659 All the developer tests can be traced back to specific interfaces. Therefore, the factors to consider in the selection of the tests to compose the sample are similar to those listed for subset selection in work-unit ATE_IND.2-6. Additionally, the evaluator may wish to employ a random sampling method to select developer tests to include in the sample.

ATE_IND.2-5 The evaluator ***shall check*** that all the actual test results are consistent with the expected test results.

1660 Inconsistencies between the developer's expected test results and actual test results will compel the evaluator to resolve the discrepancies. Inconsistencies encountered by the evaluator could be resolved by a valid explanation and resolution of the inconsistencies by the developer.

1661 If a satisfactory explanation or resolution can not be reached, the evaluator's confidence in the developer's test results may be lessened and it may even be necessary for the evaluator to increase the sample size, to regain confidence in the developer testing. If the increase in sample size does not satisfy the evaluator's concerns, it may be necessary to repeat the entire set of developer's tests. Ultimately, to the extent that the subset identified in work unit ATE_IND.2-4 is adequately tested, deficiencies with the developer's tests need to result in either corrective action to the developer's tests or in the production of new tests by the evaluator.

14.6.2.5 Action ATE_IND.2.3E

ATE_IND.2-6 The evaluator ***shall devise*** a test subset.

1662 The evaluator selects a test subset and testing strategy that is appropriate for the TOE. One extreme testing strategy would be to have the test subset contain as many interfaces as possible tested with little rigour. Another testing strategy would be to have the test subset contain a few interfaces based on their perceived relevance and rigorously test these interfaces.

1663 Typically the testing approach taken by the evaluator should fall somewhere between these two extremes. The evaluator should exercise most of the interfaces using at least one test, but testing need not demonstrate exhaustive specification testing.

1664 The evaluator, when selecting the subset of the interfaces to be tested, should consider the following factors:

a) The developer test evidence. The developer test evidence consists of: the test documentation, the available test coverage analysis, and the available depth of testing analysis. The developer test evidence will provide insight as to how the TSF has been exercised by the developer during testing. The evaluator applies this information when developing new tests to independently test the TOE. Specifically the evaluator should consider:

   1) augmentation of developer testing for interfaces. The evaluator may wish to perform more of the same type of tests by varying parameters to more rigorously test the interface.

   2) supplementation of developer testing strategy for interfaces. The evaluator may wish to vary the testing approach of a specific interface by testing it using another test strategy.

b) The number of interfaces from which to draw upon for the test subset. Where the TSF includes only a small number of relatively simple interfaces, it may be practical to rigorously test all of them. In other cases this may not be cost-effective, and sampling is required.

c) Maintaining a balance of evaluation activities. The evaluator effort expended on the test activity should be commensurate with that expended on any other evaluation activity.

1665 The evaluator selects the interfaces to compose the subset. This selection will depend on a number of factors, and consideration of these factors may also influence the choice of test subset size:

a) Rigour of developer testing of the interfaces. Those interfaces that the evaluator determines require additional testing should be included in the test subset.

b) Developer test results. If the results of developer tests cause the evaluator to doubt that an interface is not properly implemented, then the evaluator should include such interfaces in the test subset.

c) Significance of interfaces. Those interfaces more significant than others should be included in the test subset. One major factor of "significance" is the security-relevance (SFR-enforcing interfaces would be more significant than SFR-supporting interfaces, which are more significant than SFR-non-interfering interfaces; see CC Part 3 section 15.2.2.1, Security-Relevance of the Interfaces). The other major factors of "significance" is the number of SFRs mapping to this interface (as determined when identifying the correspondence between levels of abstraction in 11).

d) Complexity of interfaces. Interfaces that require complex implementation may require complex tests that impose onerous requirements on the developer or evaluator, which will not be conducive to cost-effective evaluations. Conversely, they are a likely area to find errors and are good candidates for the subset. The evaluator will need to strike a balance between these considerations.

e) Implicit testing. Testing some interfaces may often implicitly test other interfaces, and their inclusion in the subset may maximise the number of interfaces tested (albeit implicitly). Certain interfaces will typically be used to provide a variety of security functionality, and will tend to be the target of an effective testing approach.

f) Types of interfaces (e.g. programmatic, command-line, protocol). The evaluator should consider including tests for all different types of interfaces that the TOE supports.

g) Interfaces that give rise to features that are innovative or unusual. Where the TOE contains innovative or unusual features, which may feature strongly in marketing literature and guidance documents, the corresponding interfaces should be strong candidates for testing.

1666 This guidance articulates factors to consider during the selection process of an appropriate test subset, but these are by no means exhaustive.

ATE_IND.2-7 The evaluator ***shall produce*** test documentation for the test subset that is sufficiently detailed to enable the tests to be reproducible.

1667 With an understanding of the expected behaviour of the TSF, from the ST, the functional specification, and the TOE design description, the evaluator has to determine the most feasible way to test the interface. Specifically the evaluator considers:

a) the approach that will be used, for instance, whether an external interface will be tested, or an internal interface using a test harness, or will an alternate test approach be employed (e.g. in exceptional circumstances, a code inspection);

b) the interface(s) that will be used to test and observe responses;

c) the initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

d) special test equipment that will be required to either stimulate an interface (e.g. packet generators) or make observations of an interface (e.g. network analysers).

1668 The evaluator may find it practical to test each interface using a series of test cases, where each test case will test a very specific aspect of expected behaviour of that interface.

1669 The evaluator's test documentation should specify the derivation of each test, tracing it back to the relevant interface(s).

ATE_IND.2-8 The evaluator ***shall conduct*** testing.

1670 The evaluator uses the test documentation developed as a basis for executing tests on the TOE. The test documentation is used as a basis for testing but this does not preclude the evaluator from performing additional ad hoc tests. The evaluator may devise new tests based on behaviour of the TOE discovered during testing. These new tests are recorded in the test documentation.

ATE_IND.2-9 The evaluator ***shall record*** the following information about the tests that compose the test subset:

a) identification of the interface behaviour to be tested;

b) instructions to connect and setup all required test equipment as required to conduct the test;

c) instructions to establish all prerequisite test conditions;

d) instructions to stimulate the interface;

e) instructions for observing the interface;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE;

h) actual test results.

1671 The level of detail should be such that another evaluator could repeat the tests and obtain an equivalent result. While some specific details of the test

results may be different (e.g. time and date fields in an audit record) the overall result should be identical.

1672 There may be instances when it is unnecessary to provide all the information presented in this work unit (e.g. the actual test results of a test may not require any analysis before a comparison between the expected results can be made). The determination to omit this information is left to the evaluator, as is the justification.

ATE_IND.2-10 The evaluator ***shall check*** that all actual test results are consistent with the expected test results.

1673 Any differences in the actual and expected test results may indicate that the TOE does not perform as specified or that the evaluator test documentation may be incorrect. Unexpected actual results may require corrective maintenance to the TOE or test documentation and perhaps require re-running of impacted tests and modifying the test sample size and composition. This determination is left to the evaluator, as is its justification.

ATE_IND.2-11 The evaluator ***shall report*** in the ETR the evaluator testing effort, outlining the testing approach, configuration, depth and results.

1674 The evaluator testing information reported in the ETR allows the evaluator to convey the overall testing approach and effort expended on the testing activity during the evaluation. The intent of providing this information is to give a meaningful overview of the testing effort. It is not intended that the information regarding testing in the ETR be an exact reproduction of specific test instructions or results of individual tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the testing approach chosen, amount of evaluator testing performed, amount of developer tests performed, TOE test configurations, and the overall results of the testing activity.

1675 Information that would typically be found in the ETR section regarding the evaluator testing effort is:

a) TOE test configurations. The particular configurations of the TOE that were tested.

b) subset size chosen. The amount of interfaces that were tested during the evaluation and a justification for the size.

c) selection criteria for the interfaces that compose the subset. Brief statements about the factors considered when selecting interfaces for inclusion in the subset.

d) Interfaces tested. A brief listing of the interfaces that merited inclusion in the subset.

e) developer tests performed. The amount of developer tests performed and a brief description of the criteria used to select the tests.

f) verdict for the activity. The overall judgement on the results of testing during the evaluation.

1676 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the testing the evaluator performed during the evaluation.

14.6.3 Evaluation of sub-activity (ATE_IND.3)

1677 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

15 Class AVA: Vulnerability assessment

15.1 Introduction

1678 The purpose of the vulnerability assessment activity is to determine the exploitability of flaws or weaknesses in the TOE in the operational environment. This determination is based upon analysis of the evaluation evidence and a search of publicly available material by the evaluator and is supported by evaluator penetration testing.

15.2 Vulnerability analysis (AVA_VAN)

15.2.1 Evaluation of sub-activity (AVA_VAN.1)

15.2.1.1 Objectives

1679 The objective of this sub-activity is to determine whether the TOE, in its operational environment, has easily identifiable exploitable vulnerabilities.

15.2.1.2 Input

1680 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the guidance documentation;

d) the TOE suitable for testing;

e) information publicly available to support the identification of potential vulnerabilities.

1681 Other input for this sub-activity is:

a) current information regarding potential vulnerabilities (e.g. from an overseer).

15.2.1.3 Application notes

1682 The evaluator should consider performing additional tests as a result of potential vulnerabilities encountered during the conduct of other parts of the evaluation.

1683 The use of the term guidance in this sub-activity refers to the operational guidance and the preparative guidance.

1684 Potential vulnerabilities may be in information that is publicly available, or not, and may require skill to exploit, or not. These two aspects are related, but are distinct. It should not be assumed that, simply because a potential vulnerability is identifiable from information that is publicly available, it can be easily exploited.

15.2.1.4 Action AVA_VAN.1.1E

AVA_VAN.1.1C ***The TOE shall be suitable for testing.***

AVA_VAN.1-1 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state.

1685 The known state is to be one that is consistent with the ST. It is possible for the ST to specify more than one configuration for evaluation. The TOE may

be composed of a number of distinct hardware and software implementations that need to be tested in accordance with the ST. The evaluator verifies that there are test configurations consistent with each evaluated configuration described in the ST.

1686 The evaluator should consider the security objectives for the operational environment described in the ST that may apply to the test environment. There may be some security objectives in the ST that do not apply to the test environment. For example, an assumption about user clearances that is used to restrict access to the TOE may not apply; however, an assumption about a single point of connection to a network would apply.

1687 If the assurance package includes a component from the ATE_IND family, then the evaluator may refer to the result of the work unit ATE_IND.*-2 to demonstrate that this is satisfied.

1688 It is possible for the evaluator to determine the state of the TOE in a number of ways. For example, previous successful completion of the AGD_PRE.1 sub-activity will satisfy this work unit if the evaluator still has confidence that the TOE being used for testing was installed properly and is in a known state. If this is not the case, then the evaluator should follow the developer's procedures to install, generate and start up the TOE, using the supplied guidance only.

1689 If the evaluator has to perform the installation procedures because the TOE is in an unknown state, this work unit when successfully completed could satisfy work unit AGD_PRE.1-5

1690 If any test resources are used (e.g. meters, analysers) it will be the evaluator's responsibility to ensure that these resources are calibrated correctly.

15.2.1.5 Action AVA_VAN.1.2E

AVA_VAN.1-2 The evaluator examines the sources of information publicly available to support the identification of potential vulnerabilities in the TOE. There are many sources of publicly available information which the evaluator should consider using:

a) world wide web;

b) specialist publications (magazines, books);

c) research papers;

d) conference proceedings.

1691 The evaluator should not constrain their consideration of publicly available information to the above, but should consider any other relevant information available.

1692 While examining the evidence provided the evaluator will use the information in the public domain to further search for potential

vulnerabilities. Where the evaluators have identified areas of concern, the evaluator should consider information publicly available that relate to those areas of concern.

1693 The availability of information that may be readily available to an attacker that helps to identify and facilitate attacks effectively operates to substantially enhance the attack potential of a given attacker. The accessibility of vulnerability information and sophisticated attack tools on the Internet makes it more likely that this information will be used in attempts to identify potential vulnerabilities in the TOE and exploit them. Modern search tools make such information easily available to the evaluator, and the determination of resistance to published potential vulnerabilities and well known generic attacks can be achieved in a cost-effective manner.

1694 The search of the information publicly available should be focused on those sources that refer specifically to the product from which the TOE is derived. The extensiveness of this search should consider the following factors: TOE type, evaluator experience in this TOE type, expected attack potential and the level of ADV evidence available.

1695 The identification process is iterative, where the identification of one potential vulnerability may lead to identifying another area of concern that requires further investigation.

1696 The evaluator will report what actions were taken to identify potential vulnerabilities in the evidence. However, in this type of search, the evaluator may not be able to describe the steps in identifying potential vulnerabilities before the outset of the examination, as the approach may evolve as a result of findings during the search.

1697 The evaluator will report the evidence examined in completing the search for potential vulnerabilities. This selection of evidence may be derived from those areas of concern identified by the evaluator, linked to the evidence the attacker is assumed to be able to obtain, or according to another rationale provided by the evaluator.

AVA_VAN.1-3 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.

1698 It may be identified that no further consideration of the potential vulnerability is required if for example the evaluator identifies that measures in the operational environment, either IT or non-IT, prevent exploitation of the potential vulnerability in that operational environment. For instance, restricting physical access to the TOE to authorised users only may effectively render a potential vulnerability to tampering unexploitable.

1699 The evaluator records any reasons for exclusion of potential vulnerabilities from further consideration if the evaluator determines that the potential vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

1700 A list of potential vulnerabilities applicable to the TOE in its operational environment, which can be used as an input into penetration testing activities, shall be reported in the ETR by the evaluators.

15.2.1.6 Action AVA_VAN.1.3E

AVA_VAN.1-4 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.

1701 The evaluator prepares for penetration testing as necessary to determine the susceptibility of the TOE, in its operational environment, to the potential vulnerabilities identified during the search of the sources of information publicly available. The evaluator should have access to current information (e.g. from the overseer) regarding known potential vulnerabilities that may not have been considered by the evaluator, and may also have encountered potential vulnerabilities as a result of performing other evaluation activities.

1702 The evaluator will probably find it practical to carry out penetration test using a series of test cases, where each test case will test for a specific potential vulnerability.

1703 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required a basic attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers a potential vulnerability that is beyond basic attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.1-5 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include:

a) identification of the potential vulnerability the TOE is being tested for;

b) instructions to connect and setup all required test equipment as required to conduct the penetration test;

c) instructions to establish all penetration test prerequisite initial conditions;

d) instructions to stimulate the TSF;

e) instructions for observing the behaviour of the TSF;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE.

1704 The evaluator prepares for penetration testing based on the list of potential vulnerabilities identified during the search of the public domain.

1705 The evaluator is not expected to determine the exploitability for potential vulnerabilities beyond those for which a basic attack potential is required to effect an attack. However, as a result of evaluation expertise, the evaluator may discover a potential vulnerability that is exploitable only by an attacker with greater than basic attack potential. Such vulnerabilities are to be reported in the ETR as residual vulnerabilities.

1706 With an understanding of the potential vulnerability, the evaluator determines the most feasible way to test for the TOE's susceptibility. Specifically the evaluator considers:

a) the TSFI or other TOE interface that will be used to stimulate the TSF and observe responses (It is possible that the evaluator will need to use an interface to the TOE other than the TSFI to demonstrate properties of the TSF such as those described in the architectural description (as required by ADV_ARC). It should the noted, that although these TOE interfaces provide a means of testing the TSF properties, they are not the subject of the test.);

b) initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

c) special test equipment that will be required to either stimulate a TSFI or make observations of a TSFI (although it is unlikely that specialist equipment would be required to exploit a potential vulnerability assuming a basic attack potential);

d) whether theoretical analysis should replace physical testing, particularly relevant where the results of an initial test can be extrapolated to demonstrate that repeated attempts of an attack are likely to succeed after a given number of attempts.

1707 The evaluator will probably find it practical to carry out penetration testing using a series of test cases, where each test case will test for a specific potential vulnerability.

1708 The intent of specifying this level of detail in the test documentation is to allow another evaluator to repeat the tests and obtain an equivalent result.

AVA_VAN.1-6 The evaluator ***shall conduct*** penetration testing.

1709 The evaluator uses the penetration test documentation resulting from work unit AVA_VAN.1-5 as a basis for executing penetration tests on the TOE, but this does not preclude the evaluator from performing additional ad hoc penetration tests. If required, the evaluator may devise ad hoc tests as a result of information learnt during penetration testing that, if performed by the evaluator, are to be recorded in the penetration test documentation. Such tests

may be required to follow up unexpected results or observations, or to investigate potential vulnerabilities suggested to the evaluator during the pre-planned testing.

1710 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required a basic attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers a potential vulnerability that is beyond basic attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.1-7 The evaluator ***shall record*** the actual results of the penetration tests.

1711 While some specific details of the actual test results may be different from those expected (e.g. time and date fields in an audit record) the overall result should be identical. Any unexpected test results should be investigated. The impact on the evaluation should be stated and justified.

AVA_VAN.1-8 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.

1712 The penetration testing information reported in the ETR allows the evaluator to convey the overall penetration testing approach and effort expended on this sub-activity. The intent of providing this information is to give a meaningful overview of the evaluator's penetration testing effort. It is not intended that the information regarding penetration testing in the ETR be an exact reproduction of specific test steps or results of individual penetration tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the penetration testing approach chosen, amount of penetration testing performed, TOE test configurations, and the overall results of the penetration testing activity.

1713 Information that would typically be found in the ETR section regarding evaluator penetration testing efforts is:

a) TOE test configurations. The particular configurations of the TOE that were penetration tested;

b) TSFI penetration tested. A brief listing of the TSFI and other TOE interfaces that were the focus of the penetration testing;

c) verdict for the sub-activity. The overall judgement on the results of penetration testing.

1714 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the penetration testing the evaluator performed during the evaluation.

AVA_VAN.1-9 The evaluator ***shall examine*** the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing a basic attack potential.

1715 If the results reveal that the TOE, in its operational environment, has vulnerabilities exploitable by an attacker possessing less than Extended-Basic attack potential, then this evaluator action fails.

AVA_VAN.1-10 The evaluator ***shall report*** in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each:

a) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);

b) the SFR(s) not met;

c) a description;

d) whether it is exploitable in its operational environment or not (i.e. exploitable or residual);

e) identification of evaluation party (e.g. developer, evaluator) who identified it.

15.2.2 Evaluation of sub-activity (AVA_VAN.2)

15.2.2.1 Objectives

1716 The objective of this sub-activity is to determine whether the TOE, in its operational environment, has vulnerabilities exploitable by attackers possessing basic attack potential.

15.2.2.2 Input

1717 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the guidance documentation;

e) the TOE suitable for testing;

f) information publicly available to support the identification of possible potential vulnerabilities.

1718 The remaining implicit evaluation evidence for this sub-activity depends on the components that have been included in the assurance package. The evidence provided for each component is to be used as input in this sub-activity.

1719 Other input for this sub-activity is:

a) current information regarding public domain potential vulnerabilities and attacks (e.g. from an overseer).

15.2.2.3 Application notes

1720 The evaluator should consider performing additional tests as a result of potential vulnerabilities encountered during other parts of the evaluation.

15.2.2.4 Action AVA_VAN.2.1E

AVA_VAN.2.1C ***The TOE shall be suitable for testing.***

AVA_VAN.2-1 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state.

1721 The known state is to be one that is consistent with the ST. It is possible for the ST to specify more than one configuration for evaluation. The TOE may be composed of a number of distinct hardware and software implementations that need to be tested in accordance with the ST. The evaluator verifies that there are test configurations consistent with each evaluated configuration described in the ST.

1722 The evaluator should consider the security objectives for the operational environment described in the ST that may apply to the test environment. There may be some security objectives in the ST that do not apply to the test environment. For example, an assumption about user clearances that is used to restrict access to the TOE may not apply; however, an assumption about a single point of connection to a network would apply.

1723 If the assurance package includes a component from the ATE_IND family, then the evaluator may refer to the result of the work unit ATE_IND.*-2 to demonstrate that this is satisfied.

1724 It is possible for the evaluator to determine the state of the TOE in a number of ways. For example, previous successful completion of the AGD_PRE.1 sub-activity will satisfy this work unit if the evaluator still has confidence that the TOE being used for testing was installed properly and is in a known state. If this is not the case, then the evaluator should follow the developer's procedures to install, generate and start up the TOE, using the supplied guidance only.

1725 If the evaluator has to perform the installation procedures because the TOE is in an unknown state, this work unit when successfully completed could satisfy work unit AGD_PRE.1-5

1726 If any test resources are used (e.g. meters, analysers) it will be the evaluator's responsibility to ensure that these resources are calibrated correctly.

15.2.2.5 Action AVA_VAN.2.2E

AVA_VAN.2-2 The evaluator examines the sources of information publicly available to support the identification of possible potential vulnerabilities in the TOE. There are many sources of publicly available information which the evaluator should consider using:

a) world wide web;

b) specialist publications (magazines, books);

c) research papers;

d) conference proceedings.

1727 The evaluator should not constrain their consideration of publicly available information to the above, but should consider any other relevant information available.

1728 While examining the evidence provided the evaluator will use the information in the public domain to further search for potential vulnerabilities. Where the evaluators have identified areas of concern, the evaluator should consider information publicly available that relate to those areas of concern.

1729 The availability of information that may be readily available to an attacker that helps to identify and facilitate attacks may substantially enhance the attack potential of a given attacker. The accessibility of vulnerability information and sophisticated attack tools on the Internet makes it more likely that this information will be used in attempts to identify potential vulnerabilities in the TOE and exploit them. Modern search tools make such information easily available to the evaluator, and the determination of resistance to published potential vulnerabilities and well known generic attacks can be achieved in a cost-effective manner.

1730 The search of the information publicly available should be focused on those sources that refer specifically to the product from which the TOE is derived. The extensiveness of this search should consider the following factors: TOE type, evaluator experience in this TOE type, expected attack potential and the level of ADV evidence available.

1731 The identification process is iterative, where the identification of one potential vulnerability may lead to identifying another area of concern that requires further investigation.

1732 The evaluator will report what actions were taken to identify potential vulnerabilities in the evidence. However, in this type of search, the evaluator may not be able to describe the steps in identifying potential vulnerabilities before the outset of the examination, as the approach may evolve as a result of findings during the search.

1733 The evaluator will report the evidence examined in completing the search for potential vulnerabilities. This selection of evidence may be derived from those areas of concern identified by the evaluator, linked to the evidence the attacker is assumed to be able to obtain, or according to another rationale provided by the evaluator.

15.2.2.6 Action AVA_VAN.2.3E

AVA_VAN.2-3 The evaluator ***shall conduct*** a search of ST, guidance documentation, functional specification and TOE design evidence to identify possible potential vulnerabilities in the TOE.

1734 A search of the evidence should be completed whereby specifications and documentation for the TOE are analysed and then potential vulnerabilities in the TOE are hypothesised, or speculated. The list of hypothesised potential vulnerabilities is then prioritised on the basis of the estimated probability that a potential vulnerability exists and, assuming an exploitable vulnerability does exist the attack potential required to exploit it, and on the extent of control or compromise it would provide. The prioritised list of potential vulnerabilities is used to direct penetration testing against the TOE.

1735 If a potential vulnerability can be exploited in the TOE, but it not apparent that the level of attack potential required is basic, the guidance on determining the necessary attack potential can be found in Annex B.4 and B.5.1.

1736 Potential vulnerabilities hypothesised as exploitable only by attackers possessing moderate or high attack potential do not result in a failure of this evaluator action. Where analysis supports the hypothesis, these need not be considered further as an input to penetration testing. However, such vulnerabilities are reported in the ETR as residual vulnerabilities.

1737 Potential vulnerabilities hypothesised exploitable by an attacker possessing a basic attack potential, that do not result in a violation of the security objectives specified in the ST, do not result in a failure of this evaluator action. Where analysis supports the hypothesis, these need not be considered further as an input to penetration testing.

1738 Potential vulnerabilities hypothesised as exploitable by an attacker possessing a basic attack potential and resulting in a violation of the security objectives should be the highest priority potential vulnerabilities comprising the list used to direct penetration testing against the TOE.

1739 Subject to the SFRs the TOE is to meet in the operational environment, the evaluator's independent vulnerability analysis should consider generic potential vulnerabilities under each of the following headings:

a) generic potential vulnerabilities relevant for the type of TOE being evaluated, as may be supplied by the overseer;

b) bypassing;

c) tampering;

d) direct attacks;

e) misuse.

1740 Items b) - e) are explained in greater detail in Annex B.

AVA_VAN.2-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.

1741 It may be identified that no further consideration of the potential vulnerability is required if for example the evaluator identifies that measures in the operational environment, either IT or non-IT, prevent exploitation of the potential vulnerability in that operational environment. For instance, restricting physical access to the TOE to authorised users only may effectively render a potential vulnerability to tampering unexploitable.

1742 The evaluator records any reasons for exclusion of potential vulnerabilities from further consideration if the evaluator determines that the potential vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

1743 A list of potential vulnerabilities applicable to the TOE in its operational environment, which can be used as an input into penetration testing activities, shall be reported in the ETR by the evaluators.

15.2.2.7 Action AVA_VAN.2.4E

AVA_VAN.2-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.

1744 The evaluator prepares for penetration testing as necessary to determine the susceptibility of the TOE, in its operational environment, to the potential vulnerabilities identified during the search of the sources of publicly available information and the analysis of the TOE guidance and design evidence. The evaluator should have access to current information (e.g. from the overseer) regarding known potential vulnerabilities that may not have been considered by the evaluator.

1745 The evaluator will probably find it practical to carry out penetration test using a series of test cases, where each test case will test for a specific potential vulnerability.

1746 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required a basic attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers an exploitable vulnerability that is beyond basic attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.2-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include:

a) identification of the potential vulnerability the TOE is being tested for;

b) instructions to connect and setup all required test equipment as required to conduct the penetration test;

c) instructions to establish all penetration test prerequisite initial conditions;

d) instructions to stimulate the TSF;

e) instructions for observing the behaviour of the TSF;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE.

1747 The evaluator prepares for penetration testing based on the list of potential vulnerabilities identified during the search of the public domain and the analysis of the evaluation evidence.

1748 The evaluator is not expected to determine the exploitability for potential vulnerabilities beyond those for which a basic attack potential is required to effect an attack. However, as a result of evaluation expertise, the evaluator may discover a potential vulnerability that is exploitable only by an attacker with greater than basic attack potential. Such vulnerabilities are to be reported in the ETR as residual vulnerabilities.

