Guide for the Transition from CC v2.3 to CC v3.1 for ADV Requirements

Technical Editor: Bundesamt für Sicherheit in der Informationstechnik (BSI) / Federal Office for Information Security

Version 1, June 2008

General purpose:

The intention of this document is to help people involved in Common Criteria (CC) evaluations using the version 2.3 of the standard to migrate the evidence and evaluation reports of the CC development class ADV to the new version 3.1 of the CC.

Field of special use: vendors that have performed previously an evaluation of a product using CC v2.3 and are now planning to perform a re-evaluation using CC v3.1

Table of contents

1. Introduction

1.1. Structure of this document

1.2. Using this guide

2. ADV_ARC

2.1. Background of ADV_ARC

2.2. Discussion

2.3. Mapping CEM Version 3.1 work units for ADV_ARC to CC v2.3

2.3.1. ADV_ARC.1.1C

2.3.2. ADV_ARC.1.2C

2.3.3. ADV_ARC.1.3C

2.3.4. ADV_ARC.1.4C

2.3.5. ADV_ARC.1.5C

3. ADV_FSP

3.1. Background of ADV_FSP

3.2. Discussion

3.3. Mapping CEM Version 3.1 work units for ADV_FSP to CC v2.3

3.3.1. ADV_FSP.1

3.3.2. ADV_FSP.2

3.3.3. ADV_FSP.3
3.3.4. ADV_FSP.4

3.3.5. CC v2.3 work units no longer required in CC v3.1

4. ADV_TDS

4.1. Background of ADV_TDS

4.2. Discussion

4.2.1. Optional subsystem composition in CC v3.1

4.2.2. Scope of the subsystem description

4.2.3. Level of detail in module design

4.2.4. Content of module design

4.2.5. Dependencies on the IT environment

4.3. Mapping CEM Version 3.1 work units for ADV_TDS to CC v2.3

4.3.1. ADV_TDS.1

4.3.2. ADV_TDS.2

4.3.3. ADV_TDS.3

4.3.4. CC v2.3 work units no longer included in CC v3.1

5. ADV_IMP

5.1. Background of ADV_IMP

5.2. Discussion

5.3. Mapping CEM Version 3.1 work units for ADV_IMP to CC v2.3

5.3.1. ADV_IMP.1

5.3.2. CC v2.3 work units no longer included in CC v3.1

6. Transition Guide for ADV Requirements for EAL5

6.1. Summary of CC v2.3 ADV Requirements for EAL5

6.2. Summary of CC v3.1 ADV Requirements for EAL5

6.3. Summary of Differences between CC v2.3 and CC v3.1 for the ADV Class Requirements of EAL5

6.4. Detailed Analysis and Transition Guidance for EAL5

6.4.1. ADV_FSP

6.4.2. ADV_IMP
6.4.3. ADV_INT

6.4.4. ADV_TDS

6.4.5. Work Units from CEM v2.3 no longer relevant in CEM v3.1 (for EAL5)

7. References

1. Introduction

1 This document is intended to provide vendors and evaluators a guide on how to make a transition from Common Criteria version 2.3 (abbreviated with CC v2.3 in the following) to Common Criteria version 3.1 (abbreviated with CC v3.1 in the following) for the ADV class. The main use of this guide is for vendors that have performed previously an evaluation of a product using CC v2.3 and are now planning to perform a re-evaluation using CC v3.1. This guide compares the requirements and evaluator actions of CC v3.1 and CEM v3.1 to those of CC v2.3 and CEM v2.3 and shows where they can be mapped to each other and what the differences are. Guidance is provided how to address the differences between the two versions of CC and CEM.

2 The basis for this document is Release 2 of CC v3.1. Modifications and additions to the CC and CEM made there have been taken into account.

3 The following table lists the assurance components for the "Development" class (ADV) for CC v2.3 for the different assurance levels:

<table>
<thead>
<tr>
<th>Assurance Class</th>
<th>Assurance Family</th>
<th>Assurance Components by Evaluation Assurance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>ADV_FSP</td>
<td>EAL1</td>
</tr>
<tr>
<td></td>
<td>ADV_HLD</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_IMP</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_LLD</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_RCR</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_SPM</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1: ADV Components in CC v2.3

4 The same mapping for CC v3.1 is:

<table>
<thead>
<tr>
<th>Assurance class</th>
<th>Assurance Family</th>
<th>Assurance Components by Evaluation Assurance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>ADV_FSP</td>
<td>EAL1</td>
</tr>
<tr>
<td></td>
<td>ADV_ARC</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_IMP</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_SPM</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2: ADV Components in CC v3.1
Those two tables show that there are some major changes from CC v2.3 to CC v3.1 with respect to the class ADV. The most obvious changes are:

- A new family (ADV_ARC) has been introduced with CC v3.1
- The families ADV_HLD and ADV_LLD in CC v2.3 have been combined to the family ADV_TDS in CC v3.1
- The family ADV_RCR of CC v2.3 does no longer exist in CC v3.1 (showing correspondence has been integrated in the families ADV_FSP, ADV_TDS and ADV_IMP)
- The family ADV_FSP has new components added, which result in continuously increasing requirements for EAL1 to EAL5. CC v2.3 had identical requirements related to ADV_FSP for EAL1 to EAL3 (ADV_FSP.1), and increasing requirements for EAL4 (ADV_FSP.2) and EAL5 (ADV_FSP.3)
- There are no requirements for ADV_SPM for assurance levels lower than EAL6. The family ADV_SPM has only one component.
- Categorization of TSF portions into SFR-enforcing, SFR-supporting and SFR non-interfering

This document focuses on assurance levels EAL1 to EAL5 and provides guidance for the transition from CC v2.3 to CC v3.1 for both developers and evaluators.

At a first glance using the structure of the assurance levels up to EAL5, the components from the ADV class for the different versions of the Common Criteria map as follows:

<table>
<thead>
<tr>
<th>CC v3.1</th>
<th>CC v2.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_ARC.1</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.1</td>
<td>ADV_FSP.1 ADV_RCR.1</td>
</tr>
<tr>
<td>ADV_FSP.2</td>
<td>ADV_FSP.1 ADV_RCR.1</td>
</tr>
<tr>
<td>ADV_FSP.3</td>
<td>ADV_FSP.1 ADV_RCR.1</td>
</tr>
<tr>
<td>ADV_FSP.4</td>
<td>ADV_FSP.2 ADV_RCR.1</td>
</tr>
<tr>
<td>ADV_FSP.5</td>
<td>ADV_FSP.3 ADV_RCR.2</td>
</tr>
<tr>
<td>ADV_TDS.1</td>
<td>ADV_HLD.1 ADV_RCR.1</td>
</tr>
<tr>
<td>ADV_TDS.2</td>
<td>ADV_HLD.2 ADV_RCR.1</td>
</tr>
<tr>
<td>ADV_TDS.3</td>
<td>ADV_HLD.2 ADV_LLD.1</td>
</tr>
<tr>
<td>ADV_IMP.1</td>
<td>ADV_IMP.1 ADV_RCR.1</td>
</tr>
<tr>
<td></td>
<td>- ADV_SPM.1</td>
</tr>
</tbody>
</table>

Table 3: Mapping of ADV Components between CC v3.1 and CC v2.3

As this table already shows, the mapping between the components of the ADV class between the different versions of the criteria is not a one-to-one mapping. This document therefore provides a detailed analysis down to the individual work units of the CEM and attempts to identify how the concepts of CC v2.3 map to those of CC v3.1 and how evaluation evidence and results as mentioned in the work units of CC v2.3 can be reused for the work required to perform an evaluation using CC v3.1.

1.1. Structure of this document

To assist in the transition from CC v2.3 to CC v3.1, this document is structured in the following way:

For each assurance family of CC v3.1, the document discusses purpose of the family and its relation to the assurance families and components defined CC v2.3. This is done in two subchapters called "background" and "discussion".

Following those general descriptions, a mapping (with additional detailed discussion) is provided that maps the requirements of CC v3.1 and the associated work units in CEM v3.1 for the assurance levels EAL1 to EAL4 to those of CC v2.3 and CEM v2.3. A table summarizes this mapping for each assurance component of the ADV class included in CC v3.1 for EAL1 to EAL4 (a separate chapter is provided for the EAL5 ADV components). The mapping is performed down to the work unit level of the CEM, where each work unit of CEM v2.3 for the component is mapped to those work units of CEM v2.3 that in some way address the topic of the CEM v3.1 work unit. A comment provides an indication how much of a contribution can be expected. The comment values have the following meaning:
• Identical The text of the CEM v3.1 work unit is identical to the text of the CEM v2.3 work unit. Evaluators should be able to reuse the results of the work unit easily.

• Mainly identical The text of the CEM v3.1 work unit has only some slight textual differences to the CEM v2.3 work unit but the intention is identical. Nevertheless due to a paradigm shift between the two versions of the CC there may be some differences in the evidence required and the evaluation work, but they are usually very small. Evaluators should be able to reuse the results of the work unit easily.

• Essence The text of the CEM v3.1 work differs to the one in the CEM v2.3 work unit but there is a large overlap in the intention of the work units. One should be able to reuse the evidence and main arguments used in a CC v2.3 evaluation for some major parts of the CEM v3.1 work unit but it is unlikely that the CEM v3.1 work unit is fully covered by the evidence and work provided for the CEM v2.3 work unit.

• Major The corresponding work unit of CEM v2.3 may include a major contribution for the CEM v3.1 work unit and should be considered when doing a transition from a CC v2.3 to CC 3.1 in an evaluation.

• Partly The corresponding work unit of CEM v2.3 may include a contribution to some specific aspects of the CEM v3.1 work unit and should be considered when doing a transition from a CC v2.3 to CC 3.1 in an evaluation.

• May contribute The corresponding work unit of CEM v2.3 may include a minor contribution to some specific aspects of the CEM v3.1 work unit. It should be considered when doing a transition from a CC v2.3 to CC 3.1 in an evaluation and one was not able to address the aspects of the CEM v3.1 work unit completely using the other CEM v2.3 work units mapped to it.

12 A separate chapter deals with the assurance requirements of EAL5. Since the CEM for CC v2.3 does not cover assurance requirements for ADV that are beyond EAL4, the work units defined in the AIS 34 document for the German CC scheme have been used in this transition guide. Since the AIS34 is only mandatory in the German CC scheme, the transition guide for EAL5 is defined in a separate chapter of this document.

1.2. Using this guide

13 This guide has been developed for developers and evaluators that have a profound knowledge of CC v2.3 and want to know what the differences to CC v3.1 with respect to the components of the ADV class included in assurance levels up to EAL5 are. Of special importance is the guide for developers that have performed a CC v2.3 evaluation of a product and now want to undergo a CC v3.1 evaluation for a new version of the product which has some (potentially minor) changes to the CC v2.3 evaluated version. The developer is interested in the differences in the criteria that may require different or new evidence. With this guide he should be able to identify the aspects of ADV that are handled differently in CC v3.1 and then identify where his design documentation addresses those or where additional or other evidence is required.

14 The evaluator of the product in addition has an interest to identify those work units in the CC v2.3 evaluation that (to some extent) address the work units of CEM v3.1. With this mapping he can easily identify those areas in the CC v2.3 evaluation report where he can find the arguments for those functions of the product that have not changed. This allows an evaluator in such a CC v3.1 re-evaluation reuse as much as possible from the previous CC v2.3 evaluation and to concentrate on the new or modified functionality of the product as well as on the new aspects required by CC v3.1 rather than performing a product evaluation from scratch.

15 The guide provides useful information for developers and evaluators that have a good knowledge of CC v2.3 even in the case where no re-evaluation based on a CC v2.3 evaluation is planned. The sections "background" and "discussion" help to understand the differences in the approach between CC v2.3 and CC v3.1 and thereby help to get an understanding for developers on the requirements for the evidence required as well as an understanding for the evaluators for the modifications in the evaluation work.

16 In order to use this guide to perform a CC v3.1 re-evaluation of a product previously evaluated using CC v2.3 one should use this guide in the following way:

17 Developer:

18 As a developer one needs to understand the differences in the required evidence that needs to be provided. For a developer that has documentation that has been used in a previous CC v2.3 (or earlier version) evaluation one should do the following:
• Identify the Target assurance level of CC v3.1 and the components of the ADV class included. Table 2 in this document can be used to identify the components of the ADV class included in each evaluation assurance level up to EAL5.

• Identify the corresponding components from CC v2.3 using table 3 and verify, that the component was included in the previous CC v2.3 evaluation.

• Read the "background" and "discussion" section of the corresponding CC v3.1 component to identify the differences and new aspects of CC v3.1 related to the component.

• Check the evidence to identify, if the new and modified aspects are addressed in the evidence.

19 Evaluator:

20 In addition to be able to identify the evidence required for a component, an evaluator also needs to identify evaluation work he may be able to reuse (in the case the relevant aspect of the product have not changed or changed only slightly). In addition to the method described above that helps to identify the general differences between CC v3.1 and CC v2.3 components of the ADV class, an evaluator is also interested to identify easily those parts in a CC v2.3 evaluation report that correspond to evaluator activities required by CC v3.1. Therefore this document contains for each ADV component of the ADV class in CC v3.1 that is included in an assurance level up to EAL4 a mapping of the related CEM work units from CEM v3.1 to those of CEM v2.3. This mapping identifies work units in CEM v2.3 that may address work aspects of the CC v2.3 evaluation that may provide useful input for the CC v3.1 evaluation. The evaluator identifies (via the evidence used for the CC v2.3 work unit) which developer evidence was used in the CC v2.3 evaluation in the individual work units and identifies which aspects of the evidence and work performed may potentially be reusable. The tables included for the ADV components in the "mapping" section help to identify the related work units in CEM v2.3 and the text provides additional guidance on what aspect may be reused.

21 For evaluations under the German CC scheme the same method can be used for the EAL5 assurance level. A separate chapter in this document maps the work units defined in AIS 34 to the ones defined in CEM v3.1 for the EAL5 assurance level.

2. ADV_ARC

>2.1. Background of ADV_ARC

22 ADV_ARC is a new family in the Development (ADV) class introduced in CC v3.1. The purpose of this family is to address security architecture properties of self-protection, domain isolation, and non-bypassability. The reason to address those properties separately from the security functionality is "because self-protection and non-bypassability largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are achieved through the design of the TOE and TSF, and enforced by the correct implementation of that design" (CC v3.1, part 2, para 216). CC v3.1 therefore introduced the family ADV_ARC, which requires the documentation of a "security architecture description" that describes how a TOE achieves the properties of self-protection, domain isolation and non-bypassability. Since the mechanisms used to achieve those properties are normally not isolated functions, but a sum of TOE internal mechanisms that may be spread throughout the TSF, a functional decomposition as required for ADV_FSP, ADV_HLD, and ADV_LLD (CC v2.3) resp. ADV_FSP and ADV_TDS (CC v3.1) is not required. Instead of the description of a functional decomposition, the security architecture description is supposed to provide arguments why the TOE achieves the above listed objectives, referencing the other design documentation where necessary.

23 In addition to the properties mentioned above, ADV_ARC also requires a description of the TSF initialization process which demonstrates that the TSF are brought into a secure initial state upon power-up or reset.

24 The ADV_ARC family has just one component (ADV_ARC.1) which is required starting from the EAL2 assurance level. Similar to ADV_RCR in CC v2.3, the requirements increase implicitly with higher assurance level, since the level of detail in the description of the security architecture description is tight to the level of description required for ADV_TDS and therefore increases when requirements increase in this family. This is explained in detail in CC v3.1, part 2, Appendix A.1.2, paragraph 521 to 524.

25 As a consequence of the introduction of ADV_ARC in CC v3.1, the Security Functional Requirements in the families FPT_RVM and FPT_SEP as described in part 2 of CC v2.3 have been removed in CC v3.1. The functional requirements from those families are now intended to be addressed by the ADV_ARC assurance family. Therefore, although the family ADV_ARC is introduced in CC v3.1 as a new family, the aspects in ADV_FSP,
ADV_HLD, and ADV_LLD that address the security functional requirements from the FPT_RVM and FPT_SEP families should provide some coverage of the requirements of the new ADV_ARC family.

26 CC v2.3 also mentions an "architectural description" as documentation required for the components in the ADV_INT family. The purpose of the architectural description in CC v2.3 was to describe the internal structure of the TSF in terms of modularity and layering to demonstrate that the TSF are well structured with minimized complexity. As part 3 of CC v2.3 explains in paragraphs 343 and 344:

Minimising the amount of functionality in the TSF that does not enforce the TSP, reduces the possibility of flaws in the TSF. In combination with modularity and layering, it allows the evaluator to focus only on that functionality which is necessary for TSP enforcement.

Design complexity minimisation contributes to the assurance that the code is understood -- the less complex the code in the TSF, the greater the likelihood that the design of the TSF is comprehensible. Design complexity minimisation is a key characteristic of a reference validation mechanism.

27 This shows that the "architectural description" as mentioned in CC v2.3 had a very different objective to the "security architecture description" required in CC v3.1. While the requirements for the "architectural description" in CC v2.3 were driven by software engineering ideas (describing modularization, layering, and minimization of complexity), the "security architecture description" in CC v3.1 has its background in the section of operating system architectures, where separation of address spaces, protection of memory areas, and the aspect of software operating with different hardware controlled privileges together with the switch from low-privilege to a higher privilege level (usually named a "system call") are aspects crucial for security. CC v3.1 now refers to the documentation that describes the software engineering aspects required in ADV_INT as the "TSF internals description" to avoid confusion with the "security architecture description".

2.2. Discussion

28 Since ADV_ARC is a completely new family within the ADV class addressing aspects that have been covered very differently (if at all) in CC v3.1, it can not be expected that there is a simple mapping showing how the requirements of ADV_ARC have been addressed in an evaluation using CC v2.3. Nevertheless, as we have identified in the background section above, the aspects of separation and reference mediation were not neglected in CC v2.3 but have been addressed as security functional requirements (if claimed for the TOE). The problem that have been identified in evaluations claiming those SFRs was that the CC paradigm of mapping SFRs to interfaces of the functional specification, to subsystems of the high-level design and to modules of the low-level design was very difficult most of the time. Separation, reference mediation and self-protection of the TSF (in the sense of having the TSF in a domain separated from all untrusted domains) are properties rather than functions although there are of course functions within the TSF that contribute to achieve those properties. Usually those functions have little to none TSFI of its own, although they are involved in the protection of all the other security functions.

29 The requirements of ADV_ARC are largely driven by the security requirements of operating system type products. In those products a TOE consists of the TSF and of potentially untrusted parts ("user processes") that may attempt to tamper with the TSF or bypass the TSF to access protected resources. In addition operating system usually also have different user processes within the TOE that may be mutually hostile, i. e. where one of those processes may attempt to tamper with the other ones in order to get access to their resources. An operating system therefore has to have the TSF in some kind of domain separated from all untrusted user processes and also has to ensure that different user processes operate in separated domains. Calling TSF services by user processes requires TSF interfaces that pass domain boundaries and the functions called within the TSF are usually executed with different and increased privileges compared to the calling function. Those interfaces are therefore very often the target of attacks, where the caller attempts to use side-effects of the function called to increase his own rights and privileges or to "disturb" the TSF. Typical examples are the use of buffer overflow attacks either to implant hostile code into the higher privilege domain or to overwrite critical TSF data thereby either increasing the caller's rights or privileges or just cause the TSF to malfunction, resulting in a denial of service. Other attacks use insufficient parameter checking by the TSF called, which may give the calling process access to data (e. g. passwords or cryptographic keys) that it has no direct access to, since they are in a domain that only the TSF can access. In addition to those aspects the ADV_ARC family also requires the description of the start-up or initialization process of the TSF to show that the TSF reaches a secure state when start-up or initialization is finished.
30 While it is obvious that those aspects are crucial for security when trusted and untrusted portions of a TOE exist – regardless of the security objectives – they have not been specifically addressed in CC v2.3. Instead they were assumed to be covered by the design documentation – in some way.

31 According to the CEM the developer documentation for ADV_ARC needs to address the following (CEM, chapter “Application notes” for ADV_ARC):

The security architecture description describes how domains are defined and how the TSF keeps them separate. It describes what prevents untrusted processes from getting to the TSF and modifying it. It describes what ensures that all resources under the TSF's control are adequately protected and that all actions related to the SFRs are mediated by the TSF. It explains any role the environment plays in any of these (e.g. presuming it gets correctly invoked by its underlying environment, how is its security functionality invoked?). In short, it explains how the TOE is considered to be providing any kind of security service.

32 On the other hand, CC v2.3 requires for FSP_SEP:

The components of this family ensure that at least one security domain is available for the TSF's own execution and that the TSF is protected from external interference and tampering (e.g. by modification of TSF code or data structures) by untrusted subjects. Satisfying the requirements of this family makes the TSF self-protecting, meaning that an untrusted subject cannot modify or damage the TSF.

This family requires the following:

a) The resources of the TSF's security domain ("protected domain") and those of subjects and unconstrained entities external to the domain are separated such that the entities external to the protected domain cannot observe or modify TSF data or TSF code internal to the protected domain.

b) The transfers between domains are controlled such that arbitrary entry to, or return from, the protected domain is not possible.

c) The user or application parameters passed to the protected domain by addresses are validated with respect to the protected domain's address space, and those passed by value are validated with respect to the values expected by the protected domain.

d) The security domains of subjects are distinct except for controlled sharing via the TSF.

33 For FPT_RVM, CC v2.3 states:

The requirements of this family address the “always invoked” aspect of a traditional reference monitor. The goal of this family is to ensure, with respect to a given SFP, that all actions requiring policy enforcement are validated by the TSF against the SFP. If the portion of the TSF that enforces the SFP also meets the requirements of appropriate components from Domain separation (FPT_SEP) and TSF internals (ADV_INT), then that portion of the TSF provides a “reference monitor” for that SFP.

A TSF that implements a SFP provides effective protection against unauthorised operation if and only if all enforceable actions (e.g. accesses to objects) requested by untrusted subjects with respect to any or all of that SFP are validated by the TSF before succeeding. If an action that could be enforceable by the TSF, is incorrectly enforced or incorrectly bypassed, the overall enforcement of the SFP could be compromised. Subjects could then bypass the SFP in a variety of unauthorised ways (e.g. circumvent access checks for some subjects or objects, bypass checks for objects whose protection was assumed by applications, retain access rights beyond their intended lifetime, bypass auditing of audited actions, or bypass authentication). Note that some subjects, the so called “trusted subjects” with respect to a specific SFP, might be trusted to enforce the SFP by themselves, and bypass the mediation of the SFP.