1749 With an understanding of the potential vulnerability, the evaluator determines the most feasible way to test for the TOE's susceptibility. Specifically the evaluator considers:

a) the TSFI or other TOE interface that will be used to stimulate the TSF and observe responses (It is possible that the evaluator will need to use an interface to the TOE other than the TSFI to demonstrate properties of the TSF such as those described in the architectural description (as required by ADV_ARC). It should the noted, that although these TOE interfaces provide a means of testing the TSF properties, they are not the subject of the test.);

b) initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

c) special test equipment that will be required to either stimulate a TSFI or make observations of a TSFI (although it is unlikely that specialist equipment would be required to exploit a potential vulnerability assuming a basic attack potential);

d) whether theoretical analysis should replace physical testing, particularly relevant where the results of an initial test can be extrapolated to demonstrate that repeated attempts of an attack are likely to succeed after a given number of attempts.

1750 The evaluator will probably find it practical to carry out penetration testing using a series of test cases, where each test case will test for a specific potential vulnerability.

1751 The intent of specifying this level of detail in the test documentation is to allow another evaluator to repeat the tests and obtain an equivalent result.

AVA_VAN.2-7 The evaluator ***shall conduct*** penetration testing.

1752 The evaluator uses the penetration test documentation resulting from work unit AVA_VAN.2-6 as a basis for executing penetration tests on the TOE, but this does not preclude the evaluator from performing additional ad hoc penetration tests. If required, the evaluator may devise ad hoc tests as a result of information learnt during penetration testing that, if performed by the evaluator, are to be recorded in the penetration test documentation. Such tests may be required to follow up unexpected results or observations, or to investigate potential vulnerabilities suggested to the evaluator during the pre-planned testing.

1753 Should penetration testing show that a hypothesised potential vulnerability does not exist, then the evaluator should determine whether or not the evaluator's own analysis was incorrect, or if evaluation deliverables are incorrect or incomplete.

1754 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required a basic attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers an exploitable vulnerability that is beyond basic attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.2-8 The evaluator ***shall record*** the actual results of the penetration tests.

1755 While some specific details of the actual test results may be different from those expected (e.g. time and date fields in an audit record) the overall result should be identical. Any unexpected test results should be investigated. The impact on the evaluation should be stated and justified.

AVA_VAN.2-9 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.

1756 The penetration testing information reported in the ETR allows the evaluator to convey the overall penetration testing approach and effort expended on this sub-activity. The intent of providing this information is to give a meaningful overview of the evaluator's penetration testing effort. It is not intended that the information regarding penetration testing in the ETR be an exact reproduction of specific test steps or results of individual penetration tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the penetration testing approach chosen, amount of penetration testing performed, TOE test configurations, and the overall results of the penetration testing activity.

1757 Information that would typically be found in the ETR section regarding evaluator penetration testing efforts is:

a) TOE test configurations. The particular configurations of the TOE that were penetration tested;

b) TSFI penetration tested. A brief listing of the TSFI and other TOE interfaces that were the focus of the penetration testing;

c) Verdict for the sub-activity. The overall judgement on the results of penetration testing.

1758 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the penetration testing the evaluator performed during the evaluation.

AVA_VAN.2-10 The evaluator ***shall examine*** the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing a basic attack potential.

1759 If the results reveal that the TOE, in its operational environment, has vulnerabilities exploitable by an attacker possessing less than an Extended-Basic attack potential, then this evaluator action fails.

AVA_VAN.2-11 The evaluator ***shall report*** in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each:

a) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);

b) the SFR(s) not met;

c) a description;

d) whether it is exploitable in its operational environment or not (i.e. exploitable or residual);

e) identification of evaluation party (e.g. developer, evaluator) who identified it.

15.2.3 Evaluation of sub-activity (AVA_VAN.3)

15.2.3.1 Objectives

1760 The objective of this sub-activity is to determine whether the TOE, in its operational environment, has vulnerabilities exploitable by attackers possessing Extended-Basic attack potential.

15.2.3.2 Input

1761 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the implementation subset selected;

e) the guidance documentation;

f) the TOE suitable for testing;

g) information publicly available to support the identification of possible potential vulnerabilities.

1762 The remaining implicit evaluation evidence for this sub-activity depends on the components that have been included in the assurance package. The evidence provided for each component is to be used as input in this sub-activity.

1763 Other input for this sub-activity is:

a) current information regarding public domain potential vulnerabilities and attacks (e.g. from an overseer).

15.2.3.3 Application notes

1764 During the conduct of evaluation activities the evaluator may also identify areas of concern. These are specific portions of the TOE evidence that the evaluator has some reservation about, although the evidence meets the requirements for the activity with which the evidence is associated. For example, a particular interface specification looks particularly complex, and therefore may be prone to error either in the construction of the TOE or in the operation of the TOE. There is no potential vulnerability apparent at this stage, further investigation is required. This is beyond the bounds of encountered, as further investigation is required.

1765 The focused approach to the identification of potential vulnerabilities is an analysis of the evidence with the aim of identifying any potential vulnerabilities evident through the contained information. It is an

unstructured analysis, as the approach is not predetermined. Further guidance on focused vulnerability analysis can be found in Annex B.2.2.2.2.

15.2.3.4 Action AVA_VAN.3.1E

AVA_VAN.3.1C ***The TOE shall be suitable for testing.***

AVA_VAN.3-1 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state.

1766 The known state is to be one that is consistent with the ST. It is possible for the ST to specify more than one configuration for evaluation. The TOE may be composed of a number of distinct hardware and software implementations that need to be tested in accordance with the ST. The evaluator verifies that there are test configurations consistent with each evaluated configuration described in the ST.

1767 The evaluator should consider the security objectives for the operational environment described in the ST that may apply to the test environment. There may be some security objectives in the ST that do not apply to the test environment. For example, an assumption about user clearances that is used to restrict access to the TOE may not apply; however, an assumption about a single point of connection to a network would apply.

1768 If the assurance package includes a component from the ATE_IND family, then the evaluator may refer to the result of the work unit ATE_IND.*-2 to demonstrate that this is satisfied.

1769 It is possible for the evaluator to determine the state of the TOE in a number of ways. For example, previous successful completion of the AGD_PRE.1 sub-activity will satisfy this work unit if the evaluator still has confidence that the TOE being used for testing was installed properly and is in a known state. If this is not the case, then the evaluator should follow the developer's procedures to install, generate and start up the TOE, using the supplied guidance only.

1770 If the evaluator has to perform the installation procedures because the TOE is in an unknown state, this work unit when successfully completed could satisfy work unit AGD_PRE.1-5

1771 If any test resources are used (e.g. meters, analysers) it will be the evaluator's responsibility to ensure that these resources are calibrated correctly.

15.2.3.5 Action AVA_VAN.3.2E

AVA_VAN.3-2 The evaluator examines the sources of information publicly available to support the identification of possible potential vulnerabilities in the TOE. There are many sources of publicly available information which the evaluator should consider using:

a) world wide web;

b) specialist publications (magazines, books);

c) research papers;

d) conference proceedings.

1772 The evaluator should not constrain their consideration of publicly available information to the above, but should consider any other relevant information available.

1773 While examining the evidence provided the evaluator will use the information in the public domain to further search for potential vulnerabilities. Where the evaluators have identified areas of concern, the evaluator should consider information publicly available that relate to those areas of concern.

1774 The availability of information that may be readily available to an attacker that helps to identify and facilitate attacks may substantially enhance the attack potential of a given attacker. The accessibility of vulnerability information and sophisticated attack tools on the Internet makes it more likely that this information will be used in attempts to identify potential vulnerabilities in the TOE and exploit them. Modern search tools make such information easily available to the evaluator, and the determination of resistance to published potential vulnerabilities and well known generic attacks can be achieved in a cost-effective manner.

1775 The search of the information publicly available should be focused on those sources that refer to the technologies used in the construction of the product from which the TOE is derived. The extensiveness of this search should consider the following factors: TOE type, evaluator experience in this TOE type, expected attack potential and the level of ADV evidence available.

1776 The identification process is iterative, where the identification of one potential vulnerability may lead to identifying another area of concern that requires further investigation.

1777 The evaluator will report what actions were taken to identify potential vulnerabilities in the evidence. However, in this type of search, the evaluator may not be able to describe the steps in identifying potential vulnerabilities before the outset of the examination, as the approach may evolve as a result of findings during the search.

1778 The evaluator will report the evidence examined in completing the search for potential vulnerabilities. This selection of evidence may be derived from those areas of concern identified by the evaluator, linked to the evidence the attacker is assumed to be able to obtain, or according to another rationale provided by the evaluator.

15.2.3.6 Action AVA_VAN.3.3E

AVA_VAN.3-3 The evaluator ***shall conduct*** a focused search of ST, guidance documentation, functional specification, TOE design and implementation representation to identify possible potential vulnerabilities in the TOE.

1779 A flaw hypothesis methodology should be used whereby specifications and development and guidance evidence are analysed and then potential vulnerabilities in the TOE are hypothesised, or speculated.

1780 The evaluator should use the knowledge of the TOE design and operation gained from the TOE deliverables to conduct a flaw hypothesis to identify potential flaws in the construction of the TOE and potential errors in the specified method of operation of the TOE.

1781 The following provide some examples of hypotheses that may be created when examining the evidence:

a) consideration of malformed input for interfaces available to an attacker at the external interfaces;

b) examination of a key security mechanism, such as TOE security function separation, hypothesising internal buffer overflows that may lead to degradation of separation;

c) search to identify any objects created in the TOE implementation representation that are then not fully controlled by the TSF, and could be used by an attacker to undermine SFRs.

1782 The approach taken is directed by areas of concern identified during examination of the evidence during the conduct of evaluation activities and ensuring a representative sample of the development and guidance evidence provided for the evaluation is searched.

1783 For guidance on sampling see Annex A.2. This guidance should be considered when selecting the subset, giving reasons for:

a) the approach used in selection;

b) qualification that the evidence to be examined supports that approach.

1784 The evidence to be considered during the vulnerability analysis may be linked to the evidence the attacker is assumed to be able to obtain. For example, the developer may protect the TOE design and implementation representations, so the only information assumed to be available to an attacker is the functional specification and guidance (available publicly available). So, although the objectives for assurance in the TOE ensure the TOE design and implementation representation requirements are met, these design representations may only be searched to further investigate areas of concerns.

1785 On the other hand, if the source is available publicly available it would be reasonable to assume that the attacker has access to the source and can use this in attempts to attack the TOE. Therefore, the source should be considered in the focused examination approach.

1786 The following indicates examples for the selection of the subset of evidence to be considered:

a) For an evaluation where all levels of design abstraction from functional specification to implementation representation are provided, examination of information in the functional specification and the implementation representation may be selected, as the functional specification provides detail of interfaces available to an attacker, and the implementation representation incorporates the design decisions made at all other design abstractions. Therefore, the TOE design information will be considered as part of the implementation representation.

b) Examination of a particular subset of information in each of the design representations provided for the evaluation.

c) Coverage of particular SFRs through each of the design representations provided for the evaluation.

d) Examination of each of the design representations provided for the evaluation, considering different SFRs within each design representations.

e) Examination of aspects of the evidence provided for the evaluation relating to current potential vulnerability information the evaluator has received (e.g. from a scheme).

1787 This approach to identification of potential vulnerabilities is to take an ordered and planned approach; applying a system to the examination. The evaluator is to describe the method to be used in terms of what evidence will be considered, the information within the evidence that is to be examined, the manner in which this information is to be considered and the hypothesis that is to be created.

1788 The following provide some examples that a hypothesis may take:

a) consideration of malformed input for interfaces available to an attacker at the external interfaces;

b) examination of a key security mechanism, such as TOE security function separation, hypothesising internal buffer overflows that may lead to degradation of separation;

c) search to identify any objects created in the TOE implementation representation that are then not fully controlled by the TSF, and could be used by an attacker to undermine SFRs.

1789 For example, the evaluator may identify that interfaces are a potential area of weakness in the TOE and specify an approach to the search that "all interface specifications provided in the functional specification and TOE design will be searched to hypothesise potential vulnerabilities" and go on to explain the methods used in the hypothesis.

1790 The identification process is iterative, where the identification of one potential vulnerability may lead to identifying another area of concern that requires further investigation.

1791 The evaluator will report what actions were taken to identify potential vulnerabilities in the evidence. However, in this type of search, the evaluator may not be able to describe the steps in identifying potential vulnerabilities before the outset of the examination, as the approach may evolve as a result of findings during the search.

1792 The evaluator will report the evidence examine in completing the search for potential vulnerabilities. This selection of evidence may be derived from those areas of concern identified by the evaluator, linked to the evidence the attacker is assumed to be able to obtain, or according to another rationale provided by the evaluator.

1793 Subject to the SFRs the TOE is to meet in the operational environment, the evaluator's independent vulnerability analysis should consider generic potential vulnerabilities under each of the following headings:

a) generic potential vulnerabilities relevant for the type of TOE being evaluated, as may be supplied by the overseer;

b) bypassing;

c) tampering;

d) direct attacks;

e) misuse.

1794 Items b) - e) are explained in greater detail in Annex B.

AVA_VAN.3-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.

1795 It may be identified that no further consideration of the potential vulnerability is required if for example the evaluator identifies that measures in the operational environment, either IT or non-IT, prevent exploitation of the potential vulnerability in that operational environment. For instance, restricting physical access to the TOE to authorised users only may effectively render a potential vulnerability to tampering unexploitable.

1796 The evaluator records any reasons for exclusion of potential vulnerabilities from further consideration if the evaluator determines that the potential

vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

1797 A list of potential vulnerabilities applicable to the TOE in its operational environment, which can be used as an input into penetration testing activities, shall be reported in the ETR by the evaluators.

15.2.3.7 Action AVA_VAN.3.4E

AVA_VAN.3-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.

1798 The evaluator prepares for penetration testing as necessary to determine the susceptibility of the TOE, in its operational environment, to the potential vulnerabilities identified during the search of the sources of publicly available information and the analysis of the TOE guidance and design evidence. The evaluator should have access to current information (e.g. from the overseer) regarding known potential vulnerabilities that may not have been considered by the evaluator.

1799 The evaluator will probably find it practical to carry out penetration test using a series of test cases, where each test case will test for a specific potential vulnerability.

1800 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required an Extended-Basic attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers an exploitable vulnerability that is beyond Extended-Basic attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.3-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include:

a) identification of the potential vulnerability the TOE is being tested for;

b) instructions to connect and setup all required test equipment as required to conduct the penetration test;

c) instructions to establish all penetration test prerequisite initial conditions;

d) instructions to stimulate the TSF;

e) instructions for observing the behaviour of the TSF;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE.

1801 The evaluator prepares for penetration testing based on the list of potential vulnerabilities identified during the search of the public domain and the analysis of the evaluation evidence.

1802 The evaluator is not expected to determine the exploitability for potential vulnerabilities beyond those for which an Extended-Basic attack potential is required to effect an attack. However, as a result of evaluation expertise, the evaluator may discover a potential vulnerability that is exploitable only by an attacker with greater than Extended-Basic attack potential. Such vulnerabilities are to be reported in the ETR as residual vulnerabilities.

1803 With an understanding of the potential vulnerability, the evaluator determines the most feasible way to test for the TOE's susceptibility. Specifically the evaluator considers:

a) the TSFI or other TOE interface that will be used to stimulate the TSF and observe responses (It is possible that the evaluator will need to use an interface to the TOE other than the TSFI to demonstrate properties of the TSF such as those described in the architectural description (as required by ADV_ARC). It should the noted, that although these TOE interfaces provide a means of testing the TSF properties, they are not the subject of the test.);

b) initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

c) special test equipment that will be required to either stimulate a TSFI or make observations of a TSFI (although it is unlikely that specialist equipment would be required to exploit a potential vulnerability assuming an Extended-Basic attack potential);

d) whether theoretical analysis should replace physical testing, particularly relevant where the results of an initial test can be extrapolated to demonstrate that repeated attempts of an attack are likely to succeed after a given number of attempts.

1804 The evaluator will probably find it practical to carry out penetration testing using a series of test cases, where each test case will test for a specific potential vulnerability.

1805 The intent of specifying this level of detail in the test documentation is to allow another evaluator to repeat the tests and obtain an equivalent result.

AVA_VAN.3-7 The evaluator ***shall conduct*** penetration testing.

1806 The evaluator uses the penetration test documentation resulting from work unit AVA_VAN.3-6 as a basis for executing penetration tests on the TOE,

but this does not preclude the evaluator from performing additional ad hoc penetration tests. If required, the evaluator may devise ad hoc tests as a result of information learnt during penetration testing that, if performed by the evaluator, are to be recorded in the penetration test documentation. Such tests may be required to follow up unexpected results or observations, or to investigate potential vulnerabilities suggested to the evaluator during the pre-planned testing.

1807 Should penetration testing show that a hypothesised potential vulnerability does not exist, then the evaluator should determine whether or not the evaluator's own analysis was incorrect, or if evaluation deliverables are incorrect or incomplete.

1808 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required an Extended-Basic attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers an exploitable vulnerability that is beyond Extended-Basic attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.3-8 The evaluator ***shall record*** the actual results of the penetration tests.

1809 While some specific details of the actual test results may be different from those expected (e.g. time and date fields in an audit record) the overall result should be identical. Any unexpected test results should be investigated. The impact on the evaluation should be stated and justified.

AVA_VAN.3-9 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.

1810 The penetration testing information reported in the ETR allows the evaluator to convey the overall penetration testing approach and effort expended on this sub-activity. The intent of providing this information is to give a meaningful overview of the evaluator's penetration testing effort. It is not intended that the information regarding penetration testing in the ETR be an exact reproduction of specific test steps or results of individual penetration tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the penetration testing approach chosen, amount of penetration testing performed, TOE test configurations, and the overall results of the penetration testing activity.

1811 Information that would typically be found in the ETR section regarding evaluator penetration testing efforts is:

a) TOE test configurations. The particular configurations of the TOE that were penetration tested;

b) TSFI penetration tested. A brief listing of the TSFI and other TOE interfaces that were the focus of the penetration testing;

c) Verdict for the sub-activity. The overall judgement on the results of penetration testing.

1812 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the penetration testing the evaluator performed during the evaluation.

AVA_VAN.3-10 The evaluator ***shall examine*** the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing an Extended-Basic attack potential.

1813 If the results reveal that the TOE, in its operational environment, has vulnerabilities exploitable by an attacker possessing less than Moderate attack potential, then this evaluator action fails.

AVA_VAN.3-11 The evaluator ***shall report*** in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each:

a) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);

b) the SFR(s) not met;

c) a description;

d) whether it is exploitable in its operational environment or not (i.e. exploitable or residual);

e) identification of evaluation party (e.g. developer, evaluator) who identified it.

15.2.4 Evaluation of sub-activity (AVA_VAN.4)

15.2.4.1 Objectives

1814 The objective of this sub-activity is to determine whether the TOE, in its operational environment, has vulnerabilities exploitable by attackers possessing moderate attack potential.

15.2.4.2 Input

1815 The evaluation evidence for this sub-activity is:

a) the ST;

b) the functional specification;

c) the TOE design;

d) the implementation representation;

e) the architectural design;

f) the guidance documentation;

g) the TOE suitable for testing.

1816 The remaining implicit evaluation evidence for this sub-activity depends on the components that have been included in the assurance package. The evidence provided for each component is to be used as input in this sub-activity.

1817 Other input for this sub-activity is:

a) current information regarding public domain potential vulnerabilities and attacks (e.g. from an overseer).

15.2.4.3 Application notes

1818 The methodical analysis approach takes the form of a structured examination of the evidence. This method requires the evaluator to specify the structure and form the analysis will take (i.e. the manner in which the analysis is performed is predetermined, unlike the focused analysis). The method is specified in terms of the information that will be considered and how/why it will be considered. Further guidance on methodical vulnerability analysis can be found in Annex B.2.2.2.3.

15.2.4.4 Action AVA_VAN.4.1E

AVA_VAN.4.1C ***The TOE shall be suitable for testing.***

AVA_VAN.4-1 The evaluator ***shall examine*** the TOE to determine that it has been installed properly and is in a known state.

1819 The known state is to be one that is consistent with the ST. It is possible for the ST to specify more than one configuration for evaluation. The TOE may be composed of a number of distinct hardware and software implementations that need to be tested in accordance with the ST. The evaluator verifies that there are test configurations consistent with each evaluated configuration described in the ST.

1820 The evaluator should consider the security objectives for the operational environment described in the ST that may apply to the test environment. There may be some security objectives in the ST that do not apply to the test environment. For example, an assumption about user clearances that is used to restrict access to the TOE may not apply; however, an assumption about a single point of connection to a network would apply.

1821 If the assurance package includes a component from the ATE_IND family, then the evaluator may refer to the result of the work unit ATE_IND.*-2 to demonstrate that this is satisfied.

1822 It is possible for the evaluator to determine the state of the TOE in a number of ways. For example, previous successful completion of the AGD_PRE.1 sub-activity will satisfy this work unit if the evaluator still has confidence

that the TOE being used for testing was installed properly and is in a known state. If this is not the case, then the evaluator should follow the developer's procedures to install, generate and start up the TOE, using the supplied guidance only.

1823 If the evaluator has to perform the installation procedures because the TOE is in an unknown state, this work unit when successfully completed could satisfy work unit AGD_PRE.1-5

1824 If any test resources are used (e.g. meters, analysers) it will be the evaluator's responsibility to ensure that these resources are calibrated correctly.

15.2.4.5 Action AVA_VAN.4.2E

AVA_VAN.4-2 The evaluator examines the sources of information publicly available to support the identification of possible potential vulnerabilities in the TOE. There are many sources of publicly available information which the evaluator should consider using:

a) world wide web;

b) specialist publications (magazines, books);

c) research papers;

d) conference proceedings.

1825 The evaluator should not constrain their consideration of publicly available information to the above, but should consider any other relevant information available.

1826 While examining the evidence provided the evaluator will use the information in the public domain to further search for potential vulnerabilities. Where the evaluators have identified areas of concern, the evaluator should consider information publicly available that relate to those areas of concern.

1827 The availability of information that may be readily available to an attacker that helps to identify and facilitate attacks may substantially enhance the attack potential of a given attacker. The accessibility of vulnerability information and sophisticated attack tools on the Internet makes it more likely that this information will be used in attempts to identify potential vulnerabilities in the TOE and exploit them. Modern search tools make such information easily available to the evaluator, and the determination of resistance to published potential vulnerabilities and well known generic attacks can be achieved in a cost-effective manner.

1828 The search of the information publicly available should be focused on those sources that refer to the technologies used in the construction of the product from which the TOE is derived. The extensiveness of this search should consider the following factors: TOE type, evaluator experience in this TOE type, expected attack potential and the level of ADV evidence available.

1829 The identification process is iterative, where the identification of one potential vulnerability may lead to identifying another area of concern that requires further investigation.

1830 The evaluator will describe the approach to be taken to identify potential vulnerabilities in the publicly available material, detailing the search to be performed. This may be driven by factors such as areas of concern identified by the evaluator, linked to the evidence the attacker is assumed to be able to obtain. However, it is recognised that in this type of search the approach may further evolve as a result of findings during the search. Therefore, the evaluator will also report any actions taken in addition to those described in the approach to further investigate issues thought to lead to potential vulnerabilities, and will report the evidence examined in completing the search for potential vulnerabilities.

15.2.4.6 Action AVA_VAN.4.3E

AVA_VAN.4-3 The evaluator ***shall conduct*** a methodical analysis of ST, guidance documentation, functional specification, TOE design and implementation representation to identify possible potential vulnerabilities in the TOE.

1831 Guidance on methodical vulnerability analysis is provided in Annex B.2.2.2.3.

1832 This approach to identification of potential vulnerabilities is to take an ordered and planned approach. A system is to be applied in the examination. The evaluator is to describe the method to be used in terms of the manner in which this information is to be considered and the hypothesis that is to be created.

1833 A flaw hypothesis methodology should be used whereby the ST, development (functional specification, TOE design and implementation representation) and guidance evidence are analysed and then vulnerabilities in the TOE are hypothesised, or speculated.

1834 The evaluator should use the knowledge of the TOE design and operation gained from the TOE deliverables to conduct a flaw hypothesis to identify potential flaws in the construction of the TOE and potential errors in the specified method of operation of the TOE.

1835 The approach taken is directed by the results of the evaluator's assessment of the development and guidance evidence. Therefore, the investigation of the evidence for the existence of potential vulnerabilities may be directed by areas of concern identified during examination of the evidence during the conduct of evaluation activities. Or, in the absence of the evaluator identifying areas of concern in the construction of the TOE, “ad-hoc” examination of the evidence to hypothesise potential vulnerabilities in the TOE.

1836 The following provide some examples of hypotheses that may be created when examining the evidence:

a) consideration of malformed input for interfaces available to an attacker at the external interfaces;

b) examination of a key security mechanism, such as TOE security function separation, hypothesising internal buffer overflows that may lead to degradation of separation;

c) search to identify any objects created in the TOE implementation representation that are then not fully controlled by the TSF, and could be used by an attacker to undermine SFRs.

1837 For example, the evaluator may identify that interfaces are a potential area of weakness in the TOE and specify an approach to the search that 'all interface specifications in the evidence provided will be searched to hypothesise potential vulnerabilities' and go on to explain the methods used in the hypothesis.

1838 In addition, areas of concern the evaluator has identified during examination of the evidence during the conduct of evaluation activities. Areas of concern may also be identified during the conduct of other work units associated with this component, in particular AVA_VAN.4-7, AVA_VAN.4-5 and AVA_VAN.4-6) where the development and conduct of penetration tests may identify further areas of concerns for investigation, or potential vulnerabilities.

1839 However, examination of only a subset of the development and guidance evidence or their contents is not permitted in this level of rigour. The approach description should provide a demonstration that the methodical approach used is complete, providing confidence that the approach used to search the deliverables has considered all of the information provided those deliverables.

1840 This approach to identification of potential vulnerabilities is to take an ordered and planned approach; applying a system to the examination. The evaluator is to describe the method to be used in terms of how the evidence will be considered; the manner in which this information is to be considered and the hypothesis that is to be created. This approach should be agreed with the overseer, and the overseer should provide detail of any additional approaches the evaluator should take to the vulnerability analysis and identify any additional information that should be considered by the evaluator.

1841 Although a system to identifying potential vulnerabilities is predefined, the identification process may still be iterative, where the identification of one potential vulnerability may lead to identifying another area of concern that requires further investigation.

1842 Subject to the SFRs the TOE is to meet in the operational environment, the evaluator's independent vulnerability analysis should consider generic potential vulnerabilities under each of the following headings:

a) generic potential vulnerabilities relevant for the type of TOE being evaluated, as may be supplied by the overseer;

b) bypassing;

c) tampering;

d) direct attacks;

e) misuse;

f) covert channels.

1843 Items b) - e) are explained in greater detail in Annex B.

1844 Guidance for item f) should be sought from the scheme. However, if there are no Unobservability (FDP_UNO) or information flow policies (expressed through Access control (FDP_ACC)) requirements in the ST, it is not applicable to consider unintended channels of communication (e.g. covert channels) during the conduct of the vulnerability analysis. The lack of these requirements in the ST reflects that there are no objectives either to prevent one user of the TOE from observing activity associated with another user of the TOE, or to ensure that information flows cannot be used to achieve illicit data signals.

AVA_VAN.4-4 The evaluator ***shall record*** in the ETR the identified potential vulnerabilities that are candidates for testing and applicable to the TOE in its operational environment.

1845 It may be identified that no further consideration of the potential vulnerability is required if for example the evaluator identifies that measures in the operational environment, either IT or non-IT, prevent exploitation of the potential vulnerability in that operational environment. For instance, restricting physical access to the TOE to authorised users only may effectively render a potential vulnerability to tampering unexploitable.

1846 The evaluator records any reasons for exclusion of potential vulnerabilities from further consideration if the evaluator determines that the potential vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

1847 A list of potential vulnerabilities applicable to the TOE in its operational environment, which can be used as an input into penetration testing activities, shall be reported in the ETR by the evaluators.

15.2.4.7 Action AVA_VAN.4.4E

AVA_VAN.4-5 The evaluator ***shall devise*** penetration tests, based on the independent search for potential vulnerabilities.

1848 The evaluator prepares for penetration testing as necessary to determine the susceptibility of the TOE, in its operational environment, to the potential

vulnerabilities identified during the search of the sources of publicly available information and the analysis of the TOE guidance and design evidence. The evaluator should have access to current information (e.g. from the overseer) regarding known potential vulnerabilities that may not have been considered by the evaluator.

1849 The evaluator will probably find it practical to carry out penetration test using a series of test cases, where each test case will test for a specific potential vulnerability.

1850 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required a Moderate attack potential. In some cases, however, it will be necessary to carry out a test before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers an exploitable vulnerability that is beyond Moderate attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.4-6 The evaluator ***shall produce*** penetration test documentation for the tests based on the list of potential vulnerabilities in sufficient detail to enable the tests to be repeatable. The test documentation shall include:

a) identification of the potential vulnerability the TOE is being tested for;

b) instructions to connect and setup all required test equipment as required to conduct the penetration test;

c) instructions to establish all penetration test prerequisite initial conditions;

d) instructions to stimulate the TSF;

e) instructions for observing the behaviour of the TSF;

f) descriptions of all expected results and the necessary analysis to be performed on the observed behaviour for comparison against expected results;

g) instructions to conclude the test and establish the necessary post-test state for the TOE.

1851 The evaluator prepares for penetration testing based on the list of potential vulnerabilities identified during the search of the public domain and the analysis of the evaluation evidence.

1852 The evaluator is not expected to determine the exploitability for potential vulnerabilities beyond those for which a Moderate attack potential is required to effect an attack. However, as a result of evaluation expertise, the evaluator may discover a potential vulnerability that is exploitable only by an attacker with greater than Moderate attack potential. Such vulnerabilities are to be reported in the ETR as residual vulnerabilities.

1853 With an understanding of the potential vulnerability, the evaluator determines the most feasible way to test for the TOE's susceptibility. Specifically the evaluator considers:

a) the TSFI or other TOE interface that will be used to stimulate the TSF and observe responses (It is possible that the evaluator will need to use an interface to the TOE other than the TSFI to demonstrate properties of the TSF such as those described in the architectural description (as required by ADV_ARC). It should the noted, that although these TOE interfaces provide a means of testing the TSF properties, they are not the subject of the test.);

b) initial conditions that will need to exist for the test (i.e. any particular objects or subjects that will need to exist and security attributes they will need to have);

c) special test equipment that will be required to either stimulate a TSFI or make observations of a TSFI;

d) whether theoretical analysis should replace physical testing, particularly relevant where the results of an initial test can be extrapolated to demonstrate that repeated attempts of an attack are likely to succeed after a given number of attempts.

1854 The evaluator will probably find it practical to carry out penetration testing using a series of test cases, where each test case will test for a specific potential vulnerability.

1855 The intent of specifying this level of detail in the test documentation is to allow another evaluator to repeat the tests and obtain an equivalent result.

AVA_VAN.4-7 The evaluator ***shall conduct*** penetration testing.

1856 The evaluator uses the penetration test documentation resulting from work unit AVA_VAN.4-6 as a basis for executing penetration tests on the TOE, but this does not preclude the evaluator from performing additional ad hoc penetration tests. If required, the evaluator may devise ad hoc tests as a result of information learnt during penetration testing that, if performed by the evaluator, are to be recorded in the penetration test documentation. Such tests may be required to follow up unexpected results or observations, or to investigate potential vulnerabilities suggested to the evaluator during the pre-planned testing.

1857 Should penetration testing show that a hypothesised potential vulnerability does not exist, then the evaluator should determine whether or not the evaluator's own analysis was incorrect, or if evaluation deliverables are incorrect or incomplete.

1858 The evaluator is not expected to test for potential vulnerabilities (including those in the public domain) beyond those which required a Moderate attack potential. In some cases, however, it will be necessary to carry out a test

before the exploitability can be determined. Where, as a result of evaluation expertise, the evaluator discovers an exploitable vulnerability that is beyond Moderate attack potential, this is reported in the ETR as a residual vulnerability.

AVA_VAN.4-8 The evaluator ***shall record*** the actual results of the penetration tests.

1859 While some specific details of the actual test results may be different from those expected (e.g. time and date fields in an audit record) the overall result should be identical. Any unexpected test results should be investigated. The impact on the evaluation should be stated and justified.

AVA_VAN.4-9 The evaluator ***shall report*** in the ETR the evaluator penetration testing effort, outlining the testing approach, configuration, depth and results.

1860 The penetration testing information reported in the ETR allows the evaluator to convey the overall penetration testing approach and effort expended on this sub-activity. The intent of providing this information is to give a meaningful overview of the evaluator's penetration testing effort. It is not intended that the information regarding penetration testing in the ETR be an exact reproduction of specific test steps or results of individual penetration tests. The intention is to provide enough detail to allow other evaluators and overseers to gain some insight about the penetration testing approach chosen, amount of penetration testing performed, TOE test configurations, and the overall results of the penetration testing activity.

1861 Information that would typically be found in the ETR section regarding evaluator penetration testing efforts is:

a) TOE test configurations. The particular configurations of the TOE that were penetration tested;

b) TSFI penetration tested. A brief listing of the TSFI and other TOE interfaces that were the focus of the penetration testing;

c) Verdict for the sub-activity. The overall judgement on the results of penetration testing.

1862 This list is by no means exhaustive and is only intended to provide some context as to the type of information that should be present in the ETR concerning the penetration testing the evaluator performed during the evaluation.

AVA_VAN.4-10 The evaluator ***shall examine*** the results of all penetration testing to determine that the TOE, in its operational environment, is resistant to an attacker possessing a Moderate attack potential.

1863 If the results reveal that the TOE, in its operational environment, has vulnerabilities exploitable by an attacker possessing less than a High attack potential, then this evaluator action fails.

AVA_VAN.4-11 The evaluator ***shall report*** in the ETR all exploitable vulnerabilities and residual vulnerabilities, detailing for each:

a) its source (e.g. CEM activity being undertaken when it was conceived, known to the evaluator, read in a publication);

b) the SFR(s) not met;

c) a description;

d) whether it is exploitable in its operational environment or not (i.e. exploitable or residual);

e) identification of evaluation party (e.g. developer, evaluator) who identified it.

15.2.5 Evaluation of sub-activity (AVA_VAN.5)

1864 There is no general guidance; the scheme should be consulted for guidance on this sub-activity.

16 Class ACO: Composition

16.1 Introduction

1865 The goal of this activity is to determine whether the components can be integrated in a secure manner, as defined in the ST for the composed TOE. This is achieved through examination and testing of the interfaces between the components, supported by examination of the design of the components and the conduct of vulnerability analysis.

16.2 Application notes

1866 The Reliance of dependent component (ACO_REL) family identifies where the dependent component is reliant upon IT in its operational environment (satisfied by a base component in the composed TOE evaluation) in order to provide its own security services. This reliance is identified in terms of the interfaces expected by the dependent component to be provided by the base component. Development evidence (ACO_DEV) then determines which interfaces of the base component were considered (as TSFI) during the base component evaluation. A typical example would be a database management system (DBMS) (which has been evaluated) that relies upon its underlying operating system (OS).

1867 During the evaluation of the DBMS, there will be an assessment made of the security properties of that DBMS (to whatever degree of assurance is dictated by the assurance components used in the evaluation): its TSF boundary will be identified, its functional specification will be assessed to determine whether it describes the interfaces to the security services provided by the TSF, perhaps additional information about the TSF (its design, architecture, internal structure) will be provided, the TSF will be tested, aspects of its life-cycle and its guidance documentation will be assessed, etc.

1868 However, the DBMS evaluation will not call for any evidence concerning the dependency the DBMS has on the OS. The ST of the DBMS will state assumptions about the OS in its Assumptions section, or state security objectives for the OS in its Environment section, or even instantiate those objectives for the environment in terms of SFRs for the OS, but there will not be a specification that mirrors the detail in the functional specification, architecture description, or other Class ADV: Development evidence. Reliance of dependent component (ACO_REL) will fulfil that need.

1869 It should be noted that Reliance of dependent component (ACO_REL) does not cover other evidence that may be needed to address the composition problem (e.g. descriptions of non-TSF interfaces of the OS, rules for integration, etc.). This is outside the security assessment of the composition and is a functional composition issue.

1870 As part of Base TOE testing (ACO_TBT) the evaluator will perform tests of a subset of the interfaces to the base component to confirm they operate as specified. The subset selected will consider the possible effects of changes made to the configuration/use of the base component from the base component evaluation, in addition to factors such as the security functionality invoked through the interface. The developer will provide test evidence for each of the base component interfaces (the requirements for coverage are consistent with those specified for Evaluation of sub-activity (ATE_COV.2)).

1871 Composition rationale (ACO_COR) requires the evaluator to determine whether the appropriate assurance measures have been applied to the base component, and whether the base component is being used in its evaluated configuration. This includes determination of whether all security functionality required by the dependent component was within the TSF of the base component. The Composition rationale (ACO_COR) requirement may be met through the production of evidence that each of these is demonstrated to be upheld. This evidence may be in the form of a certification report and security target. If, on the other hand, one of the above have not been upheld, then it may be possible that an argument can be made as to why the assurance gained during an original evaluation is unaffected. If this is not possible then additional evaluation evidence for those aspects of the base component not covered may have to be provided. This material is then assessed in Development evidence (ACO_DEV).

1872 For example, it may be the case as described in the Interactions between entities (see B.3, Interactions between composed IT entities in CC Part 3) that the dependent component requires the base component to provide more security functionality than that considered in the base component evaluation. This would be determined during the application of the Reliance of dependent component (ACO_REL) and Development evidence (ACO_DEV) families. In this case the composition information evidence provided for Composition rationale (ACO_COR) would demonstrate that the assurance gained during the base component evaluation is unaffected. This may be achieved by means including:

a) Performing a re-evaluation of the base component focusing on the evidence relating to the extended part of the TSF;

b) Demonstrating that the extended part of the TSF cannot affect other portions of the TSF, and providing evidence that the extended part of the TSF provides the necessary security functionality.

16.3 Composition rationale (ACO_COR)

16.3.1 Evaluation of sub-activity (ACO_COR.1)

16.3.1.1 Input

1873 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the composition information;

c) the reliance information;

d) the development information;

e) unique identifier.

16.3.1.2 Action ACO_COR.1.1E

ACO_COR.1.1C ***The composition information shall demonstrate that the base component, when configured as required to support the TSF of the dependent component, provides a level of assurance in that support functionality at least as high as that of the dependent component.***

ACO_COR.1-1 The evaluator ***shall perform*** a correspondence analysis between the development information and the reliance information to identify the interfaces that are relied upon by the dependent component which are not detailed in the development information.

1874 The evaluator's goal in this work unit is to determine which interfaces relied upon by the dependent component have had the appropriate assurance measures applied.

1875 The evaluator may use the correspondence tracing in the development information developed during the Development evidence (ACO_DEV) activities (e.g. ACO_DEV.1-2, ACO_DEV.2-6, ACO_DEV.3-8) to help identify the interfaces identified in the reliance information that are not considered in the development information.

1876 The evaluator will record the SFR-enforcing interfaces described in the reliance information that are not included in the development information. These will provide input to ACO_COR.1-3 work unit, helping to identify the portions of the base component in which further assurance is required.

ACO_COR.1-2 The evaluator ***shall examine*** the composition information to determine, for those included base component interfaces on which the dependent TSF relies, whether the interface was considered during the evaluation of the base component.

1877 The ST, certification report and guidance documents for the base component all provide information on the scope and boundary of the base component.

The ST provides details of the logical scope and boundary of the composed TOE, allowing the evaluator to determine whether an interface relates to a portion of the product that was within the scope of the evaluation. The guidance documentation provides details of use of all interfaces for the composed TOE. Although the guidance documentation may include details of interfaces in the product that are not within the scope of the evaluation, any such interfaces should be identifiable, either from the scoping information in the ST or through a portion of the guidance that deals with the certified configuration. The certification report will provide any additional constraints on the use of the composed TOE that are necessary.

1878 Therefore, the combination of these inputs allows the evaluator to determine whether an interface described in the composition information has the necessary assurance associated with it, or whether further assurance is required. The evaluator will record those interfaces of the base component for which additional assurance is required, for consideration during ACO_COR.1-3.

ACO_COR.1-3 The evaluator ***shall examine*** the composition information to determine that the necessary assurance measures have been applied to the base component.

1879 The evaluation verdicts, and resultant assurance, for the base component can be reused provided the same portions of the base component are used in the composed TOE and they are used in a consistent manner.

1880 In order to determine where the necessary assurance measures have already been applied to the component, and the portions of the component for which assurance measures still need to be applied, the evaluator should use the output of the ACO_DEV.*.2E action and the work units ACO_COR.1-1 and ACO_COR.1-2:

a) For those interfaces identified in the reliance information (Reliance of dependent component (ACO_REL)), but not discussed in development information (Development evidence (ACO_DEV)), additional information is required. (Identified in ACO_COR.1-1.)

b) For those interfaces used inconsistently in the composed TOE from the base component (difference between the information provided in Development evidence (ACO_DEV) and Reliance of dependent component (ACO_REL) the impact of the differences in use need to be considered. (Identified in ACO_DEV.*.2E.)

c) For those interfaces identified in composition information for which no assurance has previously been gained, additional information is required. (Identified in ACO_COR.1-2.)

d) For those interfaces consistently described in the reliance information, composition information and the development information, no further action is required as the results of the base component evaluation can be re-used.

1881 The interfaces of the base component reported to be required by the reliance information but not included in the development information indicate the portions of the base component where further assurance is required. The interfaces identify the entry points into the base component for which associated internals need to be described to the level of detail required by the TOE design (ADV_TDS) components of the base component evaluation.

1882 For those interfaces included in both the development information and reliance information, the evaluator is to determine whether the interfaces are being used in the composed TOE in a manner that is consistent with the base component evaluation. The method of use of the interface will be considered during the Development evidence (ACO_DEV) activities to determine that the use of the interface is consistent in both the base component and the composed TOE. The remaining consideration is the determination of whether the configurations of the base component and the composed TOE are consistent. To determine this, the evaluator will consider the guidance documentation of each to ensure they are consistent (see further guidance below regarding consistent guidance documentation). Any deviation in the documentation will be further analysed by the evaluation to determine the possible effects.

1883 For those interfaces that are consistently described in the reliance information and development information, and for which the guidance is consistent for the base component and the composed TOE, the required level of assurance has been provided.

1884 The following subsections provide guidance on how to determine consistency between assurance gained in the base component, the evidence provided for the composed TOE, and the analysis performed by the evaluator in the instances where inconsistencies are identified.

16.3.1.2.1 Development

1885 If an interface identified in the reliance information is not identified in the development information, then the design information for that interface and internals necessary to satisfy the Class ADV: Development requirements of the base component must be obtained. This is likely to require deliverables from the base component developer. With the design evidence of the interface(s) the evaluator will ensure that the Class ADV: Development requirements of the base component are satisfied.

1886 If an interface identified in the reliance information is identified in the development information, but there are inconsistencies between the descriptions, further analysis is required. The evaluator identifies the differences in use of the base component in the base component evaluation and the composed TOE evaluation. The evaluator will devise testing to be performed (during the conduct of Base TOE testing (ACO_TBT)) to test the interface.

16.3.1.2.2 Guidance

1887 The guidance for the composed TOE is likely to make substantial reference out to the guidance for the individual components. The minimal guidance expected to be necessary is the identification of any ordering dependencies in the application of guidance for the dependent and base components, particularly during the preparation (installation) of the composed TOE.

1888 In addition to the application of the Preparative user guidance (AGD_PRE) and Operational user guidance (AGD_OPE) families to the guidance for the composed TOE, it is necessary to analyse the consistency between the guidance for the components and the composed TOE, to identify any deviations.

1889 If the composed TOE guidance refers out to the base component and dependent component guidance, then the consideration for consistency is limited to consistency between the guidance documentation provided for each of the components (i.e. consistency between the base component guidance and the dependent component guidance). However, if additional guidance is provided for the composed TOE, to that provided for the components, greater analysis is required, as consistency is also required between the guidance documentation for the components and guidance documentation for the composed TOE.

1890 *Consistent* in this instance is understood to mean that either the guidance is the same or it places additional constraints on the operation of the individual components when combined, in a similar manner to *refinement* of functional/assurance components.

1891 With the information available (that used as input for Development evidence (ACO_DEV) or the development aspects discussed above) the evaluator may be able to determine all possible impacts of the deviation from the certified configuration of the base component. However, for high EALs (where the base component evaluation included TOE design (ADV_TDS) requirements) it is possible that, unless detailed design abstractions for the base component are delivered as part of the development information for the composed TOE, the possible impacts of the modification to the guidance cannot be fully determined as the internals are unknown. In this case the evaluator will report the residual risk of the analysis.

1892 These residual risks are to be included in any certification report for the composed TOE.

1893 The evaluator will note these variances in the guidance for input into evaluator independent testing activities (Base TOE testing (ACO_TBT)).

1894 The guidance for the composed TOE may add to the guidance for the components, particularly in terms of installation and the ordering of installation steps for the base component in relation to the installation steps for the dependent component. The ordering of the steps for the installation of the individual components should not change, however they may need to be interleaved. The evaluator will examine this guidance to ensure that it still

meets the requirement of the AGD_PRE activity performed during the evaluations of the components.

1895 It may be the case that the reliance information identifies that interfaces of the base component, in addition to those identified as TSFI of the base component, are relied upon by the dependent component are identified in the reliance information. It may be necessary for guidance to be provided for the use of any such additional interfaces in the base component. Provided the consumer of the composed TOE is to receive the guidance documentation for the base component, then the results of the AGD_PRE and AGD_OPE verdicts for the base component can be reused for those interfaces considered in the evaluation of the base component. However, for the additional interfaces relied upon by the dependent component, the evaluator will need to determine that the guidance documentation for the base component meets the requirements of AGD_PRE and AGD_OPE, as applied in the base component evaluations.

1896 For those interfaces considered during the base component evaluation, and therefore, for which assurance has already been gained, the evaluator will ensure that the guidance for the use of each interface for the composed TOE is consistent with that provided for the base component. To determine the guidance for the composed TOE is consistent with that for the base component, the evaluator should perform a mapping for each interface to the guidance provided for both the composed TOE and the base component. The evaluator then compares the guidance to determine consistency.

1897 Examples of additional constraints provided in composed TOE guidance that would be considered to be consistent with component guidance are (guidance for a component is given followed by an example of guidance for a composed TOE that would be considered to provide additional constraints):

- Component: The password length must be set to a minimum of 8 characters length, including alphabetic and numeric characters.

- Composed TOE: The password length must be set to a minimum of 10 characters in length, including alphabetic and numeric characters and *at least one of the following special characters: ( ) { } ^ < > - _*

- NOTE: It would only be acceptable to increase the password length to [*integer > 8*] characters while removing the mandate for the inclusion of both alphabetic and numeric characters for the composed TOE, if the same or a higher metric was achieved for the strength rating (taking into account the likelihood of the password being guessed).

- Component: The following services are to be disabled in the registry settings: WWW Publishing Service and ICDBReporter service.

- Composed TOE: The following services are to be disabled in the registry settings: *Publishing Service, ICDBReporter service, Remote Procedure Call (RPC) Locator and Procedure Call (RPC) Service*.

- Component: Select the following attributes to be included in the accounting log files: date, time, type of event, subject identity and success/failure.

- Composed TOE: Select the following attributes to be included in the accounting log files: date, time, type of event, subject identity, success/failure, *event message and process thread*.

1898 If the guidance for the composed TOE deviates (is not a refinement) from that provided for the base component, the evaluator will assess the potential risks of the modification to the guidance. The evaluator will use the information available (including that provided in the public domain, the architectural description of the base component in the certification report, the context of the guidance from the remainder of the guidance documentation) to identify likely impact of the modification to the guidance on the TSP of the composed TOE.

1899 If during the dependent component evaluation the trial installation used the base component to satisfy the environment requirements of the dependent component this work unit is considered to be satisfied. If the base component was not used in satisfaction of the work unit AGD_PRE.1-5 during the dependent component evaluation, the evaluator will apply the user procedures provided for the composed TOE to prepare the composed TOE, in accordance with the guidance specified in AGD_PRE.1-5. This will allow the evaluator to determine that the preparative guidance provided for the composed TOE is sufficient to prepare the composed TOE and its operational environment securely.

16.3.1.2.3 Life-cycle

Delivery

1900 If there is a different delivery mechanism used for the delivery of the composed TOE (i.e. the components are not delivered to the consumer in accordance with the secure delivery procedures defined and assessed during the evaluation of the components), the delivery procedures for the composed TOE will require evaluation against the Delivery (ALC_DEL) requirements applied during the components evaluations.

1901 The composed TOE may be delivered as an integrated product or may require the components to be delivered separately.

1902 If the components are delivered separately, the results of the delivery of the base component and dependent component are reused. The delivery of the base component is checked during the evaluator trial installation of the dependent component, using the specified guidance and checking the aspects of delivery that are the responsibility of the user, as described in the guidance documentation for the base component.

1903 If the composed TOE is delivered as a new entity, then the method of delivery of that entity must be considered in the composed TOE evaluation activities.

1904 The assessment of the delivery procedures for composed TOE items is to be performed in accordance with the methodology for Delivery (ALC_DEL) as for any other (component) TOE, ensuring any additional items (e.g. additional guidance documents for the composed TOE) are considered in the delivery procedures.

CM Capabilities

1905 The unique identification of the composed TOE is considered during the application of Evaluation of sub-activity (ALC_CMC.1) and the items from which that composed TOE is comprised are considered during the application of Evaluation of sub-activity (ALC_CMS.2).

1906 Although additional guidance may be produced for the composed TOE, the unique identification of this guidance (considered as part of the unique identification of the composed TOE during Evaluation of sub-activity (ALC_CMC.1)) is considered sufficient control of the guidance.

1907 The verdicts of the remaining (not considered above) Class ALC: Life-cycle support activities can be reused from the base component evaluation, as no further development is performed during integration of the composed TOE.

1908 There are no additional considerations for development security as the integration is assumed to take place at either the consumer's site or, in the instance that the composed TOE is delivered as an integrated product, at the site of the dependent component developer. Control at the consumer's site is outside the consideration of the CC. No additional requirements or guidance are necessary if integration is at the same site as that for the dependent component, as all components are considered to be configuration items for the composed TOE, and should therefore be considered under the developer's security procedures anyway.

1909 Tools and techniques adopted during integration will be considered in the evidence provided by the dependent component developer. Any tools/techniques relevant to the base component will have been considered during the evaluation of the base component. For example, if the base component is delivered as source code and requires compilation by the consumer (e.g. dependent developer who is performing integration) the compiler would have been specified and assessed, along with the appropriate arguments, during the base component evaluation activities.

1910 There is no life-cycle definition applicable to the composed TOE, as no further development of items is taking place.

1911 The results of flaw remediation for a component are not applicable to the composed TOE. If flaw remediation is included in the assurance package for the composed TOE, then the Flaw remediation (ALC_FLR) requirements are

to be applied during the composed TOE evaluation (as for any augmentation).

16.3.1.2.4 Tests

1912 The composed TOE will have been tested during the conduct of the Class ATE: Tests activities of the dependent component evaluation, as the configurations used for testing of the dependent component should have used the base component to satisfy the requirements for IT in the operational environment. If the base component was not used in the testing of the dependent component, or the configuration of either component varied, then the developer testing performed during the dependent component evaluation to satisfy the Class ATE: Tests requirements is to be repeated on the composed TOE.

16.4 Development evidence (ACO_DEV)

16.4.1 Evaluation of sub-activity (ACO_DEV.1)

16.4.1.1 Objectives

1913 The objective of this sub-activity is to determine that the appropriate security functionality is provided by the base component to support the dependent component. This is achieved through examination of the interfaces of the base component to determine that they are consistent with the interfaces specified in the reliance information; those required by the dependent component.

1914 The description of the interfaces into the base component is to be provided at a level of detail consistent with Evaluation of sub-activity (ADV_FSP.2) although not all of the aspects necessary for satisfaction of Evaluation of sub-activity (ADV_FSP.2) are required for Evaluation of sub-activity (ACO_DEV.1), as once the interface has been identified and the purpose described the remaining detail of the interface specification can be reused from the base component evaluation.

16.4.1.2 Input

1915 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the development information;

c) the reliance information.

16.4.1.3 Action ACO_DEV.1.1E

ACO_DEV.1.1C ***The development information shall describe the purpose of each interface of the base component used in the composed TOE.***

ACO_DEV.1-1 The evaluator ***shall examine*** the development information to determine that it describes the purpose of each interface.