34 Those statements imply that the design documentation provided for a CC v2.3 evaluation for a TOE that includes a component from FPT_RVM and/or FPT_SEP needs to address in some way the aspects listed in CC v2.3 in the introduction to those families. A more detailed discussion what was required for a CC v2.3 evaluation to address those aspects and how this maps to the requirements of CC/CEM Version 3.1 is provided in the following chapter.
2.3. Mapping CEM Version 3.1 work units for ADV_ARC to CC v2.3

35 The following section lists the work units of the CEM for ADV_ARC.1 and discusses, if and where related information can be found in documentation of a CC v2.3 evaluation.

2.3.1.ADV_ARC.1.1C

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

37 ADV_ARC.1-1

The evaluator shall examine the security architecture description 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 functional specification and TOE design document.

38 This work unit is not related to the content of the security architecture description but related to the level of detail in this description.

39 Since the aspects of the "architectural design" in CC v2.3 have been addressed using the security functional requirements of the FPT_SEP and FPT_RVM families (if claimed for the TOE), the design documentation had to address the design and implementation of the functions that implemented those SFRs down to the level of detail required for the target assurance level. In how far the requirements on the level of detail of ADV_TDS and ADV_IMP in CC v3.1 map to the requirements for the level of detail of ADV_HLD, ADV_LLD and ADV_IMP of CC v2.3 for the different assurance levels will be discussed in the mapping for those families.

2.3.2.ADV_ARC.1.2C

40 The security architecture description shall describe the security domains maintained by the TSF consistently with the SFRs.

41 ADV_ARC.1-2

The evaluator shall examine the security architecture description to determine that it describes the security domains maintained by the TSF.

42 There is no explicit requirement in CC v2.3 to identify the security domains maintained by the TSF, although there is the requirement for a TOE that includes a component from the FPT_SEP family to explain in the high-level design and (if required) the low-level design how the requirements of FPT_SEP is enforced. Of particular importance here is the work unit ADV_HLD.2-10 in CEM Version 2.3 that requires to check the high-level design for the description of the separation of the TOE into TSP-enforcing and other subsystems. This separation may be identical with the separation into domains with different levels of privileges, although the structure of a TOE into subsystems in a CC v2.3 evaluation may not be identical to the structure into domains as CC v3.1 requires for the security architecture description.

43 Another area where information for this work unit of CEM Version 3.1 can be found is in the evidence for work units ADV_HLD.1-9 and ADV_HLD.1-10 resp. ADV_HLD.2-11 and ADV_HLD.2-12 in CEM Version 2.3. Those work units require that the evidence provided is an accurate and complete instantiation of the security functional requirements, which includes the components from FPT_SEP and FPT_RVM if they are claimed in the ST. Since FPT_SEP requires the TSF to be in a domain for its own execution that is protected from external interference and tampering by untrusted subjects (see CC v2.3, part 2, paragraph 427), the high-level design needs to address those issues when FPT_SEP is claimed.

44 In addition work unit ADV_LLD.1-6 in CEM Version 2.3 requires that the low-level design describes how each of the TSP-enforcing functions is provided. Since this also includes the functions implementing FPT_SEP, the low-level design for a CC v2.3 evaluation also needs to address the structure of the TOE into security domains.
The evidence used for the work units mentioned above for ADV_HLD and ADV_LLD in a CC v2.3 evaluation will provide at least some evidence for the security domains maintained by the TSF but the details required for this description for CC v3.1 differ from the requirements for evidence of FPT_SEP in a CC v2.3 evaluation.

2.3.3.ADV_ARC.1.3C

The security architecture description shall describe how the TSF initialisation process is secure.

46 The security architecture description shall describe how the TSF initialisation process is secure.

47 ADV_ARC.1-3

The evaluator shall examine the security architecture description to determine that the initialisation process preserves security.

48 The initialization or start-up process of the TOE was not explicitly mentioned in CC v2.3. Although this is a very important aspect of a secure system, the words "start-up", "initialisation" or "power-up" are not used in CC v2.3 to require a description of the functions that bring the TSF into its initial state where it starts to react to external requests. Although design descriptions used for a CC v2.3 evaluation may contain a description of the TSF initialization process, there is no requirement for this documentation in CC v2.3 and there are no work units in CEM Version 2.3 that would assess the description of this process. Therefore work unit ADV_ARC.1-3 cannot be mapped to any work unit of CEM Version 2.3.

2.3.4.ADV_ARC.1.4C

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

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

50 ADV_ARC.1-4

The evaluator shall examine the security architecture description to determine that it contains information sufficient to support a determination that the TSF is able to protect itself from tampering by untrusted active entities.

51 For a TOE that includes a component from FPT_SEP in a CC v2.3 evaluation, the CC requires that the following aspects are described (see CC v2.3, part 2, paragraph 428):

a) The resources of the TSF's security domain ("protected domain") and those of subjects and unconstrained entities external to the domain are separated such that the entities external to the protected domain cannot observe or modify TSF data or TSF code internal to the protected domain.

b) The transfers between domains are controlled such that arbitrary entry to, or return from, the protected domain is not possible.

c) The user or application parameters passed to the protected domain by addresses are validated with respect to the protected domain's address space, and those passed by value are validated with respect to the values expected by the protected domain.

d) The security domains of subjects are distinct except for controlled sharing via the TSF.

52 All those are aspects that demonstrate that the TSF protects itself from external entities. Again work units ADV_HLD.1-9 and ADV_HLD.1-10 resp. ADV_HLD.2-11 and ADV_HLD.2-12 as well as work unit ADV_LLD.1-6 in CEM Version 2.3 should provide details how a) to d) are satisfied.

53 The explanation given for work unit ADV_ARC.1-4 in CEM Version 3.1 also mentions that self-protection is often achieved using hardware-based means (see paragraphs 533 and 534 of CEM Version 3.1). This is especially true for operating system type TOEs. CC v2.3 addressed those aspects in work units ADV_HLD.1-5 and ADV_HLD.1-6 resp. ADV_HLD.2-5 and ADV_HLD.2-6. Nevertheless there is an important difference here between the requirements of CC v2.3 and CC v3.1: while CC v2.3 just requires that the "functions provided by the supporting protection mechanisms implemented in the underlying hardware, firmware, or software" are presented (which is usually done in the processor manual), CC v3.1 requires that the security architecture description
describes how those hardware based functions are used by the TSF to protect itself from tampering by untrusted entities. Therefore CC v3.1 requires for example for an operating system type of TOE a description how memory protection mechanisms provided by the underlying hardware are used by the TSF to define a domain for itself that can not be accessed by untrusted entities in a way that may compromise the functions or data of the TSF. This is more than CC v2.3 has required, although one may argue that work units ADV_HLD.1-5 and ADV_HLD.1-6 resp. ADV_HLD.2-5 and ADV_HLD.2-6 together with work units ADV_HLD.1.9 and ADV_HLD.1.10 resp. ADV_HLD.2-11 and ADV_HLD.2-12 as well as work unit ADV_LLD.1-6 in CEM Version 2.3 should also cover this aspect.

54 **Note:** The guidance text for this work unit in CEM v3.1 uses in paragraph 537 the term "security architectural description". This means the security architecture description and should not be confused with the "architectural description of the TOE", which is defined as the overview of the TOE components and architecture produced by the evaluator as part of the ETR.

### 2.3.5.ADV_ARC.1.5C

55 The security architecture description shall demonstrate that the TSF prevents bypass of the SFR-enforcing functionality.

56 **ADV_ARC.1-5**

The evaluator **shall examine** the security architecture description to determine that it presents an analysis that adequately describes how the SFR-enforcing mechanisms cannot be bypassed.

57 This work unit addresses the non-bypassability property, which CC v2.3 covered by the SFR FPT_RVM.1 (there is only one component in the FPT_RVM family). A ST for a CC v2.3 evaluation that includes FPT_RVM.1 needs to address the issue of non-bypassability in its design documentation, since both the high-level design as well as the low-level design need to describe how the security functional requirements are implemented (high-level design) or how the TSP is enforced (low-level design). See the discussion of ADV_ARC.1-4 for the work units of CEM Version 2.3 that require this.

58 Part 2 of CC v2.3 provides in paragraph 423 the following guidance to be used in the description of how FPT_RVM is achieved:

* A TSF that implements a SFP provides effective protection against unauthorised operation if and only if all enforceable actions (e.g. accesses to objects) requested by untrusted subjects with respect to any or all of that SFP are validated by the TSF before succeeding. If an action that could be enforceable by the TSF, is incorrectly enforced or incorrectly bypassed, the overall enforcement of the SFP could be compromised. Subjects could then bypass the SFP in a variety of unauthorised ways (e.g. circumvent access checks for some subjects or objects, bypass checks for objects whose protection was assumed by applications, retain access rights beyond their intended lifetime, bypass auditing of audited actions, or bypass authentication). Note that some subjects, the so called "trusted subjects" with respect to a specific SFP, might be trusted to enforce the SFP by themselves, and bypass the mediation of the SFP.

59 While this seems to cover the aspect of non-bypassability in an adequate way, no specific guidance is provided how this aspect needs to be addressed in the design documentation. There is also a significant difference between this description and the guidance provided for work unit ADV_ARC.1-5 in CEM Version 3.1. The guidance focuses very much (or almost exclusively) on the aspect that an untrusted entity may use interfaces provide by the TSF that bypass the security checks like access control and allow access to protected objects. While for a CC v2.3 evaluation some more general statements in the design documentation arguing how non-bypassability is achieved were sufficient, the guidance for ADV_ARC.1-4 in CEM Version 3.1 explains in the following statement (see paragraph 548):

* The evaluator should also ensure that the description is comprehensive, in that each interface is analysed with respect to the entire set of claimed SFRs. 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 SFRs 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.
It is worth to note that the security architecture description is an important input also for the vulnerability analysis. In the security architecture description the developer provides his arguments how the TSF ensure non-bypassability and protect against tampering within the intended environment of the TOE. The evaluator should therefore consider those aspects of the security architecture description including his analysis of the architecture when he performs work units AVA_VAN.2-4, AVA_VAN.3-4 resp. AVA_VAN.4-4 with respect to bypassing and tampering.

3. ADV_FSP

3.1. Background of ADV_FSP

The purpose of the ADV_FSP family is the analysis of the external interfaces of the TSF (called TSFI) and the verification that they correctly implement the TSF as defined in the ST and do not provide a way for users of those interfaces to bypass or otherwise violate any SFR defined in the ST. So the analysis of the functional specification has two major aspects:

- Verification that all the security functions that require external interfaces have such interfaces (i.e. there is no security functionality described in the ST that can not be used due to missing interfaces)
- Verification that the interfaces do not provide functionality in a way contradicting any SFR defined in the ST

There is a third aspect that was introduced in CC v2.3 with ADV_FSP.2:

- Verification that all the external interfaces of the TSF have been identified.

Together with the second bullet point above this would guarantee that the TSF do not have any interface that can be used to violate any SFR. The first two bullets above are not sufficient to verify that the full set of TSFI is compliant with the set of SFRs of the TOE.

To perform the analysis of the functional specification, the interfaces need to be described with their method of use, overall function, possible input parameter, action performed when the function is called and effect of the function. The action and effect will usually depend on the "state" the function is called in and the parameter the function is called with. The action can be described in form of internal state changes and handling of TSF internal data, the effect can be described in terms of output parameter, effects on user data managed by the TSF, or exceptions raised. A special aspect of output parameter are error messages and a special case of state change can be the reaction on an error either caused by calling the function inadequately or by an error that occurred during the processing of the function.

There is no difference in the general purpose of the scope of the functional specification and its analysis between CC v2.3 and CC v3.1, although there is a major difference in the definition what the TSFI is and how it needs to be determined. This is discussed in detail in chapter 3.2.

CC v3.1 presents more components in the ADV_FSP family. While CC v2.3 only defined four components in this family, CC v3.1 presents six components. The two additional components are defined for the lower assurance levels, allowing to differentiate the content and presentation elements as well as the evaluator actions elements for EAL1 to EAL3. In CC v2.3 EAL1 to EAL3 all included the same component (ADV_FSP.1) resulting into the same requirements and evaluator actions related to the functional specification.

There is significant more guidance presented in CC v3.1 for the functional specification. New is the introduction of a classification of the TSFI into "SFR-enforcing", "SFR-supporting" and "SFR-non-interfering" (CC v3.1, part 3, paragraph 225). This replaces the notion of "direct interfaces to the TSF" and "indirect interfaces to the TSF", which are interfaces to "non-TSF portions of the TOE" that will eventually call TSF. As we will see in the discussion this is a fundamental difference that may lead to very different definitions of what the TSFI are.

New is also the classification of errors (and error messages) into "direct" error messages (which result from errors that can be tied to a specific TSFI invocation), "indirect errors" (which can not be tied to specific TSFI invocation) and "remaining errors" (some kind of "catch situations that should not occur"). The definition states that such error messages should never been seen (but in reality they occur more often than one would hope. Examples are Unix "kernel panic" errors and Windows "blue screens").
Another difference is that CC v3.1 now explicitly requires to describe the "method of use" for an interface which describes how the interface can be invoked.

Additional, more detailed differences can be found in the details of the requirements and work units which are discussed in the next chapter.

### 3.2. Discussion

As stated in the background chapter, in CC v3.1 the overall intention of the scope of a functional specification its analysis has not changed from CC v2.3. Still there are a number of differences in the individual requirements and work units.

The first obvious difference is of course the additional components of CC v3.1 in the ADV_FSP family. CC v2.3 had just four components in the ADV-FSP family with ADV_FSP.3 required at EAL5 and EAL6 and ADV_FSP.4 required at EAL7. For EAL1 to EAL3 CC v2.3 required ADV_FSP.1 and ADV_FSP.2 was required at EAL4. The difference between ADV_FSP.1 and ADV_FSP.2 in CC v2.3 is that ADV_FSP.2 requires the "complete description of all effects, exceptions and error messages" (emphasis added) as well as a rationale that the TSF is completely represented by the functional specification.

CC v3.1 now has 6 components in the ADV_FSP family. EAL1 needs to satisfy the requirements of ADV_FSP.1, EAL2 the requirements of ADV_FSP.2, EAL3 the requirements of ADV_FSP.3 and EAL4 the requirements of ADV_FSP.4. So, the old ADV_FSP.1 of CC v2.3 has been split into three different components (ADV_FSP.1 to ADV_FSP.3). The differences come partly in with the differentiation introduced in CC v3.1 between SFR-enforcing, SFR-supporting and SFR-non-interfering. ADV_FSP.1 in CC v3.1 requires the description of all SFR-enforcing and SFR-supporting TSFI and does not require the description of the SFR-non-interfering TSFI. The developer is just required to present "rationale for the implicit categorization of interfaces as SFR-non-interfering" (ADV_FSP.1.3C). There is also no requirement for the functional specification to completely represent the TSF and there is no requirement to describe all parameter of the security-enforcing and security-supporting interfaces. They just need to be identified.

This requirement is introduced in ADV_FSP.2 of CC v3.1. In addition in ADV_FSP.2 the developer has to present all TSFI, not just the security-enforcing and the security-supporting ones. In addition, all parameter need to be described, not just "identified". Concerning the "effect" of the functions associated with the interfaces, only the "SFR-enforcing actions" need to be described, together with the "direct error messages resulting from processing associated with the SFR-enforcing actions".

In ADV_FSP.3 of CC v3.1, a requirement is added to describe for the SFR-enforcing TSFI also the "exceptions associated with invocation of the TSFI". No guidance is provided how to deal with this addition in an evaluation and when in doubt how to handle this aspect in a specific evaluation the certification body should be consulted. The type of exceptions that may occur may differ largely for different types of TOEs and therefore it is hard to provide general guidance that applies for all types of TOEs. The only difference is in the guidance to the work units, where the guidance for ADV_FSP.3-7 states:

*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 actions.*

CC requirement ADV_FSP.3.5C only talks about "direct error messages resulting from security enforcing effects and exceptions associated with invocation of the TSFI" and where a "direct error message" is defined as "a security-relevant response that can be tied to a specific TSFI invocation" (CC v3.1, part 3, paragraph 236), the text of the work unit talks about "all error messages". In this case one shall assume that the text within the work unit refers to the type of error messages listed in the associated requirement and therefore applies to the direct error messages only.

ADV_FSP in Version 3.1 of the CC requires a mapping of the TSFI to the SFRs while ADV_FSP in Version 2.3 (and previous) of the CC required a mapping of the TSFI to the TSF as described in the TOE summary specification (TSS) of the ST. In CC v2.3 the ST was required to map the TSF as described in the TSS to the SFRs. So, by transitivity, the TSFI could be mapped to the SFRs also in CC v2.3. In CC v2.3 the TSS was often used to describe additional details of security functions that were hard to formulate as SFRs or would have resulted in highly complex SFR statements helping to bridge the gap between the abstract SFRs and the TSFI. Security Targets differed very much with respect to the additional details of the TSF being added in the TSS. In a transition from a CC v2.3 evaluation to a CC v3.1 evaluation the mapping needs to be redone, where the mapping
of the SFRs to the TSF (as described in the TSS) provided in the ST of the CC v2.3 evaluation may help to generate the mapping of the SFR to the TSFI as required by the CC v3.1 evaluation.

78 There is also potential difference in the scope of the TSFI: CC v3.1 explicitly does not view interfaces to the operational environment as part of the functional specification. Those interfaces need to be addressed either by ADV_TDS or ACO_REL. (see CC v3.1, part 3, paragraph 222). Together with the next discussion point this shows that the TSFI as viewed in CC v2.3 is different from the TSFI as viewed in CC v3.1.

79 There is another major difference between CC v2.3 and CC v3.1 in the characterization of the TSFI. CC v3.1 states in section A.2.1 "Determining the TSFI":

In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must first be identified. This identification is actually a part of the TOE design (ADV_TDS) analysis, but is also performed implicitly (through identification and description of the TSFI) by the developer in cases where TOE design (ADV_TDS) is not included in the assurance package. In this analysis, a portion of the TOE must be considered to be in the TSF if it contributes to the satisfaction of an SFR in the ST (in whole or in part). This includes, for example, everything in the TOE that contributes to TSF run-time initialisation, such as software that runs prior to the TSF being able to protect itself because enforcement of the SFRs has not yet begun (e.g., while booting up). Also included in the TSF are all parts of the TOE that contribute to the architectural principles of TSF self-protection, domain isolation, and non-bypassability (see Security Architecture (ADV_ARC)).

Once the TSF has been defined, the TSFI are identified. The TSFI consist of all means for users to invoke a service from the TSF (by supplying data that is processed by the TSF) and the corresponding responses to those service invocations. These service invocations and responses are the means of crossing the TSF boundary.

80 This makes it very clear that Version 3.1 of the CC just views those interfaces as part of the TSFI "that cross the TSF boundary".

81 CC v2.3 does not provide much guidance on what it views to be the TSFI. Instead this guidance is provided in the CEM in the explanation to work unit ADV_FSP.1-3 resp. ADV_FSP.2-3. There the functional specification is bound to the "external TOE security function interfaces" where it is explained that this may also include "interfaces to non-TSF portions of the TOE" (see CEM v2.3, paragraph 1304 and figure 12). According to the CEM v2.3 the TSFI consists of "direct interfaces to the TSF" and "indirect interfaces to the TSF" (the later being defined as an interface to a non-TSF portion of the TOE, where TSF functions are called as part of the processing). It is very clear that any interface to a non-TSF portion of the TOE is not viewed as a TSFI in CC v3.1. This is also reflected in the definition of TSFI. CC v3.1 defines the TSFI as follows:

TSF interface (TSFI) - a means by which external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF. (CC v3.1 part 1, paragraph 93)

82 While Version 2.3 defines TSFI as follows:

TOE security functions interface (TSFI) - a set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which TOE resources are accessed, mediated by the TSF, or information is obtained from the TSF. (CC v2.3 part 1, paragraph 63)

83 As a result, the interfaces that make up the TSFI may be defined as a different set depending on the version of the CC used.

84 In practice the definition of TSFI as stated in CC v2.3 has (hopefully) never been applied that way. In most cases the term TSFI has been used basically with the definition given in CC v3.1 and only interfaces accessible to users that cross the TSF boundary are considered to be part of the TSFI. This is independent from the fact that the TSF boundary may not be that easy to identify or may even vary with the operation of the TOE (an example of a dynamic TSF boundary is a process in Unix that operates with UID 0 and then performs a setuid to a non-zero UID (the standard way to start a user shell). As long as the process operates with UID 0, it is part of the TSF, as soon as it changes its UID to a non-zero value it is no longer part of the TSF.

85 As a summary of this discussion, in a transition from a CC v2.3 evaluation of a TOE to a CC v3.1 evaluation of the same TOE (or a newer version of it), the way the TSFI have been derived needs to be reviewed and checked for compliance with the definition of the TSFI in CC v3.1. Appendix A.2 of part 3 of CC v3.1 provides detailed
guidance and examples how to determine the TSF boundary and the TSFI. It is strongly suggested that when porting the evidence used for a CC v2.3 evaluation to serve as input for a CC v3.1 evaluation the developer shall use this guidance to check if the interfaces defined to make up the TSFI for the CC v2.3 evaluation also satisfy the new criteria for determining the TSFI for a CC v3.1 evaluation.

3.3. Mapping CEM Version 3.1 work units for ADV_FSP to CC v2.3

86 The following section lists the work units of the CEM for ADV_FSP.1 to ADV_FSP.4 and discusses, if and where related information can be found in documentation of a CC v2.3 evaluation. The structure of this chapter is to address the individual requirements as stated in the CC and CEM version 3.1 and attempt to map those – where possible – to corresponding (or closely corresponding) requirements and work units of CC and CEM Version 2.3. Differences are discussed with their implications on the evidence the developer has to provide and the work the evaluator has to perform.

87 CC v3.1 has included different components of FSP in each assurance level. This results of course also in differences in the associated work units of CEM v3.1. The way CEM v3.1 is structured, work units for different components do not necessarily map directly to each other. The following table shows how the work units for ADV_FSP.1 to ADV_FSP.4 of CEM v3.1 can be mapped. An (a) identifies where a work unit has been augmented, i. e. requires more compared to the one of the previous assurance level. Work units where the cell in the column for the previous assurance level is empty identifies new work units.

<table>
<thead>
<tr>
<th>ADV_FSP.1 (EAL1)</th>
<th>ADV_FSP.2 (EAL2)</th>
<th>ADV_FSP.3 (EAL3)</th>
<th>ADV_FSP.4 (EAL4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_FSP.1.1</td>
<td>ADV_FSP.2.2(a)</td>
<td>ADV_FSP.3.2</td>
<td>ADV_FSP.4.2</td>
</tr>
<tr>
<td>ADV_FSP.1.2</td>
<td>ADV_FSP.2.3(a)</td>
<td>ADV_FSP.3.3</td>
<td>ADV_FSP.4.3</td>
</tr>
<tr>
<td>ADV_FSP.1.3</td>
<td>ADV_FSP.2.4(a)</td>
<td>ADV_FSP.3.4</td>
<td>ADV_FSP.4.5</td>
</tr>
<tr>
<td>ADV_FSP.1.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ADV_FSP.1.5</td>
<td>ADV_FSP.2.8</td>
<td>ADV_FSP.3.9</td>
<td>ADV_FSP.4.10</td>
</tr>
<tr>
<td>ADV_FSP.1.6</td>
<td>ADV_FSP.2.9</td>
<td>ADV_FSP.3.10</td>
<td>ADV_FSP.4.11</td>
</tr>
<tr>
<td>ADV_FSP.1.7</td>
<td>ADV_FSP.2.10</td>
<td>ADV_FSP.3.11</td>
<td>ADV_FSP.4.12</td>
</tr>
<tr>
<td>ADV_FSP.2.1</td>
<td>ADV_FSP.3.11</td>
<td>ADV_FSP.4.1</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.2.2</td>
<td>ADV_FSP.3.5</td>
<td>ADV_FSP.4.5</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.2.3</td>
<td>ADV_FSP.3.6</td>
<td>ADV_FSP.4.7(a)</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.2.4</td>
<td>ADV_FSP.3.7(a)</td>
<td>ADV_FSP.4.8(a)</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.2.5</td>
<td>ADV_FSP.3.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.2.7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) in this table indicates that the work unit description has been augmented resulting in additional work to be performed by the evaluator compared to the corresponding work unit of the lower assurance level.

- In this table indicates that the work unit is no longer required at the higher level

3.3.1. ADV_FSP.1

88 ADV_FSP.1 in CC v3.1 is the component in the ADV_FSP family used in EAL1. In CC v2.3 the assurance level EAL1 used the component ADV_FSP.1 of CC v2.3. Although they have the same name those components have been defined differently in the different versions of the CC, resulting in different work units in the corresponding versions of the CEM. Those are analyzed in the following sections.

89 The following table provides an overview how the CEM work units for ADV_FSP.1 of CC v3.1 map to the work units of ADV_FSP.1 of CC v2.3. This mapping is relevant for the assurance level EAL1 where CC v3.1 includes the component ADV_FSP.1 (of CC v3.1) and CC v2.3 includes the component ADV_FSP.1 (of CC v2.3)

<table>
<thead>
<tr>
<th>CC/CEM v3.1</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_FSP.1.1E</td>
<td>ADV_FSP.1-1</td>
<td>ADV_FSP.1-5</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_FSP.1.1C</td>
<td>ADV_FSP.1-2</td>
<td>ADV_FSP.1-5</td>
<td>may contribute</td>
</tr>
</tbody>
</table>
Table 5: Mapping work units for ADV_FSP.1 to CEM v2.3

| ADV_FSP.1.3C | ADV_FSP.1-4 | - | essence |
| ADV_FSP.1.4C | ADV_FSP.1-5 | ADV_RCR.1-1 | essence |
| ADV_FSP.1.2E | ADV_FSP.1-6 | ADV_FSP.1-7 | identical |
| | ADV_FSP.1-7 | ADV_FSP.1-8 | identical |

90 This table shows that work unit ADV_FSP.1-5 in CEM v2.3 was the central work unit for ADV_FSP.1 in CC v2.3. To some extent it was "overloaded" and has been structured into several work units in CEM v3.1.

91 ADV_FSP.1.1C

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

93 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.

94 Related work units from CC v2.3

95 There is no directly corresponding work unit in CC v2.3 that has the same (or a very similar) text. The work unit that is closest in covering is:

96 ADV_FSP.1-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the behaviour of the TOE at each external interface describing effects, exceptions and error messages.

97 Discussion

98 One can assume that a description of the "behaviour" of the TOE at each external interface implicitly includes a description of the purpose of the function but in CC v2.3 this was not a requirement. A description of a function that explains all parameter, effects, and error messages would be acceptable in CC v2.3 even when no general description of the purpose of a function is provided.

99 There is no doubt that a general description of the purpose of a function helps very much in understanding and using a function. Therefore the description of most functional specifications can be expected to have such a general description of the purpose of each function. Nevertheless, in a transition from CC v2.3 to CC v3.1 the evaluator needs to verify that such a general description exists for each function where this is required.

100 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.

101 ADV_FSP.1.2C

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

103 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.

**104 Related work units from CC v2.3**

105 **ADV_FSP.1-5**

The evaluator **shall examine** the presentation of the TSFI to determine that it adequately and correctly describes the behaviour of the TOE at each external interface describing effects, exceptions and error messages.

**106 Discussion**

107 The "method of use" is not mentioned explicitly in the work units of CEM Version 2.3 but is mentioned in ADV_FSP.1.3C of CC v2.3. Implicitly the analysis of this aspect was therefore included and it was also required since the evaluator (at least when it comes to testing at the TSFI) needs to understand how the interface can be used. Therefore one can assume that the evaluator has checked this aspect also in a CC v2.3 evaluation without having an explicit statement in a CEM work unit to do so. Therefore this aspect can be regarded as a more formal aspect to examine that the method of use is adequately described for the user of the TOE who neither has the in-depth understanding of the TOE as the evaluator nor has access to internal developer documentation that may provide this information.

108 The analysis of the parameter of an interface is also not explicitly mentioned but is of course implicitly part of the analysis of the "effects" of a function (assuming that the parameters have an influence on the effect). It is quite normal in interface descriptions to describe such parameter related effects with the parameter itself. It is also required to describe each parameter in a way that the user is able to correctly provide the parameter when he uses the function.

109 Therefore CC v3.1 for the aspects described above provides explicit requirements for the content and presentation of the functional specification that implicitly existed also in CC v2.3. The additional evaluator action now is to verify that the description of the method of use and the description of the parameter is appropriate to use the function with its parameter.

110 There are two major difference between CC v3.1 and CC v2.3 with the requirement ADV_FSP.1.2C:

- The first one is the restriction to the SFR-enforcing and SFR-supporting functions. Since CC v2.3 did not make the distinction between those different types, there is no distinction between the interfaces in CC v2.3. Instead CC v2.3 in ADV_FSP.1-4 requires that the functional specification describes all of the TOE external interfaces and makes no distinction in the level of description required for individual interfaces. CC v3.1 in component ADV_FSP.1 therefore weakens somewhat the requirements of CC v2.3
- The second one is the requirement to describe all parameter of the SFR-enforcing and SFR-supporting interfaces. CC v2.3, which does not make a distinction between different "types" of TSFI, in component ADV_FSP.1 does not include the requirement to describe all parameter. CC v3.1 therefore has a stronger requirement with respect to the description of the parameter of SFR-enforcing and SFR-supporting TSFI.

111 To cope with those differences when migrating from CC v2.3 to CC v3.1 a developer needs to:

- Either analyze which of the TSFI of his TOE are SFR-enforcing or SFR-supporting and check that all parameter of those interfaces are described
- Or provide a description of all parameter for all TSFI and let the evaluator decide which of those interfaces are SFR-enforcing and SFR-supporting

112 In the case where the developer has not provided a categorization of the TSFI into SFR-enforcing, SFR-supporting and SFR-non-interfering, the evaluator needs to analyze the TSFI and do the categorization. He then needs to verify (as an activity additional to CC v2.3) that all parameter of the SFR-enforcing and SFR-supporting functions are fully and correctly described.

113 It is worth to note that due to the different definition of the TSFI in CC v2.3 and CC v3.1, the set of TSFI for the same TOE may be different depending on the version of the CC used. See the general discussion for details.

**114 ADV_FSP.1.3C**
115 The functional specification shall provide rationale for the implicit categorisation of interfaces as SFR-non-interfering.

116 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.

117 Related work units from CC v2.3

118 None.

119 Discussion

120 Since there is no differentiation between SFR-enforcing, SFR-supporting and SFR-non-interfering in CC v2.3, this work unit is new in CC v3.1 and has no corresponding activity in a CC v2.3 evaluation.

121 To satisfy this work unit the developer has to provide some kind of rationale that allows an evaluator to determine, which TSFI are SFR-non-interfering and follow the arguments provided by the developer, why those interfaces and the functions they call do not interfere with the security functions. This is an additional activity for both the developer (to provide the rationale) and the evaluator (to examine the rationale).

122 ADV_FSP.1.4C

123 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

124 ADV_FSP.1-5

The evaluator shall check that the tracing links the SFRs to the corresponding TSFIs.

125 Related work units from CC v2.3

126 ADV_RCR.1-1

The evaluator shall examine the correspondence analysis between the TOE summary specification and the functional specification to determine that the functional specification is a correct and complete representation of the TOE security functions.

127 Discussion

128 There are three major differences between CC v2.3 and CC v3.1 with this aspect:

- First, CC v2.3 required the developer to present a correspondence analysis between two adjacent layers of the design. This was more than just a tracing. Nevertheless a correspondence analysis developed in accordance with CC v2.3 should cover all the aspects CC v3.1 require for the tracing.

- Second, the tracing in CC v3.1 is between the SFRs and the TSFI while the correspondence analysis in CC v2.3 was between the TOE security functions as described in the TSS and the TSFI. Although CC v2.3 also requires the TOE security functions to be traced to the SFRs, this does not guarantee that the combination of the two mappings done for CC v2.3 results in a correct SFR to TSFI tracing. The TOE security function description in the TSS in a CC v2.3 Security Target, although being a refinement of the SFRs may still combine several SFRs into one security function (showing that all the details mentioned in the SFRs are addressed). Combining the two mappings may then result in mapping TSFIs to SFRs they are unrelated to. Therefore the developer has to carefully check the combined mapping of SFRs to security functions and security functions to TSFI and adjust this mapping where required.

- Third, the evaluator is just required to "check" the mapping in the CC v3.1 evaluation for ADV_FSP.1 and not "examine" the mapping as required in CC v2.3. Nevertheless work unit ADV_FSP.1-6 and ADV_FSP.1-7 in CC v3.1 still require the evaluator to examine the completeness and accuracy of the functional specification with respect to the SFRs. In evaluations using CC v2.3 the corresponding work units have often been combined with the correspondence analysis between SFRs and functional
specification. With this in mind, the type of correspondence analysis between the SFRs and the
functional specification performed for a CC v3.1 evaluation should not differ from the work performed in
a CC v2.3 evaluation and therefore most of the results from the CC v2.3 evaluation of this aspect should
be reusable within a CC v3.1 evaluation.

129 It is worth to be noted that work units ADV_FSP.1-7 and ADV_FSP.1-8 of CC v2.3 require the evaluator to
check the functional specification against the SFRs. In a transition from CC v2.3 to CC v3.1, the work performed
by the evaluator in the CC v2.3 evaluation for the work units mentioned above may provide a good starting point
for the tracing required in CC v3.1.

130 ADV_FSP.1.2E

131 ADV_FSP.1-6

The evaluator shall examine the functional specification to determine that it is a complete instantiation of the
SFRs.

132 ADV_FSP.1-7

The evaluator shall examine the functional specification to determine that it is an accurate instantiation of the
SFRs.

133 Related work units from CC v2.3

134 ADV_FSP.1-7

The evaluator shall examine the functional specification to determine that it is a complete instantiation of the
TOE security functional requirements.

135 ADV_FSP.1-8

The evaluator shall examine the functional specification to determine that it is an accurate instantiation of the
TOE security functional requirements.

136 Discussion

137 Those work units have identical text in both CC v2.3 and CC v3.1. CEM Version 3.1 provides more guidance
for an evaluator how to perform those work units. This should be taken into account in a transition from CC v2.3
to CC v3.1.

138 The different definition of TSFI between CC v2.3 and CC v3.1 may result in a different set of TSFI for the
same TOE. As with all other work units this may also have an effect on this work unit.

3.3.2.ADV_FSP.2

139 ADV_FSP.2 is the component from the ADV_FSP family in CC v3.1 that is used in the assurance level EAL2.
In CC v2.3 there was no difference in the requirements for the functional specification between EAL1 and EAL2.
EAL2 in CC v2.3 included the component ADV_FSP.1 defined in CC v2.3.

140 This section will only discuss the differences arising from the changes in ADV_FSP.2 compared to ADV_FSP.1
of CC v3.1. All other aspects have been discussed in section 3.1 above.

<table>
<thead>
<tr>
<th>CC/CEN v3.1</th>
<th>CC/CEN v2.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC Requirement</td>
<td>CEM Work Unit</td>
</tr>
<tr>
<td>ADV_FSP.2.1E</td>
<td>ADV_FSP.2.1E</td>
</tr>
<tr>
<td>ADV_FSP.2.1C</td>
<td>ADV_FSP.2.1C</td>
</tr>
<tr>
<td>ADV_FSP.2.2C</td>
<td>ADV_FSP.2.2C</td>
</tr>
</tbody>
</table>
141 ADV_FSP.2.1C

142 The functional specification shall completely represent the TSF.

143 ADV_FSP.2-1

The evaluator shall examine the functional specification to determine that the TSF is fully represented.

144 Related work units from CC v2.3

145 ADV_FSP.1-6

The evaluator shall examine the functional specification to determine that the TSF is fully represented.

146 Discussion

147 The text of the requirement and the related work unit are identical in CC/CEM Version 2.3 and CC/CEM Version 3.1. There is some difference in the content of the guidance of the work units. It is worth to note that CC v3.1 has significant more guidance material how to determine the TSF boundary and the TSFI. Together with the different definition of the TSFI in the two versions of the criteria, in a transition from CC v2.3 to CC v3.1 a thorough analysis needs to be performed to verify that the determination of the TSFI results in the same set of interfaces. If this is the case, the developer evidence as well as the evaluator actions for this requirement can be easily transferred from a CC v2.3 evaluation to a CC v3.1 evaluation.

148 ADV_FSP.2.2C

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

150 ADV_FSP.2-2

The evaluator shall examine the functional specification to determine that it states the purpose of each TSFI.

151 ADV_FSP.2-3

The evaluator shall examine the functional specification to determine that the method of use for each TSFI is given.

152 Related work units from CC v2.3

153 ADV_FSP.1-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the behaviour of the TOE at each external interface describing effects, exceptions and error messages.

154 Discussion
155 The work units exist also for ADV_FSP.1 of CC v3.1, but there they are restricted to the SFR-enforcing and SFR-supporting TSFI. In ADV_FSP.2 of CC v3.1 they are extended to all TSFI, including also the SFR-non-interfering TSFI. As explained in section 3.1 above, those work units do not exist with the same wordings in CC/CEM Version 2.3, but it can be assumed from the text of the CC v2.3 and the conduct of a CC v2.3 evaluation that the evidence and work required by this CC v3.1 work units has been presented/Performed also in a CC v2.3 evaluation. The details of the discussion for those work units in section 3.1 above applies.

156 ADV_FSP.2.3C

157 The functional specification shall identify and describe all parameters associated with each TSFI.

158 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.

159 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.

160 Related work units from CC v2.3

161 ADV_FSP.1-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the behaviour of the TOE at each external interface describing effects, exceptions and error messages.

162 Discussion

163 Work unit ADV_FSP.2-4 in CEM Version 3.1 is identical to work unit ADV_FSP.1-3 of CEM Version 3.1 except that it now addressed all TSFI and not only the SFR-enforcing and SFR-supporting ones. As discussed in section 3.1 above, this is a stricter requirement than the related one from CC v2.3 and therefore may require additional evidence as well as additional work by the evaluator. This is definitively an aspect where CC v3.1 has increased the requirements compared to the same assurance level in CC v2.3. Completeness of the parameter of a TSFI may be hard to verify with the design information provided at the EAL2 assurance level. The guidance provided for this work unit in CEM Version 3.1 states that the evaluator should check the evidence provided (functional specification, guidance, design, and implementation representation) to check for completeness. It is worth to note that the implementation representation is not necessarily provided at an assurance level that includes ADV_FSP.2.

164 Work unit ADV_FSP.2-5 in CEM Version 3.1 has no counterpart in the work units for ADV_FSP.1 in CEM Version 2.3. It strengthens the requirement to have a full description of all parameter at least with their purpose and possible values. This is stronger than work unit ADV_FSP.1-5 of CEM Version 2.3 requires, since the guidance for this work unit states that "All security relevant user input parameters (or a characterization of those parameters) should be identified." Applying this guidance, ADV_FSP.1 in CC v2.3 did not require the full description of all parameter, but restricted itself to "identification" of the "security relevant" parameter. This requires significant more evidence to be provided for a CC v3.1 evaluation at EAL2 than was required for a CC v2.3 evaluation at EAL2. Especially for complex TOEs that have a large set of TSFI where many, if not most of those are not "security relevant", the difference between CC v2.3 and CC v3.1 for the developer evidence with respect to the level of description of parameter of TSFI is significant.

165 ADV_FSP.2.4C

166 For each SFR-enforcing TSFIs, the functional specification shall describe the SFR-enforcing actions associated with the TSFI.

167 ADV_FSP.2-6
The evaluator shall examine the presentation of the TSFI to determine that it completely and accurately describes the SFR-enforcing actions associated with the SFR-enforcing TSFIs.

168 Related work units from CC v2.3

169 ADV_FSP.1-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the behaviour of the TOE at each external interface describing effects, exceptions and error messages.

170 Discussion

171 While ADV_FSP.2 in CC v3.1 requires all TSFI to be described with all parameter, it differentiates in the extent that the "actions" need to be described. ADV_FSP.2 requires only the "SFR-enforcing actions" to be described, limiting the specification of what the function called via the interface does to those "effects" or "actions" that are directly related to security enforcing functionality of the TSF. This distinction was not made in CC v2.3 where all "effects" have been covered by the same requirement.

172 CC v2.3 did not require a developer to distinguish between "security related" and "non-security related" functionality provided at an interface, although the guidance of work unit ADV_FSP.1-5 of CEM Version 2.3 talks about "security relevant" parameter without explaining what this means. It is worth to note that such a differentiation is also not required for ADV_FSP.2 in CC v3.1. Instead ADV_FSP.2 in CC v3.1 requires the evaluator to concentrate in his analysis on those "actions" that are related to the security enforcing functionality of the TSF. While CC v3.1 for ADV_FSP.2 requires all parameter of all TSFI to be described (listed with their purpose and possible values), it does not require the same level of detail in the description of the effects that can be caused by those parameters. Parameters associated with "actions" that are not SFR-enforcing may be described in general terms while those related to SFR-enforcing actions must be described in detail.

173 This clearly shifts the focus of the analysis of the functional specification for ADV_FSP.2 of CC v3.1 and as a result more details for the "security enforcing effects" than have been presented for a CC v2.3 evaluation may be required. The evaluator needs to (implicitly) categorize the "actions" each TSFI provides into SFR-enforcing and other actions to perform this work unit, an activity that was not explicitly required in CC v2.3. Note that according to the definition in CC v3.1 each TSFI that has at least one SFR-enforcing action is automatically classified as "SFR-enforcing" regardless how many non-SFR-enforcing additional actions are provided at this interface.

174 ADV_FSP.2.5C

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

176 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 actions associated with each SFR-enforcing TSFI.

177 Related work units from CC v2.3

178 ADV_FSP.1-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the behaviour of the TOE at each external interface describing effects, exceptions and error messages.

179 Discussion

180 While CC v2.3 generally talked about "error messages" CC v3.1 is more precise and restricts the required description of error messages to those that are directly related SFR-enforcing actions. This is a consequence of the focus on SFR-enforcing actions for the analysis of the functional specification and lifts the burden to check for the complete description of possible error messages, regardless if they are related to "SFR-enforcing actions" or
not. This adds to the clear focus of the analysis defined by ADV_FSP.2 on the SFR-enforcing actions associated with the functional specification.

**181 ADV_FSP.2.6C**

182 Requirement ADV_FSP.2.6C as well as the work units ADV_FSP.2-8 to ADV_FSP.2-10 are identical to the requirement ADV_FSP.1.4C and the work units ADV_FSP.1-5 to ADV_FSP.1-7 and therefore the discussion of those in section 3.1 also applies here.

**3.3.3.ADV_FSP.3**

183 ADV_FSP.3 is the component from the ADV_FSP family in CC v3.1 that is used in the assurance level EAL3. In CC v2.3 there was no difference in the requirements for the functional specification between EAL2 and EAL3. EAL3 in CC v2.3 included the component ADV_FSP.1 defined in CC v2.3.

184 This section will only discuss the differences arising from the changes in ADV_FSP.3 compared to ADV_FSP.2 of CC v3.1. All other aspects have been discussed in section 3.3.2 above.