1916 The interfaces of the base component are to be identified at the same level as the description required for the TSFI in the functional specification by ADV_FSP.2-4.

1917 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.1.2C ***The development information shall identify those interfaces of the base component that are provided to support the TSF of the dependent component.***

ACO_DEV.1-2 The evaluator ***shall examine*** the development information to determine which interfaces of the base component support the TSF of the dependent component.

1918 The correspondence between the interfaces of the base component and the interfaces on which the dependent component relies may take the form of a matrix or table. The interfaces that are relied upon by the dependent component are identified in the reliance information (as examined during Reliance of dependent component (ACO_REL)).

1919 There is, during this activity, no requirement to determine completeness of the coverage of interfaces that are relied upon by the dependent component, only that the correspondence is correct and ensuring that interfaces of the base component are mapped to interfaces required by the dependent component wherever possible. The completeness of the coverage is considered in Composition rationale (ACO_COR) activities.

16.4.1.4 Action ACO_DEV.1.2E

ACO_DEV.1-3 The evaluator ***shall examine*** the development information and the reliance information to determine that the interfaces are described consistently.

1920 The evaluator's goal in this work unit is to determine that the interfaces described in the development information for the base component and the reliance information for the dependent component are represented consistently.

16.4.2 Evaluation of sub-activity (ACO_DEV.2)

16.4.2.1 Objectives

1921 The objective of this sub-activity is to determine that the appropriate security functionality is provided by the base component to support the dependent component. This is achieved through examination of the interfaces of the base component to determine that they are consistent with the interfaces specified in the reliance information; those required by the dependent component.

1922 The description of the interfaces into the base component is to be provided at the same level of detail as required by Evaluation of sub-activity (ADV_FSP.3).

16.4.2.2 Input

1923 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the development information;

c) reliance information.

16.4.2.3 Action ACO_DEV.2.1E

ACO_DEV.2.1C ***The development information shall describe the purpose and method of use of each interface of the base component used in the composed TOE.***

ACO_DEV.2-1 The evaluator ***shall examine*** the development information to determine that it describes the purpose of each interface.

1924 The interfaces of the base component are to be identified at the same level as the description required for the TSFI in the functional specification by ADV_FSP.3-4.

1925 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.2-2 The evaluator ***shall examine*** the development information to determine that it describes the method of use for each interface.

1926 The interfaces of the base component are to be identified at the same level as the description provided for the TSFI in the functional specification for the base component by ADV_FSP.3-5.

1927 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.2.2C ***The development information shall describe all parameters associated with each interface.***

ACO_DEV.2-3 The evaluator ***shall examine*** the development information to determine that it completely identifies all parameters associated with every interface to the base component invoked by the dependent TSF.

1928 The parameters associated with the interfaces to the base component, which enforce the SFRs of the dependent component, are to be identified at the same level as the description of parameters provided for the TSFI in the functional specification for the base component by ADV_FSP.3-6 and ADV_FSP.3-7.

1929 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.2.3C ***The development information shall describe all operations associated with each interface.***

ACO_DEV.2-4 The evaluator ***shall examine*** the development information to determine that it completely and accurately describes all SFR-enforcing operations associated with each interface to the base component invoked by the dependent TSF.

1930 The expected SFR-enforcing operations associated with the interfaces to the base component, invoked by the dependent TSF, are to be identified at the same level the expected operations of each SFR-enforcing operation associated with the TSFI in the functional specification for the base component by ADV_FSP.3-8.

1931 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.2.4C ***The development information shall describe the error messages resulting from processing associated with all operations.***

ACO_DEV.2-5 The evaluator ***shall examine*** the development information to determine that it completely and accurately describes error messages resulting from processing associated with all SFR-enforcing interfaces of the base component invoked by the dependent TSF.

1932 The error messages resulting from processing associated with the SFR-enforcing interfaces to the base component invoked by the dependent TSF are to be identified at the same level as the description provided for the error messages associated with all SFR-enforcing operations of the TSFI in the functional specification for the base component by ADV_FSP.3-9 and ADV_FSP.3-10.

1933 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.2.5C ***The development information shall identify those interfaces of the base component that are provided to support the TSF of the dependent component.***

ACO_DEV.2-6 The evaluator ***shall examine*** the development information to determine which interfaces of the base component support the TSF of the dependent component.

1934 The correspondence between the interfaces of the base component and the interfaces on which the dependent component relies may take the form of a matrix or table. The interfaces that are relied upon by the dependent component are identified in the reliance information (as examined during Reliance of dependent component (ACO_REL)).

1935 There is, during this activity, no requirement to determine completeness of the coverage of interfaces that are relied upon by the dependent component, only that the correspondence is correct and ensuring that interfaces of the base component are mapped to interfaces required by the dependent component wherever possible. The completeness of the coverage is considered in Composition rationale (ACO_COR) activities.

16.4.2.4 Action ACO_DEV.2.2E

ACO_DEV.2-7 The evaluator ***shall examine*** the development information and the reliance information to determine that the interfaces are described consistently.

1936 The evaluator's goal in this work unit is to determine that the interfaces described in the development information for the base component and the reliance information for the dependent component are represented consistently.

16.4.3 Evaluation of sub-activity (ACO_DEV.3)

16.4.3.1 Objectives

1937 The objective of this sub-activity is to determine that the appropriate security functionality is provided by the base component to support the dependent component. This is achieved through examination of the interfaces of the base component to determine that they are consistent with the interfaces specified in the reliance information; those required by the dependent component.

1938 The description of the interfaces into the base component is to be provided at the same level of detail as required by Evaluation of sub-activity (ADV_FSP.4). In addition to the interface description the architecture of the base component is to be provided to enable the evaluator to determine whether or not that interface formed part of the TSF of the base component.

16.4.3.2 Input

1939 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the development information;

c) reliance information.

16.4.3.3 Action ACO_DEV.3.1E

ACO_DEV.3.1C ***The development information shall describe the purpose and method of use of each interface of the base component used in the composed TOE.***

ACO_DEV.3-1 The evaluator ***shall examine*** the development information to determine that it describes the purpose of each interface.

1940 The interfaces of the base component are to be identified at the same level as the description required for the TSFI in the functional specification by ADV_FSP.4-4.

1941 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.3-2 The evaluator ***shall examine*** the development information to determine that it describes the method of use for each interface.

1942 The interfaces of the base component are to be identified at the same level as the description provided for the TSFI in the functional specification for the base component by ADV_FSP.4-5.

1943 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.3.2C ***The development information shall describe all parameters associated with each interface.***

ACO_DEV.3-3 The evaluator ***shall examine*** the development information to determine that it completely identifies all parameters associated with every interface to the base component invoked by the dependent TSF.

1944 The parameters associated with the interfaces to the base component, which enforce the SFRs of the dependent component, are to be identified at the same level as the description of parameters provided for the TSFI in the functional specification for the base component by ADV_FSP.4-6 and ADV_FSP.4-7.

1945 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.3.3C ***The development information shall describe all operations associated with each interface.***

ACO_DEV.3-4 The evaluator ***shall examine*** the development information to determine that it completely and accurately describes all operations associated with each interface to the base component invoked by the dependent TSF.

1946 The expected operations associated with the interfaces to the base component, invoked by the dependent TSF, are to be identified at the same level as required by ADV_FSP.4-8 for the expected operations of the TSFI in the functional specification for the base component.

1947 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.3.4C ***The development information shall describe the error messages resulting from processing associated with all operations.***

ACO_DEV.3-5 The evaluator ***shall examine*** the development information to determine that it completely and accurately describes error messages resulting from processing associated with all interfaces of the base component invoked by the dependent TSF.

1948 The error messages resulting from processing associated with the interfaces to the base component invoked by the dependent TSF are to be identified at the same level as the description provided for the error messages associated with all operations of the TSFI in the functional specification for the base component by ADV_FSP.4-9 and ADV_FSP.4-10.

1949 This work unit may be satisfied by the provision of the functional specification for the base component for those interfaces that are TSFI of the base component.

ACO_DEV.3.5C ***The development information shall describe the structure of the base component in terms of components.***

ACO_DEV.3-6 The evaluator ***shall examine*** the development information to determine that it describes the structure of the base component in terms of components.

1950 The components used in the description of the structure of the composed TOE should be at a level of detail consistent with that used to satisfy the Evaluation of sub-activity (ADV_TDS.3) component for the base component evaluation. For example if the base component was evaluated to EAL4 level of assurance, the components are to be described at the same level of detail as provided for Evaluation of sub-activity (ADV_TDS.3).

ACO_DEV.3.6C ***The development information shall describe the architecture of those components that provide the interfaces of the base component that are relied upon to support the TSF of the dependent component.***

ACO_DEV.3-7 The evaluator ***shall examine*** the development information to determine that it describes the architecture of the components that provide the interfaces of the base component.

1951 The architectural description of the components that provide the interfaces of the base component relied upon to support the TSF of the dependent component should be at a level of detail consistent with that used to satisfy the Evaluation of sub-activity (ADV_ARC.1) component for the base component evaluation. For example if the base component was evaluated to EAL4 level of assurance, the components are to be described at the same level of detail as provided for Evaluation of sub-activity (ADV_ARC.1).

ACO_DEV.3.7C ***The development information shall identify those interfaces of the base component that are provided to support the TSF of the dependent component.***

ACO_DEV.3-8 The evaluator ***shall examine*** the development information to determine which interfaces of the base component support the TSF of the dependent component.

1952 The correspondence between the interfaces of the base component and the interfaces on which the dependent component relies may take the form of a matrix or table. The interfaces that are relied upon by the dependent

component are identified in the reliance information (as examined during Reliance of dependent component (ACO_REL)).

1953 There is, during this activity, no requirement to determine completeness of the coverage of interfaces that are relied upon by the dependent component, only that the correspondence is correct and ensuring that interfaces of the base component are mapped to interfaces required by the dependent component wherever possible. The completeness of the coverage is considered in Composition rationale (ACO_COR) activities.

16.4.3.4 Action ACO_DEV.3.2E

ACO_DEV.3-9 The evaluator ***shall examine*** the development information and the reliance information to determine that the interfaces are described consistently.

1954 The evaluator's goal in this work unit is to determine that the interfaces described in the composition information for the base component and the reliance information for the dependent component are represented consistently.

16.5 Reliance of dependent component (ACO_REL)

16.5.1 Evaluation of sub-activity (ACO_REL.1)

16.5.1.1 Objectives

1955 The objectives of this sub-activity are to determine whether the developer's reliance evidence provides sufficient information to determine that the necessary functionality is available in the base component, and the means by which that functionality is invoked. These are provided in terms of a high-level description.

16.5.1.2 Input

1956 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the dependent component functional specification;

c) the dependent component design;

d) the dependent component architectural design;

e) the reliance information.

16.5.1.3 Application notes

1957 A dependent component whose TSF interacts with the base component requires functionality provided by that base component (e.g., remote authentication, remote audit data storage). In these cases, those invoked services need to be described for those charged with configuring the composed TOE for end users. The rationale for requiring this documentation is to aid integrators of the composed TOE to determine what services in the base component might have adverse effects on the dependent component, and to provide information against which to determine the compatibility of the components when applying the Development evidence (ACO_DEV) family.

16.5.1.4 Action ACO_REL.1.1E

ACO_REL.1.1C ***The functional reliance information shall describe the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF.***

ACO_REL.1-1 The evaluator ***shall check*** the reliance information to determine that it describes the functionality of the base dependent hardware, firmware and/or software that is relied upon by the dependent component TSF.

1958 The evaluator assesses the description of the security functionality that the dependent component TSF requires to be provided by the base component's hardware, firmware and software. The emphasis of this work unit is on the

level of detail of this description, rather than on an assessment of the information's accuracy. (The assessment of the accuracy of the information is the focus of the next work unit.)

1959 This description of the base component's functionality need not be any more detailed than the level of the description of a component of the TSF, as would be provided in the TOE Design (TOE design (ADV_TDS))

ACO_REL.1-2 The evaluator ***shall examine*** the reliance information to determine that it accurately reflects the objectives specified for the operational environment of the dependent component.

1960 The reliance information contains the description of the base component's security functionality relied upon by the dependent component. To ensure that the reliance information is consistent with the expectations of the operational environment of the dependent component, the evaluator compares the reliance information with the statement of objectives for the environment in the ST for the dependent component.

1961 For example, if the reliance information claims that the dependent component TSF relies upon the base component to store and protect audit data, yet other evaluation evidence (e.g. the dependent component design) makes it clear that the dependent component TSF itself is storing and protecting the audit data, this would indicate an inaccuracy.

1962 It should be noted that the objectives for the operational environment may include objectives that can be met by non-IT measures. While the services that the base component environment is expected to provide may be described in the description of IT objectives for the operational environment in the dependent component ST, it is not required that all such expectations on the environment be described in the reliance information.

ACO_REL.1.2C ***The functional reliance information shall identify all interfaces through which the dependent component TSF requests services from the base component.***

ACO_REL.1-3 The evaluator ***shall examine*** the reliance information to determine that it identifies the interfaces presented by the base component through which the dependent component TSF requests services.

1963 The dependent component TSF may request services of the base component that were not within the TSF of the base component (see B.3, Interactions between composed IT entities in CC Part 3).

1964 Accuracy and completeness of the interfaces is based on the security functionality that the TSF requires to be provided by the base component, as assessed in work units ACO_REL.1-1 and ACO_REL.1-2. It should be possible to map all of the functionality described in the earlier work units to the interfaces identified in this work unit, and vice versa. An interface that does not correspond to described functionality would also indicate an inadequacy.

ACO_REL.1.3C ***The functional reliance information shall describe the purpose and method of use of each interface.***

ACO_REL.1-4 The evaluator ***shall examine*** the reliance information to determine that it states the purpose of each interface.

1965 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1966 The purpose of an interface description is a general statement summarising the functionality provided by the interface. It is not intended to be a complete statement of the operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for.

ACO_REL.1-5 The evaluator ***shall examine*** the reliance information to determine that it describes the method of use for each interface.

1967 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1968 The method of use for an interface describes how the interface is manipulated in order to invoke the operations and obtain the results associated with the interface. The evaluator should be able to determine, from reading this material in the reliance information, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each interface, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the reliance information, as well as by the evaluator evaluating the reliance information.

16.5.2 Evaluation of sub-activity (ACO_REL.2)

16.5.2.1 Objectives

1969 The objectives of this sub-activity are to determine whether the developer's reliance evidence provides sufficient information to determine that the necessary functionality is available in the base component, and the means by which that functionality is invoked. These are provided in terms of a high-level description of the purpose and method of use and also the parameters, and SFR-enforcing operations and error messages.

16.5.2.2 Input

1970 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the dependent component functional specification;

c) the dependent component design;

d) the dependent component architectural design;

e) the reliance information.

16.5.2.3 Application notes

1971 A dependent component whose TSF interacts with the base component requires functionality provided by that base component (e.g., remote authentication, remote audit data storage). In these cases, those invoked services need to be described for those charged with configuring the composed TOE for end users. The rationale for requiring this documentation is to aid integrators of the composed TOE to determine what services in the base component might have adverse effects on the dependent component, and to provide information against which to determine the compatibility of the components when applying the Development evidence (ACO_DEV) family.

16.5.2.4 Action ACO_REL.2.1E

ACO_REL.2.1C ***The functional reliance information shall describe the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF.***

ACO_REL.2-1 The evaluator ***shall check*** the reliance information to determine that it describes the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF.

1972 The evaluator assesses the description of the security functionality that the dependent component TSF requires to be provided by the base component's hardware, firmware and software. The emphasis of this work unit is on the level of detail of this description, rather than on an assessment of the information's accuracy. (The assessment of the accuracy of the information is the focus of the next work unit.)

1973 This description of the base component's functionality need not be any more detailed than the level of the description of a component of the TSF, as would be provided in the TOE Design (Evaluation of sub-activity (ADV_TDS.1))

ACO_REL.2-2 The evaluator ***shall examine*** the reliance information to determine that it accurately reflects the objectives specified for the operational environment of the dependent component.

1974 The reliance information contains the description of the base component's security functionality relied upon by the dependent component. To ensure that the reliance information is consistent with the expectations of the operational environment of the dependent component, the evaluator compares the reliance information with the statement of objectives for the environment in the ST for the dependent component.

1975 For example, if the reliance information claims that the dependent component TSF relies upon the base component to store and protect audit data, yet other evaluation evidence (e.g. the dependent component design) makes it clear that the dependent component TSF itself is storing and protecting the audit data, this would indicate an inaccuracy.

1976 It should be noted that the objectives for the operational environment may include objectives that can be met by non-IT measures. While the services that the base component environment is expected to provide may be described in the description of IT objectives for the operational environment in the dependent component ST, it is not required that all such expectations on the environment be described in the reliance information.

ACO_REL.2.2C ***The functional reliance information shall identify all interfaces through which the dependent component TSF requests services from the base component.***

ACO_REL.2-3 The evaluator ***shall examine*** the reliance information to determine that it identifies the interfaces presented by the base component through which the dependent component TSF requests services.

1977 The dependent component TSF may request services of the base component that were not within the TSF of the base component (see B.3, Interactions between composed IT entities in CC Part 3).

1978 Accuracy and completeness of the interfaces is based on the security functionality that the TSF requires to be provided by the base component, as assessed in work units ACO_REL.2-1 and ACO_REL.2-2. It should be possible to map all of the functionality described in the earlier work units to the interfaces identified in this work unit, and vice versa. An interface that does not correspond to described functionality would also indicate an inadequacy.

ACO_REL.2.3C ***The functional reliance information shall describe the purpose and method of use of each interface.***

ACO_REL.2-4 The evaluator ***shall examine*** the reliance information to determine that it states the purpose of each interface.

1979 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1980 The purpose of an interface description is a general statement summarising the functionality provided by the interface. It is not intended to be a complete statement of the operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for.

ACO_REL.2-5 The evaluator ***shall examine*** the reliance information to determine that it describes the method of use for each interface.

1981 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1982 The method of use for an interface describes how the interface is manipulated in order to invoke the operations and obtain the results associated with the interface. The evaluator should be able to determine, from reading this material in the reliance information, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each interface, as it may be possible to describe in general how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the reliance information, as well as by the evaluator evaluating the reliance information.

ACO_REL.2.4C ***The functional reliance information shall provide a parameter description for each interface.***

ACO_REL.2-6 The evaluator ***shall examine*** the reliance information to determine that it completely identifies all parameters associated with every interface to the base component.

1983 The interfaces to the base component's functionality described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1984 The evaluator examines the reliance information to ensure that all of the parameters are described for each interface to the base component that is used by the dependent TSF. Parameters are explicit inputs or outputs to 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; etc.

1985 Because only the parameters that are used by the dependent component TSF need be described, the completeness of the description is measured using the TOE description (Class ADV: Development evidence) of the dependent

component TSF. The evaluator determines which interfaces of the base component are used by the dependent component TSF, and then ensures they are described in the reliance information.

ACO_REL.2.5C ***The functional reliance information shall describe the expected operations and results associated with each SFR-enforcing interface.***

ACO_REL.2-7 The evaluator ***shall examine*** the reliance information to determine that it describes all operations associated with each SFR-enforcing interface to the base component that is invoked by the dependent component TSF.

1986 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1987 The evaluator is referred to the discussion in ADV_FSP.2-6 to determine which operations are deemed to be SFR-enforcing.

1988 The evaluator ensures that all of the operations to be provided by the base component are described.

1989 The level of description that is required is that sufficient for the reader to understand what role the TSFI operations play with respect to the SFR. The evaluator should keep in mind that the description should be detailed enough to support the generation (and assessment) of test cases against that interface. If the description is unclear or lacking detail such that meaningful testing cannot be conducted against the TSFI, it is likely that the description is inadequate.

ACO_REL.2.6C ***The functional reliance information shall describe the error handling performed as a result of the dependent component TSF's use of each SFR-enforcing interface.***

ACO_REL.2-8 The evaluator ***shall examine*** the reliance information to determine that it describes the dependent component TSF's handling of error messages received from the base component.

1990 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.2)).

1991 The evaluator is referred to the discussion in ADV_FSP.2-8 to determine which interfaces are deemed to be SFR-enforcing.

1992 The base component may issue error messages in response to calls by the dependent component TSF to the base component. The evaluator ensures that the dependent component TSF handling of such error messages is described in the reliance information.

1993 For example, the dependent component TSF might depend upon the base component to store and protect audit records that the dependent TSF sends. If the base component is unable to correctly store the audit data sent to it by the dependent component TSF (e.g. if the storage resources are exhausted), the base component would likely send an error message in response.

1994 While the functional specification of the dependent component would contain details concerning the services provided by the dependent component TSF (descriptions, interfaces, etc.), it would not be required to contain details concerning reactions to error messages. The reliance information provides this other information.

16.5.3 Evaluation of sub-activity (ACO_REL.3)

16.5.3.1 Objectives

1995 The objectives of this sub-activity are to determine whether the developer's reliance evidence provides sufficient information to determine the necessary functionality external to the dependent component TSF, and the means by which that functionality is invoked. These are provided in terms of a detailed interface definition.

16.5.3.2 Input

1996 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the dependent component functional specification;

c) the dependent component design;

d) the dependent component architectural design;

e) the dependent component implementation representation;

f) the reliance information.

16.5.3.3 Application notes

1997 A dependent component whose TSF interacts with the base component requires functionality provided by that base component (e.g., remote authentication, remote audit data storage). In these cases, those invoked services need to be described for those charged with configuring the composed TOE for end users. The rationale for requiring this documentation is to aid integrators of the composed TOE to determine what services in the base component might have adverse effects on the dependent component, and to provide information against which to determine the compatibility of the components when applying the Development evidence (ACO_DEV) family.

1998 Additionally, evaluators of the composed TOE will be able to gain assurance that all requirements of the dependent component on the base component are specified in the composed ST by analysing this information.

16.5.3.4 Action ACO_REL.3.1E

ACO_REL.3.1C ***The functional reliance information shall describe the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF.***

ACO_REL.3-1 The evaluator ***shall check*** the reliance information to determine that it describes the functionality of the base component hardware, firmware and/or software that is relied upon by the dependent component TSF.

1999 The evaluator assesses the description of the security functionality that the dependent component TSF requires to be provided by the base component's hardware, firmware and software. The emphasis of this work unit is on the level of detail of this description, rather than on an assessment of the information's accuracy. (The assessment of the accuracy of the information is the focus of the next work unit.)

2000 This description of the base component's functionality need not be any more detailed than the level of the description of a component of the TSF, as would be provided in the TOE Design (Evaluation of sub-activity (ADV_TDS.3))

ACO_REL.3-2 The evaluator ***shall examine*** the reliance information to determine that it accurately reflects the objectives specified for the operational environment of the dependent component.

2001 The reliance information contains the description of the base component's security functionality relied upon by the dependent component. To ensure that the reliance information is consistent with the expectations of the operational environment of the dependent component, the evaluator compares the reliance information with the statement of objectives for the environment in the ST for the dependent component.

2002 For example, if the reliance information claims that the dependent component TSF relies upon the base component to store and protect audit data, yet other evaluation evidence (e.g. the dependent component design) makes it clear that the dependent component TSF itself is storing and protecting the audit data, this would indicate an inaccuracy.

2003 It should be noted that the objectives for the operational environment may include objectives that can be met by non-IT measures. While the services that the base component environment is expected to provide may be described in the description of IT objectives for the operational environment in the dependent component ST, it is not required that all such expectations on the environment be described in the reliance information.

ACO_REL.3.2C ***The functional reliance information shall identify all interfaces through which the dependent component TSF requests services from the base component.***

ACO_REL.3-3 The evaluator ***shall examine*** the reliance information to determine that it identifies the interfaces presented by the base component through which the dependent component TSF requests services.

2004 The dependent component TSF may request services of the base component that were not within the TSF of the base component (see B.3, Interactions between composed IT entities in CC Part 3).

2005 Accuracy and completeness of the interfaces is based on the security functionality that the TSF requires to be provided by the base component, as assessed in work units ACO_REL.3-1 and ACO_REL.3-2. It should be possible to map all of the functionality described in the earlier work units to the interfaces identified in this work unit, and vice versa. An interface that does not correspond to described functionality would also indicate an inadequacy.

ACO_REL.3.3C ***The functional reliance information shall describe the purpose and method of use of each interface.***

ACO_REL.3-4 The evaluator ***shall examine*** the reliance information to determine that it states the purpose of each interface.

2006 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.4)).

2007 The purpose of an interface description is a general statement summarising the functionality provided by the interface. It is not intended to be a complete statement of the operations and results related to the interface, but rather a statement to help the reader understand in general what the interface is intended to be used for.

ACO_REL.3-5 The evaluator ***shall examine*** the reliance information to determine that it describes the method of use for each interface.

2008 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.4)).

2009 The method of use for an interface describes how the interface is manipulated in order to invoke the operations and obtain the results associated with the interface. The evaluator should be able to determine, from reading this material in the reliance information, how to use each interface. This does not necessarily mean that there needs to be a separate method of use for each interface, as it may be possible to describe in general

how kernel calls are invoked, for instance, and then identify each interface using that general style. Different types of interfaces will require different method of use specifications. APIs, network protocol interfaces, system configuration parameters, and hardware bus interfaces all have very different methods of use, and this should be taken into account by the developer when developing the reliance information, as well as by the evaluator evaluating the reliance information.

ACO_REL.3.4C ***The functional reliance information shall provide a parameter description for each interface.***

ACO_REL.3-6 The evaluator ***shall examine*** the reliance information to determine that it completely identifies all parameters associated with every interface to the base component.

2010 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.4)).

2011 The evaluator examines the reliance information to ensure that all of the parameters are described for each interface to the base component that is used by the dependent TSF. Parameters are explicit inputs or outputs to 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; etc.

2012 Because only the parameters that are used by the dependent component TSF need be described, the completeness of the description is measured using the TOE description (Class ADV: Development evidence) of the dependent component TSF. The evaluator determines which interfaces of the base component are used by the dependent component TSF, and then ensures they are described in the reliance information.

ACO_REL.3.5C ***The functional reliance information shall describe the expected operations and results associated with all interfaces.***

ACO_REL.3-7 The evaluator ***shall examine*** the reliance information to determine that it completely and accurately describes all operations associated with each interface to the base component that is invoked by the dependent component TSF.

2013 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.4)).