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_FSP.3.1E</td>
<td>ADV_FSP.3-1</td>
<td>ADV_FSP.1-6</td>
<td>Identical</td>
</tr>
<tr>
<td>ADV_FSP.3.1C</td>
<td>ADV_FSP.3-2</td>
<td>ADV_FSP.1-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.3.2C</td>
<td>ADV_FSP.3-3</td>
<td>ADV_FSP.1-5</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.3.3C</td>
<td>ADV_FSP.3-4</td>
<td>ADV_FSP.1-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.3.4C</td>
<td>ADV_FSP.3-5</td>
<td>ADV_FSP.1-5</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.3.5C</td>
<td>ADV_FSP.3-6</td>
<td>ADV_FSP.1-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.3.6C</td>
<td>ADV_FSP.3-7</td>
<td>ADV_FSP.1-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.3.6E</td>
<td>ADV_FSP.3-8</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.3.7C</td>
<td>ADV_FSP.3-9</td>
<td>ADV_RCR.1-1</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.3.2E</td>
<td>ADV_FSP.3-10</td>
<td>ADV_FSP.1-7</td>
<td>Identical</td>
</tr>
<tr>
<td></td>
<td>ADV_FSP.3-11</td>
<td>ADV_FSP.1-8</td>
<td>Identical</td>
</tr>
</tbody>
</table>

Table 7: Mapping work units for ADV_FSP.3 to CEM v2.3

185 When looking into the requirements of ADV_FSP.3 in CC v3.1 and comparing those with the requirements of ADV_FSP.2 in the same document, there is only one additional requirement with one additional work unit, which is discussed below. All other requirements and work units are identical to those in ADV_FSP.2 of CC/CEM Version 3.1 and therefore the discussion of those in section 3.3.2 above applies. The only exemption is work unit ADV_FSP.3-7, which now applies to all error messages of the SFR-enforcing TSFI while ADV_FSP.2-7 only applied to the error messages of the SFR-enforcing TSFI that are related to the SFR-enforcing action. The additional requirement and work unit is:

**186 ADV_FSP.3.6C**

187 The functional specification shall summarise the SFR-supporting and SFR-non-interfering actions associated with each TSFI.

**188 ADV_FSP.3-8**

The evaluator shall examine the presentation of the TSFI to determine that it summarises the SFR-supporting and SFR-non-interfering actions associated with each TSFI.

**189 Related work units from CC v2.3**

190 None.

**191 Discussion**
192 CC v2.3 did not distinguish between "SFR-enforcing", "SFR-supporting" and "SFR-non-interfering". Therefore there was no need in a CC v2.3 evaluation for a developer to present a summary of the SFR-supporting and SFR-non-interfering actions of each TSFI. The guidance on work unit ADV_FSP.1-5 of CEM Version 2.3 clearly talks about the "security relevant behaviour" that "should be reflected in the description of semantics in the functional specification". This clearly restricts the analysis of the description to what CC v3.1 calls "SFR-enforcing" functionality. With component ADV_FSP.1 of CC v2.3 there is no requirement to describe any non-security relevant behaviour and therefore it can not be expected that the developer has presented such a description. On the other hand, the developer may present a full description of some or all interfaces that make up the TSFI since those interface description serve also the purpose to explain users how to use all its functions and parameter. In those cases the developer may have included the required evidence in the functional specification he has presented, but this was probably not analyzed in a CC v2.3 evaluation. Therefore in a transition from a CC v2.3 evaluation to a CC v3.1 evaluation where transition from ADV_FSP.1 of CC v2.3 to ADV_FSP.3 of CC v3.1 is required, it needs to be analyzed if the description of each TSFI contains sufficient information about the SFR-supporting and SFR-non-interfering actions. This requires this work unit to be performed completely when doing the transition.

3.3.4.ADV_FSP.4

193 ADV_FSP.4 is the component from the ADV_FSP family in CC v3.1 that is used in the assurance level EAL4. In CC v2.3 the difference in the requirements for the functional specification between EAL3 and EAL4 was the inclusion of the component ADV_FSP.2 instead of ADV_FSP.1 (of CC v2.3).

194 Since this presents the selection of different components for both CC v3.1 and CC v2.3, all requirements and work units are discussed here, referencing the discussion in previous sections where applicable.

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_FSP.4.1E</td>
<td>ADV_FSP.4-1</td>
<td>ADV_FSP.2-6</td>
<td>Identical</td>
</tr>
<tr>
<td>ADV_FSP.4.1C</td>
<td>ADV_FSP.4-2</td>
<td>ADV_FSP.2-7</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_FSP.4.2C</td>
<td>ADV_FSP.4-3</td>
<td>ADV_FSP.2-5</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_FSP.4.4C</td>
<td>ADV_FSP.4-4</td>
<td>ADV_FSP.2-3</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.4.3C</td>
<td>ADV_FSP.4-5</td>
<td>ADV_FSP.2-4</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.4.4C</td>
<td>ADV_FSP.4-6</td>
<td>ADV_FSP.2-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.4.5C</td>
<td>ADV_FSP.4-7</td>
<td>ADV_FSP.2-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.4.6C</td>
<td>ADV_FSP.4-8</td>
<td>ADV_FSP.2-5</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.4.2E</td>
<td>ADV_FSP.4-9</td>
<td>ADV_FSP.2-5</td>
<td>Essence</td>
</tr>
<tr>
<td></td>
<td>ADV_FSP.4-10</td>
<td>ADV_RCR.1-1</td>
<td>Essence</td>
</tr>
<tr>
<td></td>
<td>ADV_FSP.4-11</td>
<td>ADV_FSP.2-8</td>
<td>Identical</td>
</tr>
<tr>
<td></td>
<td>ADV_FSP.4-12</td>
<td>ADV_FSP.2-9</td>
<td>Identical</td>
</tr>
</tbody>
</table>

Table 8: Mapping work units for ADV_FSP4 to CEM v2.3

195 ADV_FSP.4.1C

196 The functional specification shall completely represent the TSF.

197 ADV_FSP.4-1

The evaluator shall examine the functional specification to determine that the TSF is fully represented.

198 Related work units from CC v2.3

199 ADV_FSP.2-6
The evaluator shall examine the functional specification to determine that the TSF is fully represented.

200 ADV_FSP.2-7

The evaluator shall examine the functional specification to determine that it contains a convincing argument that the TSF is completely represented by the functional specification.

201 Discussion

202 Compared to the discussion in section 3.2, ADV_FSP.2-7 is an additional related work unit from CEM Version 2.3. This work unit requires the evaluator to actively and explicitly search for TSF interfaces not contained in the functional specification evidence provided by the developer. CC v3.1 states: "The TSF must be identified (done as part of the TOE design (ADV_TDS) work units) in order to identify the TSFI." Since ADV_FSP.4 in CC v3.1 has a dependency on ADV_TDS.1 one would expect to find a work unit related to ADV_TDS.1 in the CEM Version 3.1 that deals with the identification of the TSFI. It is clearly the intention of ADV_FSP.4 in CC v3.1 to require the complete set of TSFI.

203 Note that in CC/CEM Version 2.3 work units ADV_FSP.2-3 and ADV-FSP.2-4 deal with the determination of the completeness of the TSFI description. See also the discussion in section 3.3.6 of this document.

204 ADV_FSP.4.2C

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

206 ADV_FSP.4-2

The evaluator shall examine the functional specification to determine that it states the purpose of each TSFI.

207 Related work units from CC v2.3

208 ADV_FSP.2-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the complete behaviour of the TOE at each external interface describing effects, exceptions and error messages.

209 Discussion

210 Work unit of ADV_FSP.4-2 in CEM Version 3.1 is identical to the work unit ADV_FSP.2-2 in CEM Version 3.1. The related work unit in CEM Version 2.3 (ADV_FSP.2-5) is stronger than the related work unit (ADV_FSP.1-5) it has been mapped to in section 3.2. Starting from ADV_FSP.2 CC v2.3 requires the functional specification to describe the "complete behaviour" of each TSFI. This seems to be very strong and the guidance in CEM Version 2.3 for this work unit still talks about the "complete security relevant behaviour". As stated before, CC v2.3 does not require a specific section in the evidence explaining the purpose of each TSFI, but it can reasonably expected that such a section exists in the developer documentation. Still this needs to be verified in the transition from CC v2.3 to CC v3.1.

211 ADV_FSP.4-3

The evaluator shall examine the functional specification to determine that the method of use for each TSFI is given.

212 Related work units from CC v2.3

213 ADV_FSP.2-5
The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the complete behaviour of the TOE at each external interface describing effects, exceptions and error messages.

214 Discussion

215 Work unit of ADV_FSP.4-3 in CEM Version 3.1 is identical to the work unit ADV_FSP.2-3 in CEM Version 3.1. Although requirement ADV_FSP.2-5 in CEM Version 2.3 is stronger than the one discussed for ADV_FSP.2-3 in section 3.2 above, this does not have an effect on the discussion of the method of use and therefore the discussion in section 3.2 on requirement ADV_FSP.2-3 applies here to.

216 ADV_FSP.4-4

The evaluator shall examine the functional specification to determine the completeness of the TSFI.

217 Related work units from CC v2.3

218 ADV_FSP.2-3

The evaluator shall examine the functional specification to determine that it identifies all of the external TOE security function interfaces.

219 ADV_FSP.2-4

The evaluator shall examine the functional specification to determine that it describes all of the external TOE security function interfaces.

220 The identification of the TSFI and the verification of its completeness is often a complex task. CC v3.1 provides significantly enhanced guidance on this subject. Work unit ADV_FSP.4-4 requires to validate the completeness of the TSFI, i.e. that all interfaces have been identified and described.

221 Discussion

222 Assuming that completeness is given when all the interfaces that make up the TSFI are identified and described (to the level of detail required by other work units), one can view the requirements of CC v3.1 and CC v2.3 with respect to the completeness of the TSFI as identical. Nevertheless one should consider the different definition of what makes up the TSFI in the different versions of the CC as described in chapter 3.2 and verify that this does not lead to a different set of TSFI.

223 ADV_FSP.4.3C

224 The functional specification shall identify and describe all parameters associated with each TSFI.

225 ADV_FSP.4-5

The evaluator shall examine the presentation of the TSFI to determine that it completely identifies all parameters associated with every TSFI.

226 Related work units from CC v2.3

227 ADV_FSP.2-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the complete behaviour of the TOE at each external interface describing effects, exceptions and error messages.

228 Discussion
229 Work unit of ADV_FSP.4-5 in CEM Version 3.1 is identical to the work unit ADV_FSP.2-4 in CEM Version 3.1. In contrast to the discussion of this requirement in section 3.3.2, ADV_FSP.2-5 of CEM Version 2.3 (where this work unit maps to) requires the description of the "complete behaviour of the TOE at each external interface describing effects, exceptions and error messages". In contrast to this statement in the work unit, the guidance for the work unit in CEM Version 2.3 still talks about the "security relevant input parameter" and the "complete security behaviour" (emphasis added) only. On the other hand, paragraph 1311 of the CEM Version 2.3 in the guidance for this work states that the evaluator should examine all available design and guidance evidence to discover parameter or error messages that have been omitted from the functional specification. This confirms that also CC v2.3 for ADV_FSP.2 requires the evaluator to verify that all parameter for each TSFI are described. Therefore ADV_FSP.4.3C of CC v3.1 is covered by ADV_FSP.2.3C of CC v2.3.

230 ADV_FSP.4-6

The evaluator shall examine the presentation of the TSFI to determine that it completely and accurately describes all parameters associated with every TSFI.

231 Related work units from CC v2.3

232 ADV_FSP.2-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the complete behaviour of the TOE at each external interface describing effects, exceptions and error messages.

233 Discussion

234 Work unit of ADV_FSP.4-6 in CEM Version 3.1 is identical to the work unit ADV_FSP.2-5 in CEM Version 3.1. Also this work unit maps to ADV_FSP.2-5 of CEM Version 2.3 and with the discussion above for ADV_FSP.4-5, this aspect is fully addressed by ADV_FSP.2-5 of CEM Version 3.1, which deals with both the identification and the description of the TSFI.

235 ADV_FSP.4.4C

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

237 ADV_FSP.4-7

The evaluator shall examine the presentation of the TSFI to determine that it completely and accurately describes all actions associated with every TSFI.

238 Related work units from CC v2.3

239 ADV_FSP.2-5

The evaluator shall examine the presentation of the TSFI to determine that it adequately and correctly describes the complete behaviour of the TOE at each external interface describing effects, exceptions and error messages.

240 Discussion

241 This work unit is similar to work unit ADV_FSP.2-6 of CEM Version 3.1, but now requires all actions to be described. ADV_FSP.2-6 only required the SFR-enforcing actions to be described.

242 This seems to be similar to the statement in the work unit ADV_FSP.2-5 of CEM Version 2.3, but as stated in the discussions before the very broad requirement of ADV_FSP.2-5 of CEM Version 2.3 is somewhat weakened in the guidance for the work unit, which talks about the "security relevant behaviour" only. Therefore it is very likely that in a CC v2.3 evaluation including ADV_FSP.2 of those criteria, the evaluator did not check for the complete description also of the non-security relevant behavior and the developer may not have completely presented this behavior in the evidence he presented for the functional specification. In a transition from CC v2.3 to CC v3.1, in
addition to the work performed for the CC v2.3 evaluation, the evidence for the functional specification needs to be checked for the complete description of the non-security relevant behavior of each TSFI.

243 ADV_FSP.4.5C

The functional specification shall describe all direct error messages that may result from security enforcing effects and exceptions associated with an invocation of each TSFI.

245 ADV_FSP.4-8

The evaluator shall examine the presentation of the TSFI to determine that it completely describes all errors messages resulting from an invocation of each TSFI.

246 Related work units from CC v2.3

247 ADV_FSP.2-5

The evaluator shall examine the presentation of the TSFI to determine that it completely describes all errors messages resulting from an invocation of each TSFI.

248 Discussion

249 Requirement ADV_FSP.4.5C of CC v3.1 and its associated work unit ADV_FSP.4-8 present a stricter requirement than ADV_FSP.2.5C in CC v2.3. The requirement now includes all direct error messages that may result from "security enforcing effects and exceptions" rather than "SFR-enforcing actions". When looking into the associated work units and their guidance in CEM Version 3.1, the requirement seem to imply that all error messages and exceptions need to be described and it is left to the evaluator to determine if the error or exception indicated by the message may have a security enforcing effect.

250 Comparing this with the text and the guidance of work unit ADV_FSP.2-5 of CEM Version 2.3 also there a complete description of error messages and exceptions is required. Therefore requirement ADV_FSP.4.5C of CC v3.1 should be fully addressed by a CC v2.3 evaluation that includes ADV_FSP.2.

251 ADV_FSP.4-9

The evaluator shall examine the presentation of the TSFI to determine that it completely describes the meaning of all errors associated with each TSFI.

252 Related work units from CC v2.3

253 ADV_FSP.2-5

The evaluator shall examine the presentation of the TSFI to determine that it completely describes the meaning of all errors associated with each TSFI.

254 Discussion

255 If the "meaning" of an error message is accurately described may be viewed as a subjective item. Here, as in other cases, the evaluator should examine if the error message is complete and understandable for the target audience.

256 Comparing this with the text and the guidance of work unit ADV_FSP.2-5 of CEM Version 2.3 also there a complete description of error messages and exceptions (including their "meaning") is required. Therefore requirement ADV_FSP.4.5C of CC v3.1 should be fully addressed by a CC v2.3 evaluation that includes ADV_FSP.2.
257 ADV_FSP.4.6C

258 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

259 ADV_FSP.4-10

The evaluator shall check that the tracing links the SFRs to the corresponding TSFIs.

260 Related work units from CC v2.3

261 ADV_RCR.1-1

The evaluator shall examine the correspondence analysis between the TOE summary specification and the functional specification to determine that the functional specification is a correct and complete representation of the TOE security functions.

262 Discussion

263 This requirement and the associated work unit in CC/CEM Version 3.1 is identical to requirement ADV_FSP.1.4C discussed in section 3.1. The discussion there also applies to ADV_FSP.4.6C, since there is also no difference in the mapping to requirements of CC/CEM Version 2.3.

264 Action ADV_FSP.4.2E

265 ADV_FSP.4-11

The evaluator shall examine the functional specification to determine that it is a complete instantiation of the SFRs.

266 Related work units from CC v2.3

267 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.

268 Discussion

269 This work unit has identical text in both CC v2.3 and CC v3.1. CEM Version 3.1 provides more guidance for an evaluator how to perform this work unit. This should be taken into account in a transition from CC v2.3 to CC v3.1.

270 The different definition of TSFI between CC v2.3 and CC v3.1 may result in a different set of TSFI for the same TOE. As with all other work units this may also have an effect on this work unit.

271 ADV_FSP.4-12

The evaluator shall examine the functional specification to determine that it is an accurate instantiation of the SFRs.

272 Related work units from CC v2.3

273 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.
**Discussion**

This work unit has identical text in both CC v2.3 and CC v3.1. CEM Version 3.1 provides more guidance for an evaluator how to perform this work unit. This should be taken into account in a transition from CC v2.3 to CC v3.1.

The different definition of TSFI between CC v2.3 and CC v3.1 may result in a different set of TSFI for the same TOE. As with all other work units this may also have an effect on this work unit.

### 3.3.5. CC v2.3 work units no longer required in CC v3.1

This section describes work units that exists in CEM Version 2.3 which have no direct counterpart in CEM Version 3.1 or can be mapped to a subactivity of a work unit of CEM Version 3.1. The analysis performed for those work units in CC v2.3 may therefore no longer be required in a CC v3.1 evaluation:

278 ADV_FSP.1-1 and ADV_FSP.2-1

The evaluator shall examine the functional specification to determine that it contains all necessary informal explanatory text.

279 ADV_FSP.1-2 and ADV_FSP.2-2

The evaluator shall examine the functional specification to determine that it is internally consistent.

Note that it is still the overall task of an evaluator to check for inconsistencies but this is no longer expressed as a single work unit but is a work that he needs to consider in all work units he performs. The evaluator needs to come to a judgment if the description of the functional specification is understandable for the target audience and does not contain inconsistencies. Any unresolved inconsistency is the result of either an incorrectness or an incompleteness and will therefore cause the work units that check for correctness and completeness of the functional specification to fail.

### 4. ADV_TDS

#### 4.1. Background of ADV_TDS

The purpose of the ADV_TDS family is the analysis of the TOE design. As explained in the CC, the purpose of the design is to provide information (with increasing level of detail for increasing assurance levels) how the security functions are implemented within the TSF. This information provided in the design shall allow an evaluator to trace the security functional requirements down to the lowest level of design documentation required for the target assurance level and verify that the security functionality are fully and correctly implemented. In addition the design documentation shall allow the evaluator to get an understanding how the individual components defined in the design co-operate and interoperate to provide the security functionality.

While CC v2.3 had two families dealing with design aspects (ADV_HLD and ADV_LLD), CC v3.1 has combined all those into one family ADV_TDS. The reason for this – as stated in CC v3.1 – is that fairly simple products may not need different "levels of decomposition" in the TOE design specification while highly complex products may require even more than just two layers. CC v3.1 explains in the Application notes for ADV_TDS:

*The goal of design documentation is to provide sufficient information to determine the TSF boundary, and to describe how the TSF implements the Security Functional Requirements. The amount and structure of the design documentation will depend on the complexity of the TOE and the number of SFRs; in general, a very complex TOE with a large number of SFRs will require more design documentation than a very simple TOE implementing only a few SFRs. Very complex TOEs will benefit (in terms of the assurance provided) from the production of differing levels of decomposition in describing the design, while very simple TOEs do not require both high-level and low-level descriptions of its implementation.*
As with CC v2.3 also ADV_TDS uses the terms "subsystem" for the decomposition at a more general abstraction level and the term "module" for the decomposition at a lower, more detailed level. Those terms as well as their general definition and use is common between CC v2.3 and CC v3.1.

CC v3.1 has added a section (12.6.1) in the introductory part of ADV_TDS that gives some explanations of terms used throughout the ADV_TDS requirements.

4.2. Discussion

4.2.1. Optional subsystem composition in CC v3.1

At a first glance the main difference between CC v2.3 and CC v3.1 is the fact that a developer no longer has the mandatory requirement to provide the subsystem level of abstraction in cases where a decomposition into "modules" is required. As paragraph 276 of CC v3.1 explains:

Very simple TOEs, in contrast, might not require a subsystem level of description; the module might clearly describe how the TOE works.

CC v3.1 does not give specific guidance for an evaluator when the developer may skip the delivery of the subsystem decomposition. Instead CC v3.1 leaves it to the evaluator to check that the requirements for the "subsystem" level decomposition are satisfied also by just the "module" decomposition provided. As long as this is given (and has been verified by the evaluator) the decomposition at the "subsystem" level may be omitted. Still the evaluator has to perform all the work units required for the "subsystem" level and ensure that the module decomposition provided by the developer satisfies all the requirements for subsystems. This is clearly expressed in CC v3.1, part 3, Appendix A.4.2.

4.2.2. Scope of the subsystem description

In CC v2.3 only the TSF needed to be described in terms of subsystems. CC v3.1 now requires the TOE to be described in terms of subsystems, where the subsystems that build the TSF need to be identified. In addition CC v3.1 also require that more than just the SFR-enforcing functionality is described (with increasing level of rigour for ADV_TDS.2 and ADV_TDS.3). This clearly requires more design documentation at the subsystem level for CC v3.1 than was required for CC v2.3. The subsystem design of CC v3.1 is intended to cover the full TOE and its functionality with the focus on the security functions while CC v2.3 just required the subsystem decomposition of the TSF and the description of the security functionality in those subsystems.

4.2.3. Level of detail in module design

While the guidance provided in Appendix A.4 for ADV_TDS is very helpful and clarifies a number of issues that have led to discussions and potentially different interpretation in CC v2.3, some additional guidance may be useful and is provided here. This is especially useful for the requirements on the level of detail expected in a module specification. Paragraph 586 of part 3 of CC v3.1 states:

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.

This actually requires that the module interfaces and the module internals are described wit sufficient detail to allow the evaluator to make his own model how the module works. When he would be tasked to implement the module himself the description provided should be sufficient for him to do this and not leave him in a state where is unable to perform his work. Still during his work he would need to make decision on some details of the implementation (and even some details of algorithms) and if the module design does is not required to provide details about SFR-supporting or SFR-non-interfering functions his module may differ significantly from the one in the TOE with respect to those parts. This has to be taken into account when looking into the level of detail required for a module design.

Since it is not the task of an evaluator to implement a module, paragraph 582 of part 3 of CC v3.1 and paragraph 796 of CEM v3.1 are of importance, which state:
Unlike subsystems, modules describe the implementation in a level of detail that can serve as a guide to reviewing the implementation representation.

This demonstrates one of the main functions the "module" design documentation has in an evaluation. To satisfy this requirement the level of detail required may be significantly lower than it seems at a first glance. The CEM actually supports this view. For the analysis of ADV_TDS.3-8 the CEM states:

The purpose of a module provides a description indicating what function the module is fulfilling. 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 subsystems. 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 complete algorithmic description for a self-evident implementation representation).

On the other hand part 3 of the CC in appendix A.4.2 "Modules" states in paragraph 591:

By contrast, interfaces used by a module must be identified such that it can be determined which module is being invoked by the module being described. It must also be clear from the design description the algorithmic reason the invoking module is being called. For example, 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 to the following rules...". The detailed description of a module in the 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 called module.

This example (which is also repeated several times in the CEM) seems to imply a significantly more detailed module design the evaluator needs to identify if that level of detail is required to understand how the module contributes to satisfy one or more SFRs. If the evaluator is not able to understand the contribution to the SFRs without those details (and is also not able to easily clarify those details using e.g. the implementation representation) he needs to ask for the additional detail. If those details are of no importance for his analysis, he should not force the developer to provide them.

4.2.4. Content of module design

CC v2.3 focused very much on the module interfaces and module interaction in its requirements for ADV_LLD.1. Although the intent of the low-level design, as expressed in paragraph 357 of part 3 of CC v2.3 states that the intent of this requirement is that the low-level design provide a description of how each module is expected to be implemented from a design perspective, this intent was not clearly expressed in the requirements. Module flow and data structures used, which are an important aspect of design documentation where not demanded in CC v2.3 to be described and analyzed to the same level of rigour as the module interfaces. Just the guidance for work unit ADV_LLD.1-6 indicates that also CC v2.3 wanted to have some level of detail on the module's internals:

It is this description in the low-level design that is key to the assessment as to whether the low-level design is sufficiently refined to permit an implementation to be created. The evaluator should analyse the description from the point of view of an implementor. If the evaluator, using the implementor's viewpoint, is unclear on any aspect of how the module could be implemented, the description is incomplete. Note that there is no requirement that a module be implemented as a separate unit (be it a program, a subprogram, or a hardware component); but the low-level design may be sufficiently detailed to permit such an implementation.

This left much room for interpretation, especially since this guidance was not really backed up by the requirements in the CC and the text of the work units in the CEM. It is assumed that all nations that have issued CC v2.3 (and earlier versions) certificates under the Common Criteria Recognition Arrangement (CCRA) have required more information on the module design than just the specification of the module interfaces.

CC v3.1 in the text of the requirements for modules in ADV_TDS.3 still focuses very much on module interfaces and interactions between modules. On the other hand some aspects are now clarified:
1. global data accessed by a module are considered to be module parameter (see paragraph 812 in CEM v3.1). On the other hand requiring that all global data that is just read by a module is listed as parameter in the module interface description may not always be required for the understanding. Quite often also the global data modified by a module is not listed in the module interface description but instead the definition of the global data has a section indicating which modules update the data structure. When performing work unit ADV_TDS.3-9 the evaluator should identify what he needs with respect to access to global data structures to get the required level of understanding how the TSF work.

2. algorithmic descriptions are now mentioned as an example to describe the module internals. CC v2.3 had ignored this important aspect. Nevertheless one has to acknowledge that algorithmic descriptions are not always the best approach to describe the purpose and internals of a module. For hardware modules other methods are more appropriate and should be used in this case. As pointed out above, algorithmic descriptions are just one example how to describe the purpose and internals of a module.

297 Although an understandable low-level design used for a CC 2.3 probably had both a description of global data and the module flow and algorithms, those have probably not been analyzed in a CC v2.3 evaluation with the scrutiny required by CC v3.1. Some updates to the low-level design documentation used for a CC 2.3 evaluation may therefore be required to satisfy the requirements of CC v3.1. Developers that have concentrated just on module interface specifications in their CC v2.3 evidence may have more problems to adapt their design documentation to the requirements of CC v3.1.

4.2.5 Dependencies on the IT environment

298 The CC v2.3 had specific requirements in ADV_HLD to describe the dependencies on the IT environment, which includes the underlying hardware, but also includes other components in the IT environment the TOE relies upon. This has been omitted in CC v3.1. While those dependencies could in theory be covered with the new ACO class, this only applies when two or more evaluated components are combined. This is not always the case. For example the dependencies of an operating system on the protection mechanism provided by the underlying hardware or an underlying hypervisor would be required to be described in CC v2.3. As far as the dependencies with respect to self-protection, domain isolation, non-bypassability and TSF initialization are concerned, those now need to be addressed in the security architecture description.

4.3. Mapping CEM Version 3.1 work units for ADV_TDS to CC v2.3

299 The following section lists the work units of the CEM for ADV_TDS.1 to ADV_TDS.3 and discusses, if and where related information can be found in documentation of a CC v2.3 evaluation. The structure of this chapter is to address the individual requirements as stated in the CC and CEM version 3.1 and attempt to map those – where possible – to corresponding (or closely corresponding) requirements and work units of CC and CEM Version 2.3. Differences are discussed with their implications on the evidence the developer has to provide and the work the evaluator has to perform.

<table>
<thead>
<tr>
<th>None (EAL1)</th>
<th>ADV_TDS.1 (EAL2)</th>
<th>ADV_TDS.2 (EAL3)</th>
<th>ADV_TDS.3 (EAL4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.1-1</td>
<td>ADV_TDS.2-1</td>
<td>ADV_TDS.3-1</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-2</td>
<td>ADV_TDS.2-2</td>
<td>ADV_TDS.3-2</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-3</td>
<td>ADV_TDS.2-3(m)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-4</td>
<td>ADV_TDS.2-6(a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-5</td>
<td>ADV_TDS.2-7</td>
<td>ADV_TDS.3-5</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-6</td>
<td>ADV_TDS.2-8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-7</td>
<td>ADV_TDS.2-9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-8</td>
<td>ADV_TDS.2-10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-9</td>
<td>ADV_TDS.2-4</td>
<td>ADV_TDS.3-4(m)</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1-10</td>
<td></td>
<td>ADV_TDS.3-4(m,a)</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.2-5</td>
<td>ADV_TDS.3-4(m,a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.2-6</td>
<td>ADV_TDS.3-5(m,a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.2-7</td>
<td></td>
<td>ADV_TDS.3-5(m,a)</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.2-8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.2-9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.2-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.3-11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 9: Corresponding CEM 3.1 work units for ADV_TDS

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.1.1E</td>
<td>ADV_TDS.1-1</td>
<td>ADV_HLD.1-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.1.1C</td>
<td>ADV_TDS.1-2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1.2C</td>
<td>ADV_TDS.1-3</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1.3C</td>
<td>ADV_TDS.1-4</td>
<td>ADV_HLD.1-4</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.1.4C</td>
<td>ADV_TDS.1-5</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1.5C</td>
<td>ADV_TDS.1-6</td>
<td>ADV_RCR.1-2</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.1.6C</td>
<td>ADV_TDS.1-7</td>
<td>ADV_HLD.1-9</td>
<td>major</td>
</tr>
<tr>
<td>ADV_TDS.1.7C</td>
<td>ADV_HLD.1-10</td>
<td>major</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1.8C</td>
<td>ADV_HLD.1-9</td>
<td>mainly identical</td>
<td></td>
</tr>
<tr>
<td>ADV_TDS.1.9C</td>
<td>ADV_HLD.1-10</td>
<td>major</td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Mapping work units for ADV_TDS.1 to CEM v2.3

301 ADV_TDS.1.1C

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

303 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 subsystems.

304 Related work units from CC v2.3

305 ADV_HLD.1-3

The evaluator shall examine the high-level design to determine that the TSF is described in terms of subsystems.

306 Discussion

307 The difference between the two requirements is that ADV_TDS.1.1 requires the TOE to be structured into subsystems while CC v2.3 in ADV_HLD.1-3 requires this only for the TSF. Otherwise there is also no real difference between the two versions in the characterization what a subsystem is. Therefore a subsystem structure of the TSF used in a CC v2.3 evaluation should also be acceptable in a CC v3.1 evaluation (potentially with additional subsystems that are not part of the TSF).
The design shall identify all subsystems of the TSF.

The evaluator shall examine the TOE design to determine that all subsystems of the TSF are identified.

Related work units from CC v2.3

None

Discussion

This work unit now determines which of the TOE subsystems are part of the TSF (a work that was not required in CC v2.3).

The design shall describe the behaviour of each SFR-supporting or SFR-non-interfering TSF subsystem in sufficient detail to determine that it is not SFR-enforcing.

The evaluator shall examine the TOE design to determine that each SFR-supporting or SFR-non-interfering subsystem of the TSF is described such that the evaluator can determine that the subsystem is SFR-supporting or SFR-non-interfering.

Related work units from CC v2.3

None

Discussion

The classification of subsystems in SFR-enforcing, SFR-supporting and SFR-non interfering is new in CC v3.1. CC v2.3 just required to map the security functions to subsystems allowing to trace them to the high-level design. A requirement to describe the behaviour and internal of subsystems that do not implement security functions to some level of detail did not exist in CC v2.3.

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

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 subsystems.

Related work units from CC v2.3

The evaluator shall examine the high-level design to determine that it describes the security functionality of each subsystem.
The evaluator shall examine the high-level design to determine that it is an accurate instantiation of the TOE security functional requirements.

328 Discussion

329 CC v3.1 requires a "complete, accurate, and high-level description of the SFR-enforcing behaviour" while CC v2.3 just requires a description that allows to accurately map the SFRs. Security enforcing behaviour may be more than just the functions used to implement the SFR. It may also include architectural support (e.g. for separation and reference mediation).

330 ADV_TDS.1.5C

331 The design shall provide a description of the interactions among SFR-enforcing subsystems of the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.

332 ADV_TDS.1-5

The evaluator shall examine the TOE design to determine that interactions between the subsystems of the TSF are described.

333 Related work units from CC v2.3

334 None

335 Discussion

336 The interactions between subsystems have not been analyzed in CC v2.3 for ADV_HLD.1. This is one of the areas where CC v3.1 requires more information on how the security functions are implemented.

337 ADV_TDS.1.6C

338 The mapping shall demonstrate that all behaviour described in the TOE design is mapped to the TSFIIs that invoke it.

339 ADV_TDS.1-6

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 subsystems of the TSF described in the TOE design.

340 Related work units from CC v2.3

341 ADV_RCR.1-2

The evaluator shall examine the correspondence analysis between the functional specification and the high-level design to determine that the high-level design is a correct and complete representation of the functional specification.

342 Discussion

343 In CC v2.3 the developer was required to provide a correspondence analysis between the functional specification and the high-level design, which contains a mapping from the functional specification (the TSFI) to the subsystems of the high-level design. This mapping in CC v2.3 was clearly intended to be a mapping from the more abstract level of the TSF representation down to the more detailed TSF representation as is clearly stated in ADV_RCR.1.1C. The purpose was to "demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation." (Text from ADV_RCR.1.1C in CC v2.3). CC v3.1 does not require this explicit "demonstration" from the developer but still requires a mapping of the TSFI to the subsystems in a way that allows the evaluator to check the consistency of the TSFI description with that of the subsystem the TSFI is mapped to.
344 ADV_TDS.1.2E

345 ADV_TDS.1-7
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.

346 ADV_TDS.1-8
The evaluator shall examine the TOE design to determine that it is an accurate instantiation of all security functional requirements.

347 Related work units from CC v2.3

348 ADV_HLD.1-9
The evaluator shall examine the high-level design to determine that it is an accurate instantiation of the TOE security functional requirements.

349 ADV_HLD.1-10
The evaluator shall examine the high-level design to determine that it is a complete instantiation of the TOE security functional requirements.

350 Discussion

351 While slightly different in their wordings, the intention of the work units of CC v3.1 listed above is identical to the one of the work units listed from CC v2.3.

4.3.2.ADV_TDS.2

352 ADV_TDS.2 is the component from the ADV_TDS family that is included in the EAL 3 assurance level. In CC v2.3 ADV_HLD.2 was included for EAL3. The following section therefore attempts to map the work units of those two.

<table>
<thead>
<tr>
<th>CC/CCEM v3.1</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.2.1C</td>
<td>ADV_TDS.2-1</td>
<td>ADV_HLD.2-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.2.2C</td>
<td>ADV_TDS.2-2</td>
<td>-</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.2.3C</td>
<td>ADV_TDS.2-3</td>
<td>ADV_HLD.2-4</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.2.4C</td>
<td>ADV_TDS.2-4</td>
<td>ADV_HLD.2-10</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-11</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_TDS.2.5C</td>
<td>ADV_TDS.2-5</td>
<td>-</td>
<td>partly</td>
</tr>
<tr>
<td>ADV_TDS.2.6C</td>
<td>ADV_TDS.2-6</td>
<td>-</td>
<td>partly</td>
</tr>
<tr>
<td>ADV_TDS.2.7C</td>
<td>ADV_TDS.2-7</td>
<td>ADV_HLD.2-7</td>
<td>partly</td>
</tr>
<tr>
<td>ADV_TDS.2.8C</td>
<td>ADV_TDS.2-8</td>
<td>ADV_RCR.1-2</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.2.2E</td>
<td>ADV_TDS.2-9</td>
<td>ADV_HLD.2-11</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-12</td>
<td>essence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-11</td>
<td>mainly identical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-12</td>
<td>mainly identical</td>
</tr>
</tbody>
</table>

Table 11: Mapping work units for ADV_TDS.2 to CEM v2.3

353 ADV_TDS.2.1C
The design shall describe the structure of the TOE in terms of subsystems.

355 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 subsystems.

356 Related work units from CC v2.3

357 ADV_HLD.2-3

The evaluator shall examine the high-level design to determine that the TSF is described in terms of subsystems.

358 Discussion

359 See discussion for ADV_TDS.1-1.

360 ADV_TDS.2.2C

The design shall identify all subsystems of the TSF.

361 The design shall identify all subsystems of the TSF.

362 ADV_TDS.2-2

The evaluator shall examine the TOE design to determine that all subsystems of the TSF are identified.

363 Related work units from CC v2.3

364 None

365 Discussion

366 See discussion for ADV_TDS.1-2.

367 ADV_TDS.2.3C

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

369 ADV_TDS.2-3

The evaluator shall examine the TOE design to determine that each SFR-non-interfering subsystem of the TSF is described such that the evaluator can determine that the subsystem is SFR-non-interfering.

370 Related work units from CC v2.3

371 None

372 Discussion

373 See discussion for ADV_TDS.1-3.

374 ADV_TDS.2.4C

375 The design shall describe the SFR-enforcing behaviour of the SFR-enforcing subsystems.

376 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 subsystems.

377 Related work units from CC v2.3

378 ADV_HLD.2-4

The evaluator shall examine the high-level design to determine that it describes the security functionality of each subsystem.

379 ADV_HLD.2-10

The evaluator shall check that the high-level design describes the separation of the TOE into TSP-enforcing and other subsystems.

380 ADV_HLD.2-11

The evaluator shall examine the high-level design to determine that it is an accurate instantiation of the TOE security functional requirements.

381 Discussion

382 Starting with ADV_HLD.2 also the CC v2.3 required a classification of the TSF subsystems into "TSP-enforcing" and "others". This classification is equivalent to the classification of "SFR-enforcing" and "non SFR-enforcing" defined in CC v3.1. Therefore the combination of work units from CC v2.3 listed above covers the aspect addressed by work unit ADV_TDS.2-4 of CC v3.1.

383 ADV_TDS.2.5C

384 The design shall summarise the SFR-supporting and SFR-non-interfering behaviour of the SFR-enforcing subsystems.

385 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 SFR-supporting and SFR-non-interfering behaviour of the SFR-enforcing subsystems.

386 Related work units from CC v2.3

387 None

388 Discussion

389 CC v2.3 totally focused on the security functions and did not require any description of the non-SFR enforcing behaviour. This is a new element introduced by CC v3.1.

390 In order to gain a good understanding of the full TSF also some information about the SFR-supporting and SFR-non-interfering functions is required to allow the evaluator to check for potential side effects of this behaviour.

391 ADV_TDS.2.6C

392 The design shall summarise the behaviour of the SFR-supporting subsystems.

393 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 subsystems.
394 Related work units from CC v2.3

396 Discussion

397 As stated above CC v2.3 solely concentrated on the security behaviour and neglected all other functionality almost completely. For a full understanding how a TOE provides its security functionality it is also of high importance, how the supporting functions work. Also the “SFR-non interfering” functions are of importance, since a flaw in those functions may potentially allow an attacker to influence or bypass also the security functionality. It is therefore natural and understandable that CC v3.1 also requires some design information for those subsystems.

398 ADV_TDS.2.7C

399 The design shall provide a description of the interactions among all subsystems of the TSF.

400 ADV_TDS.2-7

The evaluator shall examine the TOE design to determine that interactions between the subsystems of the TSF are described.

401 Related work units from CC v2.3

402 ADV_HLD.2-7

The evaluator shall check that the high-level design identifies the interfaces to the TSF subsystems.

403 ADV_HLD.2-9

The evaluator shall examine the high-level design to determine that it describes the interfaces to each subsystem in terms of their purpose and method of use, and provides details of effects, exceptions and error messages, as appropriate.

404 Discussion

405 The work units from CC v2.3 listed above are somewhat related to the requirement of CC v3.1, but do not cover it completely. The requirement for describing the interactions between the subsystems is in line with the shift of focus in CC v3.1 to have the design describe more of the flow, algorithms and interactions and not focus almost solely on the interface descriptions.

406 ADV_TDS.2.8C

407 The mapping shall demonstrate that all behaviour described in the TOE design is mapped to the TSFIIs that invoke it.

408 ADV_TDS.2-8

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 subsystems of the TSF described in the TOE design.

409 Related work units from CC v2.3

410 ADV_RCR.1-2

The evaluator shall examine the correspondence analysis between the functional specification and the high-level design to determine that the high-level design is a correct and complete representation of the functional specification.
411 Discussion

412 See discussion for ADV_TDS.1-6 above.

413 ADV_TDS.2-9

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.

414 ADV_TDS.2-10

The evaluator shall examine the TOE design to determine that it is an accurate instantiation of all security functional requirements.

415 Related work units from CC v2.3

416 ADV_HLD.2-11

The evaluator shall examine the high-level design to determine that it is an accurate instantiation of the TOE security functional requirements.

417 ADV_HLD.2-12

The evaluator shall examine the high-level design to determine that it is a complete instantiation of the TOE security functional requirements.

418 Discussion

419 See discussion for ADV_TDS.1-7 and ADV_TDS.1-8 above.

4.3.3.ADV_TDS.3

420 ADV_TDS.3 is the component from the ADV_TDS family in CC v3.1 that is used in the assurance level EAL4. In CC v2.3 the components ADV_HLD.2 and ADV_LLD.1 were included in EAL4.

421 CC v3.1 has attempted to combine the requirements of ADV_HLD and ADV_LLD into ADV_TDS. While at EAL4, the requirements for a low-level design (ADV_LLD.1) in CC v2.3 focused very much on the interfaces of the modules, the requirements of ADV_TDS.3 included in EAL4 for CC v3.1 contains additional demands for the description of the purpose and internal structure of the modules. For CC v2.3 evaluations many schemes accepted a low-level design only when also sufficient details had been provided for the purpose and internals of modules that were "TSP enforcing" (term used in CC v2.3). Evaluations at EAL4 performed under those schemes will most likely be based on low-level design documentation that satisfies most if not all of the requirements of ADV_TDS.3 for the description of the purpose and internals of "SFR-enforcing" modules.

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.3.1E</td>
<td>ADV_TDS.3-1</td>
<td>ADV_HLD.2-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.3.1C</td>
<td>ADV_TDS.3-2</td>
<td>ADV_HLD.1-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.3.2C</td>
<td>ADV_TDS.3-3</td>
<td>-</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.3.3C</td>
<td>ADV_TDS.3-4</td>
<td>ADV_HLD.2-4</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.3.4C</td>
<td>ADV_TDS.3-5</td>
<td>ADV_HLD.2-7</td>
<td>partly</td>
</tr>
<tr>
<td>ADV_TDS.3.5C</td>
<td>ADV_TDS.3-6</td>
<td>ADV_HLD.2-9</td>
<td>partly</td>
</tr>
<tr>
<td>ADV_TDS.3.6C</td>
<td>ADV_TDS.3-7</td>
<td>ADV_RCR.1-3</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.3.7C</td>
<td>ADV_TDS.3-8</td>
<td>ADV_HLD.1-4</td>
<td>mainly identical</td>
</tr>
</tbody>
</table>
Table 12: Mapping work units for ADV_TDS.3 to CEM v2.3

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.3.8C</td>
<td>ADV_TDS.3-9</td>
<td>ADV_LLD.1-7</td>
<td>partly</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-9</td>
<td>essence</td>
</tr>
<tr>
<td>ADV_TDS.3.9C</td>
<td>ADV_TDS.3-10</td>
<td>ADV_LLD.1-5</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-9</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_TDS.3-11</td>
<td>ADV_LLD.1-4</td>
<td>ADV_LLD.1-5</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-9</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_TDS.3-12</td>
<td>ADV_LLD.1-5</td>
<td>partly</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-9</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_TDS.3.10C</td>
<td>ADV_TDS.3-13</td>
<td>ADV_HLD.2-11</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-12</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-11</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-12</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_RCR.1-2</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_RCR.1-3</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_TDS.3.2E</td>
<td>ADV_TDS.3-14</td>
<td>ADV_HLD.2-11</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-12</td>
<td>major</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-11</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-11</td>
<td>major</td>
</tr>
<tr>
<td>ADV_TDS.3-15</td>
<td>ADV_HLD.2-11</td>
<td>major</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_HLD.2-12</td>
<td>may contribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-11</td>
<td>major</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADV_LLD.1-12</td>
<td>may contribute</td>
</tr>
</tbody>
</table>

422 ADV_TDS.3.1C

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

424 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 subsystems.

425 Related work units from CC v2.3

426 ADV_HLD.2-3

The evaluator shall examine the high-level design to determine that the TSF is described in terms of subsystems.

427 Discussion

428 See discussion for ADV_TDS.1-1.

429 ADV_TDS.3.2C

430 The design shall describe the TSF in terms of modules.

431 ADV_TDS.3-2

The evaluator shall examine the TOE design to determine that the entire TSF is described in terms of modules.

432 Related work units from CC v2.3
The evaluator shall check the low-level design to determine that it describes the TSF in terms of modules.

**Discussion**

The difference here is with the word "examine" that is used in CC v3.1, while CC v2.3 used the weaker term "check".

**ADV_TDS.3.3C**

The design shall identify all subsystems of the TSF.

**ADV_TDS.3-3**

The evaluator shall examine the TOE design to determine that all subsystems of the TSF are identified.

**Related work units from CC v2.3**

None.

**Discussion**

See discussion for ADV_TDS.1-2.