2014 The evaluator ensures that all of the operations to be provided by the base component are described.

2015 The level of description that is required is that sufficient for the reader to understand what role the TSFI operations play with respect to the SFR. The evaluator should keep in mind that the description should be detailed enough to support the generation (and assessment) of test cases against that interface. If the description is unclear or lacking detail such that meaningful testing cannot be conducted against the TSFI, it is likely that the description is inadequate.

ACO_REL.3.6C ***The functional reliance information shall describe the error handling performed as a result of the dependent component TSF's use of all interfaces.***

ACO_REL.3-8 The evaluator ***shall examine*** the reliance information to determine that it completely and accurately describes the dependent component TSF's handling of error messages received from the base component.

2016 The interfaces to the base component's functionality is described at the same level as the description of the interfaces to the dependent component TSF functionality, as would be provided in the functional specification (Evaluation of sub-activity (ADV_FSP.4)).

2017 The base component may issue error messages in response to calls by dependent component TSF to the base component. The evaluator ensures that the dependent component TSF handling of such error messages is described in the reliance information.

2018 For example, the dependent component TSF might depend upon the base component to store and protect audit records that the dependent TSF sends. If the base component is unable to correctly store the audit data sent to it by the dependent component TSF (e.g. if the storage resources are exhausted), the base component would likely send an error message in response.

2019 While the functional specification of the dependent component would contain details concerning the services provided by the dependent component TSF (descriptions, interfaces, etc.), it would not be required to contain details concerning reactions to error messages. The reliance information provides this other information.

16.6 Base TOE testing (ACO_TBT)

16.6.1 Evaluation of sub-activity (ACO_TBT.1)

16.6.1.1 Objectives

2020 The objective of this sub-activity is to determine whether the developer correctly performed and documented tests for each of the base component interfaces on which the dependent component relies. As part of this determination the evaluator repeats a sample of the tests performed by the developer.

16.6.1.2 Input

2021 The evaluation evidence for this sub-activity is:

a) the base component suitable for testing;

b) the reliance information;

c) the development information.

16.6.1.3 Action ACO_TBT.1.1E

ACO_TBT.1.1C ***The base component shall be suitable for testing.***

ACO_TBT.1-1 The evaluator ***shall examine*** the base component to determine that it has been installed properly and is in a known state.

2022 To determine that the base component has been installed properly and is in a known state the ATE_IND.2-1 and ATE_IND.2-2 work units will be applied to the base component. The test configuration for the base component is to be consistent with the description of the composed TOE and its environment in the ST for the composed TOE.

ACO_TBT.1.2C ***The base component test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.***

ACO_TBT.1-2 The evaluator ***shall examine*** the base component test documentation provided by the developer to determine that it consists of test plans, test procedure descriptions, expected test results and actual test results and meets the requirements of Evaluation of sub-activity (ATE_FUN.1).

2023 All work units necessary for the satisfaction of ATE_FUN.1.1E will be applied to determine:

a) that the test documentation consist of test plans, test procedure descriptions, expected test results and actual test results;

b) that the test documentation contains the information necessary to ensure the tests are repeatable;

c) the level of developer effort that was applied to testing of the base component.

ACO_TBT.1-3 The evaluator ***shall examine*** the set of resources provided by the developer to determine that they are equivalent to the set of resources used by the developer to functionally test the base component.

2024 To determine that the set of resources provided are equivalent to those used to functionally test the base component as used in the composed TOE, the ATE_IND.2-3 work unit will be applied.

ACO_TBT.1.3C ***The test results from the developer execution of the tests shall demonstrate that the base component interface relied upon by the dependent component behaves as specified.***

ACO_TBT.1-4 The evaluator ***shall examine*** the test plan to determine that the testing approach for each base component interface demonstrates the expected behaviour of that interface.

2025 The interface under scrutiny in this work unit is that interface of the base component upon which the dependent component relies. This is described in the reliance information assessed under the Reliance of dependent component (ACO_REL) family.

2026 The evaluator may construct a mapping between the base component interface relied upon by the dependent component and the test scenarios described by the developer in the test plan to identify which tests apply to each interface.

2027 Guidance on this work unit can be found in:

a) 14.2.1, Understanding the expected behaviour of the TOE.

b) 14.2.2, Testing vs. Alternate approaches to verify the expected behaviour of a security function.

ACO_TBT.1.4C ***The dependent component test documentation shall demonstrate that the test configuration(s) included the base component as used in the composed TOE in the environment.***

ACO_TBT.1-5 The evaluator ***shall examine*** the dependent component test documentation to determine that the dependent component testing configuration is consistent with the base component as used in the composed TOE.

2028 The test documentation provided for the dependent component evaluation (and therefore meeting the applicable Class ATE: Tests requirements, including Evaluation of sub-activity (ATE_FUN.1)) will detail the test configuration(s) used for the developer testing of the dependent component. The evaluator will confirm that the test configuration(s) include the use of the base component to satisfy the requirements for IT in the operational environment of the dependent component.

2029 The configuration of the base component within the operational environment will be assessed to determine that it is consistent with the use of the base component in the composed TOE.

16.6.1.4 Action ACO_TBT.1.2E

ACO_TBT.1-6 The evaluator ***shall perform*** testing, in accordance with Evaluation of sub-activity (ATE_IND.2), for a subset of the interfaces to the base component to confirm they operate as specified.

2030 The evaluator will apply all work units necessary for the satisfaction of the component Evaluation of sub-activity (ATE_IND.2), reporting in the ETR for the composed TOE all analysis, results and verdicts as dictated by the work units.

2031 The evaluator can determine the extent to which the developer's tests can be used for the evaluator's independent testing on the basis of the level of developer effort that was applied to testing as determined during ACO_TBT.1-3.

16.7 Composition vulnerability analysis (ACO_VUL)

16.7.1 Evaluation of sub-activity (ACO_VUL.1)

16.7.1.1 Objectives

2032 The objective of this sub-activity is to determine whether the composed TOE, in its operational environment, has easily exploitable vulnerabilities.

2033 The developer provides an analysis of the disposition of any residual vulnerabilities reported for the base component. The evaluator performs a search of the public domain to identify any new potential vulnerabilities in the base component (i.e. those issues that have been reported in the public domain since certification of the base component) and penetration testing.

16.7.1.2 Input

2034 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the composition information;

c) the guidance documentation;

d) the composition vulnerability information;

e) information publicly available to support the identification of possible security vulnerabilities.

16.7.1.3 Application notes

2035 See the application notes for Evaluation of sub-activity (AVA_VAN.1).

16.7.1.4 Action ACO_VUL.1.1E

ACO_VUL.1.1C ***The composition vulnerability information shall demonstrate that any residual vulnerabilities identified for the base component are not exploitable in the operational environment.***

ACO_VUL.1-1 The evaluator ***shall examine*** the composition vulnerability information to determine that residual vulnerabilities in the base component are not exploitable in the operational environment.

2036 The list of vulnerabilities identified in the product during the evaluation of the base component, which were demonstrated to be non-exploitable in the composed TOE, is to be used as an input into this activity. The evaluator will consider the rationale of why these known vulnerabilities were deemed to be non-exploitable during the base component evaluation to ensure that the premise(s) on which these decisions were made are upheld in the composed TOE, or whether the combination has re-introduced the potential vulnerability. For example, if during the base component evaluation it was

assumed that a particular operating system service was disabled, which is enabled in the composed TOE evaluation, any potential vulnerabilities relating to that service previously scoped out should now be considered.

2037 Also, this list of known, non-exploitable vulnerabilities resulting from the evaluation of the base component should be considered in the light of any known, non-exploitable vulnerabilities for the other components (e.g. dependent component) within the composed TOE. This is to consider the case where a potential vulnerability that is non-exploitable in isolation is exploitable when integrated with an IT entity containing another potential vulnerability.

ACO_VUL.1.2C ***The composition vulnerability information shall demonstrate that any assumptions and objectives in the STs for the components are fulfilled by the other components.***

ACO_VUL.1-2 The evaluator ***shall examine*** the composition vulnerability information to determine that any assumptions and objectives in the STs for the components are fulfilled by the other components.

2038 The STs for the component may include assumptions about other components that may use the component to which the ST relates, e.g. the ST for an operating system used as a base component may include an assumption that any applications loaded on the operating system do not run in privileged mode. The composition vulnerability analysis will demonstrate that such assumptions and objectives are fulfilled by other components in the composed TOE.

16.7.1.5 Action ACO_VUL.1.2E

ACO_VUL.1-3 The evaluator ***shall examine*** the sources of information publicly available to support the identification of possible security vulnerabilities in the base component that have become known since certification of the base component.

2039 The evaluator will use the information in the public domain as described in AVA_VAN.1-2 to search for vulnerabilities in the base component.

2040 Those potential vulnerabilities that were publicly available prior to the certification of the base component do not have to be further investigated unless it is apparent to the evaluator that the attack potential required by an attacker to exploit the potential vulnerability has been significantly reduced. This may be through the introduction of some new technology since the base component evaluation that means the exploitation of the potential vulnerability has been simplified.

ACO_VUL.1-4 The evaluator ***shall examine*** the sources of information publicly available to support the identification of possible security vulnerabilities in the dependent component that have become known since certification of the dependent component.

2041 The evaluator will use the information in the public domain as described in AVA_VAN.1-2 to search for vulnerabilities in the dependent component.

2042 Those potential vulnerabilities that were publicly available prior to the certification of the dependent component do not have to be further investigated unless it is apparent to the evaluator that the attack potential required by an attacker to exploit the potential vulnerability has been significantly reduced. This may be through the introduction of some new technology since the dependent component evaluation that means the exploitation of the potential vulnerability has been simplified.

ACO_VUL.1-5 The evaluator ***shall record*** in the ETR the identified potential security vulnerabilities that are candidates for testing and applicable to the composed TOE in its operational environment.

2043 The ST, guidance documentation and functional specification are used to determine whether the vulnerabilities are relevant to the composed TOE in its operational environment.

2044 The evaluator records any reasons for exclusion of vulnerabilities from further consideration if the evaluator determines that the vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

2045 A list of potential vulnerabilities applicable to the composed TOE in its operational environment, which can be used as an input into penetration testing activities (i.e. ACO_VUL.1.3E), shall be reported in the ETR by the evaluators.

16.7.1.6 Action ACO_VUL.1.3E

ACO_VUL.1-6 The evaluator ***shall conduct*** penetration testing as detailed for AVA_VAN.1.3E.

2046 The evaluator will apply all work units necessary for the satisfaction of evaluator action AVA_VAN.1.3E, reporting in the ETR for the composed TOE all analysis and verdicts as dictated by the work units.

2047 The evaluator will also apply the work units for the evaluator action AVA_VAN.1.1E to determine that the composed TOE provided by the developer is suitable for testing.

16.7.2 Evaluation of sub-activity (ACO_VUL.2)

16.7.2.1 Objectives

2048 The objective of this sub-activity is to determine whether the composed TOE, in its operational environment, has vulnerabilities exploitable by attackers possessing basic attack potential.

2049 The developer provides an analysis of the disposition of any residual vulnerabilities reported for the base component and of any vulnerabilities

introduced through the combination of the base and dependent components. The evaluator performs a search of the public domain to identify any new potential vulnerabilities in the base component (i.e. those issues that have been reported in the public domain since certification of the base component). The evaluator will also perform an independent vulnerability analysis of the composed TOE and penetration testing.

16.7.2.2 Input

2050 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the composition information;

c) the reliance information;

d) the guidance documentation;

e) the composition vulnerability information;

f) information publicly available to support the identification of possible security vulnerabilities.

16.7.2.3 Application notes

2051 See the application notes for Evaluation of sub-activity (AVA_VAN.2).

16.7.2.4 Action ACO_VUL.2.1E

ACO_VUL.2.1C ***The composition vulnerability information shall demonstrate that any residual vulnerabilities identified for the base component are not exploitable in the operational environment.***

ACO_VUL.2-1 The evaluator ***shall examine*** the composition vulnerability information to determine that residual vulnerabilities in the base component are not exploitable in the operational environment.

2052 The list of vulnerabilities identified in the product during the evaluation of the base component, which were demonstrated to be non-exploitable in the composed TOE, is to be used as an input into this activity. The evaluator will consider the rationale of why these known vulnerabilities were deemed to be non-exploitable during the base component evaluation to ensure that the premise(s) on which these decisions were made are upheld in the composed TOE, or whether the combination has re-introduced the potential vulnerability. For example, if during the base component evaluation it was assumed that a particular operating system service was disabled, which is enabled in the composed TOE evaluation, any potential vulnerabilities relating to that service previously scoped out should now be considered.

2053 Also, this list of known, non-exploitable vulnerabilities resulting from the evaluation of the base component should be considered in the light of any

known, non-exploitable vulnerabilities for the other components (e.g. dependent component) within the composed TOE. This is to consider the case where a potential vulnerability that is non-exploitable in isolation is exploitable when integrated with an IT entity containing another potential vulnerability.

ACO_VUL.2.2C ***The composition vulnerability information shall demonstrate that any assumptions and objectives in the STs for the components are fulfilled by the other components.***

ACO_VUL.2-2 The evaluator ***shall examine*** the composition vulnerability information to determine that any assumptions and objectives in the STs for the components are fulfilled by the other components.

2054 The STs for the component may include assumptions about other components that may use the component to which the ST relates , e.g. the ST for an operating system used as a base component may include an assumption that any applications loaded on the operating system do not run in privileged mode. The composition vulnerability analysis will demonstrate that such assumptions and objectives are fulfilled by other components in the composed TOE.

16.7.2.5 Action ACO_VUL.2.2E

ACO_VUL.2-3 The evaluator examines the sources of information publicly available to support the identification of possible security vulnerabilities in the base component that have become known since certification of the base component.

2055 The evaluator will use the information in the public domain as described in AVA_VAN.2-2 to search for vulnerabilities in the base component.

2056 Those potential vulnerabilities that were publicly available prior to the certification of the base component do not have to be further investigated unless it is apparent to the evaluator that the attack potential required by an attacker to exploit the potential vulnerability has been significantly reduced. This may be through the introduction of some new technology since the base component evaluation that means the exploitation of the potential vulnerability has been simplified.

ACO_VUL.2-4 The evaluator ***shall examine*** the sources of information publicly available to support the identification of possible security vulnerabilities in the dependent component that have become known since certification of the dependent component.

2057 The evaluator will use the information in the public domain as described in AVA_VAN.1-2 to search for vulnerabilities in the dependent component.

2058 Those potential vulnerabilities that were publicly available prior to the certification of the dependent component do not have to be further investigated unless it is apparent to the evaluator that the attack potential

required by an attacker to exploit the potential vulnerability has been significantly reduced. This may be through the introduction of some new technology since the dependent component evaluation that means the exploitation of the potential vulnerability has been simplified.

ACO_VUL.2-5 The evaluator ***shall record*** in the ETR the identified potential security vulnerabilities that are candidates for testing and applicable to the composed TOE in its operational environment.

2059 The ST, guidance documentation and functional specification are used to determine whether the vulnerabilities are relevant to the composed TOE in its operational environment.

2060 The evaluator records any reasons for exclusion of vulnerabilities from further consideration if the evaluator determines that the vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

2061 A list of potential vulnerabilities applicable to the composed TOE in its operational environment, which can be used as an input into penetration testing activities (ACO_VUL.2.4E), shall be reported in the ETR by the evaluators.

16.7.2.6 Action ACO_VUL.2.3E

ACO_VUL.2-6 The evaluator ***shall conduct*** a search of the composed TOE ST, guidance documentation, reliance information and composition information to identify possible security vulnerabilities in the composed TOE.

2062 The consideration of the base component in the independent evaluator vulnerability analysis will take a slightly different form to that documented in AVA_VAN.2.3E for a component evaluation, as it will not necessarily consider all layers of design abstraction relevant to the assurance package. These will have already been considered during the evaluation of the base component, but the evidence may not be available for the composed TOE evaluation. However, the general approach described in the work units associated with AVA_VAN.2.3E is applicable and should form the basis of the evaluator's search for potential vulnerabilities in the composed TOE.

2063 A vulnerability analysis of the individual components used in the composed TOE will have already been performed during the conduct of the component evaluations. The focus of the vulnerability analysis during the composed TOE evaluation is to identify any vulnerabilities introduced as a result of the integration of the components or due to any changes in the use of the components between the certified component configuration to the composed TOE configuration.

2064 The evaluator will use the understanding of the component's construction as detailed in the reliance information for the dependent component and composition information for the base component, together with the dependent component design information. This information will allow the

evaluator to gain an understanding of how the base component and dependent component interact.

2065 The evaluator will consider any new guidance provided for the installation, start-up and operation of the composed TOE to identify any potential vulnerabilities introduced through this revised guidance.

2066 If the base component has been through assurance continuity activities since certification, the evaluator will consider the patch in the independent vulnerability analysis. Information related to the change provided in Maintenance Report will be the main source of input material of the change. This will be supplemented by any updates to the guidance documentation resulting from the change and any information regarding the change available in the public domain, e.g. vendor website.

2067 Any risks identified due to the lack of evidence to establish the full impact of any patches or deviations in the configuration of a component from the certified configuration are to be documented in the evaluator's vulnerability analysis.

16.7.2.7 Action ACO_VUL.2.4E

ACO_VUL.2-7 The evaluator ***shall conduct*** penetration testing as detailed for AVA_VAN.2.4E.

2068 The evaluator will apply all work units necessary for the satisfaction of evaluator action AVA_VAN.2.4E, reporting in the ETR for the composed TOE all analysis and verdicts as dictated by the work units.

2069 The evaluator will also apply the work units for the evaluator action AVA_VAN.2.1E to determine that the composed TOE provided by the developer is suitable for testing.

16.7.3 Evaluation of sub-activity (ACO_VUL.3)

16.7.3.1 Objectives

2070 The objective of this sub-activity is to determine whether the composed TOE, in its operational environment, has vulnerabilities exploitable by attackers possessing extended-basic attack potential.

2071 The developer provides an analysis of the disposition of any residual vulnerabilities reported for the base component and of any vulnerabilities introduced through the combination of the base and dependent components. The evaluator performs a search of the public domain to identify any new potential vulnerabilities in the base component (i.e. those issues that have been reported in the public domain since certification of the base component). The evaluator will also perform an independent vulnerability analysis of the composed TOE and penetration testing.

16.7.3.2 Input

2072 The evaluation evidence for this sub-activity is:

a) the composed ST;

b) the composition information;

c) the reliance information;

d) the guidance documentation;

e) the composition vulnerability information;

f) information publicly available to support the identification of possible security vulnerabilities.

16.7.3.3 Application notes

2073 See the application notes for Evaluation of sub-activity (AVA_VAN.3).

16.7.3.4 Action ACO_VUL.3.1E

ACO_VUL.3.1C ***The composition vulnerability information shall demonstrate that any residual vulnerabilities identified for the base component are not exploitable in the operational environment.***

ACO_VUL.3-1 The evaluator ***shall examine*** the composition vulnerability information to determine that residual vulnerabilities in the base component are not exploitable in the operational environment.

2074 The list of vulnerabilities identified in the product during the evaluation of the base component, which were demonstrated to be non-exploitable in the composed TOE, is to be used as an input into this activity. The evaluator will consider the rationale of why these known vulnerabilities were deemed to be non-exploitable during the base component evaluation to ensure that the premise(s) on which these decisions were made are upheld in the composed TOE, or whether the combination has re-introduced the potential vulnerability. For example, if during the base component evaluation it was assumed that a particular operating system service was disabled, which is enabled in the composed TOE evaluation, any potential vulnerabilities relating to that service previously scoped out should now be considered.

2075 Also, this list of known, non-exploitable vulnerabilities resulting from the evaluation of the base component should be considered in the light of any known, non-exploitable vulnerabilities for the other components (e.g. dependent component) within the composed TOE. This is to consider the case where a potential vulnerability that is non-exploitable in isolation is exploitable when integrated with an IT entity containing another potential vulnerability.

ACO_VUL.3.2C ***The composition vulnerability information shall demonstrate that any assumptions and objectives in the STs for the components are fulfilled by the other components.***

ACO_VUL.3-2 The evaluator ***shall examine*** the composition vulnerability information to determine that any assumptions and objectives in the STs for the components are fulfilled by the other components.

2076 The STs for the component may include assumptions about other components that may use the component to which the ST relates, e.g. the ST for an operating system used as a base component may include an assumption that any applications loaded on the operating system do not run in privileged mode. The composition vulnerability analysis will demonstrate that such assumptions and objectives are fulfilled by other components in the composed TOE.

2077 In any instances where it is identified that the assumptions and objectives of an individual component are not fulfilled by the other components in the composed TOE should be added to the list for consideration in the independent vulnerability analysis (ACO_VUL.3.3E).

16.7.3.5 Action ACO_VUL.3.2E

ACO_VUL.3-3 The evaluator examines the sources of information publicly available to support the identification of possible security vulnerabilities in the base component that have become known since certification of the base component.

2078 The evaluator will use the information in the public domain as described in AVA_VAN.3-2 to search for vulnerabilities in the base component.

2079 Those potential vulnerabilities that were publicly available prior to the certification of the base component do not have to be further investigated unless it is apparent to the evaluator that the attack potential required by an attacker to exploit the potential vulnerability has been significantly reduced. This may be through the introduction of some new technology since the base component evaluation that means the exploitation of the potential vulnerability has been simplified.

ACO_VUL.3-4 The evaluator ***shall examine*** the sources of information publicly available to support the identification of possible security vulnerabilities in the dependent component that have become known since certification of the dependent component.

2080 The evaluator will use the information in the public domain as described in AVA_VAN.1-2 to search for vulnerabilities in the dependent component.

2081 Those potential vulnerabilities that were publicly available prior to the certification of the dependent component do not have to be further investigated unless it is apparent to the evaluator that the attack potential required by an attacker to exploit the potential vulnerability has been

significantly reduced. This may be through the introduction of some new technology since the dependent component evaluation that means the exploitation of the potential vulnerability has been simplified.

ACO_VUL.3-5 The evaluator ***shall record*** in the ETR the identified potential security vulnerabilities that are candidates for testing and applicable to the composed TOE in its operational environment.

2082 The ST, guidance documentation and functional specification are used to determine whether the vulnerabilities are relevant to the composed TOE in its operational environment.

2083 The evaluator records any reasons for exclusion of vulnerabilities from further consideration if the evaluator determines that the vulnerability is not applicable in the operational environment. Otherwise the evaluator records the potential vulnerability for further consideration.

2084 A list of potential vulnerabilities applicable to the composed TOE in its operational environment, which can be used as an input into penetration testing activities (ACO_VUL.3.4E), shall be reported in the ETR by the evaluators.

16.7.3.6 Action ACO_VUL.3.3E

ACO_VUL.3-6 The evaluator ***shall conduct*** a search of the composed TOE ST, guidance documentation, reliance information and composition information to identify possible security vulnerabilities in the composed TOE.

2085 The consideration of the base component in the independent evaluator vulnerability analysis will take a slightly different form to that documented in AVA_VAN.3.3E for a component evaluation, as it will not necessarily consider all layers of design abstraction relevant to the assurance package. These will have already been considered during the evaluation of the base component, but the evidence may not be available for the composed TOE evaluation. However, the general approach described in the work units associated with AVA_VAN.3.3E is applicable and should form the basis of the evaluator's search for potential vulnerabilities in the composed TOE.

2086 A vulnerability analysis of the individual components used in the composed TOE will have already been performed during the conduct of the component evaluations. The focus of the vulnerability analysis during the composed TOE evaluation is to identify any vulnerabilities introduced as a result of the integration of the components or due to any changes in the use of the components between the certified component configuration to the composed TOE configuration.

2087 The evaluator will use the understanding of the component's construction as detailed in the reliance information for the dependent component and composition information for the base component, together with the dependent component design information. This information will allow the

evaluator to gain an understanding of how the base component and dependent component interact.

2088 The evaluator will consider any new guidance provided for the installation, start-up and operation of the composed TOE to identify any potential vulnerabilities introduced through this revised guidance.

2089 If the base component has been through assurance continuity activities since certification, the evaluator will consider the patch in the independent vulnerability analysis. Information related to the change provided in Maintenance Report will be the main source of input material of the change. This will be supplemented by any updates to the guidance documentation resulting from the change and any information regarding the change available in the public domain, e.g. vendor website.

2090 Any risks identified due to the lack of evidence to establish the full impact of any patches or deviations in the configuration of a component from the certified configuration are to be documented in the evaluator's vulnerability analysis.

16.7.3.7 Action ACO_VUL.3.4E

ACO_VUL.3-7 The evaluator ***shall conduct*** penetration testing as detailed for AVA_VAN.3.4E.

2091 The evaluator will apply all work units necessary for the satisfaction of evaluator action AVA_VAN.3.4E, reporting in the ETR for the composed TOE all analysis and verdicts as dictated by the work units.

2092 The evaluator will also apply the work units for the evaluator action AVA_VAN.3.1E to determine that the composed TOE provided by the developer is suitable for testing.

A General evaluation guidance (normative)

A.1 Objectives

2093 The objective of this chapter is to cover general guidance used to provide technical evidence of evaluation results. The use of such general guidance helps in achieving objectivity, repeatability and reproducibility of the work performed by the evaluator.

A.2 Sampling

2094 This section provides general guidance on sampling. Specific and detailed information is given in those work units under the specific evaluator action elements where sampling has to be performed.

2095 Sampling is a defined procedure of an evaluator whereby some subset of a required set of evaluation evidence is examined and assumed to be representative for the entire set. It allows the evaluator to gain enough confidence in the correctness of particular evaluation evidence without analysing the whole evidence. The reason for sampling is to conserve resources while maintaining an adequate level of assurance. Sampling of the evidence can provide two possible outcomes:

a) The subset reveals no errors, allowing the evaluator to have some confidence that the entire set is correct.

b) The subset reveals errors and therefore the validity of the entire set is called into question. Even the resolution of all errors that were found may be insufficient to provide the evaluator the necessary confidence and as a result the evaluator may have to increase the size of the subset, or stop using sampling for this particular evidence.

2096 Sampling is a technique which can be used to reach a reliable conclusion if a set of evidence is relatively homogeneous in nature, e.g. if the evidence has been produced during a well defined process.