**ADV_TDS.3.4C**

The design shall provide a description of each subsystem of the TSF.

**ADV_TDS.3-4**

The evaluator shall examine the TOE design to determine that each subsystem of the TSF describes its role in the enforcement of SFRs described in the ST.

**Related work units from CC v2.3**

ADV_HLD.2-4

The evaluator shall examine the high-level design to determine that it describes the security functionality of each subsystem.

ADV_HLD.2-10

The evaluator shall check that the high-level design describes the separation of the TOE into TSP-enforcing and other subsystems.

ADV_HLD.2-11

The evaluator shall examine the high-level design to determine that it is an accurate instantiation of the TOE security functional requirements.

**ADV_HLD.2-12**

The evaluator shall examine the high-level design to determine that it is a complete instantiation of the TOE security functional requirements.

**Discussion**
Starting with ADV_HLD.2 the CC v2.3 required a classification of the TSF subsystems into "TSP-enforcing" and "others". This classification is equivalent to the classification of "SFR-enforcing" and "non SFR-enforcing" defined in CC v3.1. For ADV_TDS.3.1 the CC v3.1 now requires a description of each subsystem of the TSF. This is a significantly stronger requirement than what was included in CC v2.3.

**ADV_TDS.3.5C**

The design shall provide a description of the interactions among all subsystems of the TSF.

**ADV_TDS.3-5**

The evaluator shall examine the TOE design to determine that interactions between the subsystems of the TSF are described.

**Related work units from CC v2.3**

ADV_HLD.2-7

The evaluator shall check that the high-level design identifies the interfaces to the TSF subsystems.

ADV_HLD.2-9

The evaluator shall examine the high-level design to determine that it describes the interfaces to each subsystem in terms of their purpose and method of use, and provides details of effects, exceptions and error messages, as appropriate.

**Discussion**

See discussion for ADV_TDS.2-7 above. ADV_TDS.3.5C requires the description of the interaction between subsystems. Note that interaction between modules is required in ADV_TDS.3.5C for SFR-enforcing modules and in ADV_TDS.3.9C for the interaction between SFR-supporting and SFR-non interfering modules with all other modules. The reason to separate this the different level of detail required at the module level in ADV_TDS.3 for SFR-enforcing and SFR-supporting or SFR-non-interfering modules. See the discussion later in this chapter for the related work units that check the interaction between modules.

**ADV_TDS.3.6C**

The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

**ADV_TDS.3-6**

The evaluator shall examine the TOE design to determine that the mapping between the subsystems of the TSF and the modules of the TSF is complete.

**Related work units from CC v2.3**

ADV_RCR.1-3

The evaluator shall examine the correspondence analysis between the high-level design and the low-level design to determine that the low-level design is a correct and complete representation of the high-level design.

**Discussion**

As discussed previously with the mapping from the functional specification to the design, CC v2.3 had a stricter requirement for the developer to provide a "correspondence analysis" where this mapping was a part of. In the case where a low-level design (at the module level) was included, such a correspondence analysis was also required for the mapping from the subsystems of the high-level design to the modules of the low-level design.
The mapping that CC v3.1 requires is just a subset of what was required for the correspondence analysis in CC v2.3.

468 Note that the mapping from the subsystems to the modules is restricted to the TSF subsystems.

469 It is also worth to note that ADV_TDS.3.2D requires a mapping from the TSFI to the modules. This leads to the following mappings required in CC v3.1 for ADV_TDS.3:

470 TSFI à modules

471 Subsystems à modules

472 A mapping from the TSFI to the subsystems is not required for ADV_TDS.3. This is in contrast to CC v2.3, which required the following mappings between the TSFI and the related design components included in EAL4 (ADV_HLD.2, ADV_LLD.1, ADV_RCR.1):

473 TSFI à subsystems

474 Subsystems à modules

475 This shows that the direct mapping from the TSFI to the modules was not part of a CC v2.3 evaluation. Such a mapping can not be always be easily constructed by combining the two mappings. The information contained in the correspondence analysis of a CC v2.3 evaluation may not always allow to identify the module that implements a TSFI. Such a mapping from the TSFI to the module that is invoked when the TSFI is called may also not always be very useful. For example in many operating systems a system call may initially be handled by an interrupt service module that perform common aspects of all systems calls (saving the environment, disabling specific interrupts, securely copy parameter etc.) before it then calls the module that handles a specific TSFI. Mapping all TSFI to the interrupt service module is not a useful mapping in this case.

476 ADV_TDS.3-7

The evaluator shall examine the TOE design to determine that the mapping between the subsystems of the TSF to the modules of the TSF is accurate.

477 Related work units from CC v2.3

478 ADV_RCR.1-3

The evaluator shall examine the correspondence analysis between the high-level design and the low-level design to determine that the low-level design is a correct and complete representation of the high-level design.

479 Discussion

480 The accuracy of the mapping between the subsystems and the modules is also part of the correspondence analysis.

481 ADV_TDS.3.7C

482 The design shall describe each SFR-enforcing module in terms of its purpose and interaction with other modules.

483 ADV_TDS.3-8

The evaluator shall examine the TOE design to determine that the description of the purpose of each SFR-enforcing module is complete and accurate.

484 Related work units from CC v2.3
The evaluator shall examine the low-level design to determine that it describes the purpose of each module.

**Discussion**

CC v3.1 requires the description of the purpose of all modules that comprise the TSF (see ADV_TDS.3.9C) while more details about the internals are required for the SFR-enforcing modules. This allows therefore to identify the purpose of all modules and therefore is in sum equivalent to work unit ADV_LLD.1-4 of CC v2.3.

**ADV_TDS.3.8C**

The design shall describe each SFR-enforcing module in terms of its SFR-related interfaces, return values from those interfaces, interactions with and called interfaces to other modules.

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 SFR-related parameters, the invocation conventions for each interface, and any values returned directly by the interface.

**Related work units from CC v2.3**

The evaluator shall check that the low-level design identifies the interfaces to the TSF modules.

The evaluator shall examine the low-level design to determine that it describes the interfaces to each module in terms of their purpose and method of use, and provides details of effects, exceptions and error messages, as appropriate.

**Discussion**

Here is clearly a difference between CC v3.1 and CC v2.3. While CC v3.1 differentiates between “SFR-related parameter” and other parameter, CC v2.3 does not differentiate. On the other hand, the CC v2.3 work unit ADV_LLD.1-9 has the text “as appropriate” at the end, leaving the decision what is considered to be appropriate to the evaluator. Focusing on the “SFR-related” parameter would for sure be considered as an appropriate method, but there is no guarantee that this has been used in this way in evaluations.

Combining the requirements of ADV_TDS.3.7C and ADV_TDS.3.8C, version 3.1 of the CC requires for each SFR-enforcing module to describe its purpose, its interaction with other modules and its SFR-related interfaces.

**ADV_TDS.3.9C**

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

The evaluator shall examine the TOE design to determine that SFR-supporting and SFR-non-interfering modules are correctly categorised.

The evaluator shall examine the TOE design to determine that the description of the purpose of each SFR-supporting or SFR-non-interfering module is complete and accurate.
502 ADV_TDS.3-12

The evaluator \textit{shall examine} the TOE design to determine that the description of a SFR-supporting or SFR-non-interfering module's interaction with other modules is complete and accurate.

503 Related work units from CC v2.3

504 ADV_LLD.1-4

The evaluator shall examine the low-level design to determine that it describes the purpose of each module.

505 ADV_LLD.1-5

The evaluator shall examine the low-level design to determine that it defines the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

506 ADV_LLD.1-9

The evaluator shall examine the low-level design to determine that it describes the interfaces to each module in terms of their purpose and method of use, and provides details of effects, exceptions and error messages, as appropriate.

507 Discussion

508 CC v2.3 did not require a categorization of the modules but implicitly only required those modules to be described that included SFR-enforcing functions. Nevertheless in most cases the developer provided more module design specifications than just for the SFR-enforcing modules, leaving it to the evaluator to identify the SFR-enforcing functions in the individual modules. As mentioned previously, the requirements on the description of the interaction between modules in CC v2.3 was lower than in CC v3.1. Therefore an modules specification delivered for a CC v2.3 evaluation has to be analyzed if the description of the module interactions is sufficient.

509 ADV_TDS.3.10C

510 The mapping \textit{shall demonstrate} that all behaviour described in the TOE design is mapped to the TSFIs that invoke it.

511 ADV_TDS.3-13

The evaluator \textit{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.

512 ADV_TDS.3-14

The evaluator \textit{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.

513 ADV_TDS.3-15

The evaluator \textit{shall examine} the TOE design to determine that it is an accurate instantiation of all security functional requirements.

514 Related work units from CC v2.3

515 ADV_HLD.2-11

The evaluator \textit{shall examine} the high-level design to determine that it is an accurate instantiation of the TOE security functional requirements.
The evaluator shall examine the high-level design to determine that it is a complete instantiation of the TOE security functional requirements.

The evaluator shall examine the low-level design to determine that it is an accurate instantiation of the TOE security functional requirements.

The evaluator shall examine the low-level design to determine that it is a complete instantiation of the TOE security functional requirements.

The evaluator shall examine the correspondence analysis between the functional specification and the high-level design to determine that the high-level design is a correct and complete representation of the functional specification.

The evaluator shall examine the correspondence analysis between the high-level design and the low-level design to determine that the low-level design is a correct and complete representation of the high-level design.

At a first glance the mapping from the TSFI to the modules looks reasonable, but when checked with reality it may not be that reasonable any more. For example in an operating system one can assign system calls to "subsystems" where one abstracts from the fact that all system calls first go to one common module for parameter checking and setting up the environment. At the module level this abstraction no longer holds and now all system call TSFI map to a single module! Mapping the TSFI to modules may not always be useful.

Work units ADV_TDS.3-14 and ADV_TDS.3-15 also are clearly enough expressed. The mapping of the SFRs to the subsystems of the TOE design may well be just the mapping from the SFRs to the TSF subsystems and not to the modules. If this was intended one could have taken the same work units as in ADV_TDS.2. If a mapping to the module level is intended, this should be clearly expressed.

4.3.4.CC v2.3 work units no longer included in CC v3.1

CC v2.3 had the following requirement and associated work units related to the IT environment:

The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

The evaluator shall check the high-level design to determine that it identifies all hardware, firmware, and software required by the TSF.
The evaluator **shall examine** the high-level design to determine that it includes a presentation of the functions provided by the supporting protection mechanisms implemented in the underlying hardware, firmware, or software.

528 This required the design to describe the TOE and its security functions in the context of his IT environment, explaining dependencies on functions of the IT environment. It is unclear why this is no longer addressed in CC v3.1 in ADV_TDS and which other class of the CC v3.1 is intended to cover this important aspect.

5. ADV_IMP

5.1. Background of ADV_IMP

529 The Class ADV_IMP is related to the analysis of the "implementation representation" of the TSF, which, dependent on the technology used to implement the TOE, can be "software source code, firmware source code, hardware diagrams and/or IC hardware design language code or layout data" (CC v3.1, part3).

530 Both CC v3.1 as well as CC v2.3 have included the component ADV_IMP.1 at the EAL4 assurance level. For lower levels of assurance in both versions of the CC the developer was not required to provide all or part of the implementation representation and there were (of course) also no evaluator actions for analyzing the implementation representation.

531 The purpose of the analysis of the implementation representation is to allow the evaluator to analyze the details of security functionality down to a "design" level where additional details are only added by automated tools (like compilers), but no design decision are made. The "implementation representation" is usually the lowest level of the design that is easily understandable by a human. Due to the level of detail expressed there the "implementation representation" may be highly complex. It is not impossible that the implementation representation of the TSF of a complex TOE may consists of several hundred thousand lines of source code or a very large number of statements of IC design language. Due to this complexity, a full analysis that traces all security functionality down to this level and searching the full implementation representation of the TSF for potential vulnerabilities may likely to be impracticable for such complex TOEs. Therefore the analysis of the implementation representation is restricted to a sample at ADV_IMP.1, the component included in the EAL4 assurance level. This applies for both CC v3.1 as well as CC v2.3.

532 At the level ADV_IMP.1 both CC v3.1 as well as CC v2.3 require only the analysis of a sample of the implementation representation. In the selection of this sample the evaluator has (as is discussed in the next section in more detail) more freedom with CC v3.1 than he had with CC v2.3.

5.2. Discussion

533 Looking into the differences between CC v3.1 and CC v2.3 with respect to ADV_IMP.1 two aspects are immediately obvious:

1. In CC v2.3 the developer was only required to present a sample of the implementation representation of the TSF while in CC v3.1 the developer is required to present the full implementation representation of the TSF.

2. CC v3.1 provides significantly more guidance material on what the implementation representation is and what the evaluator has to get from the developer. This includes the requirement to present all the parameter and options used by the tools that transform the implementation representation into the actual implementation (e. g. compiler options, precompiler settings, transformation scripts). CC v3.1 also requires that the evaluator has access to the form of the implementation representation as used by the developer. The intention is to have the implementation representation in a form that is understandable by a human.

534 Those differences are related to the items the developer has to deliver. Although both requirements at a first glance seem to be a major increase on the items to be delivered by the developer, they should not impose any real additional technical effort for the developer, since all the information requested must be available to pass the CC requirements of the ALC class for EAL4.

535 While there may be no technical difficulties, there still be legal restrictions that may make it harder for the developer to satisfy the first aspect. For example a developer may have used a library developed by a third party...
in the development of the TSF. While the developer has the implementation representation and the design documentation of this library, he may have signed a non-disclosure agreement that does not allow him to give third parties access to this implementation representation. In CC v2.3 this library could be excluded from the sample provided to the evaluator (if the evaluator accepted that the library was not part of the sample where he wanted to trace the design down to the implementation representation). With CC v3.1 the implementation representation has to be made available to the evaluator, regardless if is part of the sample the evaluator selects for analysis. The developer therefore should perform a careful analysis before he initiates an evaluation at the EAL4 level on potential legal implications of the requirement to provide the evaluators access to the complete implementation representation of the TSF.

536 On the other hand, requiring the presentation of the full implementation representation avoids discussions with the developer on the size and the parts of the sample he has to provide. The only guidance on how to select this sample for CC v2.3 was given in the CEM work unit ADV_IMP.1-2. Still the requirements there gave very little advice for the case where the evaluator would have liked to see the implementation representation of a specific part of the TSF and the developer was reluctant to provide this part (for whatever reason). For example if the evaluator had a suspicion that the implementation of a specific part of the security functionality contained a flaw that was not easily detectable by testing, the developer could reject the request for including this part in the sample (potentially to hide the fact that this part is actually flawed).

537 Both versions of the CC require only the analysis of a sample of the implementation representation. The guidance on sampling is also quite similar in both versions (provided in Appendix A.2 of the CEM in both versions). The main difference identified is that Version 2.3 at a first glance seems to give a more precise value for the size of the sample (a minimum size of 20% is stated in Appendix A.2 of CEM v2.3) while Version 3.1 does not specify such a minimum value for the sample size. Looking closer into this aspect one recognizes that Appendix A.2 of CEM v2.3 in the case of the sample of the implementation representation does not specify sufficiently what the base for the sampling is and how those 20% should be selected. A software vendor could well provide a sample of significantly more than 20% of the source of the TSF, but the way this is selected would make the sample completely useless. For example just presenting the first 50% of the implementation representation of each module would not result in a useful sample, although the size of the sample is significantly larger than suggested by the guidance in appendix A.2 of CEM v2.3. From this aspect, requesting the vendor to present the full implementation representation and leaving it completely to the evaluator to select the sample is an advantage of CC v3.1 over CC v2.3.

538 The second aspect requires providing the implementation representation in an understandable form. While there may be a significant amount of discussion what this actually means, there are a number of examples where one definitely state that – although the data presented is without doubt the "implementation representation" – its form is not suitable for human analysis. Such examples include the implementation representation presented in a high-level programming language but generated by some complex preprocessor, which e. g. includes automatically generated, non-mnemonic variable names or presenting the implementation representation as the assembler language output produced by a high-level language compiler. CC v3.1 names this "shrouded" or "obfuscated" forms of the implementation representation and gives the evaluator the possibility to reject this as an inadequate form of the implementation representation.

539 The intention here is to prohibit that the developer delivers his implementation representation in a form that cannot be understood and evaluated. Still there are some concerns with non-cooperative developers that are not addressed in CC v3.1 that may eventually result in large difficulties for the evaluator to perform his job. Those concerns are:

- For highly complex TOEs just providing access to the implementation representation and not giving access to tools used by the developer to more easily analyze and navigate through the implementation representation can result in very inefficient evaluator work. In cases where the developer does not allow the evaluation facility to install the implementation representation on a system under their control, the evaluator will also in most cases not be able to use his own tools for analyzing the implementation representation and make navigation through it easier.
- It can not be expected that the evaluation facility has access to tools for analyzing an implementation representation for all kind of languages used there. In such cases, even when the developer allows that the evaluation facility installs the implementation representation on their own system, analyzing and navigating it without the tools used by the developer may result in a very low efficiency of the evaluator work.
Both aspects may then lead to a situation where the sample chosen by the evaluator is unnecessary small, since otherwise the overall effort spent for the work units of ADV_IMP.1 would be very high up to being impracticable.

In those cases it is therefore recommended that the evaluator requests access to the tools used by the developer himself to analyze and navigate through the implementation representation to perform this work unit.

5.3. Mapping CEM Version 3.1 work units for ADV_IMP to CC v2.3

The following section lists the work units of the CEM for ADV_IMP.1 and discusses, if and where related information can be found in documentation of a CC v2.3 evaluation. The structure of this chapter is to address the individual requirements as stated in the CC and CEM version 3.1 and attempt to map those – where possible – to corresponding (or closely corresponding) requirements and work units of CC and CEM Version 2.3. Differences are discussed with their implications on the evidence the developer has to provide and the work the evaluator has to perform.

<table>
<thead>
<tr>
<th>CC/CEM v3.1</th>
<th>CC/CEM v2.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_IMP.1.1E</td>
<td>ADV_IMP.1-1</td>
</tr>
<tr>
<td>ADV_IMP.1.1C</td>
<td>ADV_IMP.1-2</td>
</tr>
<tr>
<td>ADV_IMP.1.2C</td>
<td>ADV_IMP.1-3</td>
</tr>
<tr>
<td>ADV_IMP.1.3C</td>
<td>ADV_RCR.1-4</td>
</tr>
</tbody>
</table>

Table 13: Mapping work units for ADV_IMP.1 to CEM v2.3

5.3.1. ADV_IMP.1

ADV_IMP.1 in CC v3.1 is the component in the ADV_IMP family used in EAL4. In CC v2.3 the assurance level EAL4 also used the component ADV_IMP.1 of CC v2.3. There is no requirement from the ADV_IMP family in EAL1 to EAL3 both in CC v3.1 as well as CC v2.3.

544 ADV_IMP.1.1C

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

546 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.

547 Related work units from CC v2.3

548 ADV_IMP.1-1

The evaluator shall examine the implementation representation to determine that it unambiguously defines the TSF to a level of detail such that the TSF can be generated without any further design decisions.

549 Discussion

550 The difference between the two requirements is that ADV_IMP.1-1 in CC v2.3 requires the evaluator to examine the implementation representation for this aspect, while CC v3.1 only requires the evaluator to perform a check. See he definitions of "check" and "examine" in chapter 4 of CEM v3.1.

551 ADV_IMP.1.2C

552 The implementation representation shall be in the form used by the development personnel.
The evaluator shall check that the implementation representation is in the form used by development personnel.

**554 Related work units from CC v2.3**

None

**556 Discussion**

This work unit is new in CC v3.1 and shall ensure that the evaluator receives the implementation representation in a form suitable for his analysis. The discussion section for further details and concerns related to this aspect.

**558 ADV_IMP.1.3C**

The mapping between the TOE design description and the sample of the implementation representation shall demonstrate their correspondence.

The evaluator shall examine the mapping between the TOE design description and the sample of the implementation representation to determine that it is accurate.

**561 Related work units from CC v2.3**

The evaluator shall examine the implementation representation subset to determine that it accurately instantiates those TOE security functional requirements relevant to the subset.

The evaluator shall examine the correspondence analysis between the low-level design and the subset of the implementation representation to determine that the subset is a correct and complete representation of those portions of the low-level design that are refined in the implementation representation.

**564 Discussion**

In CC v2.3 the developer was required to provide a "correspondence analysis" between the low-level design and the sample of the implementation representation used by the evaluator. This was stronger than just the "mapping" as required by CC v3.1. Such a mapping (as required by CC v3.1) was either a part of the correspondence analysis or could be easily derived from the correspondence analysis.

In practice most developers just provided a mapping from the low-level design to the source code as a correspondence analysis. In those cases this mapping can also be used directly in a CC v3.1 evaluation.

**5.3.2. CC v2.3 work units no longer included in CC v3.1**

CC v2.3 had the following requirement and associated work units related to the IT environment:

The evaluator shall examine the implementation representation provided by the developer to determine that it is sufficiently representative.
This work unit is no longer required, since the developer now has to present the implementation representation of the whole TSF.

ADV_IMP.1-3

The evaluator shall examine the implementation representation to determine that it is internally consistent.

Checking for consistency is an inherent activity the evaluator is required to perform. Therefore this work unit has been removed in CC v3.1 as have been similar work units related to checking for internal consistency in other components of ADV.

6. Transition Guide for ADV Requirements for EAL5

The CEM for CC v2.3 did not include the evaluation guidance for requirements of the ADV class that were not included in the levels EAL1 to EAL4. This has changed with CC v3.1, where the CEM also contains the evaluation methodology for components included in assurance levels higher than EAL4.

Except for components of the ALC_FLR family, with CC v2.3 the evaluation methodology for assurance components not included in EAL1 to EAL4 was left to the national schemes. In Germany, the evaluation methodology for the assurance components included in the EAL5 level of CC v2.3 was defined by the AIS34 document. This transition guide references the AIS34 for all aspects related to the evaluation methodology and therefore provides direct guidance related to the transition of the evaluation methodology only with respect to the German CC scheme. Other schemes may have their own methodology for assurance components not included in CEM Version 2.3 and therefore statements made in this document may not necessarily apply when performing a transition from CC v2.3 to CC v3.1 for evaluations at the EAL5 level performed under a scheme different from the German one. A vendor or evaluation facility that wants to perform such a transition should therefore seek additional guidance from the scheme they use for the evaluation.