2097 Sampling in the cases identified in the CC, and in cases specifically covered in CEM work items, is recognised as a cost-effective approach to performing evaluator actions. Sampling in other areas is permitted only in exceptional cases, where performance of a particular activity in its entirety would require effort disproportionate to the other evaluation activities, and where this would not add correspondingly to assurance. In such cases a rationale for the use of sampling in that area will need to be made. Neither the fact that the TOE is large and complex, nor that it has many security functional requirements, is sufficient justification, since evaluations of large, complex TOEs can be expected to require more effort. Rather it is intended that this exception be limited to cases such as that where the TOE development

approach yields large quantities of material for a particular CC requirement that would normally all need to be checked or examined, and where such an action would not be expected to raise assurance correspondingly.

2098 Sampling needs to be justified taking into account the possible impact on the security objectives and threats of the TOE. The impact depends on what might be missed as a result of sampling. Consideration also needs to be given to the nature of the evidence to be sampled, and the requirement not to diminish or ignore any security functions.

2099 It should be recognised that sampling of evidence directly related to the implementation of the TOE (e.g. developer test results) requires a different approach to sampling, then sampling related to the determination of whether a process is being followed. In many cases the evaluator is required to determine that a process is being followed, and a sampling strategy is recommended. The approach for sampling a developer's test results will differ. This is because the former case is concerned with ensuring that a process is in place, and the latter deals with determining correct implementation of the TOE. Typically, larger sample sizes should be analysed in cases related to the correct implementation of the TOE than would be necessary to ensure that a process is in place.

2100 In certain cases it may be appropriate for the evaluator to give greater emphasis to the repetition of developer testing. For example if the independent tests left for the evaluator to perform would be only superficially different from those included in an extensive developer test set (possibly because the developer has performed more testing than necessary to satisfy the Coverage (ATE_COV) and Depth (ATE_DPT) criteria) then it would be appropriate for the evaluator to give greater focus to the repetition of developer tests. Note that this does not necessarily imply a requirement for a high percentage sample for repetition of developer tests; indeed, given an extensive developer test set, the evaluator may be able to justify a low percentage sample.

2101 Where the developer has used an automated test suite to perform functional testing, it will usually be easier for the evaluator to re-run the entire test suite rather than repeat only a sample of developer tests. However the evaluator does have an obligation to check that the automatic testing does not give misrepresentative results. The implication is thus that this check must be performed for a sample of the automatic test suite, with the principles for selecting some tests in preference to others and ensuring a sufficient sample size applying equally in this case.

2102 The following principles should be followed whenever sampling is performed:

a) Sampling should not be random, rather it should be chosen such that it is representative of all of the evidence. The sample size and composition must always be justified.

b) When sampling relates to the correct implementation of the TOE, the sample should be representative of all aspects relevant to the areas that are sampled. In particular, the selection should cover a variety of components, interfaces, developer and operational sites (if more than one is involved) and hardware platform types (if more than one is involved). The sample size should be commensurate with the cost effectiveness of the evaluation and will depend on a number of TOE dependent factors (e.g. the size and complexity of the TOE, the amount of documentation).

c) Also, when sampling relates to specifically gaining evidence that the developer testing is repeatable and reproducible the sample used must be sufficient to represent all distinct aspects of developer testing, such as different test regimes. The sample used must be sufficient to detect any systematic problem in the developer's functional testing process. The evaluator contribution resulting from the combination of repeating developer tests and performing independent tests must be sufficient to address the major points of concern for the TOE.

d) Where sampling relates to gaining evidence that a process (e.g. visitor control or design review) the evaluator should sample sufficient information to gain reasonable confidence that the procedure is being followed.

e) The sponsor and developer should not be informed in advance of the exact composition of the sample, subject to ensuring timely delivery of the sample and supporting deliverable, e.g. test harnesses and equipment to the evaluator in accordance with the evaluation schedule.

f) The choice of the sample should be free from bias to the degree possible (one should not always choose the first or last item). Ideally the sample selection should be done by someone other than the evaluator.

2103 Errors found in the sample can be categorised as being either systematic or sporadic. If the error is systematic, the problem should be corrected and a complete new sample taken. If properly explained, sporadic errors might be solved without the need for a new sample, although the explanation should be confirmed. The evaluator should use judgement in determining whether to increase the sample size or use a different sample.

A.3 Dependencies

2104 In general it is possible to perform the required evaluation activities, sub-activities, and actions in any order or in parallel. However, there are different kinds of dependencies which have to be considered by the evaluator. This section provides general guidance on dependencies between different activities, sub-activities, and actions.

A.3.1 Dependencies between activities

2105 For some cases the different assurance classes may recommend or even require a sequence for the related activities. A specific instance is the ST activity. The ST evaluation activity is started prior to any TOE evaluation activities since the ST provides the basis and context to perform them. However, a final verdict on the ST evaluation may not be possible until the TOE evaluation is complete, since changes to the ST may result from activity findings during the TOE evaluation.

A.3.2 Dependencies between sub-activities

2106 Dependencies identified between components in CC Part 3 have to be considered by the evaluator. Most dependencies are one way, e.g. Evaluation of sub-activity (AVA_VAN.1) claims a dependency on Evaluation of sub-activity (ADV_FSP.1) and Evaluation of sub-activity (AGD_OPE.1). There are also instances of mutual dependencies, where both components depend on each other. An example of this is Evaluation of sub-activity (ATE_FUN.1) and Evaluation of sub-activity (ATE_COV.1).

2107 A sub-activity can be assigned a pass verdict normally only if all those sub-activities are successfully completed on which it has a one-way dependency. For example, a pass verdict on Evaluation of sub-activity (AVA_VAN.1) can normally only be assigned if the sub-activities related to Evaluation of sub-activity (ADV_FSP.1) and Evaluation of sub-activity (AGD_OPE.1) are assigned a pass verdict too. In the case of mutual dependency the ordering of these components is down to the evaluator deciding which sub-activity to perform first. Note this indicates that pass verdicts can normally only be assigned once both sub-activities have been successful.

2108 So when determining whether a sub-activity will impact another sub-activity, the evaluator should consider whether this activity depends on potential evaluation results from any dependent sub-activities. Indeed, it may be the case that a dependent sub-activity will impact this sub-activity, requiring previously completed evaluator actions to be performed again.

2109 A significant dependency effect occurs in the case of evaluator-detected flaws. If a flaw is identified as a result of conducting one sub-activity, the assignment of a pass verdict to a dependent sub-activity may not be possible until all flaws related to the sub-activity upon which it depends are resolved.

A.3.3 Dependencies between actions

2110 It may be the case, that results which are generated by the evaluator during one action are used for performing another action. For example, actions for completeness and consistency cannot be completed until the checks for content and presentation have been completed. This means for example that the evaluator is recommended to evaluate the PP/ST rationale after evaluating the constituent parts of the PP/ST.

A.4 Site Visits

2111 This section provides general guidance on site visits. Specific and detailed information is given in work units for those activities where site visits are performed:

a) CM capabilities (ALC_CMC).n (with n>=3);

b) Delivery (ALC_DEL);

c) Development security (ALC_DVS).

2112 A development site visit is a useful means whereby the evaluator determines whether procedures are being followed in a manner consistent with that described in the documentation.

2113 Reasons for visiting sites include:

a) to observe the use of the CM system as described in the CM plan;

b) to observe the practical application of delivery procedures;

c) to observe the application of security measures during development.

2114 During an evaluation it is often necessary that the evaluator will meet the developer more than once and it is a question of good planning to combine the site visit with another meeting to reduce costs. For example one might combine the site visits for configuration management, for the developer's security and for delivery. It may also be necessary to perform more than one site visit to the same site to allow the checking of all development phases. It should be considered that development could occur at multiple facilities within a single building, multiple buildings at the same site, or at multiple sites.

2115 The first site visit should be scheduled early during the evaluation. In the case of an evaluation which starts during the development phase of the TOE, this will allow corrective actions to be taken, if necessary. In the case of an evaluation which starts after the development of the TOE, an early site visit could allow corrective measures to be put in place if serious deficiencies in the applied procedures emerge. This avoids unnecessary evaluation effort.

2116 Interviews are also a useful means of determining whether the written procedures reflect what is done. In conducting such interviews, the evaluator should aim to gain a deeper understanding of the analysed procedures at the development site, how they are used in practise and whether they are being applied as described in the provided evaluation evidence. Such interviews complement but do not replace the examination of evaluation evidence.

2117 To prepare for the site visit a checklist, based on the evaluation evidence provided should be generated by the evaluator. The results of the site visit should be recorded.

2118 Site visits may not be deemed necessary if e.g. the development site has recently been visited for another TOE evaluation or particular ISO 9000 procedures were confirmed as being followed. Other approaches to gain confidence should be considered that provide an equivalent level of assurance (e.g. to analyse evaluation evidence). Any decision not to make a visit should be determined in consultation with the overseer.

A.5 TOE Boundary

2119 The identity of what is evaluated will appear in the ETR, on the certificate, in the ST, and on the list of evaluated products. Although products are typically bought and sold, evaluations are concerned with TOEs. In cases where the developer of the product is also the developer of the evaluation evidence (i.e. the sponsor), this distinction is unnecessary. But because these roles may be filled by different parties, the following were agreed as the basis of definitions used in the CEM, along with their interrelationships and effects upon evaluations and certification.

A.5.1 Product and system

2120 The product is the collection of hardware and/or software that is available for use. Some purveyors might bundle a collection of products (e.g. a word processor, spreadsheet, and graphics application) into yet another product (e.g. an office automation system). But, provided that it is available for use, either by the public, by other manufacturers, or by limited customers, the resulting collection is considered to be a product.

2121 A system consists of one or more products in a known operational environment. The main difference between a product evaluation and a system evaluation is that, for a system evaluation, the evaluator takes into account the actual environment instead of theorising a hypothetical one, as done for a product evaluation.

A.5.2 TOE

2122 The TOE is the entity that is evaluated as defined by the ST. While there are cases where a TOE makes up the entire product, this need not be the case. The TOE may be a product, a part of a product, a set of products, a unique technology never to be made into a product, or combinations of all of these, in a specific configuration or set of configurations. This specific configuration or set of configurations is called the evaluated configuration. The ST clearly describes the relation between the TOE and any associated products.

2123 This evaluated configuration is identified in sufficient detail to differentiate hardware included in the evaluated configuration from hardware that is not included in the evaluated configuration, though it might be available as part of the product upon which the TOE is based. This identification makes it apparent to potential customers what product must be purchased, and what configuration options must be used, in order for the TOE to run securely.

A.5.3 TSF

2124 The TSF is the collection of those parts of the TOE that must be relied upon to enforce the security of the TOE as defined by the SR. There may be parts within the TOE that contribute nothing to the security of the TOE as defined by the ST; consequently, such parts would not be part of the TSF.

2125 The hardware portions of the TSF are described at a level of detail commensurate with the assurance requirements related to the relevant development documentation and the testing documentation. The level of hardware identification is determined by the impact that the hardware features have upon the security functions and assurances being claimed.

A.5.4 Evaluation

2126 An implicit assumption for all evaluations is that the TOE is (by definition) the product or system in its evaluated configuration; this assumption need not be explicitly included in the list of assumptions for the evaluation. The TOE undergoes the scrutiny of the evaluation: analysis is performed only within the evaluated configuration, testing is performed upon this evaluated configuration, exploitable vulnerabilities are identified in this evaluated configuration, and assumptions are relevant only in the evaluated configuration. The ease with which the TOE can exit this configuration is important, and must be considered where Operational user guidance (AGD_OPE) is called up. This will look at the robustness of the TOE configuration, and the impact of any accidental or intentional deviations from it that may occur without detection.

2127 The following example provides three TOEs, all of which are based upon the same virtual private networking (VPN) firewall product, but which yield different evaluation results because of the differences in the STs.

A VPN-firewall which is configured in such a way that the VPN functionality is turned off. All threats in the ST are concerned with access to the safe network from the unsafe network.

The TOE is the VPN-firewall configured in such a way that the VPN functionality is turned off. If the administrator were to configure the firewall such that some or all VPN functions were enabled, the product would not be in an evaluated configuration; it would therefore be considered to be unevaluated, and so nothing could be stated about its security.

A VPN-firewall, where all threats in the ST are concerned with access to the safe network from the unsafe network.

The TOE is the entire VPN-firewall. The VPN functions are part of the TOE, so one of the things to be determined during the evaluation would be whether there are means to gain access to the safe network from the unsafe network through the VPN functions.

A VPN-firewall, where all threats in the ST are concerned with either access to the safe network from the unsafe network or confidentiality of traffic on the unsafe network.

The TOE is the entire VPN-firewall. The VPN functions are part of the TOE, so one of the things to be determined during the evaluation would be whether the VPN functions permit the realisation of any of the threats described in the ST.

A.5.5 Certification

2128 From the earlier paragraphs, it is clear that evaluating the same product with different STs leads to different TOEs with different TSFs. Consequently, the Certificates, ETR, the STs, and the entries in the Evaluated Products List will have to differ among the evaluations to be of any use to potential customers.

2129 Note that, for the above example of three different firewall evaluations, the apparent differences between these Certificates would be subtle, as the three VPN-firewalls would all lead to certificates identifying the TOE as:

2130 *The XYZ Firewall product, as described in the Evaluated Configuration identified in Security Target #ABC.*

2131 with a different identifier for each ST ABC.

2132 Therefore, the evaluator has to ensure that the ST adequately describes the TOE in terms of what functionality is within the scope of the evaluation. A clear explanation is vital because prospective customers of evaluated products will consult the STs of the products that they are considering to buy in order to determine which security functionality of those products have been evaluated.

A.6 Scheme Responsibilities

2133 This CEM describes the minimum technical work that evaluations conducted under oversight (scheme) bodies must perform. However, it also recognises (both explicitly and implicitly) that there are activities or methods upon which mutual recognition of evaluation results do not rely. For the purposes of thoroughness and clarity, and to better delineate where the CEM ends and an individual scheme's methodology begins, the following matters are left up to the discretion of the schemes. Schemes may choose to provide the following, although they may choose to leave some unspecified. (Every effort has been made to ensure this list is complete; evaluators encountering a subject neither listed here nor addressed in the CEM should consult with their evaluation schemes to determine under whose auspices the subject falls.)

2134 The matters that schemes may choose to specify include:

a) what is required in ensuring that an evaluation was done sufficiently - every scheme has a means of verifying the technical competence,

understanding of work and the work of its evaluators, whether by requiring the evaluators to present their findings to the oversight body, by requiring the oversight body to redo the evaluator's work, or by some other means that assures the scheme that all evaluation bodies are adequate and comparable;

b) process for disposing of evaluation evidence upon completion of an evaluation;

c) any requirements for confidentiality (on the part of the evaluator and the non-disclosure of information obtained during evaluation);

d) the course of action to be taken if a problem is encountered during the evaluation (whether the evaluation continues once the problem is remedied, or the evaluation ends immediately and the remedied product must be re-submitted for evaluation);

e) any specific (natural) language in which documentation must be provided;

f) any recorded evidence that must be submitted in the ETR - this CEM specifies the minimum to be reported in an ETR; however, individual schemes may require additional information to be included;

g) any additional reports (other than the ETR) required from the evaluators -for example, testing reports;

h) any specific ORs that may be required by the scheme, including the structure, recipients, etc. of any such ORs;

i) any specific content structure of any written report as a result from an ST evaluation - a scheme may have a specific format for all of its reports detailing results of an evaluation, be it the evaluation of a TOE or of an ST;

j) any additional PP/ST identification information required;

k) any activities to determine the suitability of explicitly-stated requirements in an ST;

l) any requirements for provision of evaluator evidence to support re-evaluation and re-use of evidence;

m) any specific handling of scheme identifiers, logos, trademarks, etc.;

n) any specific guidance in dealing with cryptography;

o) handling and application of scheme, national and international interpretations;

p) a list or characterisations of suitable alternative approaches to testing where testing is infeasible;

q) the mechanism by which an overseer can determine what steps an evaluator took while testing;

r) preferred test approach (if any): at internal interface or at external interface;

s) a list or characterisation of acceptable means of conducting the evaluator's vulnerability analysis (e.g. flaw hypothesis methodology);

t) information regarding any vulnerabilities and weaknesses to be considered;

B Vulnerability Assessment (AVA) (normative)

2135 This annex provides an explanation of the AVA_VAN criteria and examples of their application. This annex does not define the AVA criteria, this definition can be found in CC Part 3 Section Class AVA: Vulnerability assessment.

2136 This annex consists of 2 major parts:

a) *Guidance for completing an independent vulnerability analysis*. This is summarised in section B.1, and described in more detail in section B.2 . These sections describe how an evaluator should approach the construction of an independent Vulnerability Analysis.

b) How to characterise and use assumed Attack Potential of an attacker. This is described in sections B.3 to B.6. These sections provide an example of describe how an attack potential can be characterised and should be used, and provide examples.

B.1 What is Vulnerability Analysis

2137 The purpose of the vulnerability assessment activity is to determine the existence and exploitability of flaws or weaknesses in the TOE in the operational environment. This determination is based upon analysis performed the evaluator, and is supported by evaluator testing.

2138 At the lowest levels of Vulnerability analysis (AVA_VAN) the evaluator simply performs a search of publicly available information to identify any known weaknesses in the TOE, while at the higher levels the evaluator performs an structured analysis of the TOE evaluation evidence.

2139 There are two main factors in performing a vulnerability analysis, namely;

a) the identification of potential vulnerabilities;

b) penetration testing to determine whether the potential vulnerabilities are exploitable in the operational environment of the TOE.

2140 The identification of vulnerabilities can be further decomposed into the evidence to be searched and how hard to search that evidence to identify potential vulnerabilities. In a similar manner, the penetration testing can be further decomposed into analysis of the potential vulnerability to identify attack methods and the demonstration of the attack methods.

2141 These main factors are iterative in nature, i.e. penetration testing of potential vulnerabilities may lead to the identification of further potential vulnerabilities. Hence, these are performed as a single vulnerability analysis activity.

B.2 Evaluator construction of a Vulnerability Analysis

2142 The evaluator vulnerability analysis is to determine that the TOE is resistant to penetration attacks performed by an attacker possessing a basic (for AVA_VAN.1 and AVA_VAN.2), extended-basic (for AVA_VAN.3), moderate (for AVA_VAN.4) or high (for AVA_VAN.5) attack potential. The evaluator first assesses the exploitability of all identified potential vulnerabilities. This is accomplished by conducting penetration testing. The evaluator should assume the role of an attacker with a basic (for AVA_VAN.1 and AVA_VAN.2), extended-basic (for AVA_VAN.3), moderate (for AVA_VAN.4) or high (for AVA_VAN.5) attack potential when attempting to penetrate the TOE.

2143 The evaluator considers potential vulnerabilities encountered by the evaluator during the conduct of other evaluation activities. The evaluator penetration testing determining TOE resistance to these potential vulnerabilities should be performed assuming the role of an attacker with a basic (for AVA_VAN.1 and AVA_VAN.2), extended-basic (for AVA_VAN.3), moderate (for AVA_VAN.4) or high (for AVA_VAN.5) attack potential.

2144 However, vulnerability analysis should not be performed as an isolated activity. It is closely linked with ADV and AGD. The evaluator performs these other evaluation activities with a focus on identifying potential vulnerabilities or “areas of concern”. Therefore, evaluator familiarity with the generic vulnerability guidance (provided in Section B.2.1) is required.

B.2.1 Generic vulnerability guidance

2145 The following four categories provide discussion of generic vulnerabilities.

B.2.1.1 Bypassing

2146 Bypassing includes any means by which an attacker could avoid security enforcement, by:

a) exploiting the capabilities of interfaces to the TOE, or of utilities which can interact with the TOE;

b) inheriting privileges or other capabilities that should otherwise be denied;

c) (where confidentiality is a concern) reading sensitive data stored or copied to inadequately protected areas.

2147 Each of the following should be considered (where relevant) in the evaluator's independent vulnerability analysis.

a) Attacks based on exploiting the capabilities of interfaces or utilities generally take advantage of the absence of the required security enforcement on those interfaces. For example, gaining access to

functionality that is implemented at a lower level than that at which access control is enforced. Relevant items include:

1) changing the predefined sequence of invocation of TSFI;

2) invoking an additional TSFI;

3) using a component in an unexpected context or for an unexpected purpose;

4) using implementation detail introduced in less abstract representations;

5) using the delay between time of access check and time of use.

b) Changing the predefined sequence of invocation of components should be considered where there is an expected order in which interfaces to the TOE (e.g. user commands) are called to invoke a TSFI (e.g. opening a file for access and then reading data from it). If a TSFI is invoked through one of the TOE interfaces (e.g. an access control check), the evaluator should consider whether it is possible to bypass the control by performing the call at a later point in the sequence or by missing it out altogether.

c) Executing an additional component (in the predefined sequence) is a similar form of attack to the one described above, but involves the calling of some other TOE interface at some point in the sequence. It can also involve attacks based on interception of sensitive data passed over a network by use of network traffic analysers (the additional component here being the network traffic analyser).

d) Using a component in an unexpected context or for an unexpected purpose includes using an unrelated TOE interface to bypass the TSF by using it to achieve a purpose that it was not designed or intended to achieve. Covert channels are an example of this type of attack. The use of undocumented interfaces (which may be insecure) also falls into this category (these include undocumented support and help facilities).

e) Using implementation detail introduced in lower representations again includes the use of covert channels in which an attacker takes advantage of additional functions, resources or attributes that are introduced to the TOE as a consequence of the refinement process (e.g. use of a lock variable as a covert channel). Additional functionality may also include test harness code contained in software modules.

f) Using the delay between time of check and time of use includes scenarios where an access control check is made and access granted, and an attacker is subsequently able to create conditions in which, had they applied at the time the access check was made, would have

caused the check to fail. An example would be a user creating a background process to read and send highly sensitive data to the user's terminal, and then logging out and logging back in again at a lower sensitivity level. If the background process is not terminated when the user logs off, the MAC checks would have been effectively bypassed.

g) Attacks based on inheriting privileges are generally based on illicitly acquiring the privileges or capabilities of some privileged component, usually by exiting from it in an uncontrolled or unexpected manner. Relevant items include:

1) executing data not intended to be executable, or making it executable;

2) generating unexpected input for a component;

3) invalidating assumptions and properties on which lower-level components rely.

h) Executing data not intended to be executable, or making it executable includes attacks involving viruses (e.g. putting executable code or commands in a file which are automatically executed when the file is edited or accessed, thus inheriting any privileges the owner of the file has).

i) Generating unexpected input for a component can have unexpected effects which an attacker could take advantage of. For example, if the TSF could be bypassed if a user gains access to the underlying operating system, it may be possible to gain such access following the login sequence by exploring the effect of hitting various control or escape sequences whilst a password is being authenticated.

j) Invalidating assumptions and properties on which lower level components rely includes attacks based on breaking out of the constraints of an application to gain access to an underlying operating system in order to bypass the TSF of an application. In this case the assumption being invalidated is that it is not possible for a user of the application to gain such access. A similar attack can be envisaged on an application on an underlying database management system: again the TSF could be bypassed if an attacker can break out of the constraints of the application.

k) Attacks based on reading sensitive data stored in inadequately protected areas (applicable where confidentiality is a concern) include the following issues which should be considered as possible means of gaining access to sensitive data:

1) disk scavenging;

2) access to unprotected memory;

3) exploiting access to shared writable files or other shared resources (e.g. swap files);

4) Activating error recovery to determine what access users can obtain. For example, after a crash an automatic file recovery system may employ a lost and found directory for headerless files, which are on disc without labels. If the TOE implements mandatory access controls, it is important to investigate at what security level this directory is kept (e.g. at system high), and who has access to this directory.

B.2.1.2 Tampering

2148 Tampering includes any attack based on an attacker attempting to influence the behaviour of the TSF (i.e. corruption or de-activation), for example by:

a) accessing data on whose confidentiality or integrity the TSF relies;

b) forcing the TOE to cope with unusual or unexpected circumstances;

c) disabling or delaying security enforcement.

2149 Each of the following should be considered (where relevant) in the evaluator's independent vulnerability analysis.

a) Attacks based on accessing data on whose confidentiality or integrity are protected include:

1) reading, writing or modifying internal data directly or indirectly;

2) using a component in an unexpected context or for an unexpected purpose;

3) using interfaces between components that are not visible at a higher level of abstraction.

b) Reading, writing or modifying internal data directly or indirectly includes the following types of attack which should be considered:

1) reading “secrets” stored internally, such as user passwords;

2) spoofing internal data that security enforcing mechanisms rely upon;

3) modifying environment variables (e.g. logical names), or data in configuration files or temporary files.

c) It may be possible to deceive a trusted process into modifying a protected file that it wouldn't normally access.

d) The evaluator should also consider the following “dangerous features”:

1) source code resident on the TOE along with a compiler (for instance, it may be possible to modify the login source code);

2) an interactive debugger and patch facility (for instance, it may be possible to modify the executable image);

3) the possibility of making changes at device controller level, where file protection does not exist;

4) diagnostic code which exists in the source code and that may be optionally included;

5) developer's tools left in the TOE.

e) Using a component in an unexpected context or for an unexpected purpose includes (for example), where the TOE is an application built upon an operating system, users exploiting knowledge of a word processor package or other editor to modify their own command file (e.g. to acquire greater privileges).

f) Using interfaces between components which are not visible at a higher level of abstraction includes attacks exploiting shared access to resources, where modification of a resource by one component can influence the behaviour of another (trusted) component, e.g. at source code level, through the use of global data or indirect mechanisms such as shared memory or semaphores.

g) Attacks based on forcing the TOE to cope with unusual or unexpected circumstances should always be considered. Relevant items include:

1) generating unexpected input for a component;

2) invalidating assumptions and properties on which lower-level components rely.

h) Generating unexpected input for a component includes investigating the behaviour of the TOE when:

1) command input buffers overflow (possibly “crashing the stack” or overwriting other storage, which an attacker may be able to take advantage of, or forcing a crash dump that may contain sensitive information such as clear-text passwords);

2) invalid commands or parameters are entered (including supplying a read-only parameter to an interface which expects to return data via that parameter);

3) an end-of-file marker (e.g. CTRL-Z or CTRL-D) or null character is inserted in an audit trail.

i) Invalidating assumptions and properties on which lower-level components rely includes attacks taking advantage of errors in the source code where the code assumes (explicitly or implicitly) that security relevant data is in a particular format or has a particular range of values. In these cases the evaluator should determine whether they can invalidate such assumptions by causing the data to be in a different format or to have different values, and if so whether this could confer advantage to an attacker.

j) The correct behaviour of the TSF may be dependent on assumptions that are invalidated under extreme circumstances where resource limits are reached or parameters reach their maximum value. The evaluator should consider (where practical) the behaviour of the TOE when these limits are reached, for example:

1) changing dates (e.g. examining how the TOE behaves when a critical date threshold is passed);

2) filling discs;

3) exceeding the maximum number of users;

4) filling the audit log;

5) saturating security alarm queues at a console;

6) overloading various parts of a multi-user TOE which relies heavily upon communications components;

7) swamping a network, or individual hosts, with traffic;

8) filling buffers or fields.

k) Attacks based on disabling or delaying security enforcement include the following items:

1) using interrupts or scheduling functions to disrupt sequencing;

2) disrupting concurrence;

3) using interfaces between components which are not visible at a higher level of abstraction.

l) Using interrupts or scheduling functions to disrupt sequencing includes investigating the behaviour of the TOE when:

1) a command is interrupted (with CTRL-C, CTRL-Y, etc.);

2) a second interrupt is issued before the first is acknowledged.

m) The effects of terminating security critical processes (e.g. an audit daemon) should be explored. Similarly, it may be possible to delay the logging of audit records or the issuing or receipt of alarms such that it is of no use to an administrator (since the attack may already have succeeded).

n) Disrupting concurrence includes investigating the behaviour of the TOE when two or more subjects attempt simultaneous access. It may be that the TOE can cope with the interlocking required when two subjects attempt simultaneous access, but that the behaviour becomes less well defined in the presence of further subjects. For example, a critical security process could be put into a resource-wait state if two other processes are accessing a resource which it requires.

o) Using interfaces between components which are not visible at a higher level of abstraction may provide a means of delaying a time-critical trusted process.