Because of this difference, the transition guidance for assurance components of the ADV class included in EAL5 but not in lower assurance levels is handled separately in this document.

To show the differences from EAL 4, the tables showing the assurance components for ADV included in the different assurance levels in CC v3.1 and CC v2.3 are repeated in the following tables, with specific emphasis on EAL5.

<table>
<thead>
<tr>
<th>Assurance Class</th>
<th>Assurance Family</th>
<th>Assurance Components by Evaluation Assurance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EAL1</td>
</tr>
<tr>
<td>Development</td>
<td>ADV_FSP</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_HLD</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_IMP</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_INT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_LLD</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_RCR</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>ADV_SPM</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 14: ADV Components in CC v2.3 (emphasis on EAL5)

The same mapping for CC v3.1 is:

<table>
<thead>
<tr>
<th>Assurance Class</th>
<th>Assurance Family</th>
<th>Assurance Components by Evaluation Assurance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>EAL1</td>
</tr>
<tr>
<td>Development</td>
<td>ADV_ARC</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 15: ADV Components in CC v3.1 (emphasis on EAL5)

6.1. Summary of CC v2.3 ADV Requirements for EAL5

577 Table 14 shows that in CC v2.3 the assurance level EAL5 includes the following new components (compared to EAL4) from the ADV class:

- ADV_FSP.3
- ADV_HLD.3
- ADV_IMP.2
- ADV_INT.1
- ADV_RCR.2
- ADV_SPM.3

578 To summarize, the additional requirements compared to EAL4 for CC v2.3 are:

579 ADV_FSP.3

It is required that the functional specification is provided in a semiformal style, supported by informal, explanatory text where appropriate.

580 ADV_HLD.3

It is required that the high-level design is presented in a semiformal style.

It is required that the high-level design describes the complete details of all effects, exceptions and error messages.

581 ADV_IMP.2

It is required that the full implementation representation of the TSF is provided.

It is required that the evaluator performs an analysis of the full implementation representation of the TSF.

582 ADV_INT.1

Assurance levels EAL1 to EAL4 do not include any component from the ADV_INT family. So all the requirements stated in ADV_INT.1 are new at EAL5. In summary ADV_INT.1 requires the TSF to be structured in a modular fashion (described in an "architectural description", which should not be confused with the "security architecture description" as defined in CC v3.1) where unnecessary interactions between modules are avoided. Further ADV_INT.1 requires a description of the purpose, interfaces, parameters, and effects of each module of the TSF.

583 ADV_RCR.2

It is required that the demonstration of correspondence between the formal model and the functional specification as well as the demonstration of the correspondence between the functional specification and the high-level design is semiformal.

A full correspondence analysis between the low-level design and the implementation representation is required.
It is required to present a formal model of the TOE Security Policy.

As one can see from this summary there is some overlap in CC v2.3 between ADV_INT.1 and ADV_LLD.1, which both require a description at the module level and a description of the module purpose, interfaces and effects. This has been taken into account in AIS 34 and will be discussed in the detailed analysis.

6.2. Summary of CC v3.1 ADV Requirements for EAL5

Table 15 shows that in CC v3.1 the assurance level EAL5 included the following new components (compared to EAL4) from the ADV class:

- ADV_FSP.5
- ADV_INT.2
- ADV_TDS.4

As one can see immediately from this list, CC v3.1 does not include any requirement for a Security Policy Model at EAL5. Other differences compared to ADV requirements of EAL4 in CC v3.1 are:

6.3. Summary of Differences between CC v2.3 and CC v3.1 for the ADV Class Requirements of EAL5

Before going into the detailed analysis of the differences a short discussion of the major differences that show up already in this summary. As already mentioned, the immediately obvious difference in the ADV requirements for EAL5 between CC v2.3 and CC v3.1 is the omission of the requirement for a formal security policy model in CC v3.1. In addition also the semiformal correspondence analysis between the functional specification and the high-level design is no longer required. Instead the mapping between the functional specification and the design required in CC v3.1 is identical to the requirements for EAL4.

6.4. Detailed Analysis and Transition Guidance for EAL5

6.4.1. ADV_FSP

In CC v3.1 the assurance level EAL5 includes the component ADV_FSP.5. Compared to the component ADV_FSP.4 which included in the EAL4 assurance level of CC v3.1, the following requirements have been added:
ADV_FSP.5.2C The functional specification shall describe the TSFI using a semi-formal style.

ADV_FSP.5.7C The functional specification shall describe all error messages that do not result from an invocation of a TSFI.

ADV_FSP.5.8C The functional specification shall provide a rationale for each error message contained in the TSF implementation that does not result from an invocation of a TSFI.

Note: There is a typographic error in ADV_FSP.5.8C. The word "yet" has been replaced by "that" in the above citation of the requirement.

The following table maps the work units of ADV_FSP.5 in CEM v3.1 to the work units for ADV_FSP.3 as defined in AIS 34.

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>AIS Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_FSP.5.1E</td>
<td>ADV_FSP.5-1</td>
<td>ADV_FSP.3-8</td>
<td>Identical</td>
</tr>
<tr>
<td>ADV_FSP.5.1C</td>
<td>ADV_FSP.5-2</td>
<td>ADV_FSP.3-9</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_FSP.5-2C</td>
<td>ADV_FSP.5-3</td>
<td>ADV_FSP.3-2</td>
<td>Essence</td>
</tr>
<tr>
<td>ADV_FSP.5.3C</td>
<td>ADV_FSP.5-4</td>
<td>ADV_FSP.3-7</td>
<td>may contribute</td>
</tr>
<tr>
<td>ADV_FSP.5-4A</td>
<td>ADV_FSP.3-5</td>
<td>Essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-5</td>
<td>ADV_FSP.3-6</td>
<td>Essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-6</td>
<td>ADV_FSP.3-7</td>
<td>Essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-7</td>
<td>ADV_FSP.3-7</td>
<td>Essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-8</td>
<td>ADV_FSP.3-7</td>
<td>Essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-9</td>
<td>ADV_FSP.3-7</td>
<td>Essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-10</td>
<td>ADV_FSP.3-7</td>
<td>may contribute</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-11</td>
<td>ADV_FSP.3-7</td>
<td>may contribute</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-12</td>
<td>ADV_RCR.1-1</td>
<td>essence</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-13</td>
<td>ADV_FSP.3-10</td>
<td>identical</td>
<td></td>
</tr>
<tr>
<td>ADV_FSP.5-14</td>
<td>ADV_FSP.3-11</td>
<td>identical</td>
<td></td>
</tr>
</tbody>
</table>

Table 16: Mapping work units for ADV_FSP5 to AIS 34

The following table shows the mapping of between the work units of ADV_FSP.4 and ADV_FSP.5 of CEM v3.1. The table shows that ADV_FSP.5 includes three new work units compared to ADV_FSP.4 while all work units of ADV_FSP.4 (except ADV_FSP.4-4, which is an editorial mistake) are fully and without additional aspects included in ADV_FSP.5.

<table>
<thead>
<tr>
<th>ADV_FSP.4 (EAL4)</th>
<th>ADV_FSP.5 (EAL5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_FSP.4-1</td>
<td>ADV_FSP.5-1</td>
</tr>
<tr>
<td>ADV_FSP.4-2</td>
<td>ADV_FSP.5-3</td>
</tr>
<tr>
<td>ADV_FSP.4-3</td>
<td>ADV_FSP.5-4</td>
</tr>
<tr>
<td>ADV_FSP.4-4</td>
<td>-</td>
</tr>
<tr>
<td>ADV_FSP.4-5</td>
<td>ADV_FSP.5-5</td>
</tr>
<tr>
<td>ADV_FSP.4-6</td>
<td>ADV_FSP.5-6</td>
</tr>
<tr>
<td>ADV_FSP.4-7</td>
<td>ADV_FSP.5-7</td>
</tr>
<tr>
<td>ADV_FSP.4-8</td>
<td>ADV_FSP.5-8</td>
</tr>
<tr>
<td>ADV_FSP.4-9</td>
<td>ADV_FSP.5-9</td>
</tr>
<tr>
<td>ADV_FSP.4-10</td>
<td>ADV_FSP.5-12</td>
</tr>
<tr>
<td>ADV_FSP.4-11</td>
<td>ADV_FSP.5-13</td>
</tr>
<tr>
<td>ADV_FSP.4-12</td>
<td>ADV_FSP.5-14</td>
</tr>
<tr>
<td>ADV_FSP.5-2</td>
<td>ADV_FSP.5-2</td>
</tr>
<tr>
<td>ADV_FSP.5-10</td>
<td></td>
</tr>
</tbody>
</table>
Table 17: Corresponding CEM 3.1 work units for ADV_FSP.4 and ADV_FSP.5

599 In CC v2.3 the assurance level EAL5 includes the component ADV_FSP.3. Compared to the component ADV_FSP.2 which is included in the EAL4 assurance level of CC v2.3, the following requirements have been modified (modifications to ADV_FSP.2 are marked in bold, no new requirement has been added):

600 ADV_FSP.3.1C The functional specification shall describe the TSF and its external interfaces using a semiformal style, supported by informal, explanatory text where appropriate.

601 As one can see, both CC v3.1 as well as CC v2.3 require a semiformal description of the TSF interfaces at the EAL5 assurance level. Differences between the two versions of the CC with respect to the ADV_FSP components included in EAL5 are:

- CC v3.1 no longer requires supporting informal explanatory text (the CEM v3.1 in section 8.2.5 contains a general requirement that the evaluator must be able to fully and completely understand the evaluation evidence. This addresses this aspect in general)
- CC v3.1 now explicitly requires the description (with rationale) of error messages that do not result from an invocation of a TSFI

602 CC v2.3 requires already in ADV_FSP.2 the "complete details of all effects, exceptions and error messages". On the other hand this was tight to the description of the TSFI. So it was not obvious from the CC v2.3 criteria whether error messages not related to a TSFI (e. g. log messages related to some internal error detected by the TSF) needed to be described or not. CC v3.1 has clarified this: Starting from ADV_FSP.5 they need to be described as part of the functional specification.

603 This raises an interesting observation. CC v3.1 states as the objective of the functional specification in paragraph 222:

This family levies requirements upon the functional specification, which describes the TSF interfaces (TSFIs). The TSFIs consist of all means for users to invoke a service from the TSF (by supplying data that is processed by the TSF) and the corresponding responses to those service invocations. It does not describe how the TSF processes those service requests, nor does it describe the communication when the TSF invokes services from its operational environment; this information is addressed by the TOE design (ADV_TDS) and Reliance of dependent component (ACO_REL) families, respectively.

604 On the other hand, CC v3.1, part 2 states in paragraph 24:

The set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which resources are accessed that are mediated by the TSF, or information is obtained from the TSF, is referred to as the TSF Interface (TSFI).

605 Error messages that do not result from an invocation of a TSFI are often returned at an interface that is not used to invoke services from the TSF (e. g. when they are returned in a log file or in a message to an administrative user).

606 The example given in the CEM v3.1, which describes an error message related to a case that should never happen, is usually also not just returned at the TSFI used to request a service. If the TSF detect such a case, a restricted state should be entered and an administrative user should be informed, since something unexpected has happened and it may not be sure that the TSF is still able to correctly enforce all SFRs. Also in this case it is very likely that the error message is issued by the TSF at an "output only" interface. Therefore such "output only" interfaces have to be addressed also as part of the TSFI, as described in CC v3.1, part 2, paragraph 24.

607 "Output only" interfaces need to be described with their purpose, method of use (how the output can be obtained) as well as the type and format of output that may be expected to turn up at the interface. Such interfaces usually output status and/or error messages. At EAL5, error messages returned by the TSF via such interfaces need to be completely described as well as all security relevant status information (e. g. information
608 The interpretation here is that such error messages need to be completely described and that the examination of this fact is performed as part of the ADV_FSP related activities. Those error messages may well be described in documents that do not contain any TSFI related information (e.g. there may be a book listing all those error messages with a description at which interfaces the TSF may issue them and potentially with guidance how to react to such errors.

609 The work units in CEM v3.1 related to the additional requirements of ADV_FSP.5 (compared to ADV_FSP.4) are:

610 ADV_FSP.5.2C

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

612 ADV_FSP.5-2

The evaluator **shall examine** the functional specification to determine that it is presented using a semiformal style.

613 Related work units from AIS34

614 ADV_FSP.3-1

The evaluator **shall examine** the functional specification to determine that the semiformal notation used for describing the TSF and its external interfaces is defined or referenced.

615 ADV_FSP.3-2

The evaluator **shall examine** all semiformal notations used to make sure that they are of a semiformal style and to justify the appropriateness of the manner how the semiformal notations are used for the TOE.

616 Discussion

617 There is little difference between the explanation of "semiformal" in CC v3.1 compared to CC v2.3. CC v3.1 explains in paragraph 211 of part 3:

> The difference between semiformal and informal documents is only a matter of formatting or presentation: a semiformal notation includes such things as an explicit glossary of terms, a standardised presentation format, etc. A semiformal specification is written to a standard presentation template. The presentation should use terms consistently if written in a natural language. The presentation may also use more structured languages/diagrams (e.g. data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). Whether based on diagrams or natural language, a set of conventions must be used in the presentation. The glossary explicitly identifies the words that are being used in a precise and constant manner; similarly, the standardised format implies that extreme care has been taken in methodically preparing the document in a manner that maximises clarity. It should be noted that fundamentally different portions of the TSF may have different semiformal notation conventions and presentation styles (as long as the number of different "semiformal notations" is small); this still conforms to the concept of a semiformal presentation.

618 CC v2.3 explains in paragraph 308 of part 3:

> A semiformal specification is written in a restricted syntax language and is typically accompanied by supporting explanatory (informal) prose. The restricted syntax language may be a natural language with restricted sentence structure and keywords with special meanings, or it may be diagrammatic (e.g. data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). Whether based on diagrams or natural language, a set of conventions must be supplied to define the restrictions placed on the syntax.
This shows that both version of the CC have an identical view of what a semiformal notation is. Therefore a semiformal description of the TSFI used successfully for a CC v2.3 evaluation should also satisfy the requirement for a semiformal description of the TSFI in a CC v3.1 evaluation.

620 ADV_FSP.5.4C

621 The functional specification shall identify and describe all parameters associated with each TSFI.

622 Within the work units associated with this requirement, work unit ADV_FSP.4-4 has not been repeated. This is a simple omission error. The evaluator needs to perform the examination of the completeness of the TSFI also for ADV_FSP.5 and it is therefore suggested that he adds the following work unit in his evaluation report:

623 ADV_FSP.5-4A

The evaluator shall examine the functional specification to determine the completeness of the TSFI.

624 Related work units from AIS34

625 ADV_FSP.3-5

The evaluator shall examine the functional specification to determine that it identifies all of the external TOE security function interfaces.

626 ADV_FSP.3-6

The evaluator shall examine the functional specification to determine that it describes all of the external TOE security function interfaces.

627 Discussion

628 Guidance how to perform this work unit can be taken from CEM v3.1, work unit ADV_FSP.4-4. The evaluator, when performing the analysis for ADV_FSP.5 needs to verify that the functional specification also is complete with respect to interfaces that can not be used to request services from the TSF but may be used by the TSF to return error messages or security relevant status information. AIS34 does not provide additional guidance to the two related CEM v2.3 work units and therefore the discussion for ADV_FSP.4-4 applies here. When examining the completeness of the TSFI, the evaluator needs also to consider interfaces where the TSF just output information and verify that the TSFI lists all such interfaces where error messages and/or security relevant status information leaves the TSF. This includes also all "output only" interfaces where such information is presented by the TSF. "Output only" interfaces not designed to return such security relevant information are not considered to be part of the TSFI at ADV_FSP.5, but may still be relevant for inclusion in the vulnerability analysis to determine that they have no security relevant side effects.

629 ADV_FSP.5.7C

630 The functional specification shall describe all error messages that do not result from an invocation of a TSFI.

631 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.

632 ADV_FSP.5.8C

633 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.
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.

### Related work units from AIS34

636 ADV_FSP.3-7

The evaluator **shall examine** the presentation of the TSFI to determine that it adequately and correctly describes the complete behaviour of the TOE at each external interface describing effects, exceptions and error messages.

### Discussion

638 Quite often products have security related error messages that do not result from an invocation of a TSFI. Examples are error messages produced by detecting an error state resulting from some TSF internal checking or checks of the environment the TSF relies upon (like an error state returned from a disk the TSF uses to store TSF data). Even in cases where the TSF may be able to identify the user's request where the error was detected as part of processing this request, it may well be inappropriate (if not even a violation of the security policy) to return the error at the TSFI that was called. An example is a firewall system that detects a disk error while processing a packet received from the external interface. Providing an error message back to the originator of the IP packet indicating that a TSF internal error was detected as part of the processing is not what one expects the product to do. Instead those error messages are quite often returned at an "output only" interface like a log file or a message sent to an administrative user.

639 CC v2.3 states in paragraph 316 of part 3:

*The functional specification is a high-level description of the user-visible interface and behaviour of the TSF. It is an instantiation of the TOE security functional requirements. The functional specification has to show that all the TOE security functional requirements are addressed.*

640 CC v2.3 therefore included all user-visible interfaces and behaviour, which also includes "output-only" interfaces. In CC v3.1, part 2, paragraph Work unit ADV_FSP.3-7 of AIS34 therefore applies also to error messages that are not a result from an invocation of a TSFI, requiring that they are described (as part of the overall set of error messages).

641 In a similar way CC v3.1, part 2, paragraph 64 defines the TSFI as:

*The set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which resources are accessed that are mediated by the TSF, or information is obtained from the TSF, is referred to as the TSF Interface (TSFI).*

642 In order to address the additional requirements for error messages included in ADV_FSP.5 of CC v3.1, one has to also identify and consider all interfaces where those error messages are "exported" from the TSF, although those interfaces usually require a simpler description since they do not involve the description of input parameter, actions etc. but just a description of the method of use and the output parameter. Work units ADV_FSP.5-10 and ADV_FSP.5-11 require to analyze for all error messages that do not result from an invocation of a TSFI:

- The conditions under which the error message is generated
- The interface used to "export" the error message from the TSF
- The definition of the format of the error message (well defined syntax, well defined semantics of the individual fields in the message)

643 As part of the analysis of the design the evaluator shall identify if the information provided about the conditions under which the error message is generated is consistent with the information provided in the design.
644 As part of the analysis of the guidance documentation the evaluator shall identify if the information provided in the error message allows the user of the interface where the message is exported to correctly interpret the message and take the appropriate action.

645 **Note:** Work unit ADV_FSP.4-4 requiring the determination of the completeness of the TSFI has not been replicated to ADV_FSP.5. This is an editorial omission in CEM v3.1 R2. An evaluator analysis of the completeness of the TSFI is also required at ADV_FSP.5.

646 The evaluator needs to perform the required analysis of the completeness of the TSFI in the same way as it is required for ADV_FSP.4 and use the guidance provided in work unit ADV_FSP.4-4. The evaluator needs to also examine that the TSFI lists all interfaces used to export error messages from the TSF with a complete list of all security relevant error messages that may be exported via those interfaces.

### 6.4.2.ADV_IMP

647 In CC v3.1 the component for ADV_IMP included in EAL5 is the same component (ADV_IMP.1) that was already included in EAL4. In CC v2.3 assurance level EAL5 included the component ADV_IMP.2, while EAL4 included component ADV_IMP.1. As was discussed above in chapter 5, the work required by the evaluator for ADV_IMP.1 of CC v2.3 covers basically all aspects required for ADV_IMP.1 of CC v3.1. The main difference is just that the developer for ADV_IMP.1 of CC v3.1 has to provide the full implementation representation of the TSF, allowing the evaluator to freely select the subset he wants to analyze. For ADV_IMP.2 in CC v2.3, the developer is also required to present the implementation representation of the entire TSF and in addition it is required that the "implementation representation shall describe the relationships between all portions of the implementation".

648 The first aspect is identical to the requirement in CC v3.1 for ADV_IMP.1, where also the implementation representation of the entire TSF is required. The second item is reflected by the following work units in AIS34:

649 ADV_IMP.2-4

The evaluator *shall examine* the implementation representation to determine that it describes the relationships between all portions of the implementation.

650 ADV_IMP.2-5

The evaluator *shall examine* the implementation representation to determine that it is an accurate instantiation of the TOE security functional requirements.

651 ADV_IMP.2-6

The evaluator *shall examine* the implementation representation to determine that it is a complete instantiation of the TOE security functional requirements.

652 Those work units, which require a complete mapping between the SFRs and the implementation representation are not included in CEM v3.1. Only a mapping between the TOE design and the sample of the implementation representation as selected by the evaluator is required.

653 In addition to those work units, AIS34 required a complete correspondence analysis between the low-level design and the implementation representation in the following work unit:

654 ADV_RCR.2-6

The evaluator *shall examine* the correspondence analysis between the low-level design and the implementation representation to determine that the implementation representation is a correct and complete representation of the low-level design.

### 655 Discussion

656 The requirements for the analysis of the implementation representation at the assurance level EAL5 are significantly lower in CC v3.1 compared with CC v2.3. Although both versions of the CC require the developer to
provide the complete implementation representation of the TSF, the evaluator in a CC v3.1 evaluation at EAL5 is only required to analyze a sample of the implementation representation and only analyze the mapping between the TOE design and this selected sample for correctness. This is already required at the EAL4 assurance level and the discussion in chapter 5 of this document has shown that the results of the analysis of the implementation representation of an EAL4 evaluation using CC v2.3 satisfy mainly the requirements of ADV_IMP.1 of CC v3.1. Since ADV_IMP.1 is also the assurance component included in EAL5, the significant additional analysis work required from the evaluator as well as the full correspondence analysis between the low-level design and the implementation representation that needs to be provided by the developer for a CC v2.3 evaluation at EAL5 are no longer required for an evaluation at the same assurance level using CC v3.1. In summary, the requirements for the analysis of the implementation representation at EAL5 have been reduced significantly with CC v3.1.