B.2.1.3 Direct attacks

2150 Direct attack includes the identification of any penetration tests necessary to test the strength of permutational or probabilistic mechanism and other mechanisms to ensure they withstand direct attack.

2151 For example, it may be a flawed assumption that a particular implementation of a pseudo-random number generator will possess the required entropy necessary to seed the security mechanism.

2152 Where a probabilistic or permutational mechanism relies on selection of security attribute value (e.g. selection of password length) or entry of data by a human user (e.g. choice of password), the assumptions made should reflect the worst case.

2153 Probabilistic or permutational mechanisms should be identified during examination of evaluation evidence required as input to this sub-activity (security target, functional specification, TOE design and implementation representation subset) and any other TOE (e.g. guidance) documentation may identify additional probabilistic or permutational mechanisms.

2154 Where the design evidence or guidance includes assertions or assumptions (e.g. about how many authentication attempts are possible per minute), the evaluator should independently confirm that these are correct. This may be achieved through testing or through independent analysis.

2155 Direct attacks reliant upon a weakness in a cryptographic algorithm should not be considered under Vulnerability analysis (AVA_VAN), as this is outside the scope of the CC. Correctness of the implementation of the cryptographic algorithm is considered during the 11 and 14 activities.

B.2.1.4 Misuse

2156 Misuse may arise from:

a) incomplete guidance documentation;

b) unreasonable guidance;

c) unintended misconfiguration of the TOE;

d) forced exception behaviour of the TOE.

2157 If the guidance documentation is incomplete the user may not know how to operate the TOE in accordance with the TSP. The evaluator should apply familiarity with the TOE gained from performing other evaluation activities to determine that the guidance is complete. In particular, the evaluator should consider the functional specification. The TSF described in this document should be described in the guidance as required to permit secure administration and use through the TSFI available to human users. In addition, the different modes of operation should be considered to ensure that guidance is provided for all modes of operation.

2158 The evaluator may, as an aid, prepare an informal mapping between the guidance and these documents. Any omissions in this mapping may indicate incompleteness.

2159 The guidance is considered to be unreasonable if it makes demands on the TOE's usage or operational environment that are inconsistent with the ST or unduly onerous to maintain security.

2160 A TOE may use a variety of ways to assist the consumer in effectively using that TOE in accordance with the TSP and prevent unintentional misconfiguration. A TOE may employ functionality (features) to alert the consumer when the TOE is in a state that is inconsistent with the TSP, whilst other TOEs may be delivered with enhanced guidance containing suggestions, hints, procedures, etc. on using the existing security features most effectively; for instance, guidance on using the audit feature as an aid for detecting when the TSP is being compromised; namely insecure.

2161 The evaluator considers the TOE's functionality, its purpose and security objectives for the operational environment to arrive at a conclusion of whether or not there is reasonable expectation that use of the use of the guidance would permit transition into an insecure state to be detected in a timely manner.

2162 The potential for the TOE to enter into insecure states may be determined using the evaluation deliverables, such as the ST, the functional specification and any other design representations provided as evidence for components included in the assurance package for the TOE (e.g. the TOE/TSF design specification if a component from TOE design (ADV_TDS) is included).

2163 Instances of forced exception behaviour of the TSF could include, but not limited to, the following:

a) behaviour of the TOE when start-up, closedown or error recovery is activated;

b) behaviour of the TOE under extreme circumstances (sometimes termed overload or asymptotic behaviour), particularly where this could lead to the de-activation or disabling of parts of the TSF;

c) any potential for unintentional misconfiguration or insecure use arising from attacks noted in the section on tampering above.

B.2.2 Identification of Potential Vulnerabilities

2164 Potential vulnerabilities may be identified by the evaluator during different activities. They may become apparent during an evaluation activity or they may be identified as a result of analysis of evidence to search for vulnerabilities.

B.2.2.1 Encountered

2165 The encountered identification of vulnerabilities is where potential vulnerabilities are identified by the evaluator during the conduct of evaluation activities, i.e. the evidence are not being analysed with the express aim of identifying potential vulnerabilities.

2166 The encountered method of identification is dependent on the evaluator's experience and knowledge; which is monitored and controlled by the Certification Authority. It is not reproducible in approach, but will be documented to ensure repeatability of the conclusions from the reported potential vulnerabilities.

2167 There is no formal analysis criteria required for this method. Potential vulnerabilities are identified from the evidence provided as a result of knowledge and experience. However, this method of identification is not constrained to any particular subset of evidence.

2168 Evaluator is assumed to have knowledge of the TOE-type technology and known security flaws as documented in the public domain. The level of knowledge assumed is that which can be gained from a security e-mail list relevant to the TOE type, the regular bulletins (bug, vulnerability and security flaw lists) published by those organisations researching security issues in products and technologies in widespread use. This knowledge is not expected to extend to specific conference proceedings or detailed theses produced by university research. However, to ensure the knowledge applied is up to date, the evaluator may need to perform a search of public domain material.

2169 Examples of how these may arise (how the evaluator may encounter potential vulnerabilities):

a) while the evaluator is examining some evidence, it sparks a memory of a potential vulnerability identified in a similar product type, that the evaluator believes to also be present in the TOE under evaluation;

b) while examining some evidence, the evaluator spots a flaw in the specification of an interface, that reflects a potential vulnerability.

2170 This may include becoming aware of a potential vulnerability in a TOE through reading about generic vulnerabilities in a particular product type in an IT security publication or on a security e-mail list to which the evaluator is subscribed.

2171 Attack methods can be developed directly from these potential vulnerabilities. Therefore, the encountered potential vulnerabilities are collated at the time of producing penetration tests based on the evaluator's vulnerability analysis. There is no explicit action for the evaluator to encounter potential vulnerabilities. Therefore, the evaluator is directed through an implicit action specified in AVA_VAN.1.2E and AVA_VAN.*.4E.

2172 Current information regarding public domain vulnerabilities and attacks may be provided to the evaluator by, for example, an overseer such as the evaluation authority. This information is to be taken into account by the evaluator when collating encountered vulnerabilities and attack methods when developing penetration tests.

B.2.2.2 Analysis

2173 The following types of analysis are presented in terms of the evaluator actions.

B.2.2.2.1 Unstructured Analysis

2174 The unstructured analysis to be performed by the evaluator (for Evaluation of sub-activity (AVA_VAN.2)) permits the evaluator to consider the generic vulnerabilities (as discussed in B.2.1). The evaluator will also apply their experience and knowledge of flaws in similar technology types.

B.2.2.2.2 Focused

2175 During the conduct of evaluation activities the evaluator may also identify areas of concern. These are specific portions of the TOE evidence that the evaluator has some reservation about, although the evidence meets the requirements for the activity with which the evidence is associated. For example, a particular interface specification looks particularly complex, and therefore may be prone to error either in the construction of the TOE or in the operation of the TOE. There is no potential vulnerability apparent at this stage, further investigation is required. This is beyond the bounds of encountered, as further investigation is required.

2176 Difference between potential vulnerability and area of concern:

a) Potential vulnerability - The evaluator knows a method of attack that can be used to exploit the weakness or the evaluator knows of vulnerability information that is relevant to the TOE.

b) Area of concern - The evaluator may be able to discount concern as a potential vulnerability based on information provided elsewhere. While reading interface specification, the evaluator identifies that due to the extreme (unnecessary) complexity of an interface a potential vulnerability may lay within that area, although it is not apparent through this initial examination.

2177 The focused approach to the identification of vulnerabilities is an analysis of the evidence with the aim of identifying any potential vulnerabilities evident through the contained information. It is an unstructured analysis, as the approach is not predetermined. This approach to the identification of potential vulnerabilities can be used during the independent vulnerability analysis required by Evaluation of sub-activity (AVA_VAN.3).

2178 This analysis can be achieved through different approaches, that will lead to commensurate levels of confidence. None of the approaches have a rigid format for the examination of evidence to be performed.

2179 The approach taken is directed by the results of the evaluator's assessment of the evidence to determine it meets the requirements of the AVA/AGD sub-activities. Therefore, the investigation of the evidence for the existence of potential vulnerabilities may be directed by any of the following:

a) areas of concern identified during examination of the evidence during the conduct of evaluation activities;

b) reliance on particular functionality to provide separation, identified during the analysis of the architectural design (as in Evaluation of sub-activity (ADV_ARC.1)), requiring further analysis to determine it cannot be bypassed;

c) representative examination of the evidence to hypothesise potential vulnerabilities in the TOE.

2180 The evaluator will report what actions were taken to identify potential vulnerabilities in the evidence. However, the evaluator may not be able to describe the steps in identifying potential vulnerabilities before the outset of the examination. The approach will evolve as a result of the outcome of evaluation activities.

2181 The areas of concern may arise from examination of any of the evidence provided to satisfy the SARs specified for the TOE evaluation. The information publicly accessible is also considered.

2182 The activities performed by the evaluator can be repeated and the same conclusions, in terms of the level of assurance in the TOE, can be reached although the steps taken to achieve those conclusions may vary. As the

evaluator is documenting the form the analysis took, the actual steps taken to achieve those conclusions are also reproducible.

B.2.2.2.3 Methodical

2183 The methodical analysis approach takes the form of a structured examination of the evidence. This method requires the evaluator to specify the structure and form the analysis will take (i.e. the manner in which the analysis is performed is predetermined, unlike the focused identification method). The method is specified in terms of the information that will be considered and how/why it will be considered. This approach to the identification of potential vulnerabilities can be used during the independent vulnerability analysis required by Evaluation of sub-activity (AVA_VAN.4) and Evaluation of sub-activity (AVA_VAN.5).

2184 This analysis of the evidence is deliberate and pre-planned in approach, considering all evidence identified as an input into the analysis.

2185 All evidence provided to satisfy the (ADV) assurance requirements specified in the assurance package are used as input to the potential vulnerability identification activity.

2186 The “methodical” descriptor for this analysis has been used in an attempt to capture the characterisation that this identification of potential vulnerabilities is to take an ordered and planned approach. A “method” or “system” is to be applied in the examination. The evaluator is to describe the method to be used in terms of what evidence will be considered, the information within the evidence that is to be examined, the manner in which this information is to be considered; and the hypothesis that is to be generated.

2187 The following provide some examples that a hypothesis may take:

a) consideration of malformed input for interfaces available to an attacker at the external interfaces;

b) examination of a security mechanism, such as domain separation, hypothesising internal buffer overflows leading to degradation of separation;

c) analysis to identify any objects created in the TOE implementation representation that are then not fully controlled by the TSF, and could be used by an attacker to undermine the TSP.

2188 For example, the evaluator may identify that interfaces are a potential area of weakness in the TOE and specify an approach to the analysis that “all interface specifications provided in the functional specification and TOE design will be analysed to hypothesise potential vulnerabilities” and go on to explain the methods used in the hypothesis.

2189 This identification method will provide a plan of attack of the TOE, that would be performed by an evaluator completing penetration testing of

potential vulnerabilities in the TOE. The rationale for the method of identification would provide the evidence for the coverage and depth of exploitation determination that would be performed on the TOE.

B.3 When is attack potential used

B.3.1 Developer

2190 Attack potential is used by a PP/ST author during the development of the PP/ST, in consideration of the threat environment and the selection of assurance components. This may simply be a determination that the attack potential possessed by the assumed attackers of the TOE is generically characterised as basic, extended-basic, moderate or high. Alternatively, the PP/ST may wish to specify particular levels of individual factors assumed to be possessed by attackers. (e.g. the attackers are assumed to be experts in the TOE technology type, with access to specialised equipment.)

2191 The PP/ST author considers the threat profile developed during a risk assessment (outside the scope of the CC, but used as an input into the development of the PP/ST in terms of the Security Problem Definition or in the case of low assurance STs, the objectives statement). Consideration of this threat profile in terms of one of the approaches discussed in the following sections will permit the specification of the attack potential the TOE is to resist.

B.3.2 Evaluator

2192 Attack potential is especially considered by the evaluator in two distinct ways during the ST evaluation and the vulnerability assessment activities.

2193 During the ST evaluation, the evaluator determines whether or not the choice of the assurance requirement components, in particular the components of the AVA class, are commensurate with the threat attack potential (see ASE_REQ.1.4C). Cases where the assurance is not commensurate may mean either that the evaluation will not provide sufficient assurance, or that the evaluation will be unnecessarily onerous.

2194 Attack potential is used by an evaluator during the conduct of the vulnerability analysis sub-activity to determine whether or not the TOE is resistant to attacks assuming a specific attack potential of an attacker. If the evaluator determines that a potential vulnerability is exploitable in the TOE, they have to confirm that it is exploitable considering all aspects of the intended environment, including the attack potential assumed by an attacker.

2195 Therefore, using the information provided in the threat statement of the Security Target, the evaluator determines the minimum attack potential required by an attacker to effect an attack, and arrives at some conclusion about the TOE's resistance to attacks. Table 1 demonstrates the relationship between this analysis and attack potential.

Vulnerability Component	TOE resistant to attacker with attack potential of:	Residual vulnerabilities only exploitable by attacker with attack potential of:
VAN.5	high	*infeasible*
VAN.4	moderate	high
VAN.3	extended-basic	moderate
VAN.2	basic	extended-basic
VAN.1	basic	extended-basic

Table 1 Vulnerability testing and attack potential

2196 The “infeasible” attack potential in the residual vulnerabilities column of the above table represents those potential vulnerabilities that would become exploitable should a countermeasure in the operational environment be removed or the operational environment develop such that the technology required to perform an exploit becomes more widely available. A vulnerability classified as residual in this instance reflects the fact that a known weakness exists in the TOE, but in the current operational environment, with the assumed attack potential, the weakness cannot be exploited.

2197 A vulnerability analysis applies to all TSFI, including ones that access probabilistic or permutational mechanisms. No assumptions are made regarding the correctness of the design and implementation of the TSFI; nor are constraints placed on the attack method or the attacker's interaction with the TOE - if an attack is possible, then it is to be considered during the vulnerability analysis. As shown in Table 1, successful evaluation against a vulnerability assurance component reflects that the TSF is designed and implemented to protect against the required level of threat.

2198 It is not necessary for an evaluator to perform an attack potential calculation for each potential vulnerability. In some cases it is apparent when developing the attack method whether or not the attack potential required to develop and run the attack method is commensurate with that assumed of the attacker in the operational environment. For any vulnerabilities for which an exploitation is determined, the evaluator performs an attack potential calculation to determine that the exploitation is appropriate to the level of attack potential assumed for the attacker.

2199 This material is not intended to select and specify a preferred method. Rather it is to provide alternatives for PP/ST authors and evaluators to consider when assuming a level of attack potential.

2200 The values given in Tables 2, 3, 4 and 5 below are examples and are not mathematically proven. Therefore, the values given in these example tables may need to be adjusted according to the technology type. Guidance from the scheme should be sought.

B.4 Weighted parameters approach

B.4.1 Application of attack potential

2201 Attack potential is a function of expertise, resources and motivation. There are multiple methods of representing and quantifying these factors. Also, there are other factors that are applicable for particular TOE types. The following material presents one method.

B.4.1.1 Treatment of motivation

2202 Motivation is an attack potential factor that can be used to describe several aspects related to the attacker and the assets the attacker desires. Firstly, motivation can imply the likelihood of an attack - one can infer from a threat described as highly motivated that an attack is imminent, or that no attack is anticipated from an un-motivated threat. However, except for the two extreme levels of motivation, it is difficult to derive a probability of an attack occurring from motivation.

2203 Secondly, motivation can imply the value of the asset, monetarily or otherwise, to either the attacker or the asset holder. An asset of very high value is more likely to motivate an attack compared to an asset of little value. However, other than in a very general way, it is difficult to relate asset value to motivation because the value of an asset is subjective - it depends largely upon the value an asset holder places on it.

2204 Thirdly, motivation can imply the expertise and resources with which an attacker is willing to effect an attack. One can infer that a highly motivated attacker is likely to acquire sufficient expertise and resources to defeat the measures protecting an asset. Conversely, one can infer that an attacker with significant expertise and resources is not willing to effect an attack using them if the attacker's motivation is low.

2205 During the course of preparing for and conducting an evaluation, all three aspects of motivation are at some point considered. The first aspect, likelihood of attack, is what may inspire a developer to pursue an evaluation. If the developer believes that the attackers are sufficiently motivated to mount an attack, then an evaluation can provide assurance of the ability of the TOE to thwart the attacker's efforts. Where the operational environment is well defined, for example in a system evaluation, the level of motivation for an attack may be known, and will influence the selection of countermeasures.

2206 Considering the second aspect, an asset holder may believe that the value of the assets (however measured) is sufficient to motivate attack against them. Once an evaluation is deemed necessary, the attacker's motivation is considered to determine the methods of attack that may be attempted, as well as the expertise and resources used in those attacks. Once examined, the developer is able to choose the appropriate assurance level, in particular the AVA requirement components, commensurate with the attack potential for the threats. During the course of the evaluation, and in particular as a result

of completing the vulnerability assessment activity, the evaluator determines whether or not the TOE, operating in its operational environment, is sufficient to thwart attackers with the identified expertise and resources.

2207 It may be possible for a PP author to quantify the motivation of an attacker, as the PP author has greater knowledge of the operational environment in which the TOE (conforming to the requirements of the PP) is to be placed. Therefore, the motivation could form an explicit part of the expression of the attack potential in the PP, along with the necessary methods and measures to quantify the motivation.

B.4.2 Characterising attack potential

2208 This section examines the factors that determine attack potential, and provides some guidelines to help remove some of the subjectivity from this aspect of the evaluation process.

B.4.2.1 Identification and exploitation

2209 For an attacker to exploit a vulnerability in the TOE, the potential vulnerability must first be identified, the attack method then developed and finally the potential vulnerability exploited using the attack method. Each of these stages of determining whether there is a vulnerability in the TOE must be considered when quantifying the factors comprising the attack potential.

2210 To illustrate this, consider a potential vulnerability that is uncovered following months of analysis by an expert, but requires use of a simple attack method published on the Internet to exploit. Compare this with a potential vulnerability that is well known, but requires enormous time and resource to exploit.

2211 When a vulnerability is identified by an evaluator, the evaluator must determine the attack potential associated with the vulnerability. The evaluator may have performed considerable analysis to identify the vulnerability. However, the evaluator must consider the effect of the vulnerability becoming publicly known. That is, an attacker would not have to repeat the analysis to identify the vulnerability, but would only have to perform the exploitation. In some instances knowledge of the vulnerability would not immediately facilitate exploitation because considerable further analysis would be required to permit the development of an attack method

2212 In direct attacks against probabilistic or permutational mechanisms, the issue of exploitation will normally be the most important, since potential vulnerabilities in these mechanisms will often be self evident. Note, however, that this may not always be the case. With cryptographic mechanisms, for example, knowledge of subtle potential vulnerabilities may considerably affect the effectiveness of a brute force attack. Knowledge that users of a system tend to choose first names as passwords will have a similar effect. For vulnerability testing above AVA_VAN.1, the initial identification of potential vulnerabilities will become a much more important

consideration, since the existence of difficult to uncover potential vulnerabilities may be promulgated, often rendering exploitation trivial.

B.4.2.2 Factors to be considered

2213 The following factors should be considered during analysis of the attack potential required to exploit a vulnerability:

a) Time taken to identify and exploit (***Elapsed Time***);

b) Specialist technical expertise required (***Specialist Expertise***);

c) Knowledge of the TOE design and operation (***Knowledge of the TOE***);

d) ***Window of opportunity***;

e) ***IT hardware/software or other equipment*** required for exploitation.

2214 These factors provide characterisation of other quantifiers that can be used to describe the attack potential posed by an attacker, such as motivation and collusion.

2215 Motivation and the value of the asset are intrinsically linked, as they give an indication of the lengths to which an attacker will go in order to subvert the TSP of the TOE. If the asset is of high value to the attacker, either in monetary value or in prestige of attaining possession, the attacker is likely to have a high motivation and will sustain his efforts in subverting the TOE. This may be demonstrated by seeking to increase his access to the required knowledge of the TOE, related technology and/or attack methods (either by increasing his own or looking to external sources), increasing the sophistication of the equipment available for the attack, and by dedicating a large amount of time to the attack. The ability to increase these factors and sustain the effort applied to the attack is likely to depend upon the funds at the attacker's disposal, as many of the factors (***Knowledge of the TOE***, ***Equipment***, and even ***Elapsed Time*** to a certain extent) can be purchased by the attacker.

2216 The attacker may also seek to increase the factors by colluding with others. For this reason the attack potential is calculated as that possessed by the combination of the people involved in an attack, providing a characterisation of the role of the attacker. Therefore, the different types of expertise required at each stage of the attack, and within each stage of the attack, must be considered. Different levels of expertise may be required between the identification of the potential vulnerability, the development of the attack method and the realisation of the attack. Within each of these stages a number of different types of expertise may be required. Therefore the highest level of required expertise must be assumed when applying.

2217 In many cases these factors are not independent, but may be substituted for each other in varying degrees. For example, expertise or hardware/software

may be a substitute for time. A discussion of these factors follows. (The levels of each factor are discussed in increasing order of magnitude.)

2218 Elapsed time is the total amount of time taken by an attacker to identify that a particular potential vulnerability may exist in the TOE, to develop an attack method and to sustain effort required to mount the attack against the TOE. When considering this factor, the worst case scenario should be used to estimate the amount of time required.

2219 For example, the time taken to identify a potential vulnerability may be the time taken to locate the potential vulnerability in the information that is publicly available or may be the time required to analyse the design information to identify a potential vulnerability. In addition to this time taken for identification, consideration of the time required to develop an attack method (which may also be publicly available) and successfully run the attack method on the TOE to exploit the vulnerability must be included in the ***Elapsed Time*** factor.

2220 For the purposes of this discussion within minutes means an attack can be identified or exploited in less than an hour; within hours means an attack can succeed in less than a day; within days means an attack can succeed in less than a week, within weeks means an attack can succeed in less than a month, and in months means a successful attack requires up to three months.

2221 Specialist expertise refers to the level of generic knowledge of the underlying principles, product type or attack methods (e.g. Internet protocols, Unix operating systems, buffer overflows). The identified levels are as follows:

a) Laymen are unknowledgeable compared to experts or proficient persons, with no particular expertise;

b) Proficient persons are knowledgeable in that they are familiar with the security behaviour of the product or system type;

c) Experts are familiar with the underlying algorithms, protocols, hardware, structures, security behaviour, principles and concepts of security employed, techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc. implemented in the product or system type.

2222 When describing the expertise required, the total number of experts required must be included; the number of people for each type of expertise required and access to the expertise (dissemination) must be considered when describing the expertise required. Therefore, if expertise in both techniques for types of attack applicable to the TOE and underlying algorithms and protocols is required, then the highest level of ***Specialist Expertise*** characterisation should be assumed.

2223 ***Knowledge of the TOE*** refers to specific expertise in relation to the TOE. This is distinct from generic expertise, but not unrelated to it. Identified levels are as follows:

a) Public information concerning the TOE (e.g. as gained from the Internet);

b) Restricted information concerning the TOE (e.g. knowledge that is controlled within the developer organisation and shared with other organisations under a non-disclosure agreement)

c) Sensitive information about the TOE (e.g. knowledge that is shared between discreet teams within the developer organisation, access to which is constrained only to members of the specified teams);

d) Critical information about the TOE (e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need to know basis and individual undertaking).

2224 The knowledge of the TOE may graduate according to design abstraction, although this can only be done on a TOE by TOE basis. Some TOE designs may be public source (or heavily based on public source) and therefore even the design representation would be classified as public or at most restricted, while the implementation representation for other TOEs is very closely controlled as it would give an attacker information that would aid an attack and is therefore considered to be sensitive or even critical.

2225 Care should be taken here to ensure the highest level of knowledge of the TOE required during identification, development and running of the potential vulnerability is identified.

2226 ***Window of opportunity*** (Opportunity) is also an important consideration, and has a relationship to the ***Elapsed Time*** factor. Identification or exploitation of a vulnerability may require considerable amounts of access to a TOE that may increase the likelihood of detection. Some attack methods may require considerable effort off-line, and only brief access to the TOE to exploit. Access may also need to be continuous, or over a number of sessions.

2227 For some TOEs the ***Window of opportunity*** may equate to the number of samples of the TOE that the attacker can obtain. This is particularly relevant where attempts to penetrate the TOE and undermine the TSP may result in the destruction of the TOE preventing use of that TOE sample for further testing, e.g. hardware devices. Often in these cases distribution of the TOE is controlled and so the attacker must apply effort to obtain further samples of the TOE.

2228 For the purposes of this discussion unnecessary/unlimited access means that the attack doesn't need any kind of opportunity to be realised; easy means that access is required for less than a day or that the number of TOE samples required to perform the attack is less than ten; moderate means that access is required for less than a month or that the number of TOE samples required to perform the attack is less than fifty; difficult means that access is required for at least a month or that the number of TOE samples required to perform the attack is less than one hundred; none means that the opportunity window is not sufficient to perform the attack (the length for which the asset to be

exploited is available or is sensitive is less than the opportunity length needed to perform the attack - for example, if the asset key is changed each week and the attack needs two weeks).

2229 Consideration of this factor may result in a determining that it is not possible to complete the exploit, due to requirements for time availability that are greater than the opportunity time.

2230 ***IT hardware/software or other equipment*** refers to the equipment required to identify or exploit a vulnerability.

a) Standard equipment is readily available to the attacker, either for the identification of a vulnerability or for an attack. This equipment may be a part of the TOE itself (e.g. a debugger in an operating system), or can be readily obtained (e.g. Internet downloads, protocol analyser or simple attack scripts).

b) Specialised equipment is not readily available to the attacker, but could be acquired without undue effort. This could include purchase of moderate amounts of equipment (e.g. power analysis tools, use of hundreds of PCs linked across the Internet would fall into this category), or development of more extensive attack scripts or programs.

c) Bespoke equipment is not readily available to the public as it may need to be specially produced (e.g. very sophisticated software), or because the equipment is so specialised that its distribution is controlled, possibly even restricted. Alternatively, the equipment may be very expensive.

2231 Specialist expertise and ***Knowledge of the TOE*** are concerned with the information required for persons to be able to attack a TOE. There is an implicit relationship between an attacker's expertise (where the attacker may be one or more persons with complementary areas of knowledge) and the ability to effectively make use of equipment in an attack. The weaker the attacker's expertise, the lower the potential to use equipment (IT hardware/software or other equipment). Likewise, the greater the expertise, the greater the potential for equipment to be used in the attack. Although implicit, this relationship between expertise and the use of equipment does not always apply, for instance, when environmental measures prevent an expert attacker's use of equipment, or when, through the efforts of others, attack tools requiring little expertise to be effectively used are created and freely distributed (e.g. via the Internet).

B.4.2.3 An approach to calculation

2232 The above section identifies the factors to be considered. However, further guidance is required if evaluations are to be conducted on a consistent basis. The following approach is provided to assist in this process. The numbers have been provided with the objective of achieving ratings that are consistent with the relevant evaluation levels.

2233 Table 2 identifies the factors discussed in the previous section and associates numeric values with the total value of each factor.

2234 When this table is used by a PP/ST author the highest level of each factor assumed to be applied by an attacker is to be identified. The values associated with these levels are then identified using Table 2, and are summed to determine the overall attack potential rating of the assumed attacker. This should be performed in the context of threats against the TOE as specified in the Security Problem Definition of the ST.

2235 When determining the attack potential for a given vulnerability, a single level for each factor is selected to represent the extent of the factor required to identify the potential vulnerability, develop an attack method and perform the exploitation. The selected level should reflect the highest level of the factor required. Although in the case of Expertise if more than one type of expert is required, this factor should be iterated (see below). The values associated with the selected level for each factor should then be identified using Table 2. When selecting values the intended operational environment for the TOE should be assumed. The values are then summed, giving a single value. This value is then checked using Table 3 to determine the overall rating.

2236 If the attacker needs to have different types of expertise (example: hardware specialist and firewall expert) or different types of equipment (example: protocol analyser and very sophisticated software) to develop and perform the attack, then the different values corresponding to the factor (***Specialist Expertise*** or ***Equipment***) should be added. If you only need one type of expertise or equipment, no iteration should be performed. Elapsed time, Knowledge of the TOE and Window of Opportunity factors cannot be iterated: only the overall sequence of steps to perform the attack should be considered (identification of a potential vulnerability if any, development of attack method if any and realisation of the attack).

2237 For the ***Elapsed Time*** factor, each week is considered to be worth one point, so this factor can scale in the granularity required for the TOE.

2238 Where a factor falls close to the boundary of a range the evaluator should consider use of an intermediate value to those in the table. For example, if twenty samples are required to perform the attack then a value between one and four may be selected for that factor, or if the design is based on a publicly available design but the developer has made some alterations then a value between zero and four should be selected according to the evaluator's view of the impact of those design changes. The table is intended as a guide.

2239 The "**" specifications in the table are not to be seen as a natural progression from the timescales specified in the preceding ranges associated with a factor. These specifications identify that for a particular reason the potential vulnerability cannot be exploited in the TOE in its intended operational environment. For example, in considering ***Window of Opportunity***, unauthorised access to the TOE may be detected after a certain amount of time in a TOE with a known environment (i.e. in the case of a

system) where regular patrols are completed, and the attacker could not gain access to the TOE for the required two weeks undetected. However, this would not be applicable to a TOE connected to the network where remote access is possible, or where the physical environment of the TOE is unknown.

Factor	Range	Value
Elapsed Time	<= 1 day	0
	<= 1 week	1
	<= 1 month	4
	<= 3 months	13
	<= 6 months	26
	> 6 months	*(1)
Expertise	Layman	0
	Proficient	2
	Expert	5
Knowledge of TOE	Public	0
	Restricted	1
	Sensitive	4
	Critical	10
Window of Opportunity	Unnecessary / unlimited access	0
	Easy	1
	Moderate	4
	Difficult	12
	None	**(2)
Equipment	Standard	0
	Specialised	3
	Bespoke	7

(1) Indicates that the corresponding attack potential is beyond high attack potential.
(2) Indicates that the attack path is not exploitable due to other measures in the intended operational environment of the TOE.

Table 2 Calculation of attack potential

2240 For a given vulnerability it may be necessary to make several passes through the table for different attack scenarios (e.g. trading off, or compensating, expertise for time or equipment). The lowest value obtained for these passes should be retained, as this reflects the minimum level of attack potential required to undermine the TSP.

2241 In the case of a vulnerability that has been identified and is in the public domain, the identifying values should be selected for an attacker to uncover that vulnerability in the public domain, rather than to initially identify it.

2242 If different types of attacker are assumed by the PP/ST author, several passes through the table should be made to determine the different level of attack potential understood for each type of attacker. The PP/ST author then considers the highest value obtained when determining the level of attack the

TOE should withstand (selection of Vulnerability analysis (AVA_VAN) component).

2243 Table 3 should then be used to obtain a rating for the vulnerability/attack potential.

Range of values	Resistant to attacker with attack potential of:
0-2	No rating
3-5	Basic
6-9	Extended-Basic
10-14	Moderate
15-26	High
*	Beyond High

Table 3 Rating of vulnerabilities

2244 An approach such as this cannot take account of every circumstance or factor, but should give a better indication of the level of resistance to attack required to achieve the standard ratings. Other factors, such as the reliance on unlikely chance occurrences are not included in the basic model, but can be used by an evaluator as justification for a rating other than those that the basic model might indicate.

2245 It should be noted that whereas a number of vulnerabilities rated individually may indicate high resistance to attack, collectively the combination of vulnerabilities may indicate that overall a lower rating is applicable. The presence of one vulnerability may make another easier to exploit.

B.4.3 Examples of the application of this approach

B.4.3.1 Basic attack potential

2246 The characterisation of the least attributes required by an attacker demonstrating a Basic attack potential rating is considered to be represented by the following, giving a result of 4 from Table 2, and a rating of Basic in Table 3:

- the TOE would withstand attack for up to 1 month (4);
- layman expertise (0);
- public knowledge of the TOE (0);
- unlimited access/unlimited number of samples (0);
- standard equipment (0).

B.4.3.2 Considering Elapsed Time only

2247 The following examples consider a change only in the elapsed time taken to exploit a vulnerability in the TOE, showing the affect on the rating of attack

potential for a layman, with standard equipment, public knowledge of the TOE and unlimited access/unlimited number of samples:

a) The TOE can be broken within one week (1) = No rating.

b) The TOE can withstand the attack for up to one month (4) = Basic.

c) The TOE can withstand the attack for up to two months (8) = Extended-Basic.

d) The TOE can withstand the attack for up to three months (12) = Moderate.

e) The TOE can withstand the attack for more than three months (13+) = (High).

B.4.3.3 Comparing time and expertise

2248 The following examples illustrate the affect on the rating of attack potential when changing the expertise of an attacker and the elapsed time taken to exploit a vulnerability in the TOE. Standard equipment, public knowledge of the TOE and unlimited access/unlimited number of samples are assumed to be used in all cases.

a) The proficient attacker (2) takes between one day and one week (1) to break the TOE, with no other factors = Basic rating (3).

b) The proficient attacker (2) takes up to one month to break the TOE (4), with no other factors = Extended-Basic rating (6).

c) The proficient attacker (2) takes up to two months to break the TOE (8), with no other factors = Moderate rating (10).

d) The proficient attacker (2) takes up to three months to break the TOE (12), with no other factors = High rating (14).

e) The expert attacker (5) takes between one day and one week (1) to break the TOE, with no other factors = Extended-Basic rating (6).

f) The expert attacker (5) takes up to one month to break the TOE (4), with no other factors = Extended-Basic rating (9).

g) The expert attacker (5) takes up to two months to break the TOE (8), with no other factors = Moderate rating (13).

h) The expert attacker (5) takes up to three months to break the TOE (12), with no other factors = High rating (17).

B.4.3.4 Elapsed time, Specialist expertise, Knowledge of the TOE

2249 The following provides examples of how ratings vary according to the elapsed time, expertise and knowledge that is applied in the attack, when

standard equipment is used with unlimited access/unlimited number of samples:

a) TOE resists attack from proficient attacker (2), with restricted knowledge (1), 1 day to 1 week (1) = Basic rating (4).

b) TOE resists attack from laymen with critical information for at least one week = High (14).

c) TOE resists attack from a proficient with sensitive information between one day and one week = Extended-Basic (7).

d) TOE resists attack from a layman with critical information between one day and one week = Moderate (11).

This example reflects a calculation performed by an evaluator when a vulnerability has been found to result in a successful attack and the evaluator is to rate it as “exploitable” or “residual”.

e) The TOE can be broken by a layman with critical information in less than one day = Moderate (10).

B.5 Independent Factors Approach

2250 This approach is a variation of the Weighted Factors approach presented above. Rather than providing weighting of factors and taking the sum of the weights into account to derive the attack potential, this approach presents a set of independent parameters that require no weighting when assigning initial values to the parameters. (However, there is a sense of implicit weighting as minimum values for a given rating are set for each factor.)

B.5.1 Definitions of Independent Attack Potential Parameters

2251 In consideration of the “Nature of the TOE” and the “Life expectancy of the TOE” the following independent parameters have been identified, which affect an estimation of the resistance of a potential vulnerability against a specific attack.

2252 The “Nature of the TOE” is the term used to refer to the TOE type, the complexity of the technology used in the construction of the TOE and the information relating to the technology used in the TOE that is available in the public domain. Each of these three aspects affects the level of understanding of the “Nature of the TOE”.

2253 The “Life expectancy of the TOE” is used in consideration of the time taken to develop and run an attack method to undermine the TSP. This may be affected by aspects such as the frequency with which a key or password is updated, and the duration in which the TOE is expected to be in operational use.

B.5.1.1 Knowledge of the Technology

2254 *The kind of theoretical knowledge necessary to identify a potential vulnerability within the life expectancy of the TOE.*

2255 The main focus of this parameter is the technology on which the TOE is based. It strongly depends on the nature of the TOE. This means that the technology of a limited (simple) software product is considered to need less time to become familiar with than with a complex product like an IC or an operating system. Therefore, identifying potential vulnerabilities depends on both the nature of the TOE and the expertise of the attacker.

2256 The following levels of knowledge (expertise) of a special technology have been defined: *Inexperienced-Layman*, *Low-Experienced-Layman*, *Proficient Attacker* or *Expert*.

- An *Inexperienced-Layman* is defined as someone who does not have any useful knowledge of the technology in the subject area of the TOE whereas a *Low-Experienced-Layman* has little knowledge that can be seen as useful.

- A *Proficient Attacker* has at least intermediate knowledge of the technique. Meaning the attacker is familiar with general theory on which the TOE based in respect to its security behaviour.

- An *Expert* has the extensive and profound knowledge required to identify a potential vulnerability in accordance with the nature of the TOE. This not only includes general knowledge of the theory on which the TOE is based but also detailed knowledge on which the theory itself is based. These could be algorithms, tools, the internal structure of the theory etc.

2257 There are two possibilities to obtain knowledge of a potential vulnerability. Firstly to study the technology on which the TOE is based. Secondly to have a well known list that identifies, and possibly explains, certain vulnerabilities.

2258 In the first case, this parameter depends on the time and/or the expertise an attacker needs to gain knowledge of the vulnerability. The second case assumes that someone already has information of the potential vulnerability. Then the situation should be interpreted as a worst case, whereby the knowledge of the attacker should be estimated as an in-experienced-layman, as the attacker has been given all the knowledge required. The knowledge required to effect an attack on the TOE having gained understanding of the existence of the potential vulnerability is considered in Knowledge of Exploitation, below.

B.5.1.2 Knowledge of the TOE

2259 *The kind of knowledge, or possibility to obtain this knowledge, an attacker has or can obtain within the life expectancy of the TOE.*

2260 If a potential vulnerability is based on a special knowledge of the TOE (internals) this parameter indicates the depth of information an attacker requires. An attacker may be able to identify potential vulnerabilities in the TOE from TOE design information.

2261 The following levels of ***Knowledge of the TOE*** or parts of it have been defined: *None*, *Restricted*, *Sensitive* or *Critical*.

- This parameter is determined to be "*None*" if an attacker has no or only very little knowledge of the TOE. This knowledge is usually public e.g. available by literature and/or the Internet.
- "*Restricted*" means that the attacker possibly knows simple internal procedures of the TOE.
- In the case that an attacker knows significant relationships within the TOE and/or has special knowledge of parts of the TOE, the attacker could be seen as someone who has "*Sensitive*" information like detailed design knowledge (e.g. module level TOE design).
- This parameter is considered "*Critical*" if an attacker knows exactly how the TOE works internally (e.g. the source code) including details that can be used to exploit potential vulnerabilities.

2262 Generally two possibilities exist to obtain knowledge of the TOE. Firstly an attacker has access to the development documentation or has knowledge of its content. In this case the question arises if the development environment could be deemed to be trustworthy or if it is imaginable that development information could leave the protected area. Secondly an attacker could become familiar with details of the TOE internals during the operational work.

2263 In the second case, the time factor has to be considered. If the life expectancy (see definition above) of the TOE is shorter than the time an attacker (in consideration of his expertise) needs to get specific information of the TOE, the level of ***Knowledge of the TOE*** has to be set to a value which correlates with the respective knowledge the attacker can probably acquire during this available time.

2264 The possibility to buy such kind of knowledge is not in the scope of this parameter. The parameter "Equipment" (discussed below) deals with this issue.

B.5.1.3 Knowledge of Exploitation

2265 *The knowledge an attacker can obtain to develop and perform the exploitation of a potential vulnerability within the life expectancy of the TOE.*

2266 On the basis of Knowledge of the Technology and ***Knowledge of the TOE*** an attacker can develop and perform attacks against the TOE or part of the

TOE to exploit potential vulnerabilities. Therefore, an attacker needs expertise of methods to organise and to carry out such attacks. Knowledge of Exploitation only deals with the knowledge of planning possible attacks and the capability to carry them out, but not with physically completing the attack (this is discussed under Opportunity below).

2267 Similar to the definition of Knowledge of the Technology, four levels of knowledge (expertise) for exploitation of potential vulnerabilities are defined: *Inexperienced-Layman*, *Low-Experienced-Layman*, *Proficient Attacker* or *Expert*.

a) In this context, an *Inexperienced-Layman* is defined as someone who does not have any experience to develop, plan and/or perform attacks against the TOE whereas a *Low-Experienced-Layman* has little experience to do so, possibly assisted in similar attacks under instruction.

b) A *Proficient Attacker* has at minimum intermediate knowledge to exploit specific vulnerabilities of the TOE.

c) One can be sure to assume that an *Expert* has enough experience to design and perform (if the opportunity and equipment are available) exploitations of potential vulnerabilities.

2268 The knowledge of an expert can be seen as something that represents the state of the art knowledge concerning a certain technology. With respect to specific exploitations of potential vulnerabilities of the TOE or parts of the TOE, this leads to the fact that if there are any vulnerabilities in the TOE an expert knows exactly what has to be done for their successfully exploitation.

2269 Proceeding on the assumption that an attacker has the opportunity (see Opportunity) and the equipment (***Equipment***) to carry out an attack, Knowledge of Exploitation only depends on the capability and time required to organise this exploitation. The knowledge of the technology and of the TOE is discussed by the parameters Knowledge of the Technology and ***Knowledge of the TOE*** (see above). Therefore, an estimation of the expertise for developing, designing and performing exploitations has to be made under consideration of the life expectancy of the TOE. This means a specific attack must be practical within the available time by an attacker possessing the expertise estimated for Knowledge of Exploitation. In the case that the knowledge of an attacker is not sufficient to identify a potential vulnerability then the question arises whether the attacker is able to get this expertise within the life expectancy of the TOE. If this is possible a proficient attacker can become an expert attacker.

2270 As already mentioned for ***Knowledge of the TOE*** the possibility to buy such kind of knowledge isn't in the scope of this parameter. This subject is also addressed by parameter "***Equipment***") discussed below.

B.5.1.4 Opportunity

2271 *The likelihood of having the required access to the TOE within its life expectancy.*

2272 Exploiting a potential vulnerability in the TOE requires some level of access to the TOE. In general the opportunity to have access to the TOE depends on the operational environment in which the TOE is used.

2273 This parameter determines the following levels of access to the TOE: *easy, with some effort, difficult or improbable*. This parameter does not take into consideration whether an attacker has suitable equipment or the expertise to perform the exploitation of a potential vulnerability. These are discussed by ***Equipment*** and ***Knowledge of Exploitation respectively***.

a) If a TOE is generally available (e.g. available for members of the public to buy, without restrictions) then the opportunity to perform an attack is *easy* in principle.

b) In the case that a TOE is merely distributed or available for a limited circle but there are no high obstacles (e.g. there are no special protective measures restricting access to the TOE) to get one or more samples of it than the TOE is available with *some effort*.

c) If the operational environment of the TOE protects the TOE for example by organisational measures then it is *difficult* to get access to it.

d) It should be *improbable* to get the opportunity of access to the TOE if the TOE operates inside a high protected area with strict access rules, where only a few people that can be seen as very trustworthy have access to the TOE.

2274 The determination for the level of opportunity in the context of this parameter reflects the probability that an unauthorised person can have access to the TOE within its life expectancy. Therefore, the time required for an attacker to exploit a vulnerability is the main factor on which ***Opportunity*** depends.

2275 The nature of the TOE may also need to be considered when determining access to the TOE. For example, some TOEs may be rendered inoperable following an attempt to exploit a potential vulnerability. Therefore if multiple attempts are required, multiple copies of the TOE may also be required. The feasibility of obtaining multiple copies should be considered, as this may become cost prohibitive or there may be controls on attempts to obtain multiple copies.

B.5.1.5 Equipment

2276 *The type of equipment necessary to exploit the potential vulnerability*.

2277 To exploit a potential vulnerability of the TOE special equipment may be necessary. There are different types of equipment imaginable: Firstly the hardware and software devices needed to perform a certain attack against the TOE. Secondly the human resource required is another factor that could have an influence to drive a successful attack.

2278 Money is an additional possibility to improve the prerequisite for that. Although the financial aspect could affect also the first types, it also could be used to buy other things (e.g. industrial espionage) which are helpful to perform attacks.

2279 The following levels of equipment necessary to exploit a potential vulnerability are defined: *Standard*, *Higher-than-Average*, *Specialised* or *Bespoke*.

a) In this context the definition of "*Standard*" equipment is only the equipment (hardware, software, money or human resource) likely to be possessed by a standard user of the TOE that is necessary to perform an attack.

b) If some more effort is essential, e.g. special equipment (that means the equipment needed cannot be considered as a common object of utility by the TOE user) must be bought within a moderate price-range (one can finance this utility without major difficulty), than the value of ***Equipment*** can be set to "*Higher-than-Average*".

c) "*Specialised*" equipment does demand an expense and/or human resource that is difficult to realise by a private person.

d) In this context "*Bespoke*" means that the equipment and human resource needed is not commonly available.

2280 Before an estimation of the equipment is determined, the interrelationships between the three types of equipment mentioned above must be clear because an assessment of ***Equipment*** must be done under consideration of all types of equipment required to realise an attack. For example if an attacker has enough money to buy all hardware and software components which are necessary but it seems to be impossible to get the human resource to carry out the attack then the level of equipment should be rated as "bespoke" due to the constraints in obtaining the human resource required.

B.5.2 Determination of the Attack Potential

2281 On the basis of the independent parameter definitions it should be possible to decide whether a potential vulnerability is resistant against a certain attack potential.

2282 To determine resistance, the user shall consider the TOE and the potential vulnerability with the definitions of the independent parameters (as described in Section B.5.1.

2283 For a given potential vulnerability, the evaluator has to give a statement of the rating for each independent parameter. This rating is to be compared with the minimum rating of that parameter for a given attack potential. If for each parameter the rating assigned by the user meets or exceeds the minimum rating for the given attack potential the TOE is considered to resist the potential vulnerability.

2284 To specify the attack potential the TOE is to resist, the PP/ST author identifies for each parameter the minimum level assumed to be possessed by an attacker of the TOE. The author will then compare these ratings to the minimums specified for each of the levels of attack potential to identify the appropriate attack potential claim.

2285 A lookup table is provided, allowing a value to be identified for each independent parameter. For each parameter, the evaluator looks up the rating considered to be appropriate for the potential vulnerability in Table 4.

Independent Parameter	Rating	Value
Knowledge of the Technology	Inexperienced-Layman	0
	Low-experience-Layman	1
	Proficient	2
	Expert	3
Knowledge of the TOE	None	0
	Restricted	1
	Sensitive	2
	Critical	3
Knowledge of Exploitation	Inexperienced-Layman	0
	Low-experience-Layman	1
	Proficient	2
	Expert	3
Opportunity	Easy	0
	Some Effort	1
	Difficult	2
	Improbable	3
Equipment	Standard	0
	Higher average	1
	Specialised	2
	Bespoke	3

Table 4 Assignment of TOE's characteristic to the requirements derived from the definitions of the independent parameters

2286 Application steps using Table 4:

a) Estimate for each independent parameter the requirements, as given in Section B.5.1, met by an attacker of the TOE.

b) For each parameter make the assignment to that rating which the TOE at least fulfilled. Determine the appropriate cells in column “Rating”.

2287 There may be additional factors that prevent a vulnerability from being exploited, and therefore Table 4 does not need to be applied. For example, if a brute force attack against a permutational or probabilistic mechanism will take a longer to complete than the life-expectancy of the TOE, the evaluator reports that the vulnerability cannot be exploited in the TOE.

B.5.3 Determination of the Requirements for Attack Potential of a Potential Vulnerability

2288 The following table contains on the one hand the minimum requirements of each independent parameter (represented by the rows) which on the other hand depends on the level of the attack potential of the identified vulnerability (represented by the columns).

2289 To confirm that a vulnerability has a certain attack potential, one has to check whether the rating of each independent parameter of the analysed vulnerability fulfils the requirements necessary for a defined attack potential, i.e. for each parameter the assigned rating achieves at least the minimum stated in Table 5 for the a given attack potential.

2290 If the requirements, for a particular characteristic (expressed by an independent parameter) that is considered important to decide the attack potential rating, are not fulfilled the considered attack potential has not been achieved. This means that when applying this method it is not possible to compensate a "weak" estimated parameter by another parameter that was estimated as "High". Each independent parameter is considered equally in a way that a minimum of requirements have to be met.

Independent Parameter	Attack Potential "Basic"	Attack Potential "Extended-Basic"	Attack Potential "Moderate"	Attack Potential "High"
Knowledge of the Technology	0	1	2	3
Knowledge of the TOE	1	1	2	3
Knowledge of Exploitation	1	2	2	3
Opportunity	1	2	2	3
Equipment	1	1	2	3

Table 5 Determination of the Attack Potential

2291 Application steps using Table 5:

a) Take the assigned rating values resulted of step 2 described in Section B.5.2.

b) Make sure that for each parameter the minimum requirements displayed by the “Attack Potential” column which shall be claimed are fulfilled.

B.6 Example calculation for direct attack

2292 Mechanisms subject to direct attack are often vital for system security and developers often strengthen these mechanisms. As an example, a TOE might use a simple pass number authentication mechanism that can be overcome by an attacker who has the opportunity to repeatedly guess another user's pass number. The system can strengthen this mechanism by restricting pass numbers and their use in various ways. During the course of the evaluation an analysis of this direct attack could proceed as follows:

2293 Information gleaned from the ST and design evidence reveals that identification and authentication provides the basis upon which to control access to network resources from widely distributed terminals. Physical access to the terminals is not controlled by any effective means. The duration of access to a terminal is not controlled by any effective means. Authorised users of the system choose their own pass numbers when initially authorised to use the system, and thereafter upon user request. The system places the following restrictions on the pass numbers selected by the user:

a) the pass number must be at least four and no greater than six digits long;

b) consecutive numerical sequences are disallowed (such as 7,6,5,4,3);

c) repeating digits is disallowed (each digit must be unique).

2294 Guidance provided to the users at the time of pass number selection is that pass numbers should be as random as possible and should not be affiliated with the user in some way - a date of birth, for instance.

2295 The pass number space is calculated as follows:

a) Patterns of human usage are important considerations that can influence the approach to searching a password space. Assuming the worst case scenario and the user chooses a number comprising only four digits, the number of pass number permutations assuming that each digit must be unique is:

$$7(8)(9)(10) = 5040$$

b) The number of possible increasing sequences is seven, as is the number of decreasing sequences. The pass number space after disallowing sequences is:

$$5040 - 14 = 5026$$

2296 Based on further information gleaned from the design evidence, the pass number mechanism is designed with a terminal locking feature. Upon the sixth failed authentication attempt the terminal is locked for one hour. The failed authentication count is reset after five minutes so that an attacker can at best attempt five pass number entries every five minutes, or 60 pass number entries every hour.

2297 On average, an attacker would have to enter 2513 pass numbers, over 2513 minutes, before entering the correct pass number. The average successful attack would, as a result, occur in slightly less than:

$$7(8)(9)(10) = 5040$$

2298 Using either of the approaches to calculate attack potential described above, it is possible that a layman can defeat the mechanism within days (given easy access to the TOE), with the use of standard equipment, and with no knowledge of the TOE, giving a value of 2. Given the resulting sum, 2, the attack potential required to effect a successful attack is not rated, as it falls below that considered to be Basic.