6.4.3 ADV_INT

657 In CC v3.1 the assurance level EAL5 includes the component ADV_INT.2. No component from the ADV_INT family is included in the assurance level EAL4. Therefore all requirements of ADV_INT.2 are additional to the requirements included in EAL4. They are:

658 Developer action elements:

659 ADV_INT.2.1D The developer shall design and implement the entire TSF such that it has well-structured internals.

660 ADV_INT.2.2D The developer shall provide an internals description and justification.

661 Content and presentation elements:

662 ADV_INT.2.1C The justification shall describe the characteristics used to judge the meaning of "well-structured".

663 ADV_INT.2.2C The TSF internals description shall demonstrate that the entire TSF is well-structured.

664 CC v2.3 included component ADV_INT.1 with the following requirements:

665 Developer action elements:

666 ADV_INT.1.1D The developer shall design and structure the TSF in a modular fashion that avoids unnecessary interactions between the modules of the design.

667 ADV_INT.1.2D The developer shall provide an architectural description.

668 Content and presentation of evidence elements:

669 ADV_INT.1.1C The architectural description shall identify the modules of the TSF.

670 ADV_INT.1.2C The architectural description shall describe the purpose, interface, parameters, and effects of each module of the TSF.

671 ADV_INT.1.3C The architectural description shall describe how the TSF design provides for largely independent modules that avoid unnecessary interactions.

672 As one can easily see, both sets of requirements are quite different in their text, but have in common that they do not clearly define the meaning of "well-structured" (CC v3.1) or "modular fashion" resp. "unnecessary interaction" (CC v2.3). The intention of both sets of requirements is to require the TSF to have a structure that allows for a thorough analysis. This reflects the "small enough to be analysed" requirement of a "Reference Monitor" as defined in the Anderson Report [AND72], which was one of the fundamental basis documents for the TCSEC. In this report the three fundamental principles of a "Reference Monitor" have been defined as:

a) The reference validation mechanism must be tamper proof.
b) The reference validation mechanism must always be invoked.

c) The reference validation mechanism must be small enough to be subject to analysis and tests to assure that it is correct.

673 Condition c) was interpreted to use a method for structuring the allows for easy understanding and supports the analysis of the functionality to ensure that it is complete and correct.

674 If an IT product is well-structured can be subjective. CC v2.3 required a specific technique – modular design with minimized interaction between modules – that is just one technique supporting “well-structured”. Other techniques may also result in "well-structured" TSF and therefore CC v3.1 attempts to not prescribe the technique used for structuring the TSF. This results in more generic requirements and can make the assessment more subjective. On the other hand, all more objective measures would imply restrictions on the design methodology that can be used, an aspect the CC does not want to limit unnecessarily.

675 It is therefore hard to map the individual requirements of the two versions of the CC to each other. The following discussion attempts to provide the essence of the two versions of the criteria and then compare them on a general level.

676 ADV_INT.1 in CC v2.3

677 CC v2.3 requires for ADV_INT.1 an "architectural description" that "identifies the modules of the TSF". Since also the low-level design (ADV_LLD) is required to "describe the TSF in terms of modules", the question arises if the modules in the architectural description and in the low-level design have to be the same or if two different "module" structures of the TSF are allowed. AIS 34 solves this problem by allowing different module structures for low-level design and architectural description, but discourages to do so by requiring significant additional effort to show that the different module structures are compatible with each other in terms of interoperation between modules and the provision of the security functionality. In addition both module structures must be mapped to the implementation representation, which requires additional work. All this work can be saved when the module structure in the low-level design and the architectural description are identical.

678 Looking into the work units AIS 34 requires for ADV_INT.1 of CC v2.3, one comes up with the following list:

679 ADV_INT.1-1

The evaluator shall check the architectural description to determine that it identify a module structure and all modules of the low-level design are covered by this module structure.

680 ADV_INT.1-2

The evaluator shall examine the architectural description to determine that all interfaces of modules of the low-level design are covered in the modularity analysis.

681 ADV_INT.1-3

The evaluator shall examine the architectural description to determine that it describes how the TSF design provides for largely independent modules that avoid unnecessary interactions.

682 ADV_INT.1-4

The evaluator shall examine the architectural description to determine that the low-level design is in compliance with the architectural description.

683 ADV_INT.1-5

The evaluator shall examine the architectural description to determine that the implementation representation is in compliance with the architectural description.
As one can see, this all comes down to work unit ADV_INT.1-3 in the case the module structure of the low-level design and the architectural description are identical. In this (natural) case, the only aspect the evaluator has to examine as part of the evaluation of ADV_INT is the analysis of the dependencies and interactions between modules to verify that unnecessary dependencies and interactions are avoided. As AIS 34 points out, the necessity for interaction between modules may be caused by non security related aspects like performance aspects and this must be also accepted by the evaluator as an argument why the interaction is necessary. It is worth to be noted that ADV_LLD in CC v2.3 required the definition of "the interrelationships between the modules in terms of provided security functionality and dependencies on other modules" (ADV_LLD.1.5C), which should cover at least a part of the definition of the overall interrelationships and dependencies between modules.

ADV_INT.2 in CC v3.1

In CC v3.1 the "architectural description" of CC v2.3 has turned into a "TSF internal description" (although the CEM v3.1 in 11.6.1.2 still mentions an "architectural description" as a required input, which is then never referenced in the work units. This is most likely the "TSF internals description" referenced in the work units).

Note: the term "architectural description" is not used in a consistent way in CC v3.1. CEM 8.5.5.3.2 describes the "architectural description" as a part of the ETR where the evaluator reports the high level description of the TOE and its major components. Paragraph 537 and 546 of CEM v3.1 mention the term "architectural description" but mean the TOE security architecture description. Section 11.6.1.2 mention the "architectural description" as an input document for ADV_INT meaning the TSF internals description. In work unit ADV_TDS.4 the architectural description is mentioned as other evidence that could provide input to the work unit, meaning obviously the TOE security architecture description. Paragraph 1638 talking about composed TOEs mentions the term architectural description, meaning the evaluators report in the ETR as part of the public documentation.

The evaluator should keep in mind that the term "architectural description" in CC v3.1 is related to the overview description of the TOE architecture in the ETR. The evaluator should assign the meanings listed in the note above to the term "architectural description" in the paragraphs mentioned.

Work units related to ADV_INT.2 in CEM v3.1:

ADV_INT.2.1C

The justification shall describe the characteristics used to judge the meaning of "well-structured".

ADV_INT.2-1

The evaluator shall examine the justification to determine that it identifies the basis for determining whether the TSF is well-structured.

ADV_INT.2.2C

The TSF internals description shall demonstrate that the entire TSF is well-structured.

ADV_INT.2-2

The evaluator shall examine the TSF internals description to determine that it demonstrates that the TSF is well-structured.

ADV_INT.2-3

The evaluator shall determine that the TOE design is well-structured.

ADV_INT.2-4

The evaluator shall determine that the TSF is well-structured.
698 As one can see from those work units, CC v3.1 has shifted the focus of ADV_INT from minimizing interaction and dependencies between modules (as it was in CC v2.3) to an examination that the TSF is "well-structured". The criteria for "well-structured" are technology dependent and therefore the evaluator is advised in the guidance for ADV_INT.2-1 to seek guidance from his scheme for determining the adequacy of criteria for being "well-structured". It can be assumed that some minimization of interdependencies and interactions between modules is such an acceptable criteria for being well-structured, which would then make the arguments used in a CC v2.3 evaluation for ADV_INT.1 also acceptable for this assurance component in a CC v3.1 evaluation. In addition there may be other criteria that are acceptable, which then would allow TOEs that would have failed the requirements for ADV_INT.1 in a CC v2.3 evaluation to eventually pass the ADV_INT.2 criteria in a CC v3.1 evaluation, but this depends on the acceptance of the criteria for being "well-structured" that are applied in the evaluation. Since those criteria are not defined in CC v3.1 due to the dependence on the technology used to design and implement the TOE, an evaluator should always discuss the acceptance of the TOE specific criteria for being well-structured with his national body before performing an analysis.

6.4.4. ADV_TDS

699 In CC v3.1 the assurance level EAL5 includes the component ADV_TDS.4. Compared to the component ADV_TDS.3 which included in the EAL4 assurance level of CC v3.1, the following requirements have been modified compared to ADV_TDS.3 (modifications marked in bold):

700 ADV_TDS.4.2C The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-supporting, or SFR-non-interfering.

701 ADV_TDS.4.4C The design shall provide a semiformal description of each subsystem of the TSF, supported by informal, explanatory text where appropriate.

702 ADV_TDS.4.7C The design shall describe each SFR-enforcing and SFR-supporting module in terms of its purpose and interaction with other modules.

703 ADV_TDS.4.8C The design shall describe each SFR-enforcing and SFR-supporting module in terms of its SFR-related interfaces, return values from those interfaces, interaction with and called interfaces to other modules.

704 This shows that at EAL5 CC v3.1 basically increases the requirements for the design documentation of the SFR-supporting modules, lifting it to the level of description required for the SFR-enforcing modules already for ADV_TDS.3. The level of detail required for the SFR-non-interfering modules does not increase from ADV_TDS.3 to ADV_TDS.4. Concerning the description at the subsystem level, ADV_TDS.4 requires a semiformal description.

705 In CC v2.3, EAL5 includes the component ADV_HLD.3 while the component for the low-level design (ADV_LLD.1) is identical to the one included in EAL4.

706 ADV_HLD.3 has to following modified requirements compared to ADV_HLD.2:

707 ADV_HLD.3.1C The presentation of the high-level design shall be semiformal.

708 ADV_HLD.3.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing complete details of all effects, exceptions and error messages.

709 As one can see, both CC v3.1 as well as CC v2.3 require a semiformal description at the subsystem level. But while CC v2.3 increases the requirements on the description of the interfaces between subsystems, CC v3.1 increases the requirements on the description at the module level (the low-level design in CC v2.3). Component ADV_TDS.4 requires an explicit designation of the modules as SFR-enforcing, SFR-supporting and SFR-non-interfering (at ADV_TDS.3 this was implicit and required only the identification of the SFR-enforcing modules). The level of description for the SFR-supporting modules has been increased and requires the same level of detail as ADV_TDS.3 requires for the SFR-enforcing modules.

710 This is in line with the shift in the paradigm between CC v2.3 and CC v3.1 where CC v3.1 requires more details on the internals and interactions between modules. This also reflects the emphasis CC v3.1 puts on the description at the module level with specific focus on those modules that either directly implement security functional requirements or support modules that do.
The modifications between ADV_TDS.3 and ADV_TDS.4 in CC v3.1 are reflected in the CEM in a set of new work units and some modifications to existing work units. The new work units are related to the semiformal description at the subsystem level and the required categorization of the modules of the TSF as SFR-enforcing, SFR-supporting or SFR-non-interfering.

The following table maps the work units of ADV_TDS.4 to those of ADV_HLD.3, ADV_LLD.1 and ADV_RCR.1 of CEM v2.3/AIS 34. Work units that are new for ADV_TDS.4 compared to ADV_TDS.3 are marked in bold.

<table>
<thead>
<tr>
<th>CC Requirement</th>
<th>CEM Work Unit</th>
<th>CEM Work Unit</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.4.1E</td>
<td>ADV_TDS.4.1C</td>
<td>ADV_TDS.3-1</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.2C</td>
<td>ADV_TDS.4.3C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.3E</td>
<td>ADV_TDS.4.3C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.4C</td>
<td>ADV_TDS.4.4C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.5C</td>
<td>ADV_TDS.4.5C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.6C</td>
<td>ADV_TDS.4.6C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.7C</td>
<td>ADV_TDS.4.7C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.8C</td>
<td>ADV_TDS.4.8C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.9C</td>
<td>ADV_TDS.4.9C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.10C</td>
<td>ADV_TDS.4.10C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.11C</td>
<td>ADV_TDS.4.11C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.12C</td>
<td>ADV_TDS.4.12C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.13C</td>
<td>ADV_TDS.4.13C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.14C</td>
<td>ADV_TDS.4.14C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.15C</td>
<td>ADV_TDS.4.15C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.16C</td>
<td>ADV_TDS.4.16C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
<tr>
<td>ADV_TDS.4.17C</td>
<td>ADV_TDS.4.17C</td>
<td>ADV_TDS.4-3</td>
<td>mainly identical</td>
</tr>
</tbody>
</table>

Table 18: Mapping work units for ADV_TDS.4 to CEM v2.3/AIS 34

In addition to the table above, the following table presents a mapping of the work units of ADV_TDS.3 to those of ADV_TDS.4, showing where work units are new and where work units have been modified.

<table>
<thead>
<tr>
<th>ADV_TDS.3 (EAL4)</th>
<th>ADV_TDS.4 (EAL5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADV_TDS.3-1</td>
<td>ADV_TDS.4-1</td>
</tr>
<tr>
<td>ADV_TDS.3-2</td>
<td>ADV_TDS.4-3</td>
</tr>
<tr>
<td>ADV_TDS.3-3</td>
<td>ADV_TDS.4-5</td>
</tr>
<tr>
<td>ADV_TDS.3-4</td>
<td>ADV_TDS.4-6</td>
</tr>
<tr>
<td>ADV_TDS.3-5</td>
<td>ADV_TDS.4-7</td>
</tr>
<tr>
<td>ADV_TDS.3-6</td>
<td>ADV_TDS.4-8</td>
</tr>
</tbody>
</table>
Table 19: Corresponding CEM 3.1 work units for ADV_TDS.3 and ADV_TDS.4

(a) in this table indicates that the work unit description has been augmented resulting in additional work to be performed by the evaluator compared to the corresponding work unit of the lower assurance level.

(m) in this table indicates that the text of the work unit has been modified to express the differences in the requirements for the different assurance levels.

- in this table indicates that the work unit is no longer required at the higher level.

This shows that there are two new work units in ADV_TDS.4 and a few work units with increased or modified work for the evaluator. Those are discussed in the following sections.

715 ADV_TDS.4.1C

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

717 ADV_TDS.4-2

The evaluator shall examine the TDS documentation to determine that the semiformal notation used for describing the subsystems, modules and their interfaces is defined or referenced.

It should be noted that the work unit text references "modules and their interfaces" although the requirement ADV_TDS.4.4C requires a semiformal description only at the subsystem level. This reflects that some developers use a semiformal description not only for the externally visible interfaces (where they are at EAL5 required to provide a semiformal description) but also for TSF internal interfaces. Most developers use the same method for describing external and internal interfaces. The work unit therefore requires the evaluator to check TSF internal interfaces that are described using a semiformal notation for a correct definition or reference to the semiformal language used for the description.

CC v3.1 allow a vendor to just provide a design at the module level if the overall complexity of the TSF is low. In the case of an evaluation that includes ADV_TDS.4, in such a case a semiformal method has to be used at the module level, including the description of the interfaces between modules.

Note: This work unit is not placed in line with the requirements in part 3 of CC v3.1. While the requirement for a semiformal description at the subsystem level is contained in ADV_TDS.4.4C, the only work unit associated with this requirement in the CEM v3.1 (ADV_TDS.4-6) has not changed. Instead the new work unit ADV_TDS.4-2 has been introduced as a work unit associated with requirement ADV_TDS.4-1C, which has not changed from ADV_TDS.3. This placement should not have any influence on the work the evaluator needs to perform in the assessment of the subsystem structure and the semiformal description at the subsystem level.

721 Related work units from AIS34

722 ADV_HLD.3-1

The evaluator shall examine the high level design to determine that the semiformal notation used for describing the structure of the TSF, the security functionality of all subsystems and the interfaces of all subsystems is defined or referenced.

723 ADV_HLD.3-2
The evaluator shall examine all semiformal notations used to make sure that they are of a semiformal style and to justify the appropriateness of the manner how the semiformal notations are used for the TOE.

**724 Discussion**

725 CC v3.1 as well as CC v2.3 require a semiformal description at the subsystem level at EAL5. In the paradigm of CC v3.1 the interfaces between subsystems are not as important as in CC v2.3 while the focus is more on the interaction between subsystems. An evaluator reusing a semiformal description of the subsystems from a CC v2.3 evaluation therefore needs to examine that the interaction between the subsystems is sufficiently covered by the semiformal description. Concerning the requirements for a semiformal notation itself, there is no difference between CC v2.3 and CC v3.1 and therefore a semiformal notation acceptable for a CC v2.3 evaluation should also be acceptable for a CC v3.1 evaluation.

**726 ADV_TDS.4.2C**

**727 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-supporting, or SFR-non-interfering.**

**728 ADV_TDS.4-4**

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.

**729 Related work units from AIS34 (Note: Since EAL5 in CC v2.3 includes the same component for ADV_LLD as EAL4, AIS34 just references the CEM v2.3 for the work units related to ADV_LLD)**

730 ADV_LLD.1-10

The evaluator shall check that the low-level design describes the separation of the TOE into TSP-enforcing and other modules.

**731 Discussion**

732 The differentiation between SFR-enforcing, SFR-supporting and SFR-non-interfering is new in CC v3.1. Starting from EAL5 the developer is required to explicitly identify one of those categories for all modules of the TSF.

733 CC v2.3 only required a categorization into "TSP-enforcing" and "other" modules. "TSP-enforcing" in CC v2.3 is defined in the following way (CC v2.3, paragraph 355):

"The term "TSP-enforcing module" refers to any module that must be relied upon for correct enforcement of the TSP."

734 This is very much like the sum of the SFR-enforcing and the SFR-supporting modules. A developer that wants to reuse the low-level design documentation from a CC v2.3 evaluation must therefore analyze the "other" modules in a CC v2.3 evaluation and validate that those are "SFR-non-interfering".

**735 ADV_TDS.4.7C**

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

**737 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.

**738 Related work units from CC v2.3**
The evaluator shall examine the low-level design to determine that it describes the purpose of each module.

**Discussion**

CC v3.1 requires the description of the purpose of all modules that comprise the TSF (see ADV_TDS.4.7C and ADV_TDS.4.9C) while more details about the internals are required for the SFR-enforcing and SFR-supporting modules. This allows therefore to identify the purpose of all modules and therefore is in sum equivalent to work unit ADV_LLD.1-4 of CC v2.3.

**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 SFR-related parameters, the invocation conventions for each interface, and any values returned directly by the interface.

**Related work units from CC v2.3**

The evaluator shall check that the low-level design identifies the interfaces to the TSF modules.

The evaluator shall examine the low-level design to determine that it describes the interfaces to each module in terms of their purpose and method of use, and provides details of effects, exceptions and error messages, as appropriate.

**Discussion**

Here is clearly a difference between CC v3.1 and CC v2.3. While CC v3.1 differentiates between "SFR-related parameter" and other parameter, CC v2.3 does not differentiate. On the other hand, the CC v2.3 work unit ADV_LLD.1-9 has the text "as appropriate" at the end, leaving the decision what is considered to be appropriate to the evaluator. Focusing on the "SFR-related" parameter would for sure be considered as an appropriate method, but there is no guarantee that this has been used in this way in evaluations.

Compared to ADV_TDS.3 this work unit now also applies to the SFR-supporting modules. By definition a SFR-supporting module does not implement directly any SFR but is relied upon by SFR-enforcing modules. A SFR-supporting module by definition will not have any interface or parameter that is directly related to a SFR. Still the functions of SFR-supporting modules are important to understand that they provide their SFR-supporting functionality correctly and without security critical side effects.

A typical SFR-supporting module is a device driver for a block device. Any access control function for objects stored on this device usually has called before the device driver is called to read blocks from or write blocks to the device. Also any device management interface of the device driver is usually called after the requirements expressed in the TSF have been checked to ensure that the user upon whose request the management function should be performed has the right to perform the requested management activity. In order to perform his analysis of the SFR-supporting modules, the evaluator needs a description of all interfaces that are related to the SFR-supporting functionality. The evaluator therefore shall examine that the description of the interfaces presented by each SFR-supporting module contain an accurate and complete description of the parameters related to SFR-supporting functionality, the invocation conventions for each interface, and any values returned directly by the interface. Parameters related to SFR-supporting functionality are those parameters that are used by SFR-enforcing modules when they call a SFR-supporting module. The evaluator shall search the calls to SFR-supporting modules from SFR-enforcing modules and verify that all parameters used in those calls are described completely.
Note: The definition of the analysis the evaluator shall perform related to the interfaces and parameter of SFR-supporting modules is not fully backed by the CEM v3.1. The evaluator is therefore advised to contact his Certification Body and check if the approach described in this guide is accepted.

ADV_TDS.4.9C

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

The following three work units differ from the corresponding ones of ADV_TDS.3 just by omitting the SFR-supporting modules (which are now covered by the more detailed requirements of work units ADV_TDS.4-10 and ADV_TDS.4-11).

ADV_TDS.4-12

The evaluator shall examine the TOE design to determine that SFR-non-interfering modules are correctly categorised.

ADV_TDS.4-13

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.

ADV_TDS.4-14

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.

For a discussion of the related work units from CC v2.3/AIS34 see the section on ADV_TDS.3.8C and the associated work units ADV_TDS.3-10 to ADV_TDS.3-12.

6.4.5. Work Units from CEM v2.3 no longer relevant in CEM v3.1 (for EAL5)

As already mentioned above, CC v3.1 at EAL5 no longer requires a formal security policy model and no longer requires a semi-formal correspondence analysis between the formal security policy model and the functional specification and between the functional specification and the subsystems of the TSF (formerly known as the high-level design). Therefore all work units related to ADV_SPM.3 in AIS34 are obsolete for CC v3.1 at EAL5 (which are ADV_SPM.3-1 to ADV_SPM.3-13). In addition the following work units from AIS34 for EAL5 have no equivalent work unit in CEM v3.1 at EAL5:

- ADV_IMP.2-4
- ADV_IMP.2-5
- ADV_IMP.2-6
- ADV_RCR.2-2
- ADV_RCR.2-3
- ADV_RCR.2-6

This shows that in summary the requirements for EAL5 have been lowered significantly from CC v2.3 to CC v3.1 with respect to the security policy modeling, the analysis of the implementation representation and the demonstration of correspondence between different levels of design documentation.

7. References

The following documents have been used to develop this guide:

| AIS34 | AIS 34, Version 1.00, 01 June 2004, Evaluation Methodology for CC Assurance Classes for EAL5+
<table>
<thead>
<tr>
<th></th>
<th>October 1972, Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), L. G. Hanscom Field, Bedford, Massachusetts 01730</th>
</tr>
</thead>
</table>

[1] Note: This work unit does not show up in CEM v3.1. It is identical to the work unit ADV_FSP.4-4 and requires to check the completeness of the TSFI. See the text in the discussion section for evaluator guidance.

[2] See note above